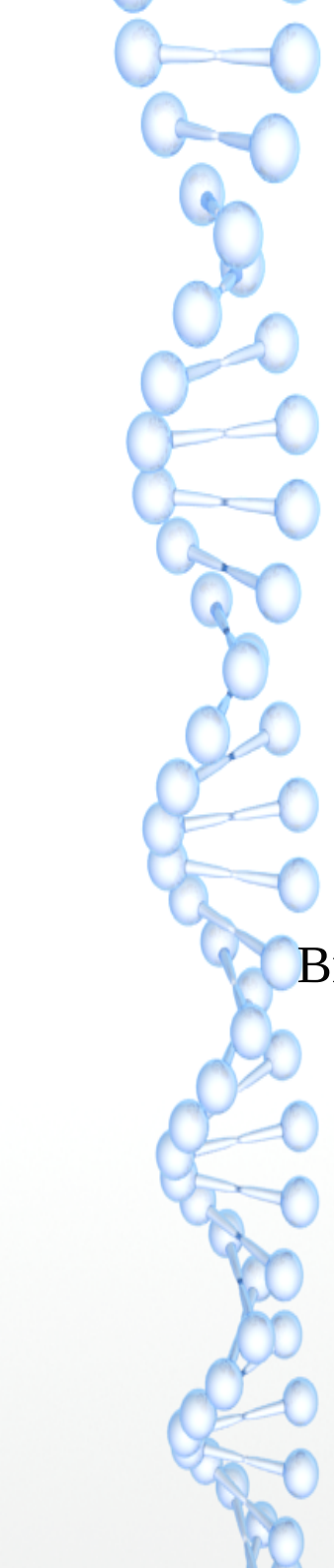


Introduction to Rosalind and Wilkins HPC

an introduction to Linux and high performance computing with SLURM

Brought to you by the TICR, sub-division of the Colorado Center for Personalized Medicine (CCPM)

Outline

- TICR, Rosalind, Wilkins
- HPC/Linux Intro
- Utilizing HPC (Bioinformatics Example)
  - More “Advanced” Linux
  - SLURM
  - Loading Modules and Local Installation

ITEMS/HANDOUTS

- Permissions PDF
- Rosalind Map PDF

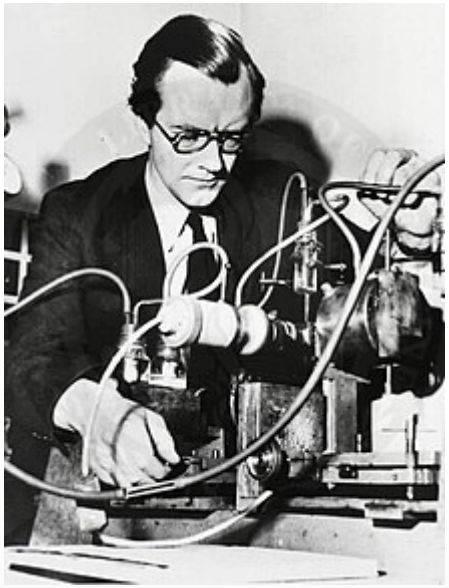
TICR, Rosalind, Wilkins

TICR
TRANSATIONAL INFORMATICS AND COMPUTATIONAL RESOURCE

OIT
DENVER | ANSCHUTZ

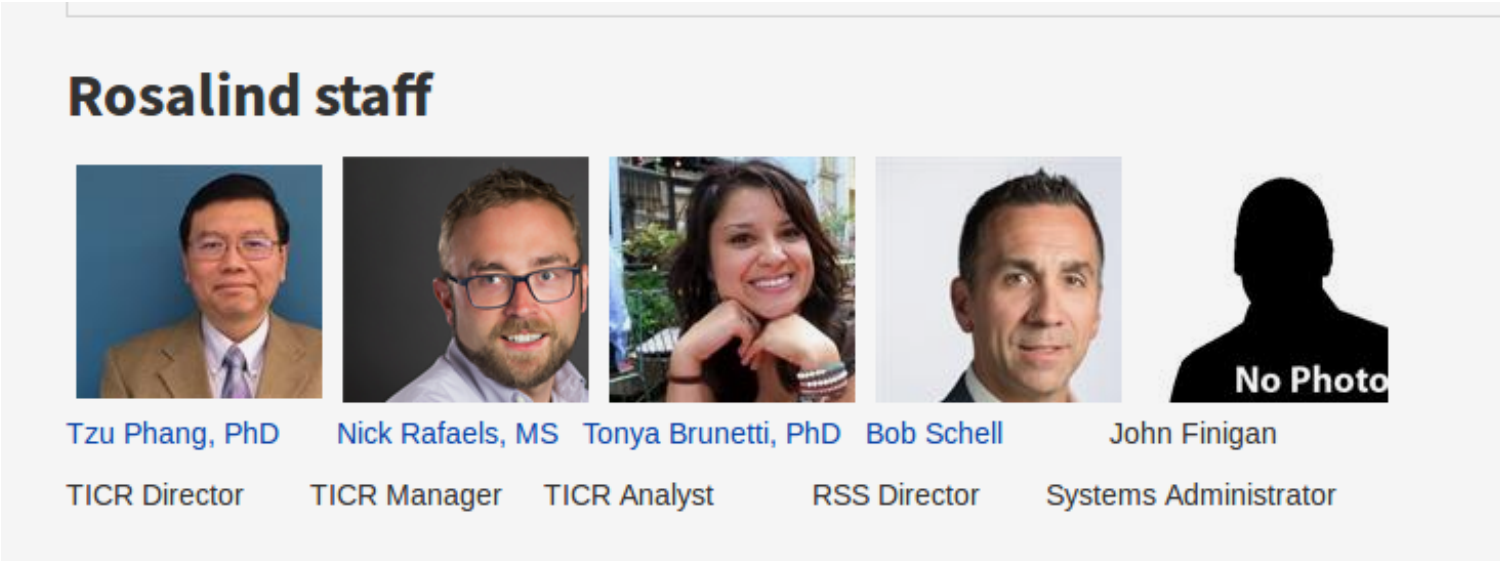
Research and Shared Services

COLORADO CENTER FOR PERSONALIZED MEDICINE

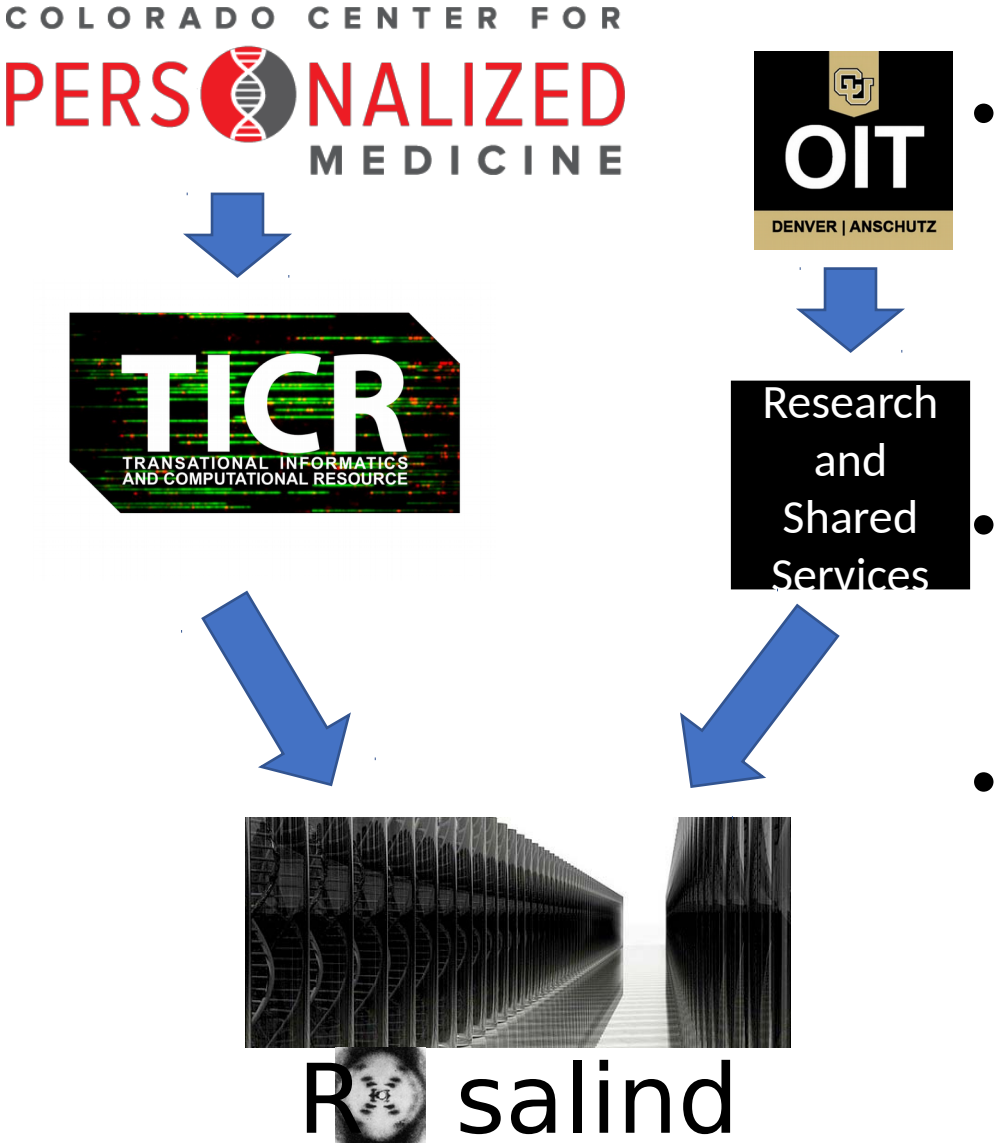


Who is TICR?

- **T**ranslational **I**nformatics and **C**omputational **R**esources
- **Roles and Responsibilities:**
  - Rosalind support and outreach
  - CCPM Biobank pipelines/analysis/validation/architecture development
  - computational informatics and programming support for CCPM
  - work closely with OIT and Health Data Compass to address CCPM and Biobank computational needs
- Vist our website at **www.ucdenver.edu/Rosalind**



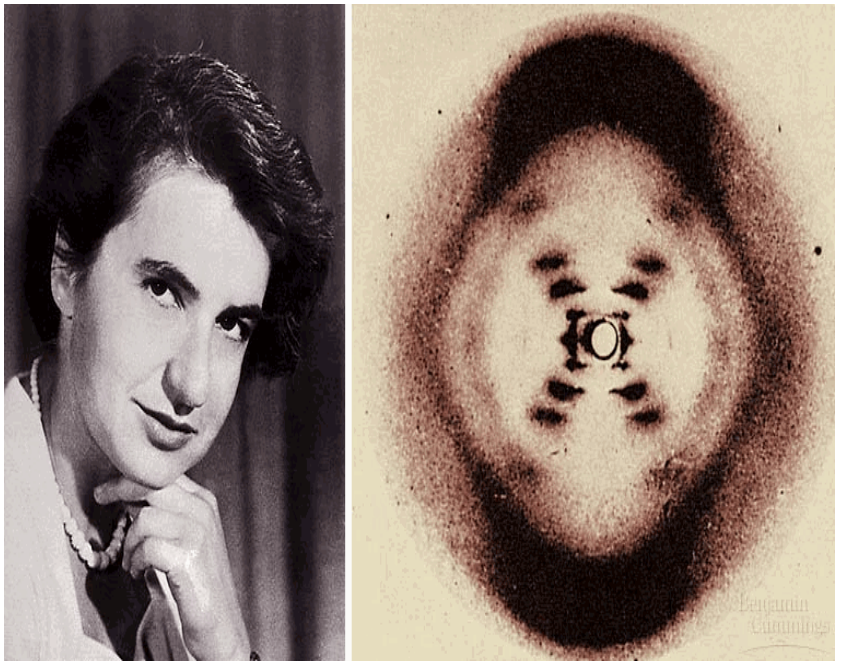
Introduction and Background

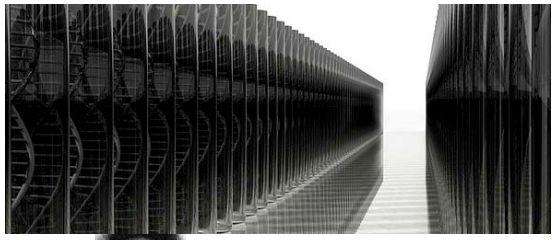


- The Translational Informatics and Computational Resource (TICR) within the Colorado Center for Personalized Medicine (CCPM) has partnered with
- Research and Shared Services (RSS) within the Office of Information Technology (OIT)
- To build a high performance computing system (HPC), Rosalind.

Who is Rosalind?

- Named after Rosalind Franklin, a chemist and X-ray crystallographer whose research was central to the discovery of the double-helix structure of DNA.
- Her colleague, Maurice Wilkins, passed along the X-ray image of DNA (on the right) to Watson and Crick in 1953, which confirmed the 3-D structure of DNA they had hypothesized
- Rosalind Franklin passed away from cancer in 1958 before Watson, Crick and Wilkins were awarded the Nobel Prize in 1962.





Rosalind

Swift
Secure
Solution to
Support
Science!

- Swift - Rosalind provides the capacity and the power to quickly process big data driven hypotheses
- Secure - Rosalind utilizes resources from OIT to store and backup your data safely, and provides a solution for analyzing highly sensitive data
- Solution - Big data queries drive innovations in science and health care
- Support - OIT and TICR provide support to facilitate your computational research needs
- Science - Choose Rosalind to discover new ways to look at science!

