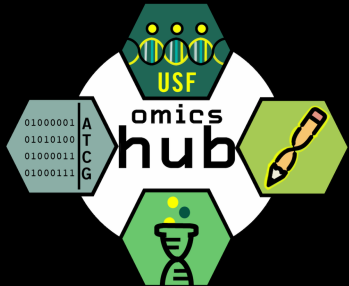


Linux for Biologists II

Jenna Oberstaller, PhD



| | | | | |
|---|--|---------------------------|--------------------|---|
| Introduction to Microbiomes and Metagenomic Data Analysis | Linux for Biologists II: <ul style="list-style-type: none"> • Shell scripting • Cluster computing at USF • Intro to Conda/Containers | Metagenomics WGS Assembly | Intro to R | Metagenomics Functional Analysis and Data Viz |
| Introduction to NGS/Illumina Sequencing | Sequence Data QC | Metagenomics WGS Binning | R for Metagenomics | Data Viz II |
| Linux for Biologists | Practice | Practice | Practice | Machine Learning for Metagenomics |
| Thursday | Friday | Monday | Tuesday | Wednesday |

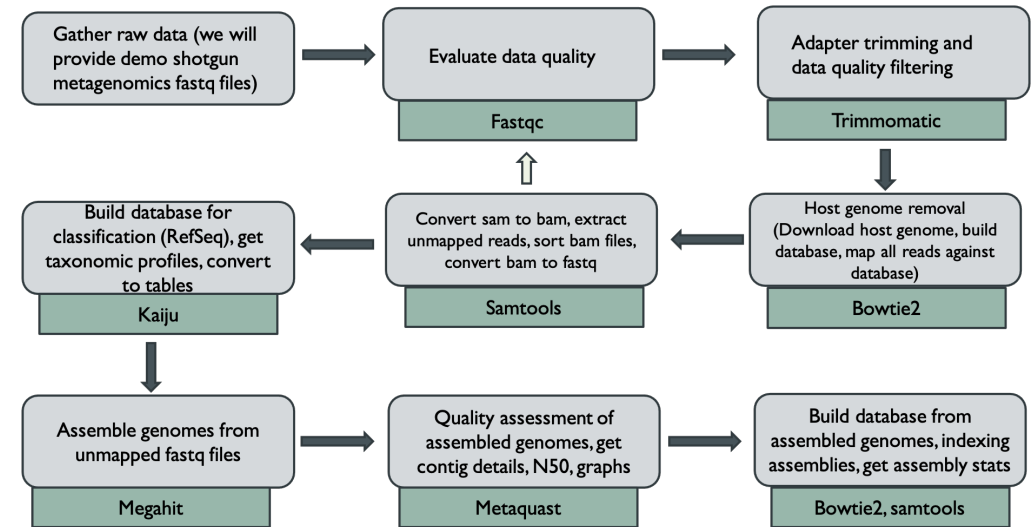
Workshop schedule

STEP I: Basic UNIX commands

??



