Incorporating Snakemake into High Performance Computing Workflows

Texas A&M HPRC

Wes Brashear

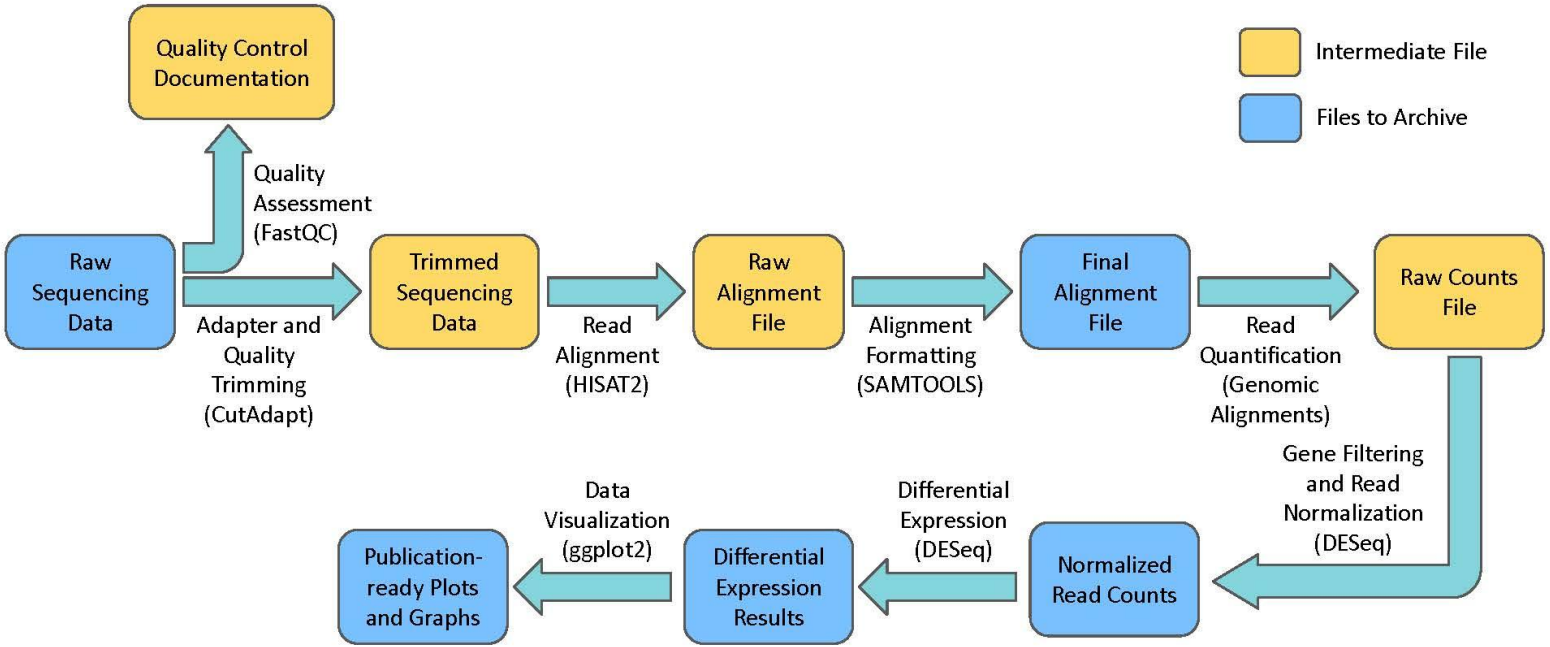
28 April 2026

High Performance
Research Computing
DIVISION OF RESEARCH

Course Outline

- Brief Introduction
- Getting on ACES
- Our first Snakemake job
- Understanding the order of rule execution
- A Complete Bioinformatics Workflow
- Submitting Snakemake jobs through Slurm
- Generating reports
- Snakemake parallelization