Rosalind HPC metrics and cost

- Compute and Storage
  - 768 cores at 128 GB RAM per node (4TB and 32 nodes total)
  - 72 cores at 1.5 TB RAM per node (3TB and 2 nodes total)
  - 3.7 PB of usable storage
- Cost
  - Compute Costs Standard Rate: $0.121 per core-hour
  - Storage Costs Standard Rate: $0.02 per GB/month
- Billing
  - Occurs on a monthly basis by speed type linked to project
  - Billing within project is sorted by individual based on compute and storage.

redhat.

slurm
workload manager

Rosalind HPC cost example

- Raw data and result storage: 500 GB
- Temporary data storage: 200 GB
- Computational needs: 40 core-hours / sample = 40 * 30 = 1,200 core-hours
- Estimated prototyping and statistical analysis computational needs: 20% of above: 240 core-hours

Sample(s)	30

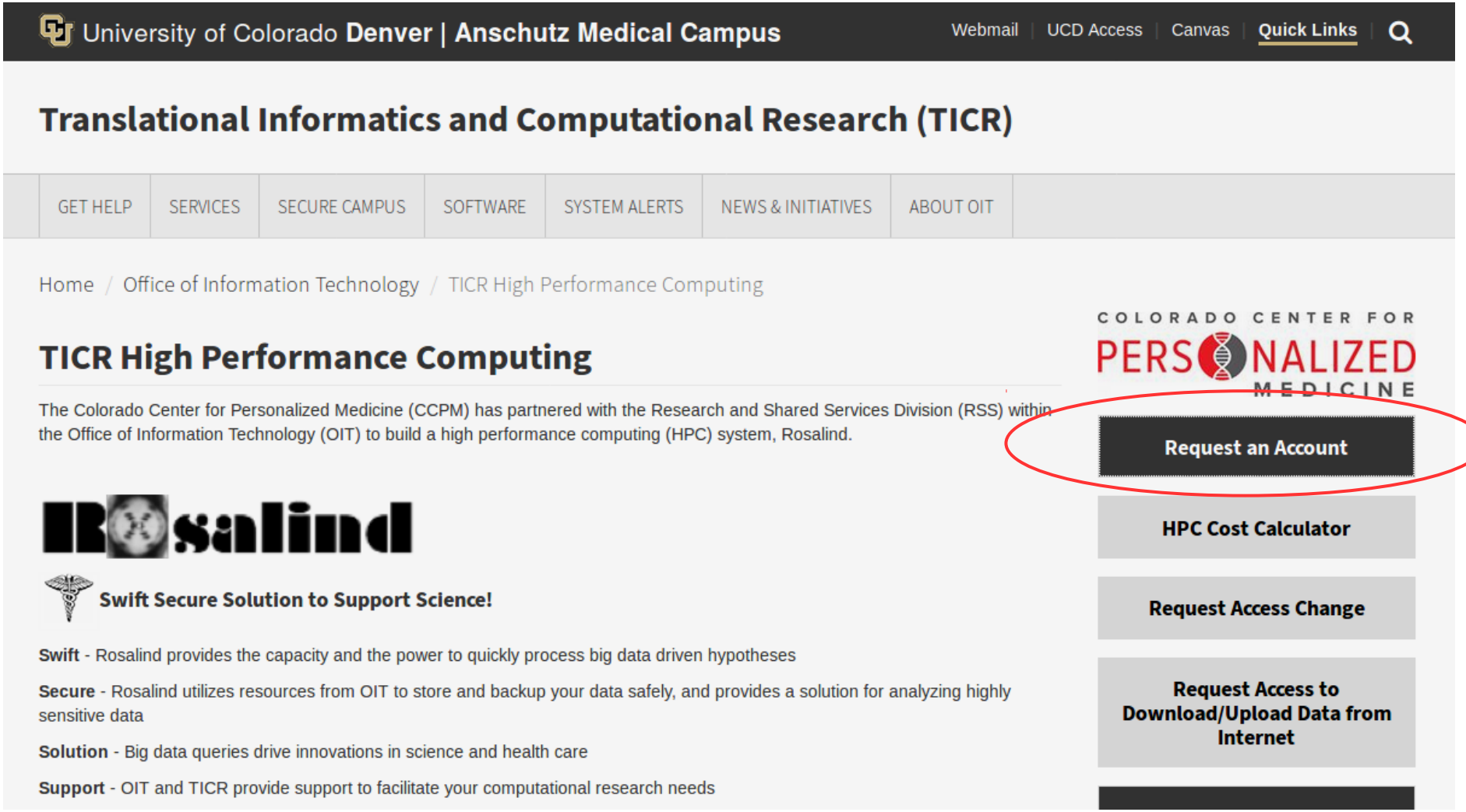
Usages	Cost	Your Needs	Time Needed	Total
Raw Data Storage	$0.02/GB/month	500	6	$60.00
Intermediate Files Storage	$0.02/GB/month	200	3	$12.00
Computation	$0.121/core-hour	1200		$145.20
Prototyping	$0.121/core-hour	240		$29.04
Estimated Cost				$246.24
Per Sample Cost				$8.21

You can price out your own project needs:

[HPC cost calculator]

How to apply for an account on Rosalind?

www.ucdenver.edu/Rosalind
CCPM-Rosalind@ucdenver.edu



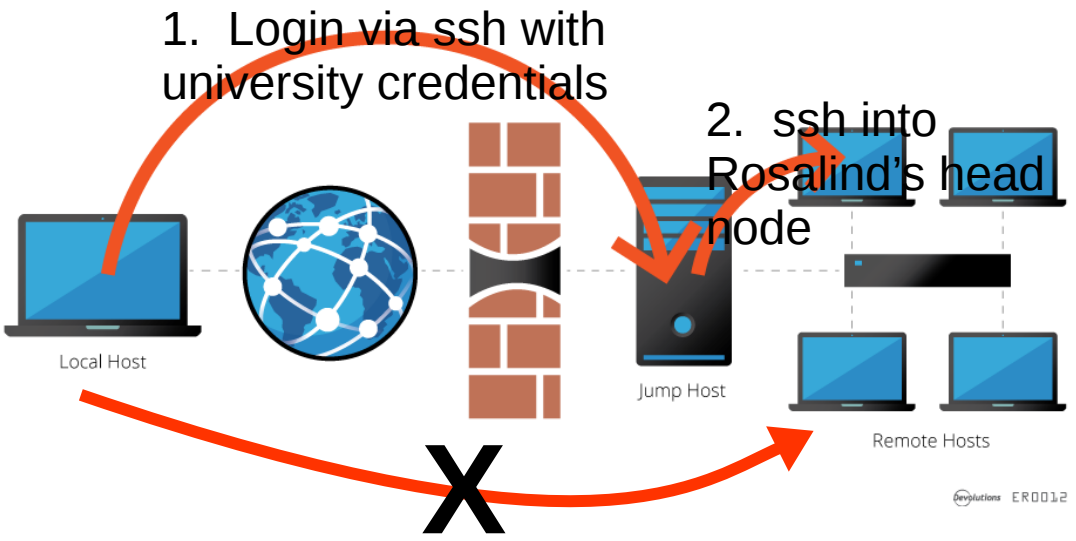
Why learn on the Wilkins demo environment when we could learn on Rosalind?

- Rosalind is HIPAA compliant
  - Secure!
  - You can put PHI on here, where most HPCs do not support this (Biobank with Compass?)
  - You can put sensitive information on here, where most HPCs do not support this
  - We are officially supported by AMC and OIT
  - We offer a lot of one-on-one, case-by-case customer service to help make your experience on the HPC better

In order to keep Rosalind HIPAA compliant we require certifications and training and Rosalind lives behind a jump host, so we mimic how Rosalind works on a non-HIPAA compliant demo system we named Wilkins

What is a jump host?

A jump host, also known as a jump server, jumpbox, or secure administrative host, is a server that allows you to connect or “jump” to other remote servers



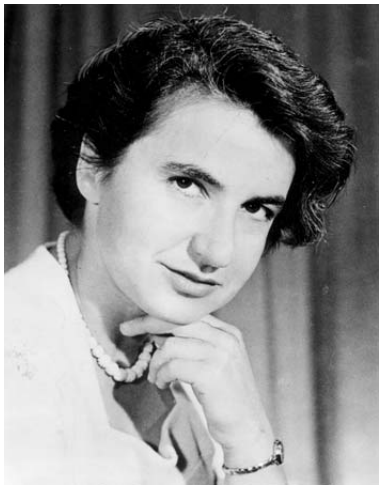
Why can’t I just connect to Rosalind directly?

- Rosalind is HIPAA-compliant meaning many users may have highly confidential and sensitive information about people such as electronic health records (EMR), protected health information (PHI), genomic data, transcripts and grades of students, etc… so we must take every precaution to protect this
- It minimizes risk to Rosalind by making malware and viruses harder to penetrate and compromise the system by forming a firewall from the Internet and outside world

Wilkins Demo vs Rosalind Environment

Goal: Mimic how Rosalind works as closely as possible on the Wilkins environment

- Not HIPAA compliant
- No jump host
- Very limited amount of RAM and space



- HIPAA compliant
- Has jump host
- Lots of RAM and space

Rosalind Support and Resources

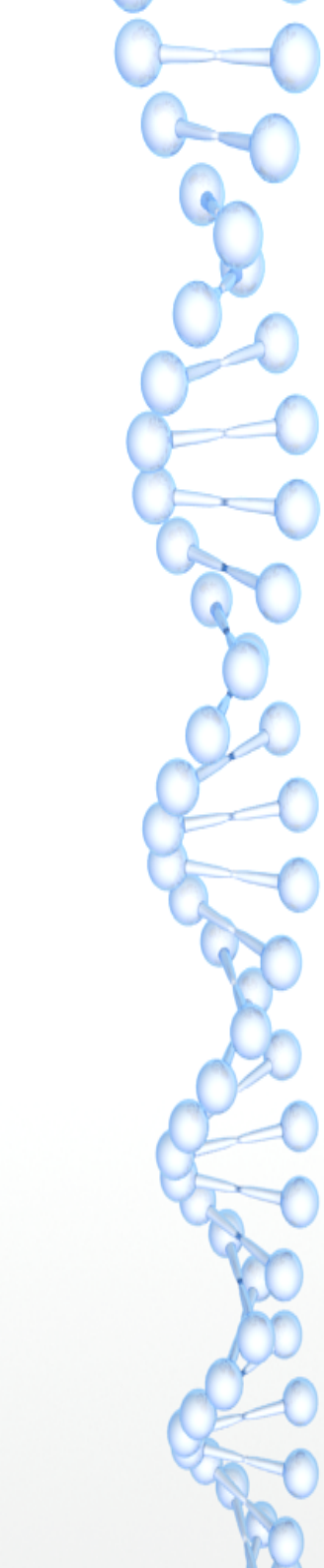
- Our Website: www.ucdenver.edu/Rosalind

- Github help page/getting started
github.com/tbrunetti/Rosalind_HPC

- Email: CCPM-Rosalind@ucdenver.edu
hpcsupport@ucdenver.edu

- Bootcamps

- Write HPC into your Grant

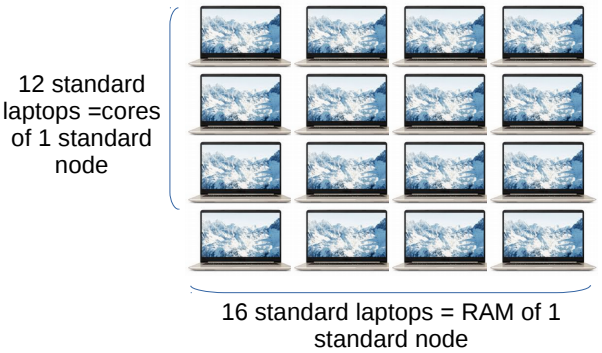


Why the HPC and why Linux?

Benefits of HPC

Standard laptop
~8GB RAM,
500GB hard drive,
dual-core
processor

Rosalind Standard Compute Node
128 GB RAM, 24 cores

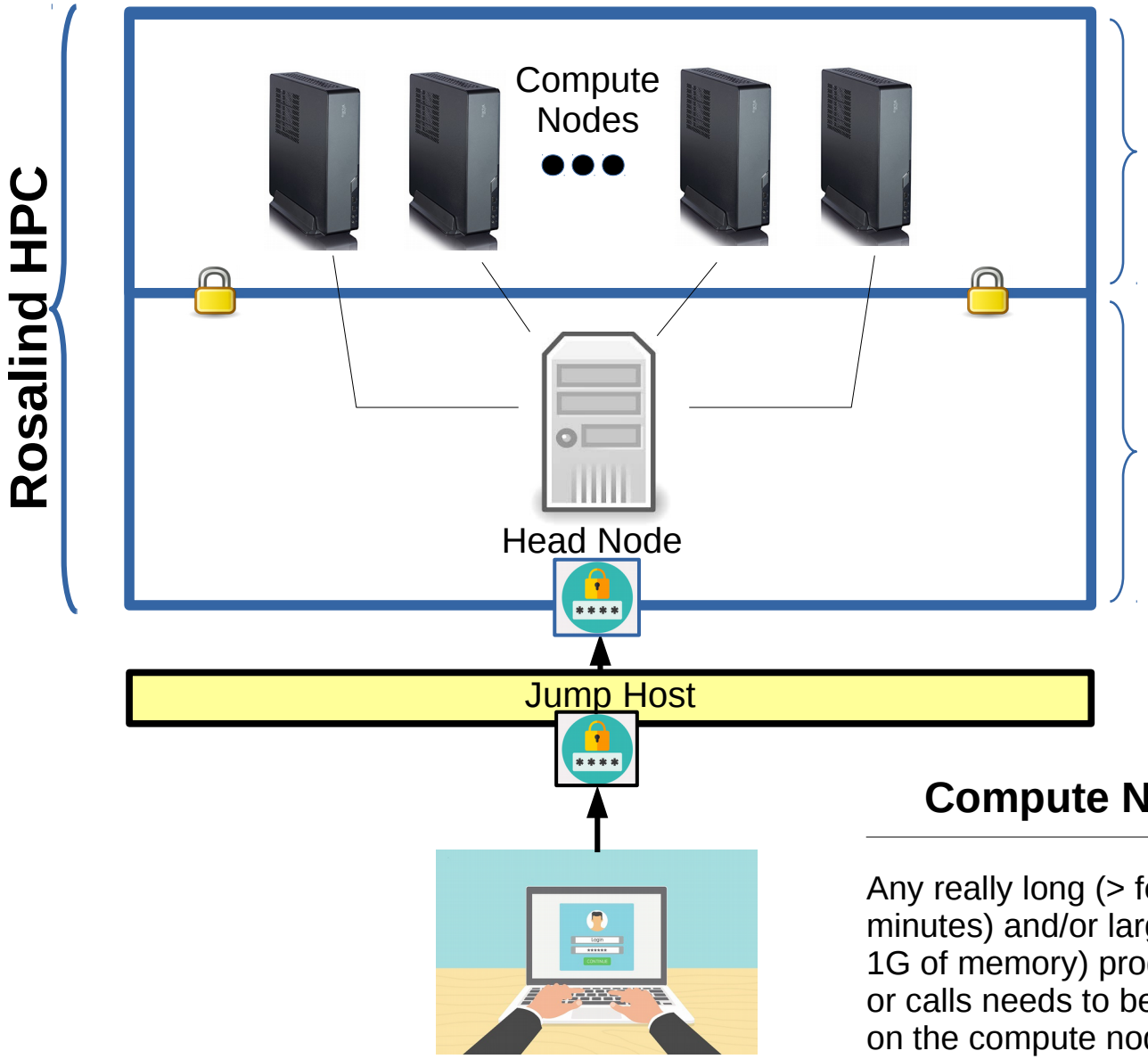


Rosalind Standard Compute Node Cluster
4096 GB RAM, 768 cores

The two high memory nodes add an additional 3TB of RAM and 72 cores!