THE PIPELINE



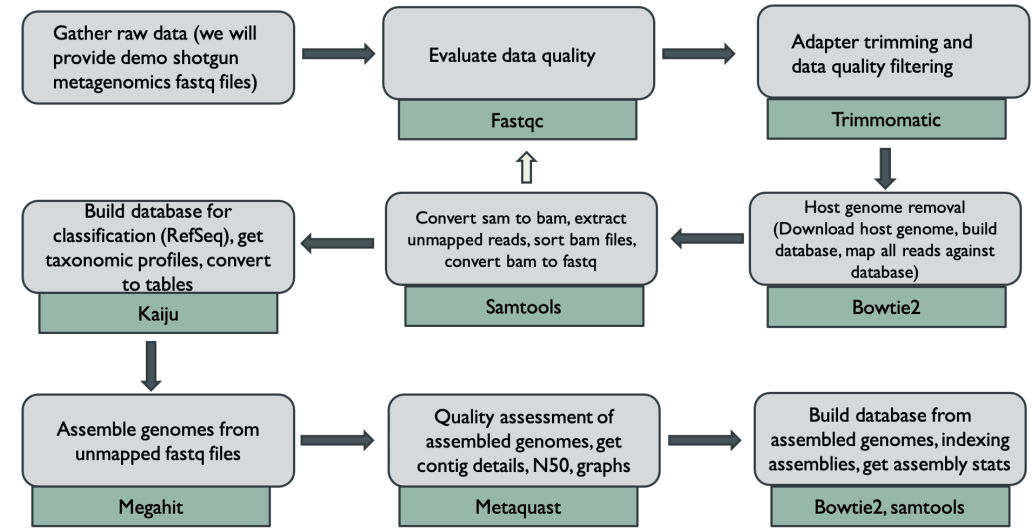
STEP I: Basic UNIX commands

??



- genomic software
- chaining them together
- specifying computational resources

THE PIPELINE



Additional downloads (for later)

- Filezilla: for easy file-downloads and uploads

<https://filezilla-project.org/>

- VSCode: plain-text editor and more

<https://code.visualstudio.com/download>

What is High-Performance Computing?

High-Performance Computing (HPC)

- Generally “large” computational tasks
- Reduce run time for single job from 12 months to 1 week
 - Example: Airplane aerodynamic simulation including fluid flow and structural mechanics within *a single analysis*



When to use High-Performance Computing

- **When your desktop is just not enough to solve part or all of your problem.**
 - Takes too long = Problem/step has too many operations
 - Break up problem into many smaller steps to run in parallel
 - Run out of memory = Problem/step too big
 - Can problem be run in distributed parallel?
 - spreads out memory load across multiple machines
 - Run on "large memory" hardware (we have several 20 core, 512GB RAM machines)



Cluster computing at USF

Navigating RRA



Resources at USF

- RRA (Research-Restricted Access)
- CIRCE (general, campus-wide)
- Research Computing Staff
 - Computational expertise – optimizing resources
 - Software installation (modules) and support
- The Hub
 - omics software/environments/pipelines
 - frequently-used omics databases, reference genomes
 - omics, “make-it-work” expertise

File management on RRA

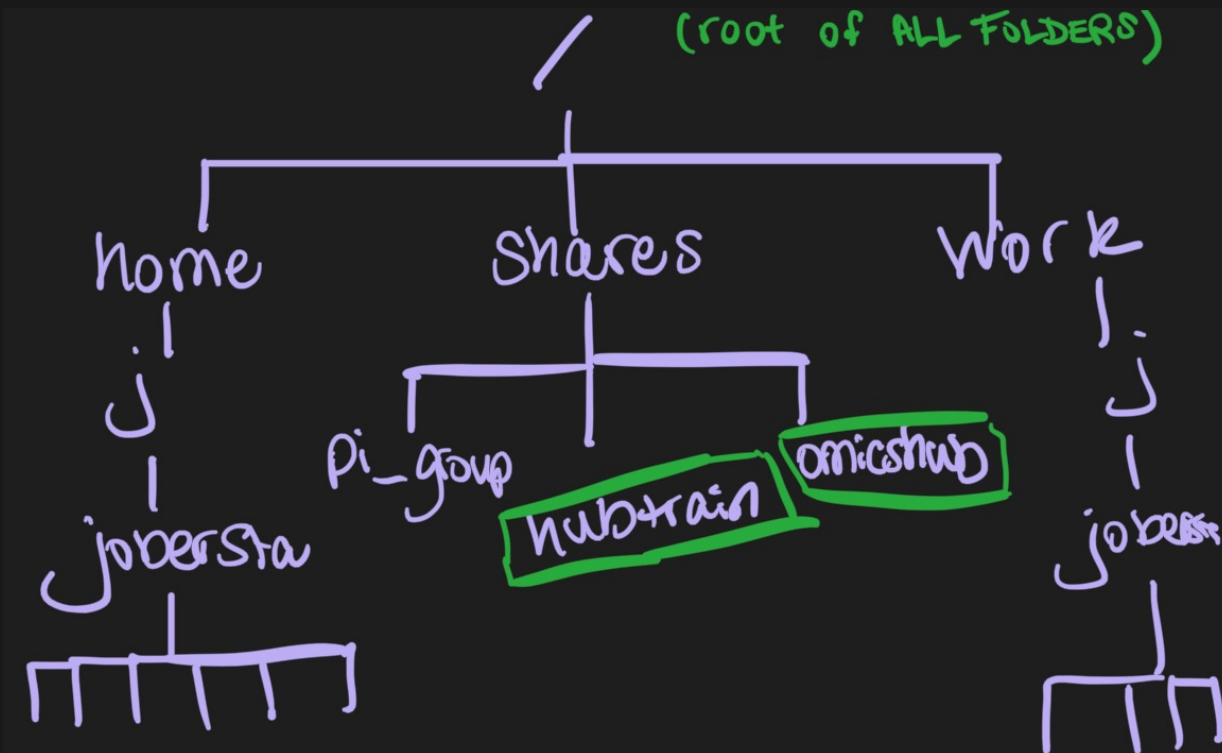
- /home vs /work
- /shares
- Storage space: du and space limits
 - rra-myquota
- permissions

