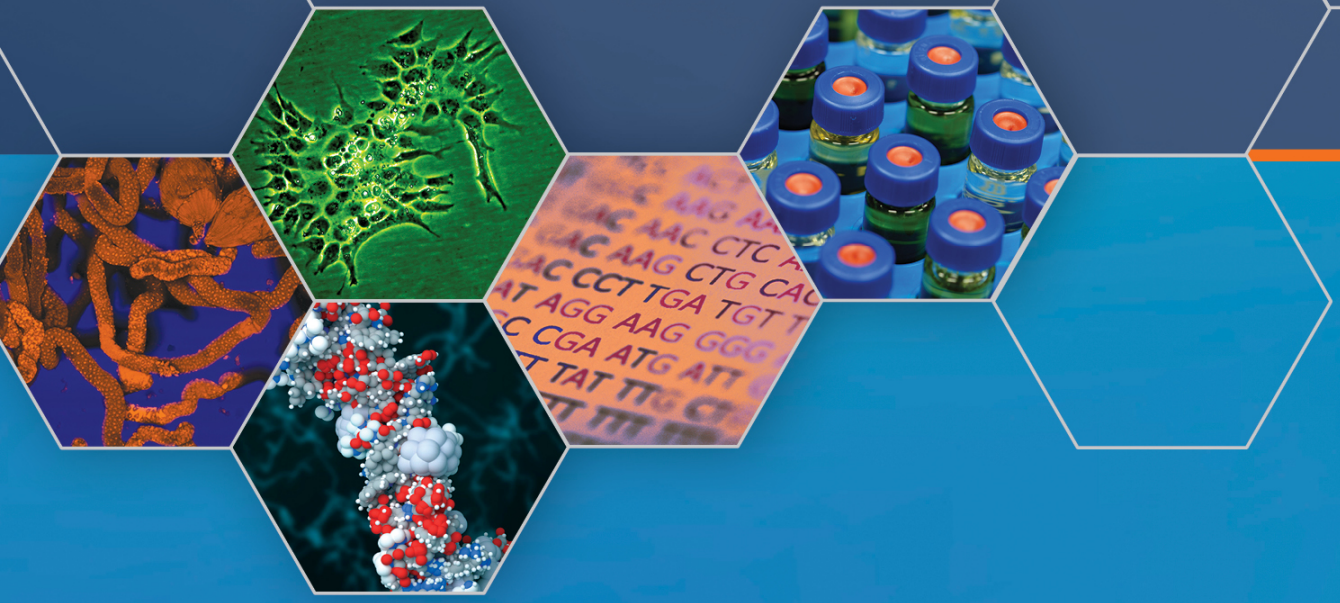
ICBR

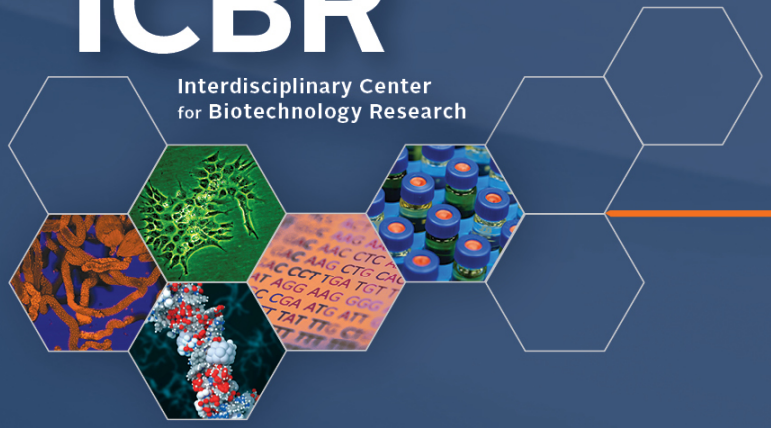
Interdisciplinary Center
for Biotechnology Research



UF | UNIVERSITY of FLORIDA

ICBR

Interdisciplinary Center for Biotechnology Research



Bioinformatics 101 – Lecture 3

Using HiPerGator

Alberto Riva (ariva@ufl.edu), J. Lucas Boatwright (jlboat@ufl.edu)

ICBR Bioinformatics Core

UF | UNIVERSITY of FLORIDA

Working on HiPerGator

- Accessible only through **ssh**. Only the four head nodes (login1 to login4) can receive ssh connections.
- Head nodes should **not** be used for intensive computations.
- Programs to run are submitted as *jobs* to an execution queue. The *scheduler* assigns jobs to available nodes according to multiple criteria: resources required, priority, etc.