GRAND TOTAL: 7,096 GB RAM (~7TB) and 840 cores with 3.7 PB storage
OR...

887 laptops to achieve the same RAM and 420 laptops to achieve the same number of cores!



The **compute nodes** are where all the power of the HPC is harnessed.

The only node the user directly interacts with on Rosalind is the **head node**.

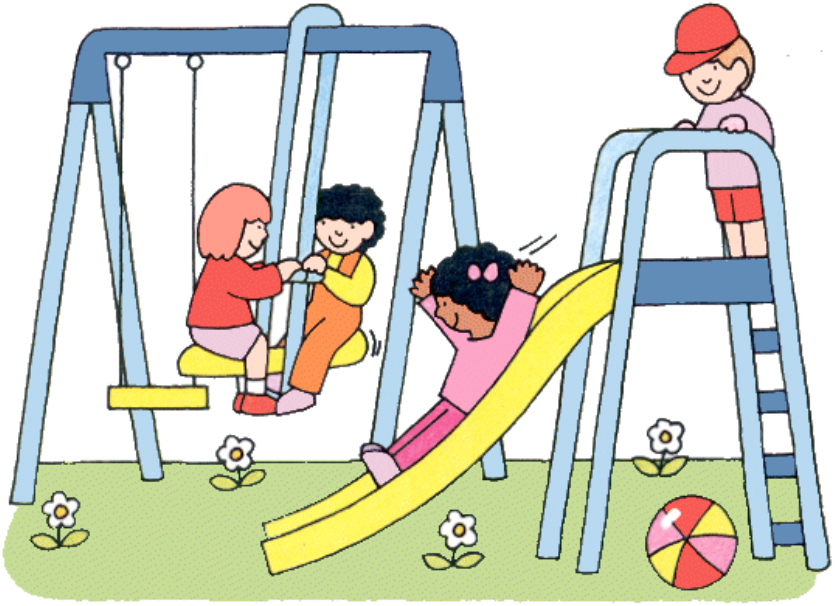
This node is the master or brain behind HPC.

Compute Node	Head Node
Any really long (> few minutes) and/or large (> 1G of memory) programs or calls needs to be used on the compute nodes. The compute nodes wait for the head node to assign them to a job or task.	Anytime a user makes calls through the command line on Rosalind, it is being performed on the head node. The head node is very small and limited in ability. Its main function is to work with the scheduler and delegate tasks and jobs to the compute nodes that users want to utilize.

Rosalind is a shared resource

What does that mean?

- This means that there are several different users and projects on Rosalind all sharing the same compute resources
- This means we have to be mindful of a few things:
    - we don’t want to install software for everyone since different groups will want their own versions
    - We want to make sure all your data is secure and only seen by you and your group members
    - We also want the resources to be shared among our users in the fairest way possible



How does that work?

Linux securities in place
Job scheduler (SLURM)

What is the difference between your personal computer and an HPC

- CPU, RAM and storage are very limited
- You have administrative master access
  - You can install anything you want for all users
- You can run anything you want at any time you want

- Large RAM/storage
- No sudo privileges
  - You can only install software for yourself in your own space, not across the whole HPC
- Fairly share resources using a resource management system

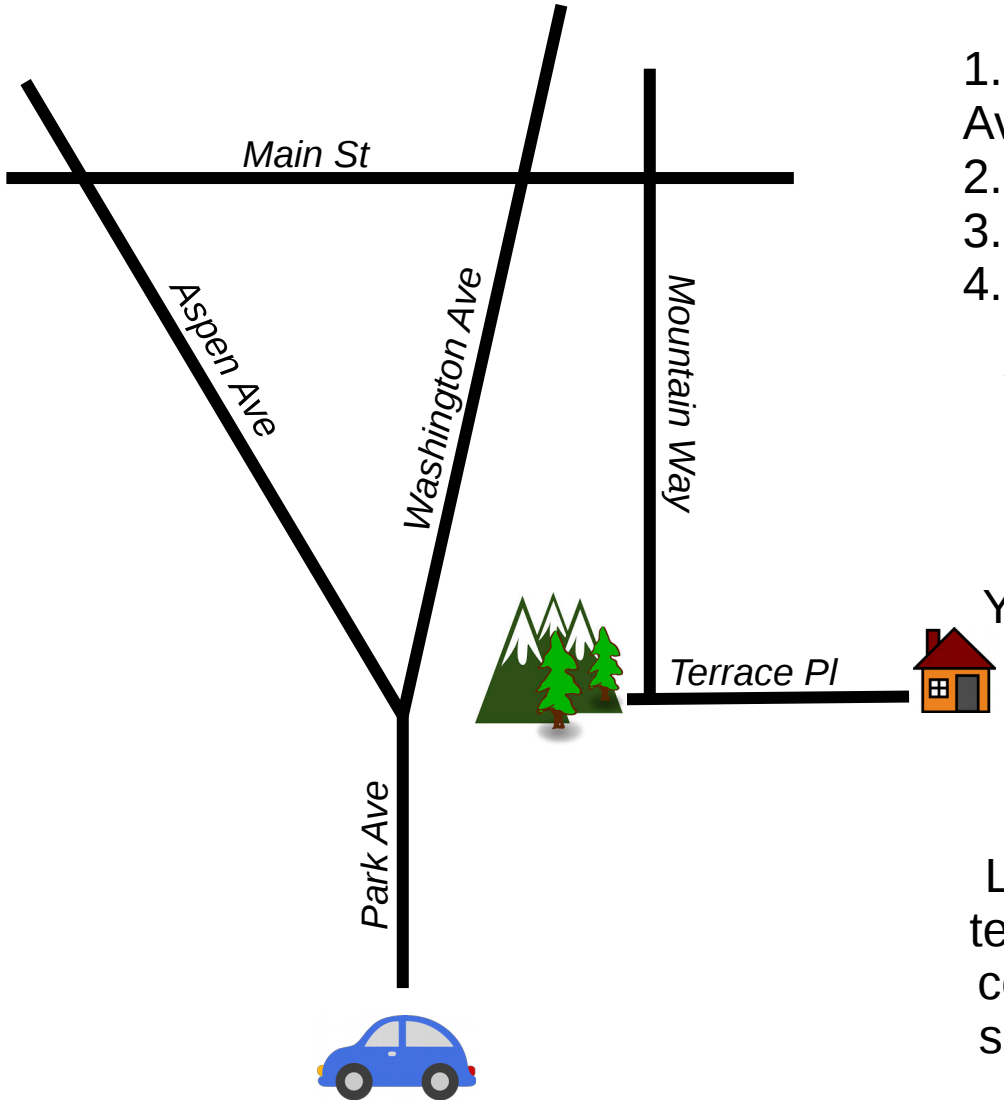
What is Linux and Why?

- Scalability
- Open Source OS
- Flexible
  - Ability to communicate with a variety of networks and OS
- Linux has has been around since 1991 with Unix making its first appearance 1983 therefore making is a fairly developed and well-understood OS

Mac OSX (Unix back end) 2001; all older version macintosh OS

Since 1985; based on Micosoft Disk Operating System (MS-DOS) - single-user single-task operating system

Navigating through directories in Linux is similar to telling a friend who has never been to your house step-by-step directions



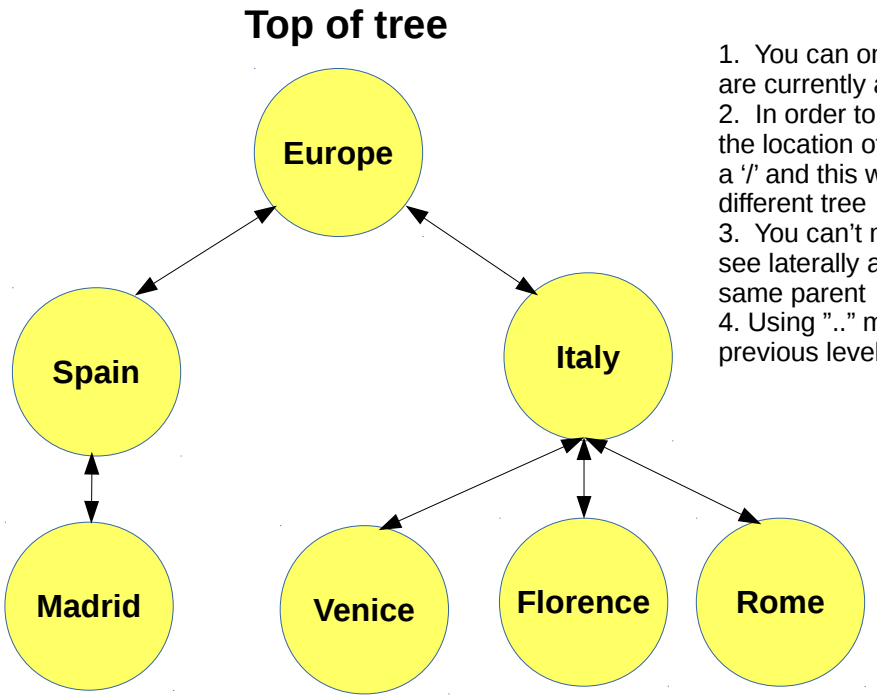
1. Take a right at the fork onto Washington Ave
2. Turn right onto Main Street
3. Turn right onto Mountain Way
4. Turn left onto Terrace Place

You wouldn't tell your friend to find Terrace Place first if they are unfamiliar with your neighborhood because they will get lost!

Your friend can only see what is right in front of them or right behind them but they can't foresee any streets beyond that

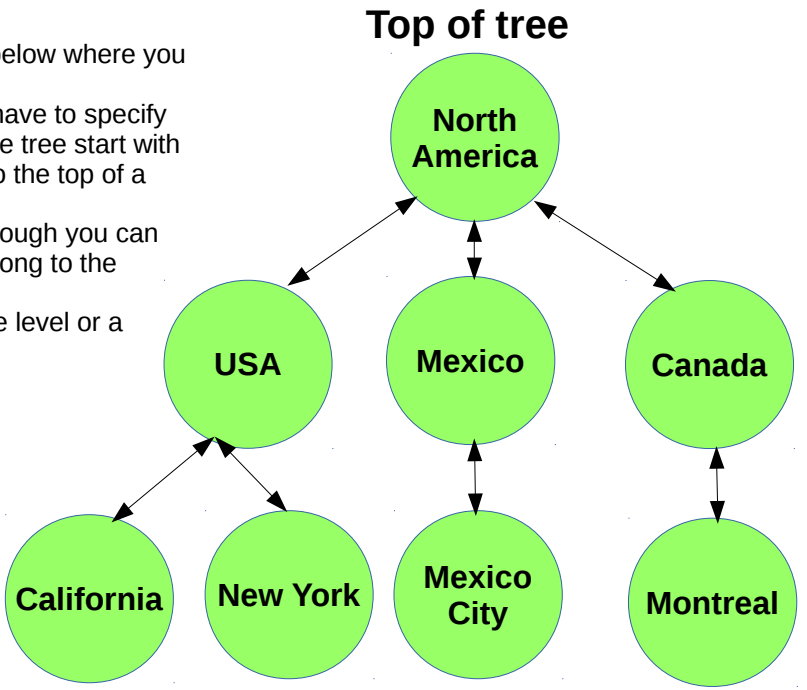
Linux path addresses are very similar. They tell the computer directions on how to get to a certain directory or file. However, it is simpler since it uses a tree architecture. Let's take a look at a simple example

Example of Linux Path Architecture



Rules:

1. You can only see one level below where you are currently at
2. In order to cross trees, you have to specify the location of the very top of the tree start with a '/' and this will "teleport" you to the top of a different tree
3. You can't move laterally, although you can see laterally as long as they belong to the same parent
4. Using ".." means to go up one level or a previous level



If I was in North America, I could get to New York by taking the following path:
USA/New York. If I then decided I wanted to go to California from New York I could do a few things:

- /North America/USA/California
- ../California ← Why does this work? Remember .. means to move up one level from current!

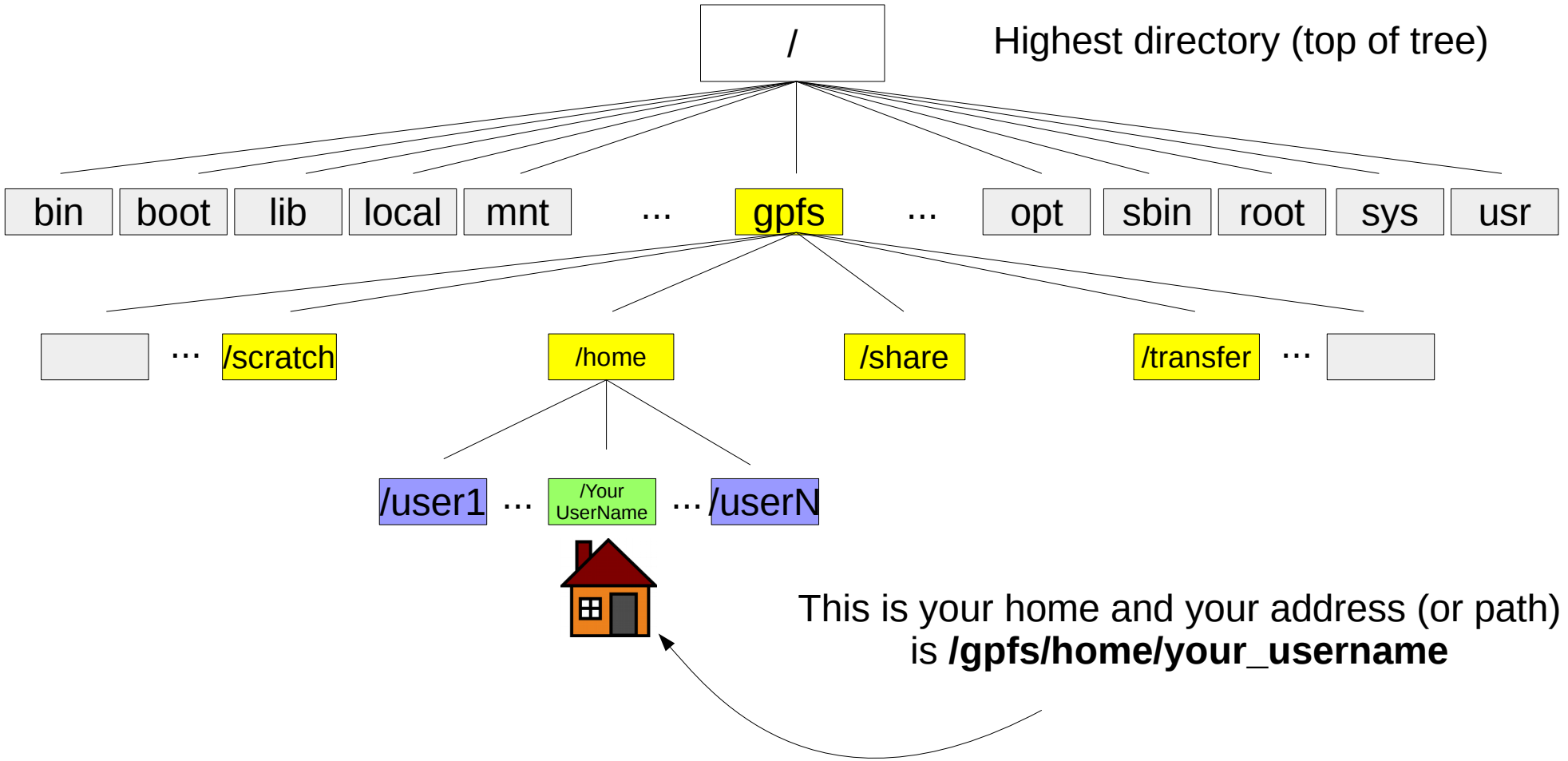
Starting at Europe, how do you get to Rome? From Rome what is the path to Venice?