File management on RRA

| | /home | /work | /shares |
|----------------|---------------------------------|--|---------------------------------------|
| purpose | Precious files you need to keep | Files you're actively analyzing/generating | Shared files (eg between lab members) |
| backed up? | Yes | No | yes |
| storage limits | 200GB | 2TB | 4TB |
| permissions | Only you | Only you | Everyone in a group |

Hardest concepts for beginners

- UNDERSTANDING FILE PATHS AND NAVIGATION

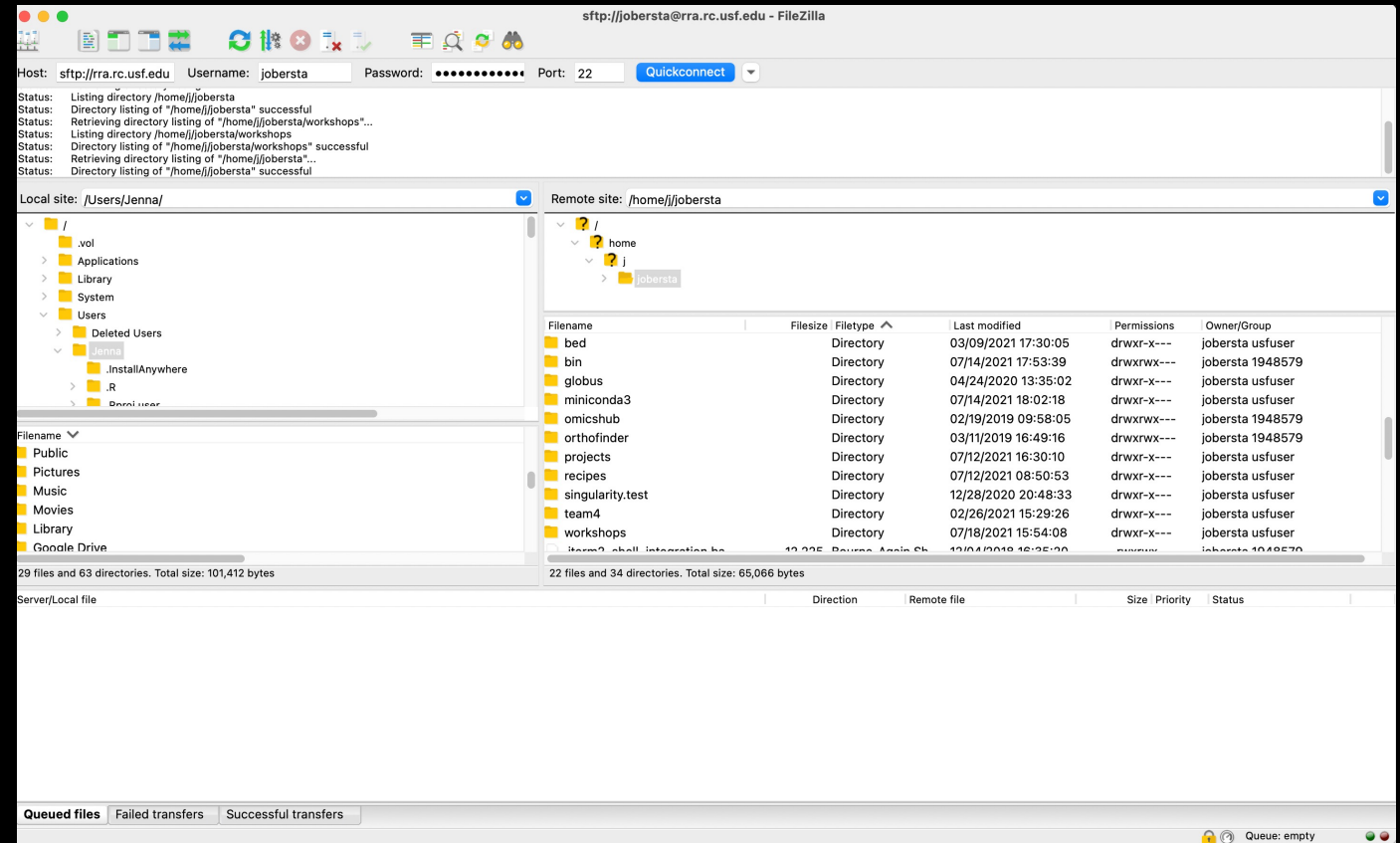


Basic Linux – How to connect

- ssh client
 - Mac – terminal, iTerm2
 - Windows – Putty*, linux subsystem
 - Linux – ssh already built in

How to manage files (a GUI way)

- Transfer files to rra.rc.usf.edu via SFTP client
 - FileZilla on Windows, Mac, or Linux
- Configuration details:
 - Host Name – rra.rc.usf.edu
 - Port 22 (default SSH port)
 - User Name – your NetID
 - Password – your NetID password
- Local Computer on Left, Remote Computer on Right



<https://filezilla-project.org/>

Exercises – Managing Files

1. Drag and drop file from RRA to local desktop
2. Edit the file using any method
3. Move file back to RRA
4. ls, cat, less, etc. the file (hint: from RRA)
5. Delete a file on RRA
6. Create a directory in your home dir

Up Next:

SLURM

What is SLURM?

Simple

Linux

Utility for

Resource

Management

- A software suite that facilitates management of computational resources, i.e. an HPC cluster.

What does SLURM do?

- Using a restaurant hostess' job as an analogy, SLURM will direct consumers to available tables (HPC resources) based upon some "key" items:
 - Availability of seating;
 - Reservations;
 - "VIP" status;
 - Wait times for seating.
-
- SLURM must also perform the role of a restaurant manager; scheduling of personnel (sudden sick calls or other crises), operational hours, special events and promotions, etc.