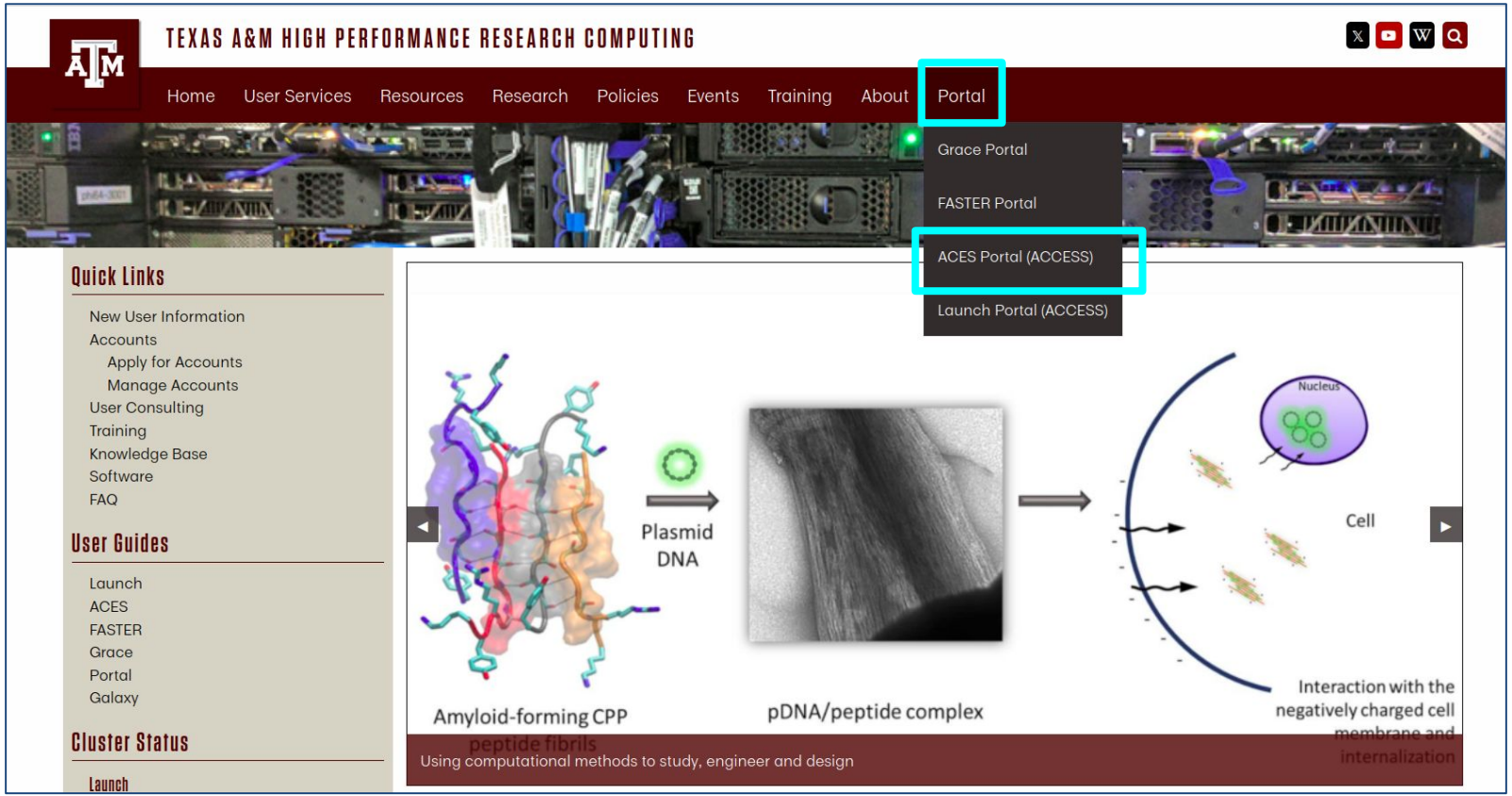
The Need for Workflow Management



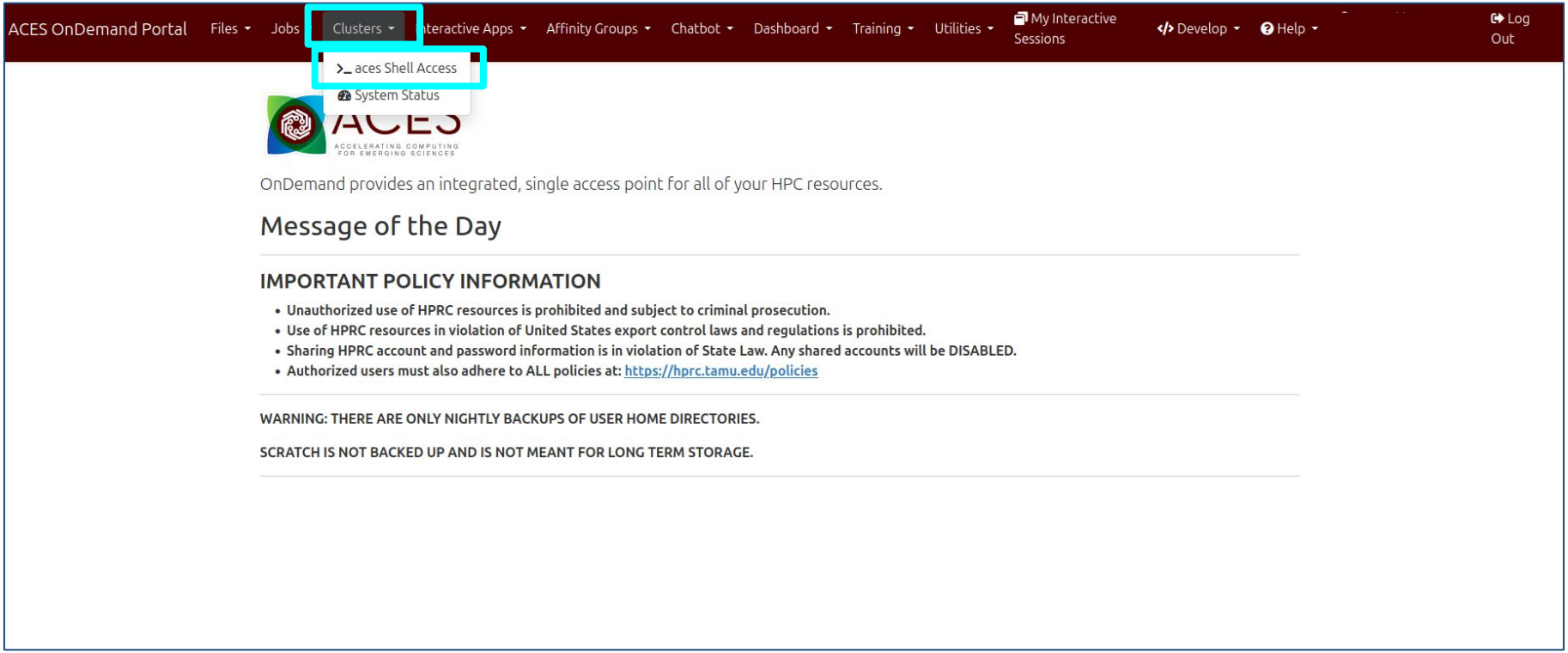
Snakemake

- Python-based workflow management system that allows users to easily create and share workflows
- Broad user community that is rapidly growing
- Provide access to curated and citable workflows
- Integrates with Jupyter (data visualization, report generation)
- Integrates well with batch scheduling systems