/Italy/Rome ../Venice **or** /Europe/Italy/Venice

If you are in Rome, how do you get to Mexico? From Mexico what cities can you see?

/North America/Mexico Mexico City

Rosalind Road Map



= sudo required

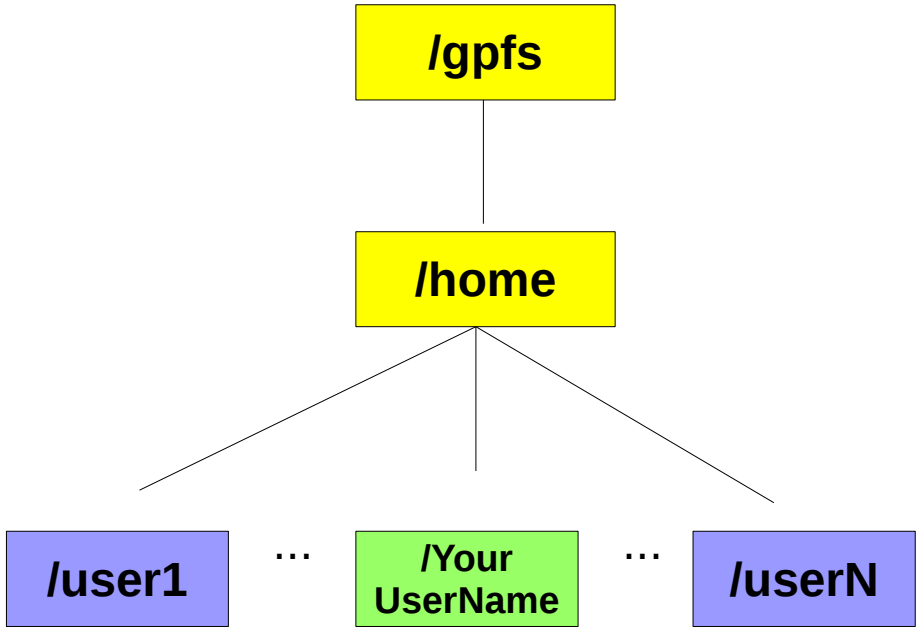
= limited access

= full access*

= no access w/o permission

But what person doesn't have a shortcut to get home?!...

Shortcuts to home



$HOME is a variable that will always take you back to /gpfs/home/your_usernname

$HOME = /gpfs/home/your_username

= sudo required
= limited access
= full access*
= no access w/o permission

Sometimes you will see /homelink/your_username and this is because we have created a link to called /homelink that links to /gpfs/home

Let’s login!

Was everyone able to install an ssh client or open a terminal that is not MS-DOS?

Open your favorite ssh client or open a terminal if you plan on using the command line to log in

Windows Users

PuTTY
MobaXterm
ZOC
Private Shell
Tera Term
LogMeTT
Cygwin w/openSSH

Mac Users

terminal.app
iTerm2
Termius
ZOC
PuTTY

Linux Users

Command prompt w/openSSH (CTRL+ALT+T) for debian/ubuntu (CTRL+SHIFT+T) for Redhat/Fedora/CentOS

```
[brunettt@myLocalComputer ~]$ ssh brunettt@cubipmtest02.ucdenver.pvt
```

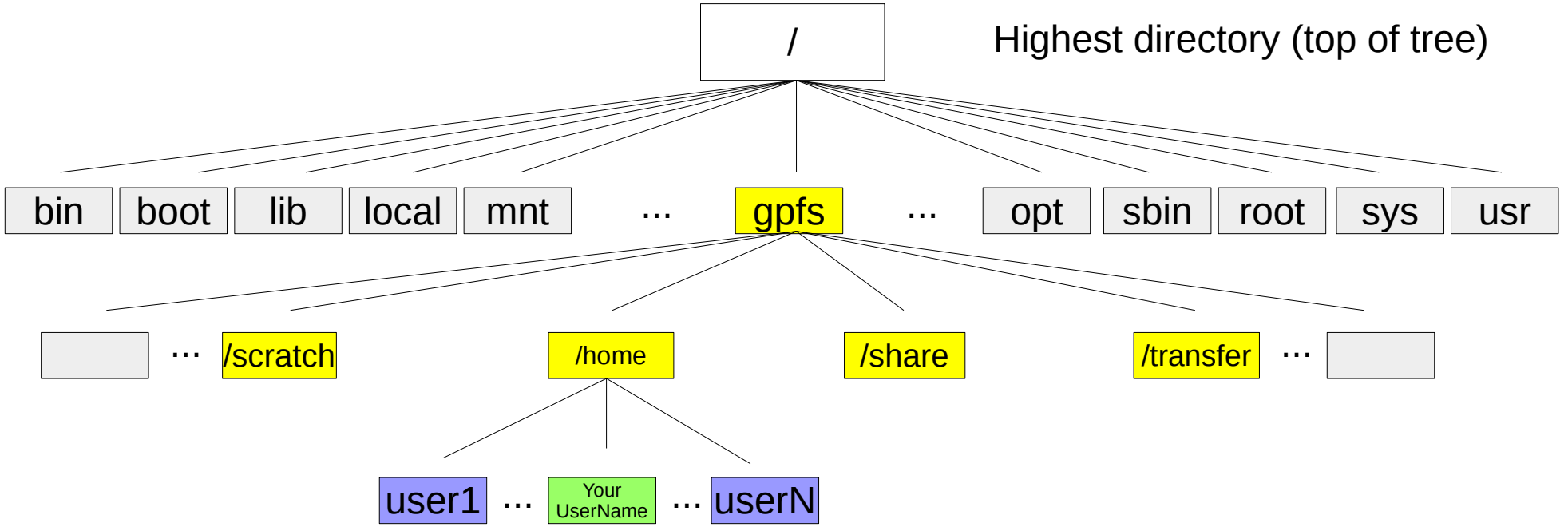
Where are you?

- **p**rint **w**orking **d**irectory

```
[brunettt@cubipmtest02 ~]$ pwd
```

```
/homelink/brunettt
```

- This shows you that the *folder (aka directory)* labeled as your username (brunettt) lives inside the directory homelink
- Each time you ssh into the head node, it will always take you to your home directory located on the head node



What should be stored in /homelink/your_username?

- small reference files
- scripts
- locally installed programs

Space is very limited in home!

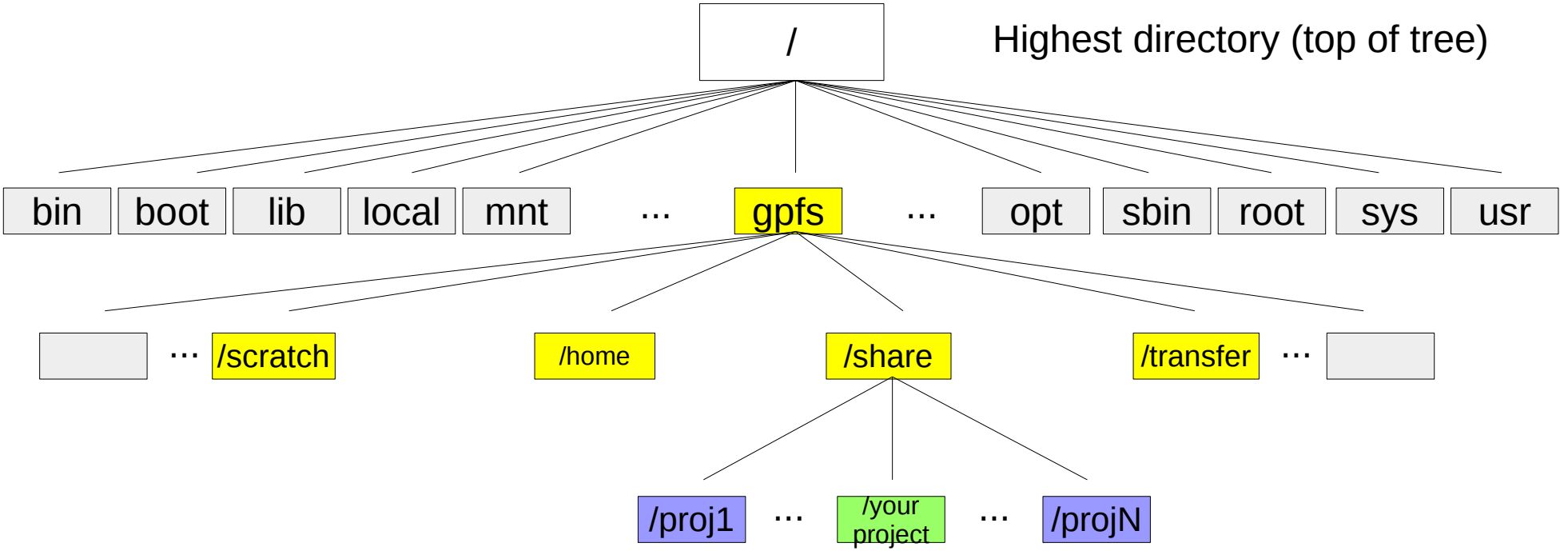
But I have a LOT of data!

= sudo required

= limited access

= full access*

= no access w/o permission



= sudo required

= limited access

= full access*

= no access w/o permission

What should be stored in /gpfs/share/your_project?

- programs, scripts, files that are shared between member of your project
- Large data sets and text files
- Output generated from running programs and scripts

LOTS OF SPACE! Most of your files and output should be stored here!

How do we get travel from home to our project share?

Navigation through Rosalind

- **c**hange **d**irectory

  cd is used to change directories. You can follow the cd command with any full length path or any relative path

- The general format for using it is the following: **`cd <your destination address>`**

```
[brunettt@cubipmtest02 ~]$ cd   /gpfs/share/training
```

```
[brunettt@cubipmtest02 training]$ pwd
/gpfs/share/training/
```

1 Using `cd`, try to navigate to your project share directory. 2 After you change directories, how can you confirm your location?

1

```
cd ../../gpfs/share/myProject
cd /gpfs/share/myProject
```

2

```
pwd
```

What items are in your project share?

- **l**ist **s**egments

```
[brunettt@cubipmtest02 training]$ ls
```

```
drosophila_fastq  drosophila_ref_files README.txt
```

- This will list all the files and folders in your present working directory
- You may be interested in knowing more about these files so we can give the `ls` command 2 optional arguments

```
[brunettt@cubipmtest02 training] ls -lh
```

```
[brunettt@cubipmtest02 training] ls -lh
```

```
[brunettt@cubipmtest02 training]
drwx--S---. 2 brunettt ticr_wilkins_user  156 Jun  8 12:30 drosophila_fastq
drwx--S---. 2 brunettt ticr_wilkins_user 4.0K Jun  8 11:20 drosophila_ref_files
-rw---S---. 1 brunettt ticr_wilkins_user  809 Jun  8 12:41 README.txt
```

Type, permissions, ownership

size

Date and time modified

File/directory name

Important note about size: If you are looking at the size of a directory, the *size does not reflect the entire size of all the contents in a directory*. In order to check that you must use the following Linux command:

- **d**isk **u**sage

```
[brunettt@cubipmtest02 training] du -sh drosophila_ref_files
```

```
379M drosophila_ref_files/
```

What is the total size of the entire drosophila_fastq directory? What is the size of the README.txt file?

drosophila_fastq: 27GB, README.txt; 809KB

How do you know when there are options?

- **man**ual

```
[brunettt@cubipmtest02 trainng]$ man ls
```

- Linux commands usually come with built-in manuals that you can access from the command line
- Tells you how to use the command and the optional arguments you can add and what they do
- Most sections are labeled as:
  - **NAME**
  - **SYNOPSIS**
  - **DESCRIPTION**
  - **AUTHOR**
  - **COPYRIGHT**
  - **SEE ALSO**

What optional argument(s) would you add to the **`ls`** command to sort the most recently used files to appear at the end of the list? What does the **`sh`** that I added to the **`du`** command in the previous slide do?

-rt

summarize and human readable sizes

Keeping your stuff secure-- giving permission and changing ownership

- You may have noticed using the **`-lh`** options resulted in being able to view permissions and file ownership

```
[brunettt@cubipmtest02 training]
drwx------. 2 brunettt ticr_wilkins_user  156 Jun  8 12:30 drosophila_fastq
drwx------. 2 brunettt ticr_wilkins_user 4.0K Jun  8 11:20 drosophila_ref_files
-rw-------. 1 brunettt ticr_wilkins_user  809 Jun  8 12:41 README.txt
```

Type, permissions, ownership

User who owns this file (default is the person who created it)

Group you want to give access to this file/folder (default is person who created it, but in HPC, you can change this to the name of your project share so users who are in the same project can share files between each other

What do the permissions mean?