Modules

- Software on HiPerGator is organized into *modules*. Module commands:

`module load <name>`	Load named module
`module spider <patt>`	Show matching module(s)
`module unload <name>`	Unload named module
`module purge`	Unload all modules

- Hundreds of modules are available. Modules may automatically load other modules they depend on.

Script submission

- A submission script is a regular shell script that includes *directives* for the scheduler.
- Submission script cannot get input from the user. They should read inputs from files and write output to files.
- Scripts are submitted using `sbatch`.
- Check execution of your script using `squeue`. Cancel a running job using `scancel`.

UF | ICBR
BIOINFORMATICS

UFHealth
CANCER CENTER

Turning a script into a job

- A job submission script is a regular shell script that contains directives for the scheduler. Example:

```
#!/bin/bash

#SBATCH --time=1:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=3
#SBATCH --mem=2G

FILE=$1
OUT=$2
cut -f 1 $FILE | sort | uniq -c > $OUT
```

Set maximum run time (default is 10 minutes!)

Number of parallel processes

Amount of memory needed

Input and output files

Turning a script into a job

Other important directives:

`--account=<acct>`	if you belong to multiple groups
`--qos=<qos>`	choose default or “burst” (-b) mode
`--mail-type=<events>`	get email when specified events occur (e.g., FAIL,END)
`--mail-user=<addr>`	where to send mail

More information can be found on the UF Research Computing site:

https://help.rc.ufl.edu/

The dibig_tools module

- The dibig_tools module provides access to a large number of tools, scripts, and pipelines developed by the Bioinformatics Core. For example:

- user-friendly replacements for sbatch (`submit`) and squeue (`qmine`).
- `tcalc.py`, a delimited file processor.
- `kut`, a more powerful version of `cut`.
- `csvtoxls.py`, to convert delimited files to Excel.

The dibig_tools module

Features of the `submit` command:

- Automatically adds start/end time to output;
- Can write file to signal job is done;
- Allows concatenating jobs:

```
submit -after <jobid1> job2.qsub
```

- Often-used options (e.g. account, email) can be saved to a configuration file;
- Comes with extensive library of ready-to-use scripts.

Pipelines

- A *pipeline* is a tool to perform a complete analysis (end-to-end).
- It normally consists of multiple steps to be performed in sequence. Each step may be local or (more often) require submitting parallel jobs.
- A pipeline manager should handle job submission, processing of the results, generation of final report.

Simple pipelines

- Simple pipelines can be created by concatenating jobs to perform consecutive processing steps.

- For example: after previous script (script1.qsub), add step to count number of lines in output (script2.qsub).

```
#!/bin/bash

#SBATCH --time=10:00
#SBATCH --ntasks=3
#SBATCH --mem=1G

FILE=$1
OUT=$2
wc -l $FILE > $OUT
```

Simple pipelines

- We can now submit these two scripts as two separate jobs, scheduling one after the other.

```
$ submit script1.qsub genes.csv chroms.txt
1234567

$ submit -after 1234567 script2.qsub chroms.txt
```

- The second job will be "held" until the first one has terminated successfully, then it will be scheduled for running.

Simple pipelines

This approach only works in very simple cases.
Limitations:

- We had to come up with a name for the intermediate file – the pipeline manager should do that.
- What if the second step needs input from more than one previous step?
- If the first step fails to produce its output, the pipeline should stop.

Pipeline managers

Modern pipeline frameworks provide all the features mentioned above, and more. Example: **nextflow**. Features:

- Reproducible – workflows can run Docker/Singularity containers, manage software version and reproduce results;
- Portable – designed to interface with job schedulers or run locally;
- Parallelization – inherently parallel with input and output defining serial or parallel runs;
- Continuous checkpoints – can resume execution at last step after stopping.

```
process Fastqc {
    publishDir "results"

    input:
    set dataset_id, file(reads) from fastqc_paired_fastq

    output:
    file "*_fastqc.{zip,html}" into fastqc_results

    clusterOptions = { "--account=bioinf_workshop
                        --qos=bioinf_workshop --time=4:00:00
                        --mem-per-cpu=2gb --cpus-per-task=1" }
    module 'fastqc'

    script:
    """
    fastqc -q $reads
    """
}
```