How can I benefit from using SLURM?

Primary benefits

- Resource "hungry" applications no longer need to be run on your laptop and/or workstation.
- Depending on the application, more CPU's and memory can be supplied, resulting in faster run times and delivery of results.
- Jobs can be submitted and forgotten to allow you to return to daily activities.

Other benefits

- Jobs can take advantage of scripting to allow for complicated workflows.
- Dependencies can be configured so that jobs will only run in a specified order.

Basic SLURM script

- A very basic SLURM submission script will look similar to the script below:

```
#!/bin/bash
#SBATCH --job-name="test job"

./myapp
```

- The lines explained
 - `#!/bin/bash` - The interpreter that SLURM will utilize
 - `#SBATCH --job-name="test job"` - The name of the job
 - `./myapp` - A fictional application that will be run by SLURM
- SLURM submission scripts can use any interpreted language (ex: sh, bash, csh, tcsh, ruby, etc.).
- NOTE: All lines containing SLURM-specific commands must start with `#SBATCH`

SLURM submission

- sbatch: submit batch jobs to SLURM

```
[johndoe@login0 ~]$ sbatch ./test-sbatch.sh  
Submitted batch job 1234567
```

- After the job has been accepted, a job ID will be presented. If there are any errors within the script syntax, an error will be displayed and the user will be notified that the job has not been accepted by the system.

- scancel: cancel existing pending/running SLURM jobs that you own

```
[johndoe@login0 ~]$ scancel 1234567
```

loading programs

aka : intro to environments

Programs we use are available via

- modules (aka “environmental modules”)
- conda environments

Programs can be complicated and depend on other programs, settings, etc. to run
Environments load all these dependencies together

Modules

What are modules?