- Open PDF: Understanding_File_Permissions.pdf
- You have control over which who can read, write/modify, or run your files and programs

```
[brunettt@cubipmtest02 training]
drwx------. 2 brunettt ticr_wilkins_user  156 Jun  8 12:30 drosophila_fastq
drwx------. 2 brunettt ticr_wilkins_user 4.0K Jun  8 11:20 drosophila_ref_files
-rw-------. 1 brunettt ticr_wilkins_user  809 Jun  8 12:41 README.txt
```

Type and Permissions

- **ch**ange **mod**e

```
[brunettt@cubipmtest02 training] chmod g+rx drosophila_fastq
```

This means I am giving the group read access to the files in the directory. Note the x means executable and this has to be flagged in order for the group to access the directory and change into it

Bit flag	Description
read	User in set can open a file to read it only
write	User in set can open a file and modify it or can create new files in the directory
execute	User in set can change into directory or can execute the file if it is a program or script

Keeping your stuff secure-- giving permission and changing ownership

- You may have noticed using the **`-lh`** options resulted in being able to view permissions and file ownership

```
[brunettt@cubipmtest02 training]
drwx------. 2 brunettt ticr_wilkins_user  156 Jun  8 12:30 drosophila_fastq
drwx------. 2 brunettt ticr_wilkins_user 4.0K Jun  8 11:20 drosophila_ref_files
-rw-------. 1 brunettt ticr_wilkins_user  809 Jun  8 12:41 README.txt
```

Type, permissions, ownership

User who owns this file (default is the person who created it)

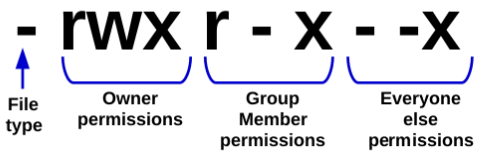
Group you want to give access to this file/folder (default is person who created it, but in HPC, you can change this to the name of your project share so users who are in the same project can share files between each other

How do can I calculate the permissions?

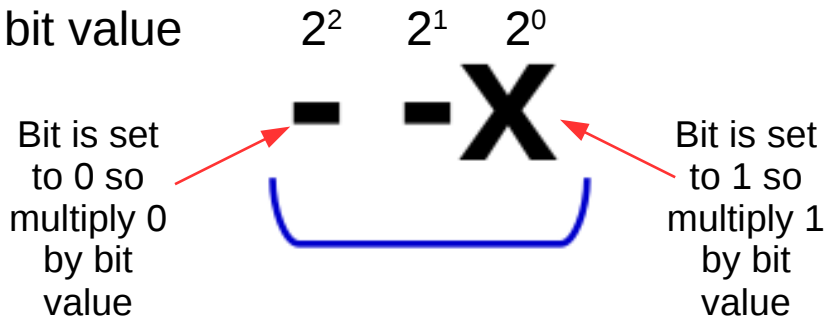
Numerically

Bit-wise calculation

Each set is comprised of 3 bits, and a bit is binary, meaning it is either set to 1 or 0.



If there is a “-” in the permissions is means that bit position is set to 0. If there is a letter in that bit position it means it that bit position is set to 1.



Everyone :$(0*2^2)+(0*2^1)+(1*2^0) = 1$

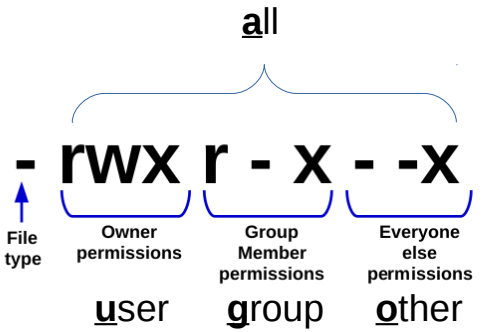
Group: $(1*2^2)+(0*2^1)+(1*2^0) = 5$

Owner: $(1*2^2)+(1*2^1)+(1*2^0) = 7$

`chmod 751 myFile.txt`

Symbolically

Each bold and underlined letter is the alphabetical abbreviation for each set of permissions



+ means to add the permission to the specified set and **-** means to subtract or remove permissions from a set

`chmod a+x,ug+r,u+w myFile.txt`

Why and how to change ownership

- **ch**ange **own**er




```
[brunettt@cubipmtest02 brunettt] chown :ticr_wilkins_user drosophila_ref_files
```

```
[brunettt@cubipmtest02 brunettt] ls -lh
```

```
[brunettt@cubipmtest02 brunettt]
drwxr-x---. 2 brunettt brunettt                156 Jun  8 12:30 drosophila_fastq
drwx------. 2 brunettt ticr_wilkins_user      4.0K Jun  8 11:20 drosophila_ref_files
-rw-------. 1 brunettt brunettt                809 Jun  8 12:41 README.txt
```

- Important because the person who owns the file or project ultimately makes decisions on who is granted access
- This would change group from **`brunettt`** to **`test`** meaning <u>any user part of the test project would have the group permissions to this file/directory</u>

General format for changing ownership:

`chown owner:group myFile.txt`

`chown :group myFile.txt`

`chown owner myFile.txt`

Based on the output above, what permissions does the group test have to drosophila_ref_files?

Trick question: None!

What command would we have to use to give the test group read, write, and executable access to drosophila_ref_files?

chmod g+rwx drosophila_ref_files or use 770

What are in these directories and files?

We have added a README.txt file that has a description of what the files mean. How can we view it?

- **less**

```
[brunettt@cubipmtest02 training] less README.txt
```

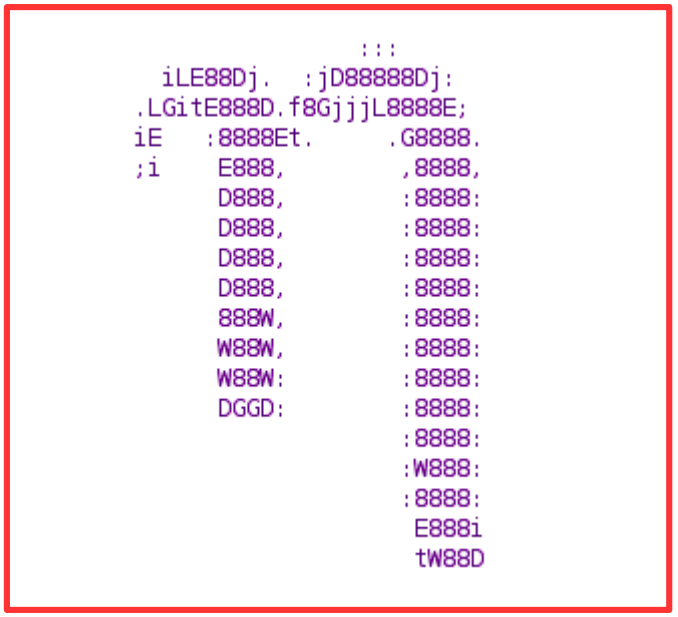
- Similar to those familiar with **more** but with added functionality.
- Allows you to open a file to read, scroll, and search only! Cannot modify!
- To exit press “q” for quit

What is the purpose of the files located in drosophila_ref_files?

Navigate to your home directory. Open the README.txt file. What items are in scripts vs software?

Manipulating files

- You may have noticed that **`less`** does not allow you to modify any files, just read them. In order to modify a file you need to use a **text editor**
- There are many flavors of text editors and many are installed on most HPCs: nano, vi/vim, emacs are some of the more commonly used editors



Most user friendly → Steeper learning curve

- Navigate to your share folder and let's open README.txt using nano

```
[brunettt@cubipmtest02 training] nano README.txt
```

```
 GNU nano 2.3.1                          File: README.txt

Welcome to the Introduction to Rosalind HPC Tutorial!  We are happy to have you here!

Here is an explanation of all the contents that are in your project share

Directory: drosophila_fastq
        -This contains sequencing data from the model organism Drosophila melanogaster in fastq format.  Fastq (.fq, .fastq) is the raw data f$


Directory: drosophila_ref_files
        -This contains all the reference files and reference file indicies (required for alignmnet) for Drosophila melanogaster.  This is what$
```

- nano is great because it gives you a cheat sheet at the bottom of how to perform everything! **"^" means to use the control button**

```
                              [ Read 10 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

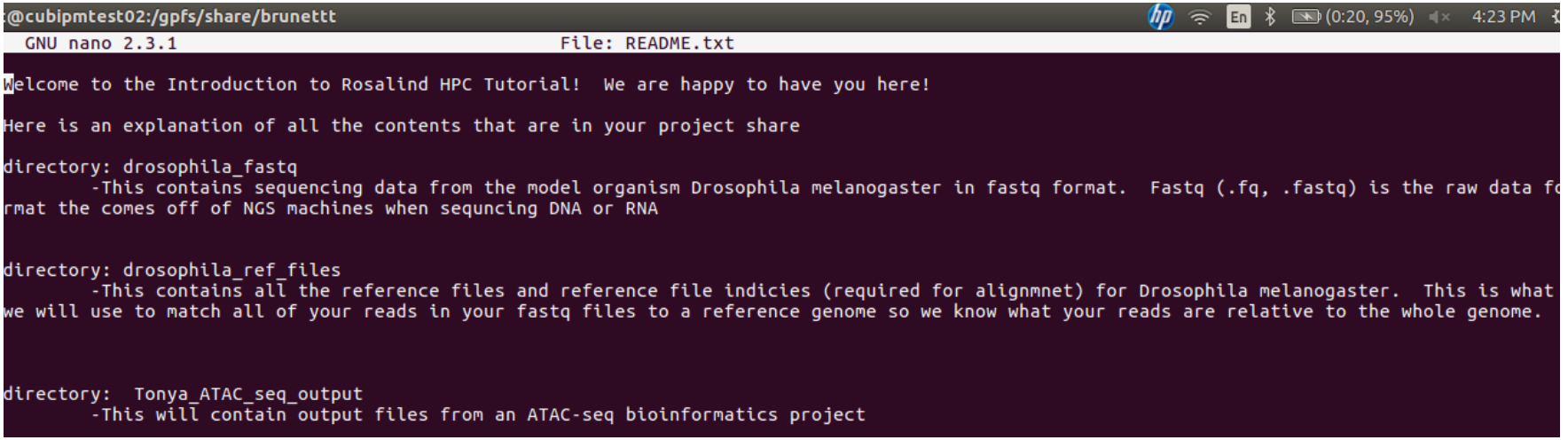
- First call the "Get Help" command (^G or ctrl+G). You will see that at the bottom it tells you how to wrap the text so that the line doesn't run over to the next page

```
(M-B)                   Backup files enable/disable
(M-F)                   Multiple file buffers enable/disable
(M-M)                   Mouse support enable/disable
(M-N)                   No conversion from DOS/Mac format enable/disable
(M-Z)                   Suspension enable/disable
(M-$)                   Soft line wrapping enable/disable

^Y Prev Page            ^P Prev Line            ^X Exit
^V Next Page            ^N Next Line
```

means after exiting help (ctrl+x) press "esc" and then press "shift+4" since that is the dollar symbol

- Now you can see the text in the README.txt files has been wrapped so all the words do not overflow off the screen.



- Now scroll to the bottom of the file and add the following lines:

```
directory:  yourname_ATAC_seq_output
        -This will contain output files from an ATAC-seq bioinformatics project
```

- Then go ahead and save your file. Give it a new file name by using ctrl+O and typing in a new file name and pressing enter. Please put your name or initials Somewhere in the file name. Then exit nano.

Other ways to find information

- **nano** does a lot such as opens, reads, modifies and searches within files. However there are instances when a text editor is not the best option.

WHY? Let's look at an example:

- Navigate into the drosophila_fastq directory located in your share folder. What are the sizes of the files in this directory?

```
[brunettt@cubipmtest02 training] cd /gpfs/share/brunettt/drosophila_fastq
[brunettt@cubipmtest02 training] ls -lh
total 27G
-rw-------. 1 brunettt ticr_wilkins_user 14G Jun  7 16:11 SRR4044399_1.fastq
-rw-------. 1 brunettt ticr_wilkins_user 14G Jun  7 16:29 SRR4044399_2.fastq
```

Relatively large files can take a long time to open/load/search, so less or a text editor may not be the best option to quickly peek in a file

- **head**

```
[brunettt@cubipmtest02 drosophila_fastq]$ head SRR4044399_1.fastq
@SRR4044399.1 V00D00575:88:HH22CADXXa:1:1101:1365:1896 length=51
NTTTTGATCAGTTAATATTGGCTCGGATCGCCTTCTACGCTGACAGCTGTC
+SRR4044399.1 V00D00575:88:HH22CADXXa:1:1101:1365:1896 length=51
#1:BDDBDFFDFHHHIIJJGIEHGIIFGAFFFBHDGICCGFDGIIIHIGIG
@SRR4044399.2 V00D00575:88:HH22CADXXa:1:1101:1287:1915 length=51
NGTCTACACGGTAACGATGCCCACAAGATGTCTCAGACTCGCCCATAACCT
+SRR4044399.2 V00D00575:88:HH22CADXXa:1:1101:1287:1915 length=51
#4:DDDFFHHGHHIGHFJJJJIJIHEFHIJJJJJJJJJJJIJJJJJJJJJJ
@SRR4044399.3 V00D00575:88:HH22CADXXa:1:1101:1407:1918 length=51
NCCGATGAGTCAGCAGTCGTGTATCTCTCACACTCCCATCACATCCCATCC
```

By default **`head`** will print out the first 10 lines of a file or the “head” of the file. What do you think **`tail`** does?

- **tail**

```
[brunettt@cubipmtest02 drosophila_fastq]$ tail SRR4044399_1.fastq
```

1. How you you print out the first 20 lines of a file?