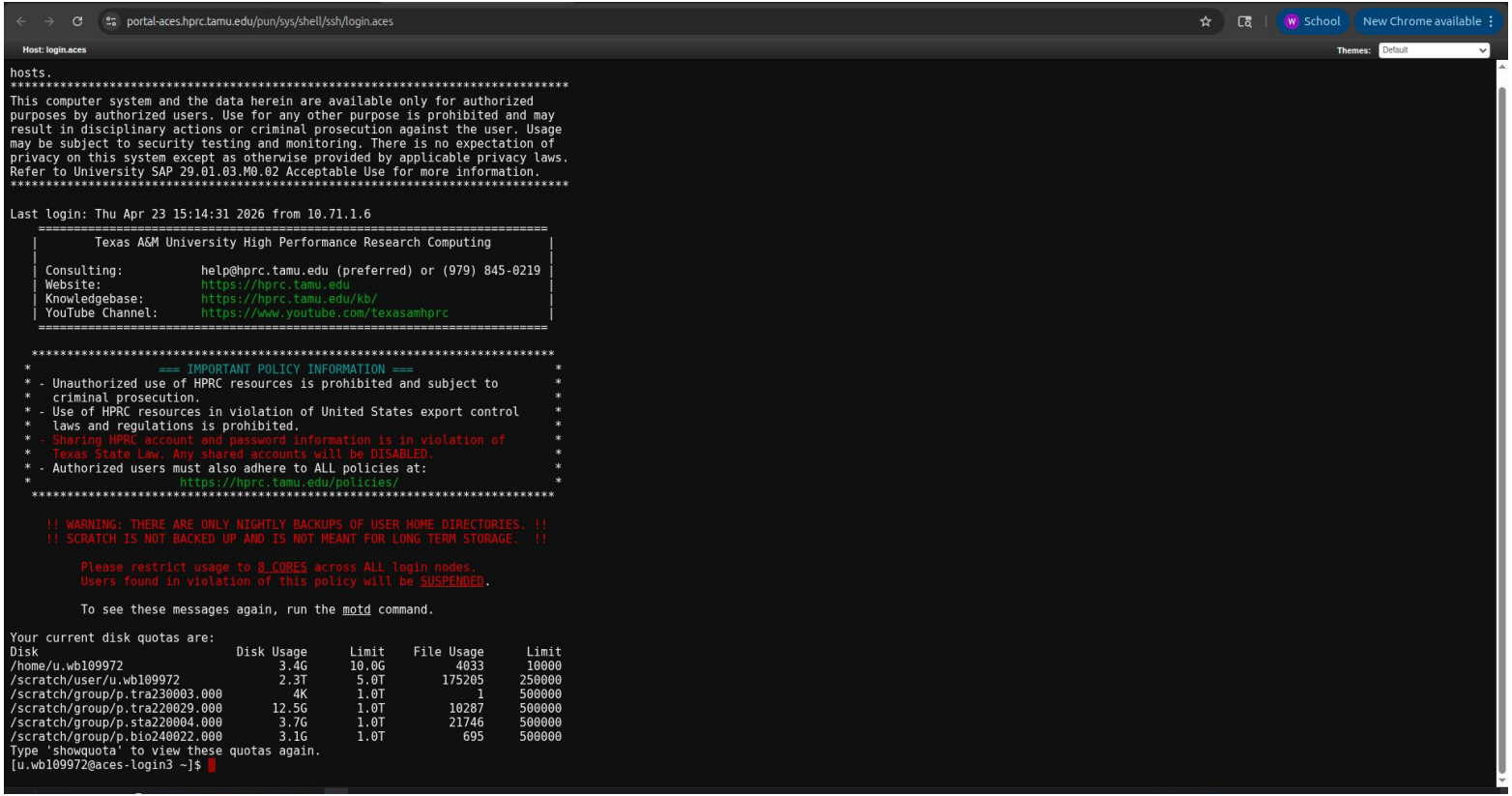
Accessing the HPRC ACES Portal



Accessing ACES Shell in OOD Portal



Accessing ACES Shell in OOD Portal



Get course data and set up the environment

- Run the following commands to change your working directory and copy the material for today's course:

```
cd $SCRATCH
```

```
cp -r /scratch/training/snakemake_course .
```

- Run the following command to load the required modules

```
module load GCC/13.2.0 OpenMPI/4.1.6 R_tamu/4.4.1 snakemake/8.28.0
```

Exercise 1 - Our first snakemake job

- Navigate to the ex01 directory:

```
cd snakemake_tutorial/ex01-03
```

- List the contents of the directory

```
ls -lth
```

Output:

```
total 28K
-rw-r--r-- 1 u.wb109972 u.wb109972 20K Apr 27 14:54 penguin.csv
-rw-r--r-- 1 u.wb109972 u.wb109972 781 Apr 27 14:54 plot_penguins.R
-rw-r--r-- 1 u.wb109972 u.wb109972 264 Apr 27 14:54 Snakefile
```

Exercise 1 - Our first snakemake job

- Look at the penguin.csv and plot_penguins.R files in the Launch file browser

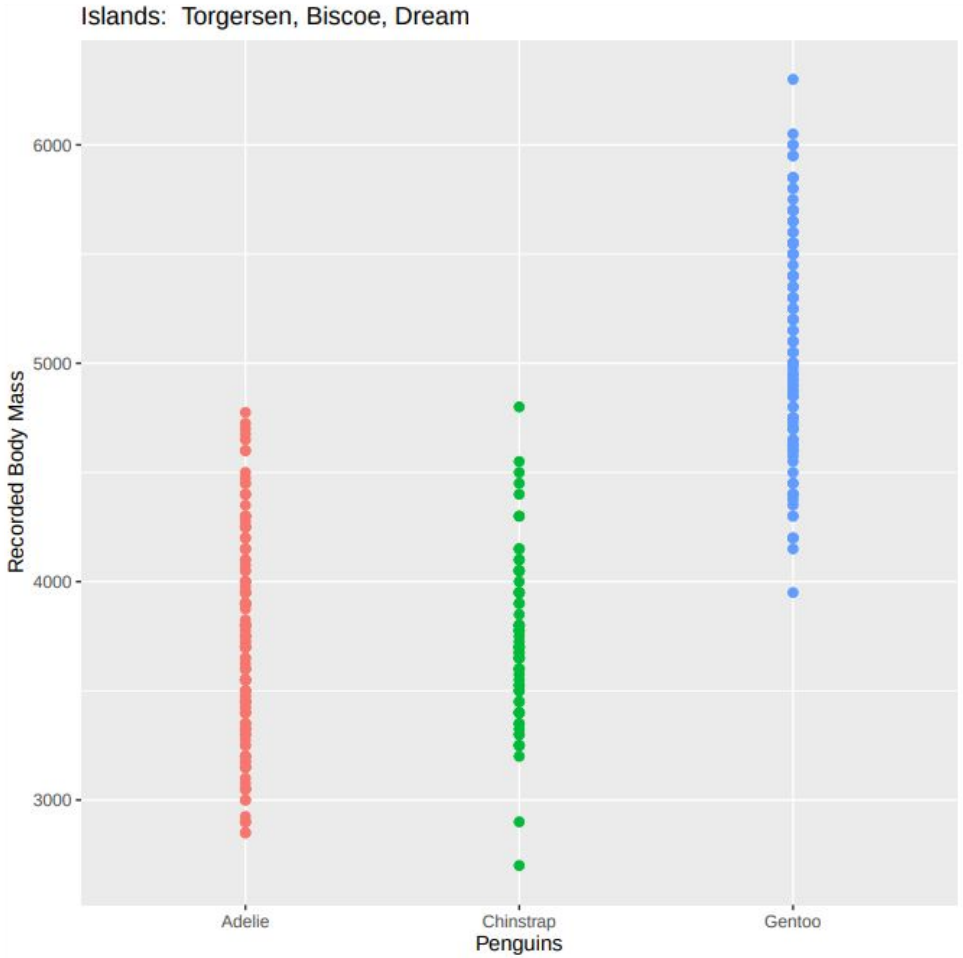
- Run the following command to execute Snakemake:

```
snakemake --cores 1 penguin_plot.pdf
```