- Modules are scripts that dynamically adjust a user's environment to run and build software

Why use them?

- Avoids issues created by hard-coding environment variables in ~/.bashrc, ~/.bash_profile, etc., scripts
 - *More often than not, users "forget" about these manual edits leading to problems running and building software*
- No need to use "export" and "unset" commands on the console.
- No need to remember explicit paths to installed software and/or specific versions of the same software, e.g. (/apps/intel/2013_sp1 vs. /apps/intel/2015/)
- Ease of use: module load apps/somesoftware/version

Modules continued

"Proper" use of modules

- Always start with a clean environment by "purging" it first: ``module purge``
- Only load the modules that are required for the task at hand

Best practices in module use

- Never bulk load modules via login scripts
 - Module load apps/matlab/r2015 apps/synopsys/hspice/C-2009.09 apps/synopsys/icc/F-2011.09-SP4
 - Bulk loading will cause issues on the Operating System level
 - Research Computing will not provide support on issues with software failures due to bulk module loading.

Module Commands

module list: Show all currently-loaded modules

```
[johndoe@login0 ~]$ module list  
No Modulefiles Currently Loaded.
```

Module Commands

module avail: Show all available modules

```
[johndoe@login0 ~]$ module avail
```

```
----- /etc/modulefiles -----  
admin/genders  
admin/module-cvs  
admin/module-info  
admin/pdsh  
admin/rsge  
admin/stress-ng  
admin/tools  
apps/R/2.11.1  
apps/R/2.15.3  
apps/R/3.0.3-pbdR  
apps/R/3.1.2  
apps/R/3.2.3  
apps/abinit/6.12.3  
apps/abinit/7.10.2  
apps/abinit/7.10.5  
apps/abinit/7.4.2  
apps/abyss/1.2.7  
apps/accelrys/MaterialsStudio-4.4  
apps/accelrys/MaterialsStudio-5.0  
apps/ads/2006u2  
apps/ads/2009  
apps/ads/2009u1  
apps/aermod/15181  
apps/aldec/2013  
apps/altera/quartus-full/13.1  
apps/altera/quartus-web/13.1  
apps/amber/12  
apps/amdapp/2.7  
apps/ansoft/designer-6.0  
apps/matlab/r2014a  
apps/matlab/r2014b  
apps/matlab/r2015a  
apps/matlab/r2015b  
apps/maya/2013  
apps/meep/1.2.1  
apps/mega/4.02  
apps/mega/5.1  
apps/meme/3.5.7  
apps/mentor/ams/14.1  
apps/mentor/calibre/2014.4_18.13  
apps/modelnet/0.99  
apps/molden/4.9  
apps/mosek/7.1.15  
apps/mosis/TDP_2011q2v2  
apps/mpb/1.4.2  
apps/mpi4py/1.2.2  
apps/mpi4py/1.3.5  
apps/mpiblast/1.5.0  
apps/mpiblast/1.6.0  
apps/mrbayes/3.1.2  
apps/mrjob/0.2.4  
apps/multidistribute/unknown  
apps/namd/2.7  
apps/ncarg/4.4.1  
apps/ncarg/6.0.0  
apps/ncarg/6.0.0a  
apps/ncbi/6.1  
apps/netbeans/6.5ss
```

Module Commands

module load: Load a module into your current environment

```
[johndoe@login0 ~]$ module list
No Modulefiles Currently Loaded.
[johndoe@login0 ~]$ module load compilers/intel/2015_cluster_xe
[johndoe@login0 ~]$ module list
Currently Loaded Modulefiles:
  1) compilers/intel/2015_cluster_xe
[johndoe@login0 ~]$
```

Module Commands

module rm: Remove a module into your current environment

```
[johndoe@login0 ~]$ module list
Currently Loaded Modulefiles:
  1) compilers/intel/2015_cluster_xe
[johndoe@login0 ~]$ module rm compilers/intel/2015_cluster_xe
[johndoe@login0 ~]$ module list
No Modulefiles Currently Loaded.
[johndoe@login0 ~]$
```