```
head -n 20 SRR4044399_1.fastq
```

2. What about the last line of a file? HINT: **`man`**

```
tail -n 1 SRR4044399_1.fastq
```

Additionally, for large files counting and searching is not always most optimal in text editors

- **w**ord **c**ount

```
[brunettt@cubipmtest02 drosophila_fastq]$ wc -l SRR4044399_1.fastq
230409868 SRR4044399_1.fastq
```

- The '-l' argument tells word count to count the number of lines. In this case we have 230,409,868 lines in our file!
- **wc** can also count just count words, bytes, characters, and the length of the longest line by adding **-w**, **-c**, **-m**, or **-L** argument, respectively

How many bytes are in SRR4044399_2.fastq?

14,435,163,464 bytes OR ~14,345 MB OR ~14.4GB

It is also useful to be able to search for strings within a file without opening it

- Navigate into the drosophila_ref_files directory. These are reference files for the model organism *Drosophila melanogaster*. It contains the full sequence of the entire fruit fly genome!!!
- Use head to see what the following file contains:
  GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna

```
[brunettt@cubipmtest02 drosophila_ref_files]]$ head GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna
>NC_004354.4 Drosophila melanogaster chromosome X
GAATTCGTCAGAAATGAgctaaacaaatttaaatcattaaatgcGAGCGGCGAATCCGGAAACAGCAACTTCAAACCAGT
CACTCTGGCTGAACTAAATGGCCTGATAAACTCACTGGAATTAAAGAAAGCCCCAGGAACTGACAATCTTAACAACAAGA
CCATAATAAACTTACCTACAAAGGCCAgaatatatttaatacttATTTATAACAACATCCTGAGAACTGGACATTTCCCG
AACAAATGGAAGCACGCTAGCATCTCAATGATTCCCAAACCAGGGAAATCACCATTTGCTCTAAATTCATACCGCCCAAT
CAGCTTACTCTCTGGTCTTTCCAAACTACTCGAAAGAATACTACTGAAACGACTGTATGACATTGACTCTTTTGCCAAAG
CAATCCCTTCCCATCAATTTGGTTTCAGAAAGGATCATGGAGCGGAACATCAGCTGGCCAGGGTGACCCAATTTATTCTA
AAAGCTTTTGATGAAAAAGATGTCTGTTCTGCCACATTCCTTGACATTACGGAAGCCTTTGACCGAGTATGGCACGACGG
CTTGCTATATAAACTATCCAGACTCATCCCCAGATACCTATTCGACCTACTTGAAAACTATTTATCTAATAGAACCTTCT
CAGTAAGGATCGACGGTGAAACAACGTCTAGGATAGGTAATATTAGAGCAGGAGTGCCCCAGGGCAGCATACTGGGACCG
```

1

2

1 Each “>” means it is the name of a new chromosome or chromosomal segment. Here you can see it is the X chromsome

2 This is the exact DNA sequence for the entire NC_004354.4 chromosome X in *Drosophila melanogaster* until you reach the next “>” indicating a new chromosome.

Don’t you wish there was a way to count all the “>” so that we knew how many different chromosomal segments are represented in this 1,799,366 line file?

Searching for patterns in files- Just **grep** it!

- **g**lobal **r**egular **e**xpression **p**rint

```
[brunettt@cubipmtest02 drosophila_ref_files]$ grep ">" GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna
```

1 How can I tell **grep** to print the line numbers associated with the pattern? 2 Is there an easy way to tell **grep** to print everything **except** lines that contain ">"?

1 -n

2 -v

Great! But it is still hard to count...but guess what it doesn't have to be with redirection and piping

Redirect any output into a file

- Unix/Linux has this amazing character: **>**
  - following a command call will redirect any output to a new file that does not exist
  - If the file exists in the directory, it **WILL OVERWRITE** the file!!!
- You can also use: **>>**
  - Following a command call this will **append** the output to the end of an existing file or create a new file if one does not already exist.

```
grep ">" GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna > yourName_total_chromosomes.txt
```

Now you will notice there is no output printed to the screen because it is now saved in the a new file: **Tonya_total_chromosome.txt**

To confirm this, use **nanos**, **less**, **head**, or **tail**, prove to yourself that the **grep** command worked. Tell me how many different chromosome segments are in the *Drosophila melanogaster* genome.

1870 chromosomal segments using **wc –l Tonya_total_chromosome.txt**

What if I don’t need to save the intermediate file output?

- The other very useful Unix/Linux character is the pipe: |
  - This is not a capital letter I, rather it is the bar character on your keyboard.
  - This lets you link multiple Linux commands together without creating intermediate files
- Let’s see how piping works from our previous example of counting the number of chromosomal segments:

```
grep “>” GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna | wc -l
```

Translation: “Take the output of the **`grep`** function and use the output as input into the **`wc -l`** function”

Remember modifying the README.txt file?

- We had you had the following to your README.txt file:

```
directory:  yourname_ATAC_seq_output

        -This will contain output files from an ATAC-seq bioinformatics project
```

- Let's go ahead and make this directory now. Navigate to /gpfs/share/training

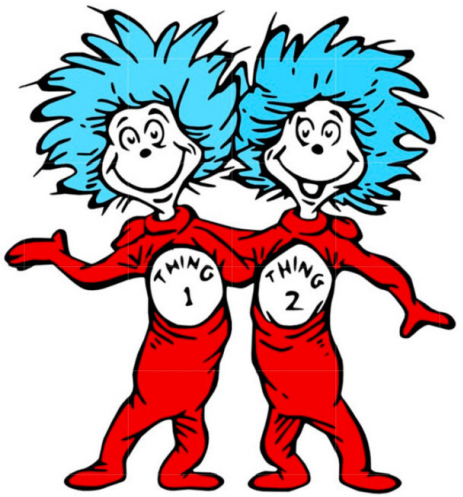
- **make directory**

```
[brunettt@cubipmtest02 training]$ mkdir Tonya_ATAC_seq_output
```

- **`mkdir`** following by a string creates a new folder or directory in your current path
- Create the same directory listed above but with your own name; after you complete this change into this directory.

Do you remember how to check your permissions on this newly created directory?

Let's set this directory up for the next segment of the tutorial



- First we need to copy some files into this new directory. Confirm you are in your newly created ATAC seq directory.

- **cop**y

  general format: **`cp <directory or file to be copied> <destination of copy>`**

```
[brunettt@cubipmtest02 Tonya_ATAC_seq_output]$ cp -R $HOME/scripts .
```

`-R` stands for recursive, and it means that if you are copying a directory, to also copy all the contents inside of the directory; note if you were just copying a single file, the **`-R`** is not necessary

This argument tells the **`cp`** command what to copy. In this case we are copying the **`scripts`** directory located in **`$HOME`**

This argument tells the **`cp`** command the location of where the copy should be made. The **`.`** is shorthand of "here" or present working directory instead of writing a full path out

Now let's build some software

- There are two bioinformatics softwares that we have not yet installed for you
  - **samtools** – allows you to manipulate and gather statistics effieciently on sequence alignment files
  - **bwa** – the Burrows-Wheeler Algorithm adapted to be a DNA aligner so we can take your raw fastq files and see how well and where those reads map to when compared to a reference genome
- Navigate to $HOME and list the contents in your $HOME directory. You will see a file called samtools-1.8.tar.bz2

Now let's build some software

- There are two bioinformatics softwares that we have not yet installed for you
  - **samtools** – allows you to manipulate and gather statistics effieciently on sequence alignment files
  - **bwa** – the Burrows-Wheeler Algorithm adapted to be a DNA aligner so we can take your raw fastq files and see how well and where those reads map to when compared to a reference genome
- Navigate to $HOME and list the contents in your $HOME directory. You will see a file called samtools-1.8.tar.bz2

Move samtools to the appropriate directory

- **mo**v**e**

  general format:

  `mv /path/to/yourfile.txt path/to/destination`

```
[brunettt@cubipmtest02 ~]$ mv samtools-1.8.tar.bz2 software/
```

When using **`mv`** if your destination location is a file (not a directory) or if the file name exists in the destination directory, **it will overwrite it!!!! Be very careful!!**

This will move the **`samtools-1.8.tar.bz2`** file to the software directory.

Wait! Samtools is not ready for use!

- Change directories into your software directory where you moved samtools
- Notice the **`.tar.bz2`** extension? This is because software that you download is generally compressed, you may see extensions such as **`.tar.gz`** or **`.tar.bz2`**
- **`.tar`** is short for tape archive which is a way to compress a directory in Unix/Linux systems (derived from backing up data to tapes). The **`.gz`** is short for gzip which can compress a file even more, and **`.bz2`** is short for BZIP2 and is another way to further compress a file.

Step 1: Decompress the file

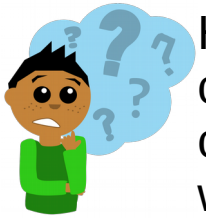
Decompressing tarballs

- **t**ape **ar**chiver

```
[brunettt@cubipmtest02 software]$ tar -jxvf samtools-1.8.tar.bz2
```

- It is important to notice the extension following the .tar extension. This is because we have to tell the tar programs the type of compression that was used in order to apply to appropriate decompression algorithm.

argument	meaning
j	Decompress a .bz2 file
x	Extract the contents of the archive;
v	Verbose output (so you can see the progress)
f	Use the file archive
z	Decompress a .gz file



How would this command change if that samtools was compressed in the following way:
samtools-1.8.tar.gz?

```
tar -zxvf samtools-1.8.tar.gz
```

Building samtools

- If you list the contents of your software directory, what do you notice?

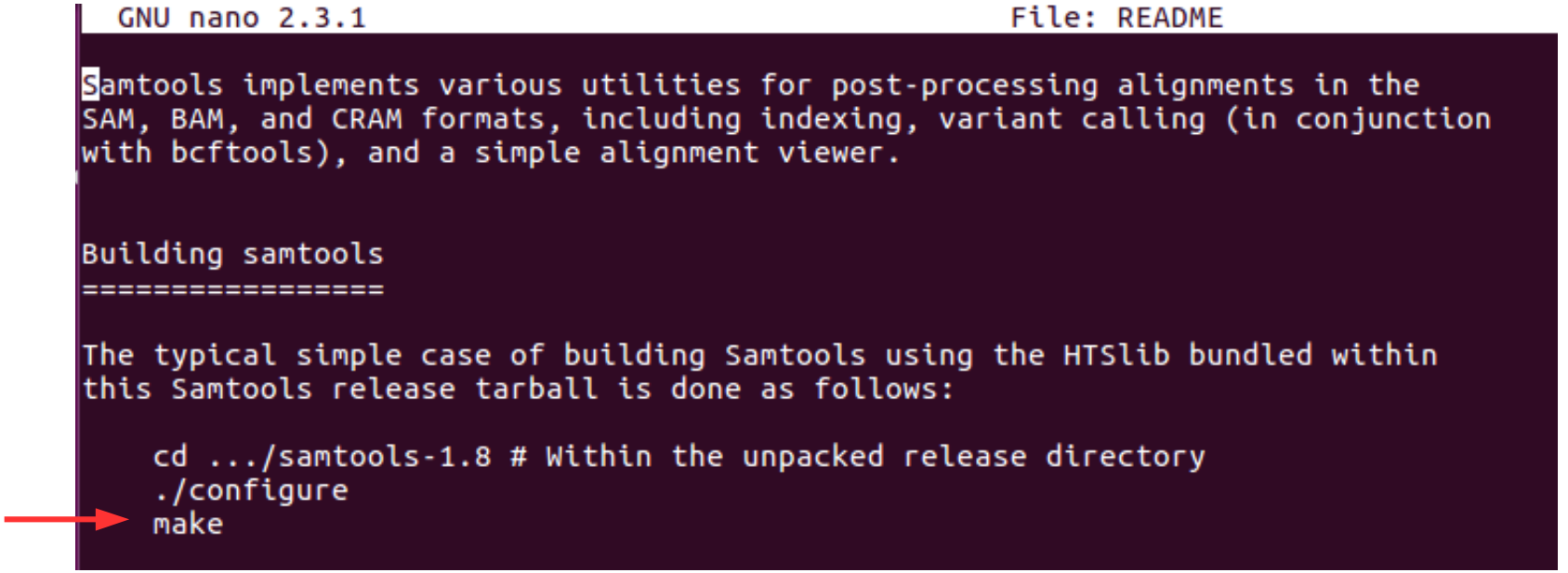
```
[brunettt@cubipmtest02 software]$ ls

bbmap                bedtools-2.27.0.tar.gz  FastQC
BBMap_38.06.tar.gz   fastqc_v0.11.7.zip      samtools-1.8
bedtools2            bwa-master.zip          samtools-1.8.tar.bz2
```

- Decompressing the file created a new directory called **`samtools-1.8`** that you can navigate into. Go ahead and navigate into this new directory. What is in it?

```
[brunettt@cubipmtest02 software]$ cd samtools-1.8
```

- There will be many files, including a README. Go ahead and open the README so we can determine if we need to perform any other actions before we can start using samtools.



Step 2: <u>ALWAYS</u> open the README of any software before assuming it is ready for use!

So we need to run **`./configure`** followed my the **`make`** command before **`samtools`** can be used.

Step 3: Execute the instructions for build and use. Reminder follow local installation instructions! If it says "sudo" you cannot use those instructions!

Exit what method you used to view the README and let's run the following:

```
[brunettt@cubipmtest02 software]$ ./configure
```

```
[brunettt@cubipmtest02 software]$ make
```

Samtools is installed!

Step 4: Check for errors and make sure installation is successful

- To double check that samtools has been successfully installed let's try and run it

```
[brunettt@cubipmtest02 samtools-1.8]$ ./samtools
```

```
Program: samtools (Tools for alignments in the SAM format)
Version: 1.8 (using htslib 1.8)

Usage:   samtools <command> [options]

Commands:
  -- Indexing
     dict           create a sequence dictionary file
     faidx          index/extract FASTA
     index          index alignment

  -- Editing
     calmd          recalculate MD/NM tags and '=' bases
     fixmate        fix mate information
     reheader       replace BAM header
     targetcut      cut fosmid regions (for fosmid pool only)
     addreplacerg   adds or replaces RG tags
     markdup        mark duplicates