Exercise 1

```
Assuming unrestricted shared filesystem usage.
host: alogin2.cluster
Building DAG of jobs...
Using shell: /usr/bin/bash
Provided cores: 1 (use --cores to define parallelism)
Rules claiming more threads will be scaled down.
Job stats:
job      count
-----  -------
plot         1
total        1

Select jobs to execute...
Execute 1 jobs...

[Mon Apr 27 16:35:58 2026]
localrule plot:
    input: penguin.csv
    output: penguin_plot.pdf
    jobid: 0
    reason: Missing output files: penguin_plot.pdf
    wildcards: data=penguin
    resources: tmpdir=/tmp

Saving 7 x 7 in image
[Mon Apr 27 16:36:03 2026]
Finished job 0.
1 of 1 steps (100%) done
Complete log: .snakemake/log/2026-04-27T163557.943573.snakemake.log
```

Exercise 1


Islands: Torgersen, Biscoe, Dream
Recorded Body Mass
6000
5000
4000
3000
Adelie
Chinstrap
Gentoo
Penguins

Exercise 1

Snakefile:

```
rule plot:
    output: "{data}_plot.pdf"
    input:  "{data}.csv"
    shell:  "Rscript plot_penguins.R -i {input} -o {output}"

rule biscoe_island:
    output: "{data}_biscoe.csv"
    input:  "{data}.csv"
    shell:  "grep -e species -e Biscoe {input} > {output}"
```

Command:

```
snakemake --cores 1 penguin_plot.pdf
```

Exercise 1

Snakefile:

Snakemake recognizes the {data} == penguin from our command

```
rule plot:
    output: "{data}_plot.pdf"  → penguin_plot.pdf
    input:  "{data}.csv"       → penguin.csv
    shell:  "Rscript plot_penguins.R -i {input} -o {output}"

rule biscoe_island:
    output: "{data}_biscoe.csv"
    input:  "{data}.csv"
    shell:  "grep -e species -e Biscoe {input} > {output}"
```

Command:

```
snakemake --cores 1 penguin_plot.pdf
```

Exercise 1 - Why isn't the second rule executed?

Snakefile:

```
rule plot:
    output: "{data}_plot.pdf"
    input:  "{data}.csv"
    shell:  "Rscript plot_penguins.R -i {input} -o {output}"

rule biscoe_island:
    output: "{data}_biscoe.csv"
    input:  "{data}.csv"
    shell:  "grep -e species -e Biscoe {input} > {output}"
```

Snakemake builds the simplest DAG (directed acyclic graph) to produce the results:

{data}.csv ⟹ Rscript ⟹ {data}_plot.pdf

Exercise 2 - Changing requested output to run both rules

- Run the following command to execute Snakemake:

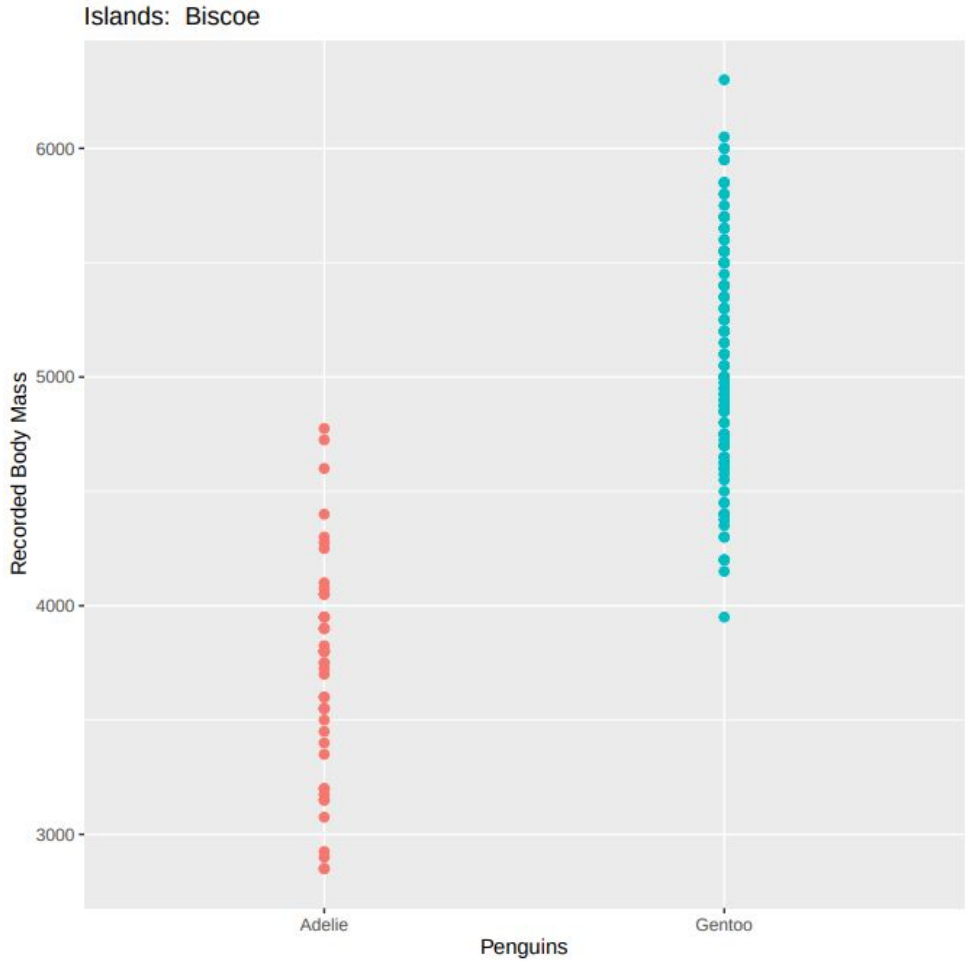
```
snakemake --cores 1 penguin_biscoe_plot.pdf
```