Module Commands

Other useful commands:

- module purge: Remove all modules from your current environment

Exercises - Modules

- Order of modules loaded matters!
 - module load apps/matlab/r2013a
 - module load apps/matlab/r2016a
 - which matlab
- Some modules pre-pend system commands
 - which python; python --version
 - module load apps/python/3.5.1
 - which python; python --version

Modules continued

Modules are lightweight, easy to use, and can be mixed and matched

- BUT can interact in unexpected ways
 - so better not to mix-and-match unless you've tested/know what you're doing
- If you need to load several programs at once—e.g. when testing a pipeline—better to use a conda environment

Hub modules and conda environments

- We primarily make our workflows available in conda environments
 - Anaconda is an environment manager and package manager in one.
 - conda environments are more isolated than modules
 - no mixing-and-matching—the environment is self-contained
 - bulkier than modules and a couple more steps are required to load them
 - more reproducible
 - incompatibilities not as big of a concern—conda resolves them
- Load the Hub's anaconda module to access the Hub's conda environments (next)

add Hub modules to your search path

```
[jobesta@rra-login1 ~]$ export  
MODULEPATH=$MODULEPATH:/shares/omicshub/modulefiles >> ~/.bash_profile  
[jobesta@rra-login1 ~]$ source ~/.bash_profile  
[jobesta@rra-login1 ~]$ module avail
```

```
----- /shares/omicshub/modulefiles -----  
hub.apps/anaconda3/2020.11          hub.apps/metabat/2.0  
hub.apps/rnaseqpipeline/april.2021  
hub.apps/basespace/april.2021      hub.apps/quast/5.0.2  
hub.apps/seekdeep/2.6.0
```

now hub modulefiles will appear at the end of the list of system modulefiles when you type `module avail`

Loading Hub conda environments

```
[jobesta@rra-login1 ~]$ module avail
```

```
----- /shares/omicshub/modulefiles -----  
hub.apps/anaconda3/2020.11          hub.apps/metabat/2.0  
hub.apps/rnaseqpipeline/april.2021  
hub.apps/basespace/april.2021      hub.apps/quast/5.0.2  
hub.apps/seekdeep/2.6.0
```


```
[jobesta@rra-login1 ~]$ module load hub.apps/anaconda3/2020.11
```

```
[jobesta@rra-login1 ~]$ conda env list
```


to show all available environments

Loading Hub conda environments

```
[jobesta@rra-login1 ~]$ conda activate multiqc  
(multiqc) [jobesta@rra-login1 ~]$ multiqc --help
```



the active environment name will
show up in your prompt in
parentheses



now you can use the program as
per its documentation
(e.g.
<https://multiqc.info/docs/#running-multiqc>)

Basic shell scripting

What is a script?

- commands saved in a file
 - executable

Why?

- you could just run commands line by line forever
 - but REALLY NOT PRACTICAL
- scripts:
 - let you chain a bunch of commands together at once
 - are more reproducible
 - can be scaled up
 - do not need you
 - are helpful for your records

Objectives

- Provide just enough background to follow along with the logic you'll see in the pipeline scripts

Recap: Basic linux commands

- pwd
- mkdir
- ls
- cd
- cp
- man

Vocab

- shell
- script
- environment
- variables
- file path
- modules
- conda
- SLURM

Basic UNIX command syntax

- Command options name_of_file_to_act_on.txt

ls -l directory_name

- Look through a “man” entry
 - ‘ls’ as an example. How would I list all files with all the date, owner, etc. information?
 - What if I wanted to list all the files in the order they were edited?

Concepts

- Application: writing a shell script
 - Exercise: write a script to do something useful—like create the directory structure, then copy all the workshop scripts into it
- Concepts:
 - Basic commands
 - Creating a text file
 - The difference between a text file and a script
 - The shebang
 - Make executable
 - How to run a shell script

PRACTICE, PART II

- submit a script to slurm
 - practice with variables
 - loading modules in a submit-script
 - loading environments in a submit-script
-
- work in /work
 - move precious analyses to /home when you're done
 - practice good data and project-management

Exercise: Write and run a shell script

- Get reads data
- Evaluate quality