  -- File operations
     collate        shuffle and group alignments by name
     cat            concatenate BAMs
     merge          merge sorted alignments
```

The `./` will execute the program. You could also have done:

`./homelink/youruserna me/sotware/samtools-1.8/samtools`

OR

$HOME/software/samtools -1.8/samtools

Summary of Compiling Software

Step 1: *Decompress the file*

Step 2: *ALWAYS open the README of any software before assuming it is ready for use*

Step 3: *Execute the instructions for build and use*

Step 4: *Check for errors and make sure installation is successful*

Now let's build some software

- There are two bioinformatics software packages that we have not yet installed for you
  - **samtools** – allows you to manipulate and gather statistics effieciently on sequence alignment files
  - **bwa** – the Burrows-Wheeler Algorithm adapted to be a DNA aligner so we can take your raw fastq files and see how well and where those reads map to when compared to a reference genome

But wait! We have not given you the compressed software for bwa yet. How do we get the software into Wilkins?...

Install BWA DNA aligner onto HPC

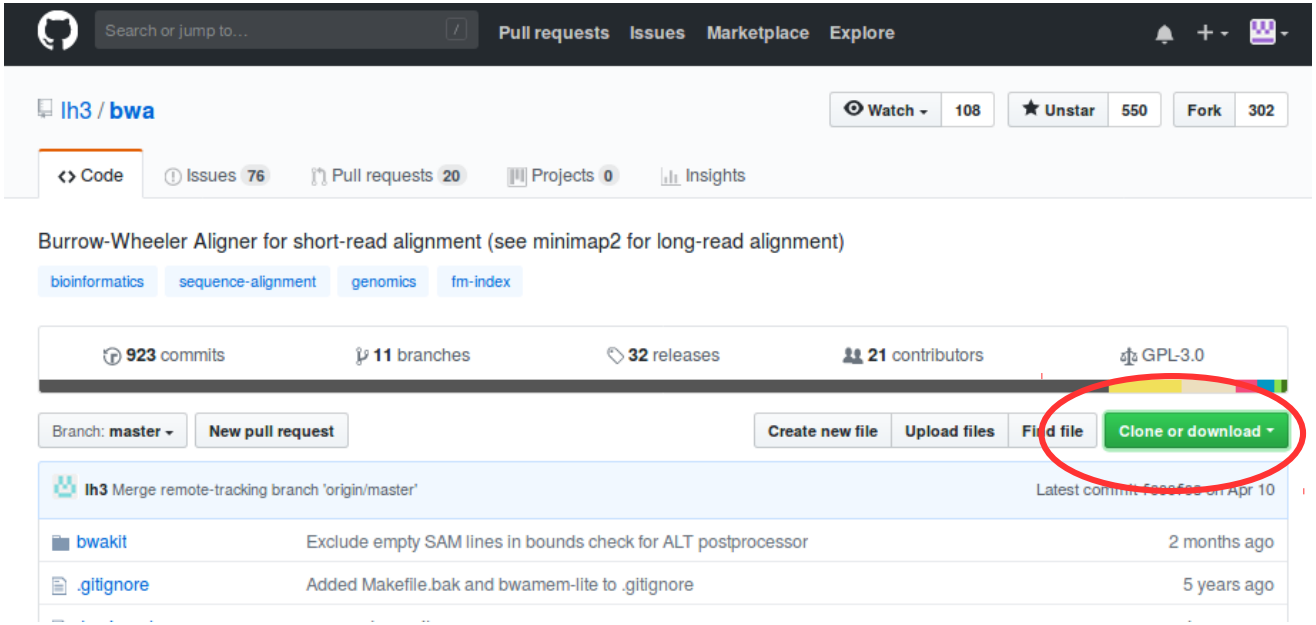
Step 1: Switch windows to your local machine. Then let's download the bwa software from github

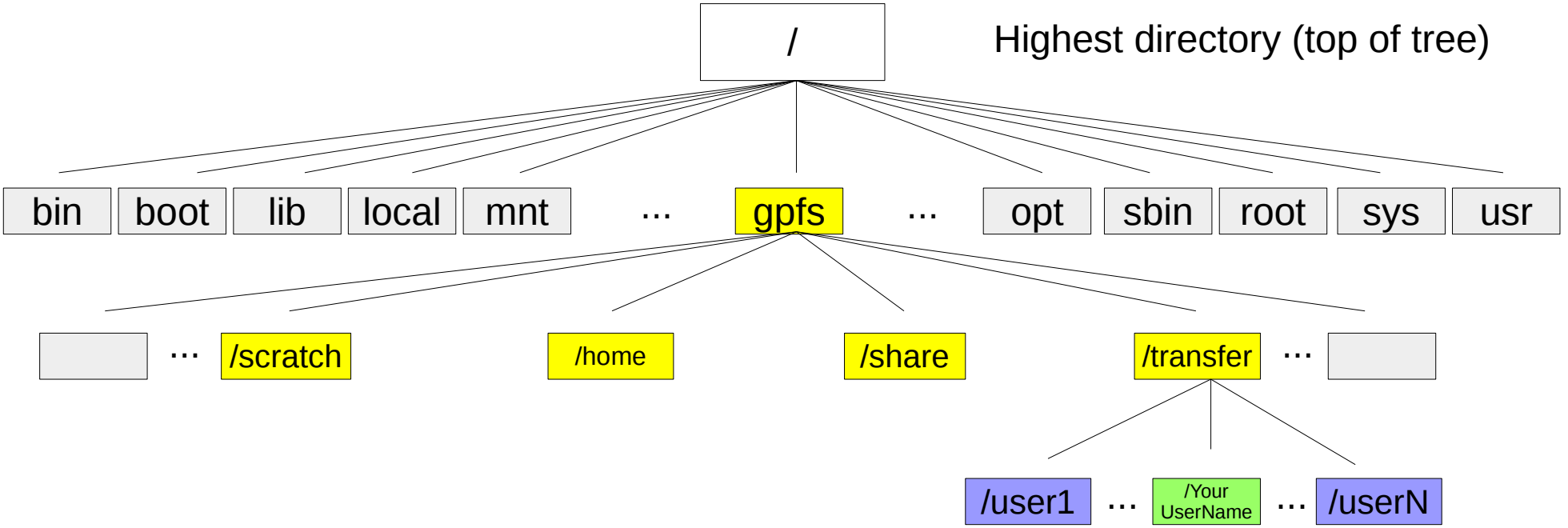
1. Go to https://github.com/lh3/bwa
2. Download the zip file
3. Go to the download location

   on your local computer and

   and open a terminal

So how do we get this into the Wilkins environment?





Rosalind and Wilkins have a separate location so that files can be uploaded to Rosalind/Wilkins and download from Rosalind/Wilkins using **SFTP**

What should be stored in /gpfs/transfer/your_username?

- Files that need to be transferred into/out of Rosalind onto a HIPAA-compliant device

= sudo required

= limited access

= full access*

= no access w/o permission

Install BWA DNA aligner onto HPC

Step 2: Open a Unix/Linux terminal on your local computer and put compressed file into Wilkins staging area

- **s**ecure **f**ile **t**ransfer **p**rotocol

```
[brunettt@local.host Downloads] sftp brunettt@cubipmtest02.ucdenver.pvt
```

```
sftp> cd /gpfs/transfer/brunettt
sftp> put bwa-master.zip
Uploading bwa-master.zip to /gpfs/transfer/brunettt/bwa-master.zip
bwa-master.zip                                100%  262KB 262.3KB/s   00:00
```

put is used to *copy a file into the transfer area from whatever location you are logged into*. Remember in this case, you logged in from your local host, so **put** will copy the file to from your local host into the staging area

You would go through the same process if you wanted to **get** something out of Rosalind except use **get** instead of **put**

```
sftp> quit
```

Install BWA DNA aligner onto HPC

Step 3: Switch windows back to the Wilkins commands line and change directories to where you would like to transfer your software and move file there

```
[brunettt@cubipmtest02 ~]$ cd $HOME/software
```

```
[brunettt@cubipmtest02 software]$ mv /gpfs/transfer/brunettt/bwa-master.zip .
```

Now we can follow the 4 basic steps for installing software...step 1 decompress, but wait, this ends in **`.zip`**...

Decompressing .zip files

- **unzip**

```
[brunettt@cubipmtest02 software]$ unzip bwa-master.zip
```

- Similar to tar files decompressing the file created a new directory called **`bwa-master`** that you can navigate into. Go ahead and navigate into this new directory. What is in it?
- What is the next step?

Step 2: Locate and open the README

```
[minimap2]: https://github.com/lh3/minimap2

## Getting started

        git clone https://github.com/lh3/bwa.git
        cd bwa; make
        ./bwa index ref.fa
        ./bwa mem ref.fa read-se.fq.gz | gzip -3 > aln-se.sam.gz
        ./bwa mem ref.fa read1.fq read2.fq | gzip -3 > aln-pe.sam.gz
```

We have already done these!

We still need to run this part

Install BWA DNA aligner onto HPC

Step 3: *Execute the instructions for build and use*

```
[brunettt@cubipmtest02 bwa-master]$ make
```

Step 4: *Check for errors and make sure installation is successful*

```
[brunettt@cubipmtest02 bwa-master]$ ./bwa
```

```
Program: bwa (alignment via Burrows-Wheeler transformation)
Version: 0.7.17-r1194-dirty
Contact: Heng Li <lh3@sanger.ac.uk>

Usage:   bwa <command> [options]

Command: index         index sequences in the FASTA format
         mem           BWA-MEM algorithm
         fastmap       identify super-maximal exact matches
         pemerge       merge overlapping paired ends (EXPERIMENTAL)
         aln           gapped/ungapped alignment
         samse         generate alignment (single ended)
         sampe         generate alignment (paired ended)
         bwasw         BWA-SW for long queries

         shm           manage indices in shared memory
         fa2pac        convert FASTA to PAC format
```

Clean up and housekeeping

- Let's go ahead and remove some files that we don't need anymore
- Since we already installed samtools-1.8.tar.bz2 and bwa-master.zip let's delete these archives
- **rem**ove

```
[brunettt@cubipmtest02 software]$ rm samtools-1.8.tar.bz2
```

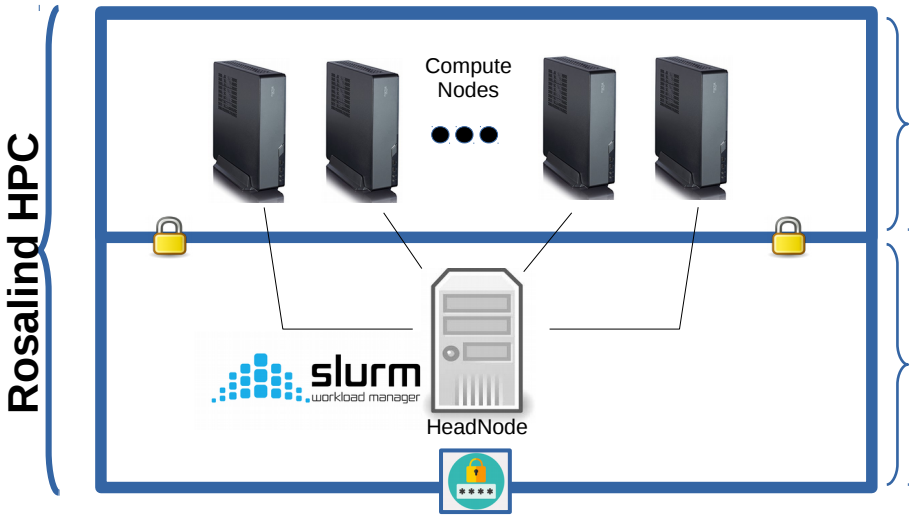
```
[brunettt@cubipmtest02 software]$ rm bwa-master.zip
```

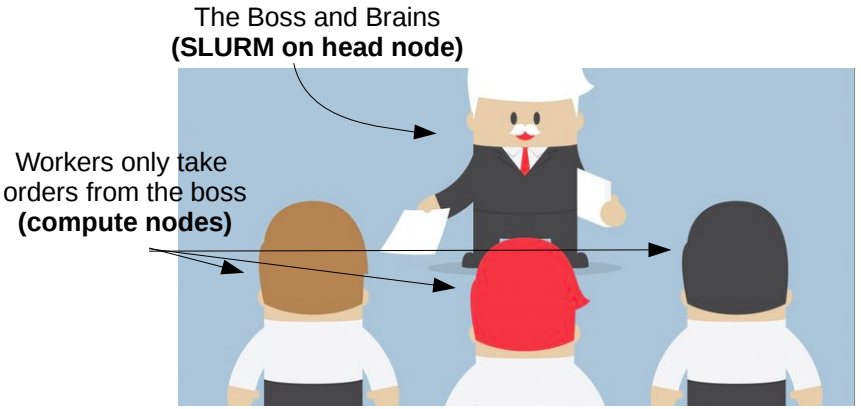
When using **`rm`** use it wisely! Once deleted it cannot be retreived!!! No recycle bin or confirmation!!! Also, if you need to remove an entire directory the **`-r`** (mean recursive) argument is required, and again, it will permanently delete the whole directory!

Scheduling and Submitting a Job...or Jobs!

Rosalind Utilizes SLURM

- **S**imple **L**inux **U**tility for **R**esource **M**anagement
- SLURM Workload Manager is just one of many flavors of job scheduling softwares available and happens to be the one that we chose for Rosalind
- SLURM lives on the head node
- In order to use any of the the compute nodes, you must first communicate your request with SLURM on the head node.
- SLURM will process your request and assign your request to an available compute node.
- You never have direct access to the compute nodes!





HPCs use job schedulers

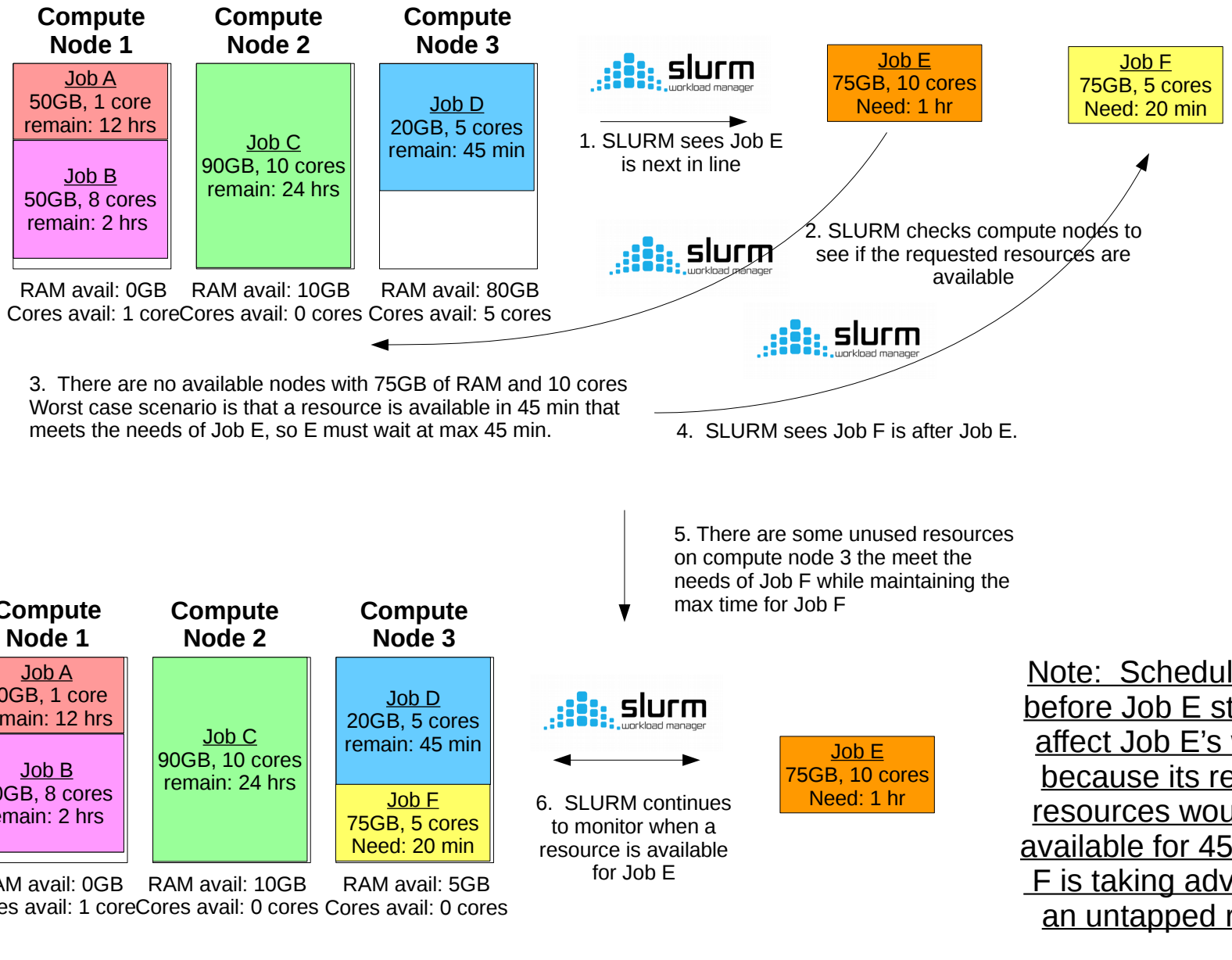
- What is a job scheduler?
  - responsible for executing multiple job requests (i.e. your programs) whether they come from the same or different user(s) and determining what resources are available, and whose jobs are next in line to be run
- Why do we need a job scheduler?
  - "Fair-Share"
    - Sophisticated algorithm to schedule jobs fairly on a shared resource
    - The scheduler knows when each user made submitted requests so it can plan jobs effectively and know how long a job has been waiting in the queue or line
    - In the event all resources are being utilized, the "fair-share" policy is implemented meaning those users who have used the least amount of computer hours/resources within a window of time, get pushed to the front of the queue.
      - Prevents one individual from constantly using all the resources as they become available

HPCs use job schedulers

- Efficiency
  - Jobs that are shorter or require less resources typically are in the queue shorter since resources become available more quickly and are more likely to fit into an untapped resource
  - Scheduler knows how many resources every user has requested and the time and memory constraints of those resources.

Let’s take a look at a very simplistic example of what we mean by efficiency

Ex: Let's assume there is an HPC with 3 compute nodes each with 10 cores. Each compute node as 100GB of RAM and have a time limit of 24 hours



Note: Scheduling Job F before Job E still doesn't affect Job E's wait time because its requested resources would not be available for 45 min! Job F is taking advantage of an untapped resource

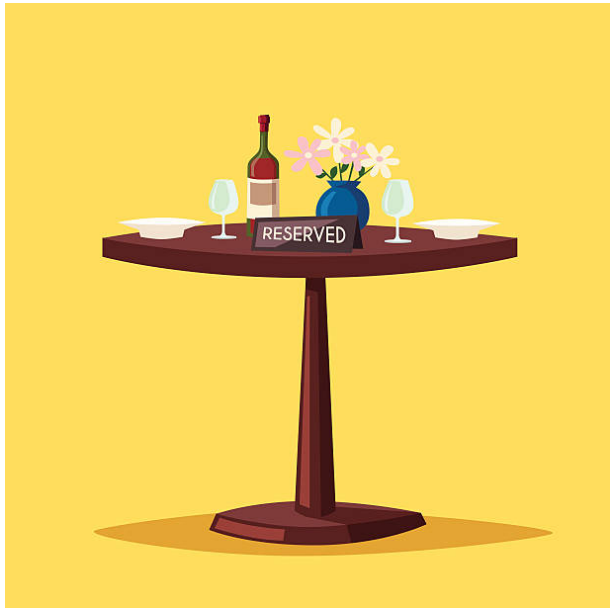
How do you communicate with SLURM?

- SLURM has its own special language that are specific to SLURM.
- Almost all SLURM commands will begin with a lowercase ‘s’
- **sbatch** tells SLURM that you are about to submit an order using a batch script to use the compute nodes to run one of your jobs.