Exercise 2

```
Assuming unrestricted shared filesystem usage.
host: alogin1.cluster
Building DAG of jobs...
Using shell: /usr/bin/bash
Provided cores: 1 (use --cores to define parallelism)
Rules claiming more threads will be scaled down.
Job stats:
job              count
-------------  -------
biscoe_island        1
plot                 1
total                2

Select jobs to execute...
Execute 1 jobs...

[Mon Apr 27 16:56:54 2026]
localrule biscoe_island:
    input: penguin.csv
    output: penguin_biscoe.csv
    jobid: 1
    reason: Missing output files: penguin_biscoe.csv
    wildcards: data=penguin
    resources: tmpdir=/tmp
```

```
[Mon Apr 27 16:56:55 2026]
Finished job 1.
1 of 2 steps (50%) done
Select jobs to execute...
Execute 1 jobs...

[Mon Apr 27 16:56:55 2026]
localrule plot:
    input: penguin_biscoe.csv
    output: penguin_biscoe_plot.pdf
    jobid: 0
    reason: Missing output files: penguin_biscoe_plot.pdf;
Input files updated by another job: penguin_biscoe.csv
    wildcards: data=penguin_biscoe
    resources: tmpdir=/tmp

Saving 7 x 7 in image
[Mon Apr 27 16:56:58 2026]
Finished job 0.
2 of 2 steps (100%) done
Complete log:
.snakemake/log/2026-04-27T165654.923742.snakemake.log
```

Exercise 2


Islands: Biscoe
Recorded Body Mass
6000
5000
4000
3000
Adelie
Gentoo
Penguins

Exercise 2 - Why were both rules executed?

Snakefile:

```
rule plot:
    output: "{data}_plot.pdf"
    input:  "{data}.csv"
    shell:  "Rscript plot_penguins.R -i {input} -o {output}"

rule biscoe_island:
    output: "{data}_biscoe.csv"
    input:  "{data}.csv"
    shell:  "grep -e species -e Biscoe {input} > {output}"
```

DAG:

{data}.csv ⇨ rule biscoe_island ⇨ {data}_biscoe.csv ⇨ rule plot ⇨ {data}_plot.pdf

Exercise 3

- Run the following command again:

```
snakemake --cores 1 penguin_plot.pdf
```

Exercise 3

- Run the following command again:

```
snakemake --cores 1 penguin_plot.pdf
```

- Output:

```
Building DAG of jobs...
Nothing to be done (all requested files are present and up to date).
```

Exercise 3 (continued)

- Delete the last line of penguin.csv and run the following command again:

```
snakemake --cores 1 penguin_plot.pdf
```

- Output:

```
host: alogin1.cluster
Building DAG of jobs...
Using shell: /usr/bin/bash
Provided cores: 1 (use --cores to define
parallelism)
Rules claiming more threads will be scaled
down.
Job stats:
job      count
-----  -------
plot         1
total        1

Select jobs to execute...
Execute 1 jobs...
```

```
[Mon Apr 27 17:14:02 2026]
localrule plot:
    input: penguin.csv
    output: penguin_plot.pdf
    jobid: 0
    reason: Updated input files: penguin.csv
    wildcards: data=penguin
    resources: tmpdir=/tmp

Saving 7 x 7 in image
[Mon Apr 27 17:14:06 2026]
Finished job 0.
1 of 1 steps (100%) done
Complete log: .snakemake/log/2026-04-27T171402.906747.snakemake.log
```

Exercise 4 - A Complete Bioinformatics Workflow

1. Change directories to ../ex04

```
cd ../ex04
```

2. Purge the modules that are currently loaded and load the modules necessary to run Snakemake, FastQC, Trim_Galore!, HISAT2, and SAMtools (Hint: you will need to ensure that all of the loaded modules have compatible dependencies).

```
# Use module avail <pkg> to find compatible versions
module load GCC/13.3.0 OpenMPI/5.0.3 snakemake/8.27.0 \
    FastQC/0.11.9-Java-11 Trim_Galore/0.6.11 \
    HISAT2/2.2.1 SAMtools/1.22
```

Exercise 4 - Snakefile

```
# Threads per step
FASTQC_THREADS = 2
TRIM_THREADS   = 8
ALIGN_THREADS  = 8
SORT_THREADS   = 8

# Samples
SAMPLES = ["Control1"]

# Ensure full pipeline is run
rule all:
    input:
        expand("FastQC/{sample}_R1_fastqc.html", sample = SAMPLES),
        expand("trimming/{sample}_R1_val_1.fq.gz", sample = SAMPLES),
        expand("align/{sample}.sam", sample = SAMPLES)
```

Exercise 4 - Snakefile (continued)

```
# Run FastQC on raw illumina data
rule fastqc:
    input:
        r1 = "raw_fastqs/{sample}_R1.fastq.gz",
        r2 = "raw_fastqs/{sample}_R2.fastq.gz"
    output:
        "FastQC/{sample}_R1_fastqc.html",
        "FastQC/{sample}_R1_fastqc.zip",
        "FastQC/{sample}_R2_fastqc.html",
        "FastQC/{sample}_R2_fastqc.zip"
    threads: FASTQC_THREADS
    shell:
        """
        mkdir -p FastQC
        fastqc -t {threads} -o FastQC {input.r1} {input.r2}
        """
```

Exercise 4 - Snakefile (continued)

```
# Trim libraries with Trim_Galore!
rule trim_galore:
    input:
        r1 = "raw_fastqs/{sample}_R1.fastq.gz",
        r2 = "raw_fastqs/{sample}_R2.fastq.gz"
    output:
        val1 = "trimming/{sample}_R1_val_1.fq.gz",
        val2 = "trimming/{sample}_R2_val_2.fq.gz",
        rep1 = "trimming/{sample}_R1.fastq.gz_trimming_report.txt",
        rep2 = "trimming/{sample}_R2.fastq.gz_trimming_report.txt"
    threads: TRIM_THREADS
    shell:
        """
        mkdir -p trimming
        trim_galore --paired --fastqc --output_dir trimming --cores {threads} {input.r1} {input.r2}
        """
```

Exercise 4 - Snakefile (continued)

```
# Map trimmed reads to genome with HISAT2
rule hisat:
    input:
        r1 = "trimming/{sample}_R1_val_1.fq.gz",
        r2 = "trimming/{sample}_R2_val_2.fq.gz"
    output: "align/{sample}.sam"
    threads: ALIGN_THREADS
    params:
        index = "../mm39/GCF_000001635.27_GRCm39_genomic"
    shell:
        """
        mkdir -p align
        hisat2 -x {params.index} -1 {input.r1} -2 {input.r2} -p {threads} \
           --no-mixed --no-discordant -S {output}
        """

# Sort sam file
rule samtools:
    input: "align/{sample}.sam"
    output: "align/{sample}_sorted.bam"
    threads: SORT_THREADS
    shell: "samtools sort -@ {threads} -O bam -o {output} {input}"
```

Exercise 4

Run snakemake:

```
snakemake --cores 8
```

Exercise 5 - Submitting snakemake jobs to Slurm

Navigate to the ex05 directory. Create a batch script to run the snakemake job from exercise 2. Submit the job to Slurm.

Exercise 5 - Submitting snakemake jobs to Slurm

Solution:

```
#!/bin/bash
#SBATCH --job-name=plotting_penguins
#SBATCH --time=00:02:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=1G
#SBATCH --output=%x.%j.stdout
#SBATCH --error=%x.%j.stderr

# Set up the environment
module purge
module load GCC/13.2.0 OpenMPI/4.1.6 R_tamu/4.4.1 snakemake/8.28.0


snakemake --cores 1 penguin_biscoe_plot.pdf
```

Exercise 6 - Submitting snakemake jobs to Slurm (multiple cores)

Create a batch script to run our RNA workflow snakemake job. Submit the job to Slurm.

Exercise 6 - Submitting snakemake jobs to Slurm (multiple cores)

Solution:

```
#!/bin/bash
#SBATCH --job-name=rna_workflow
#SBATCH --time=00:05:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=10G
#SBATCH --output=%x.%j.stdout
#SBATCH --error=%x.%j.stderr

# Set up the environment
module purge
module load GCC/12.3.0 OpenMPI/4.1.5 snakemake/8.4.2 FastQC/0.12.1-Java-11
module load Trim_Galore/0.6.10 HISAT2/2.2.1 SAMtools/1.18

snakemake --cores 8
```

Exercise 7 - Generating a simple report

1. Change your working directory to ex07
2. Run snakemake:

```
snakemake --cores 8
```

3. Generate the report:

```
snakemake --report pipeline_report.html
```

Exercise 7 - Generating a simple report

Snakefile:

```
# Threads per step
FASTQC_THREADS = 2
TRIM_THREADS   = 8
ALIGN_THREADS  = 8
SORT_THREADS   = 8

# Samples
SAMPLES = ["Control1", "Control2"]

# Ensure full pipeline is run
rule all:
      input:
            expand("FastQC/{sample}_R1_fastqc.html", sample=SAMPLES),
            expand("FastQC/{sample}_R2_fastqc.html", sample=SAMPLES),
            expand("trimming/{sample}_R1.fastq.gz_trimming_report.txt", sample=SAMPLES),
            expand("trimming/{sample}_R2.fastq.gz_trimming_report.txt", sample=SAMPLES),
```

```
# Run FastQC on raw illumina data
rule fastqc:
	input:
		r1 = "../fastqs/{sample}_R1.fastq.gz",
		r2 = "../fastqs/{sample}_R2.fastq.gz"
	output:
		report("FastQC/{sample}_R1_fastqc.html"),
		"FastQC/{sample}_R1_fastqc.zip",
		report("FastQC/{sample}_R2_fastqc.html"),
		"FastQC/{sample}_R2_fastqc.zip"
	threads: FASTQC_THREADS
	shell:
	"""
	mkdir -p FastQC
	fastqc -t {threads} -o FastQC {input.r1} {input.r2}
	"""

# Trim libraries with Trim_Galore!
rule trim_galore:
	input:
		r1 = "../fastqs/{sample}_R1.fastq.gz",
		r2 = "../fastqs/{sample}_R2.fastq.gz"
	output:
		"trimming/{sample}_R1_val_1.fq.gz",
		"trimming/{sample}_R2_val_2.fq.gz",
		report("trimming/{sample}_R1.fastq.gz_trimming_report.txt"),
		report("trimming/{sample}_R2.fastq.gz_trimming_report.txt")
	threads: TRIM_THREADS
	shell:
	"""
	mkdir -p trimming
	trim_galore --paired --fastqc --output_dir trimming --cores {threads} {input.r1} {input.r2}
	"""
```

Exercise 8 - Examining Runtimes

1. Copy the Snakefile from Exercise 7
2. Modify the input to include the following samples from the “fastqs” folder in the parent directory:
   - Control 1
   - Control 2
   - Control 3
3. Add mapping with HISAT2 as an additional rule (make sure to include the results in the report)
4. Create a batch script to submit the job to slurm using 48 cores and 120Gb of memory
5. You will need to load the module WebProxy for `snakemake --report` to work

Exercise 8 - Snakefile modification

```
# Map trimmed reads to genome with HISAT2
rule hisat:
    input:
        r1 = "trimming/{sample}_R1_val_1.fq.gz",
        r2 = "trimming/{sample}_R2_val_2.fq.gz"
    output:
        sam = "align/{sample}.sam",
        summary = report("align/{sample}_summary.txt")
    threads: ALIGN_THREADS
    params:
        index = "../mm39/GCF_000001635.27_GRCm39_genomic"
    shell:
        """
        mkdir -p align
        hisat2 -x {params.index} -1 {input.r1} -2 {input.r2} -p {threads} --no-mixed \
           --no-discordant -S {output.sam} --summary-file {output.summary}
        """
```

Exercise 8 - Slurm script

```
#!/bin/bash
#SBATCH --job-name=rna_workflow
#SBATCH --time=00:30:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=48
#SBATCH --mem=120G
#SBATCH --output=%x.%j.stdout
#SBATCH --error=%x.%j.stderr

# Set up the environment
module purge
module load GCC/13.3.0 OpenMPI/5.0.3 snakemake/8.27.0 FastQC/0.11.9-Java-11
module load Trim_Galore/0.6.11 HISAT2/2.2.1 SAMtools/1.22

snakemake --cores 48
snakemake --report
```

Exercise 9 - Examining Runtimes

1. Copy the Snakefile and batch script from Exercise 8
2. Add the following samples to the Snakefile:
    - Treatment 1
    - Treatment 2
    - Treatment 3
3. Compare runtimes and resource utilization between the runs for Exercise 8 and 9

Exercise 9 - Examining Runtimes

Three Samples:

```
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 48
CPU Utilized: 05:30:27
CPU Efficiency: 39.12% of 14:04:48
core-walltime
Job Wall-clock time: 00:17:36
Memory Utilized: 12.01 GB
Memory Efficiency: 3.24% of 371.00 GB
```

Six Samples:

```
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 48
CPU Utilized: 12:06:51
CPU Efficiency: 60.49% of 20:01:36
core-walltime
Job Wall-clock time: 00:25:02
Memory Utilized: 20.83 GB
Memory Efficiency: 5.61% of 371.00 GB
```

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Take our short course survey!

https://u.tamu.edu/hprc_shortcourse_survey

Help us help you. Please include details in your request for support, such as, Cluster (FASTER, Grace, ACES, Launch), NetID (UserID), Job information (Job id(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.