```
[brunettt@demo.system01 ~]$ sbatch myJob.sh
```

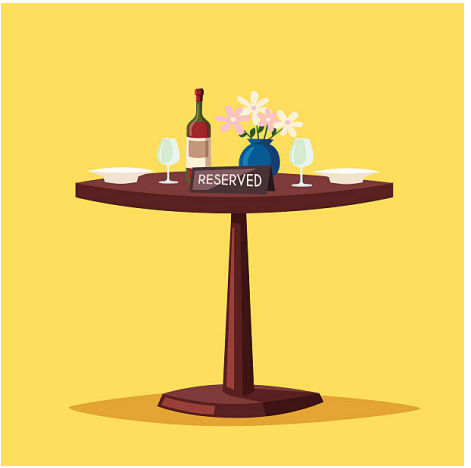
What is a BATCH script?

SLURM requires a specially formatted shell script so it knows how to allocate resources, how to utilize those resources, and how to run your program



1. Request and Allocate Resources

- We have created a short example batch script in your $HOME/scripts called myJob.sh. Open this script.
- All allocation requests are made at the top of your file after the line: #!/bin/bash
- All reservation information all must start with #SBATCH



"Make a reservation at our popular restaurant!"

```
#!/bin/bash

#SBATCH --time=5
#SBATCH --mem=1000
#SBATCH --job-name=testing
#SBATCH --output=testing.out
#SBATCH --error=testing.err
#SBATCH --node=1
#SBATCH --ntasks=1
#SBATCH -p defq
```

Max time in minutes, also take days-hours:minutes format

Max memory (RAM) in MB

Name of job

Name of file to write standard output

Name of file to write standard error

Number of Nodes needed

Number of tasks per node

The node type or "queue" to run job

- What happens if I go over the time and/or memory reserved?

  Your job will be killed! SLURM cannot “add on” or extend additional resources!!!

- Can’t I just request the max time and memory all the time?

  You can, but remember, “fair-share” policy and efficiency will potentially prevent your jobs from running in a timely manner due to lack of available requested resources. Your job will only start running after those resources are available and you are next in line to receive those resources

- How do I know what node or “queue” I should select and what are my options?


```
[brunettt@cubipmtest02 ~]$ sinfo
```

Available Nodes on Rosalind

subsets of the nodes available in defq

Node Name	Time Limit	# nodes available	Total RAM per node	CPUs per node	Cores per CPU
defq	36 hours	32	128GB	2	12
longrun	infinite	6	128GB	2	12
sas	7 days	2	128GB	2	12
matlab	7 days	4	128GB	2	12
bigmem	21 days	2	1526GB	2	18

What queues are available on the Wilkins demo environment? What is the maximum time limit and how many nodes are available?

Only defq is available and it has a 5 hour time limit and only 1 node

2. Give the script instructions and submit the request

- This is where you will place your order, i.e. tell SLURM the exact order of the exact commands you want to run on the compute nodes

```
#!/bin/bash

#SBATCH --time=3
#SBATCH --mem=1000
#SBATCH --job-name=testing
#SBATCH --output=testing.out
#SBATCH --error=testing.err
#SBATCH --node=1
#SBATCH --ntasks=1
#SBATCH -p defq

pwd
echo "Hello World!"
sleep 120
```

Write down your "order" for the waiter

```
[brunettt@demo.system01 ~]$ sbatch myJob.sh
Submitted batch job 233830
```

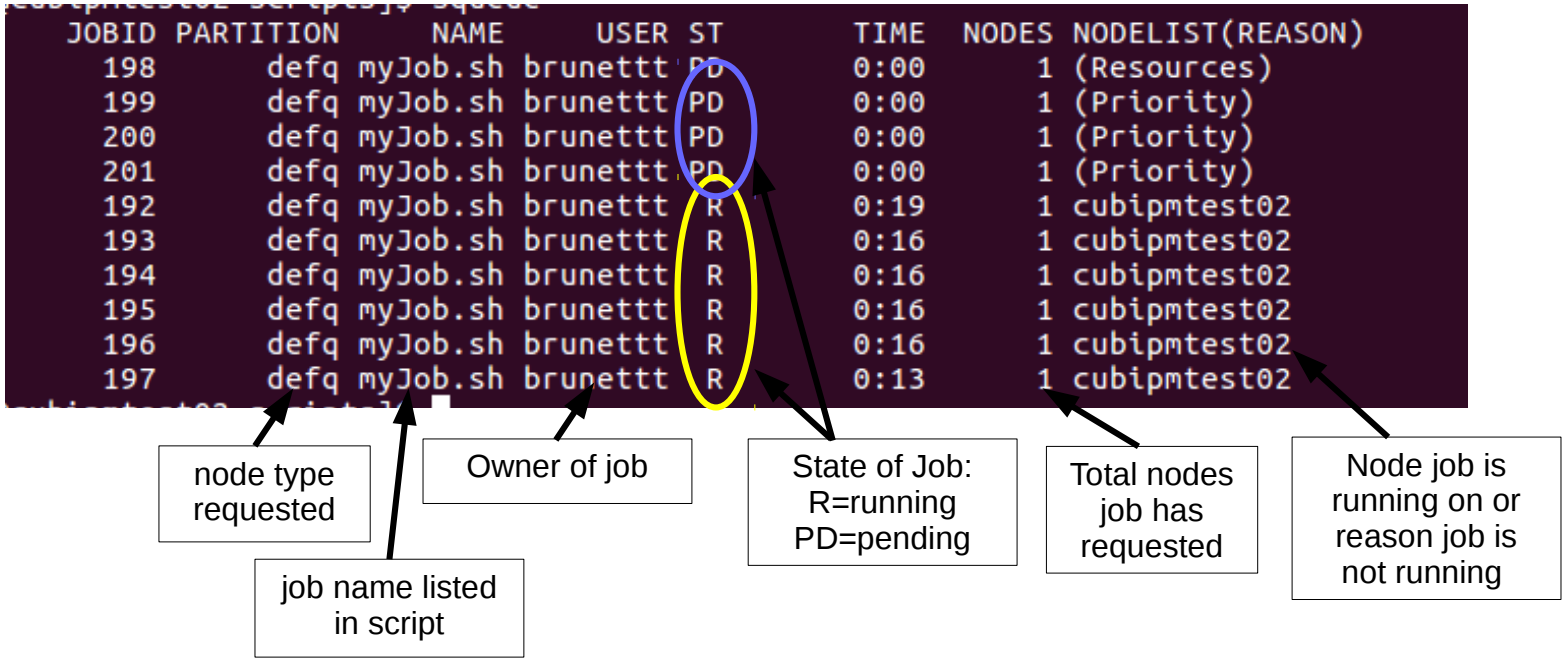
Waiter submits your "order" in the form of a shell script to the boss who will assign it to a chef, in this case are the compute nodes

Checking your place in line

```
[brunettt@cubipmtest02 ~]$ squeue
```

```
[brunettt@cubipmtest02 ~]$ squeue -u brunettt
```

```
[brunettt@cubipmtest02 ~]$ squeue --job 233839
```



Getting information about a currently running job

```
[brunettt@cubipmtest02 ~]$ sstat --job 233839.batch
```

```
[brunettt@cubipmtest02 scripts]$ sstat --job 204.batch
       JobID  MaxVMSize  MaxVMSizeNode  MaxVMSizeTask  AveVMSize     MaxRSS MaxRSSNode MaxRSSTask     AveRSS MaxPages MaxPagesNode   MaxPagesTask   AvePages     MinCPU MinCPUNode MinCPUTas
Tasks AveCPUFreq ReqCPUFreqMin ReqCPUFreqMax ReqCPUFreqGov ConsumedEnergy  MaxDiskRead MaxDiskReadNode MaxDiskReadTask  AveDiskRead MaxDiskWrite MaxDiskWriteNode MaxDiskWriteTask AveDiskWr
------------ ---------- -------------- -------------- ---------- ---------- ---------- ---------- ---------- -------- ------------ -------------- ---------- ---------- ---------- ---------
------ ---------- ------------ ------------ ------------ ------------ ------------ ------------ ---------- ---------- ------------ ------------ ---------- ---------- ---------- ---------
204.batch            0   cubipmtest02              0            0          0 cubipmtes+          0          0        0 cubipmtest02              0          0            cubipmtes+
    0          0       Unknown       Unknown       Unknown              0            0    cubipmtest02               0            0            0     cubipmtest02                 0
[brunettt@cubipmtest02 scripts]$
```

This is a lot of info and it can be hard to read, so you can parse it by the statistics you are interested in. For example:

```
[brunettt@cubipmtest02 ~]$ sstat --job 233839.batch -o MaxRSS,NTasks
```

```
[brunettt@cubipmtest02 scripts]$ sbatch myJob.sh
Submitted batch job 205
[brunettt@cubipmtest02 scripts]$ sstat --job 205.batch -o MaxRSS,NTasks
    MaxRSS   NTasks
---------- --------
         0        0
[brunettt@cubipmtest02 scripts]$
```

Create a batch script

- Navigate into the scripts directory in $HOME

- Using a text editor create and open a file called: <yourUserName>_testSlurm.sh

```
[brunettt@cubipmtest02 scripts]$ nano test_brunettt.sh
```

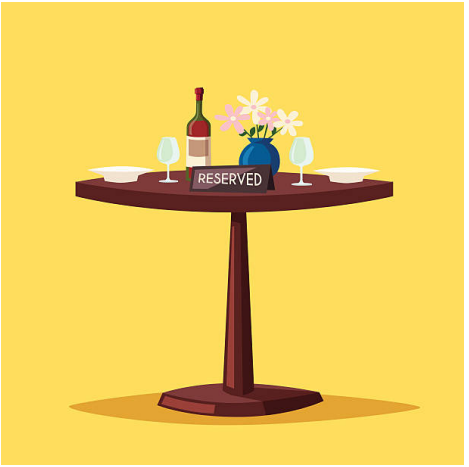
- Let's add the following line to the very top of our file:

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

- The "sha-bang" (#! --hybrid of sharp bang) followed by /bin/bash (no spaces!) tells Linux to interpret the code in this file as a bash script. This is required at the top of every batch file for SLURM!!!

1. Make your reservation

- Make your reservation for the following:
  - time of 5 minutes
  - Memory max of 1000 MB
  - Job name of your name followed by the string “_testing”, NO SPACES!
  - Output/log file of your name followed by the string “_testing.log” NO SPACES!
  - Error file of your name followed by the string “_testing.err” NO SPACES!
  - Request 1 node
  - Request 1 task
  - Tell SLURM to make a reservation on the defq node




```
#!/bin/bash

#SBATCH --time=5
#SBATCH --mem=1000
#SBATCH --job-name=Tonya_testing
#SBATCH --output=Tonya_testing.out
#SBATCH --error=Tonya_testing.err
#SBATCH --node=1
#SBATCH --ntasks=1
#SBATCH -p defq
```

2. Give the script instructions and submit the request

In your batch script, write out the following sequence (one per line) using the Linux commands we just learned.

- Change into your ATACseq project directory
- print current path address
- Wait 60 seconds before running the next command
- Change to your home directory
- Get current path address
- List all the directories and files in home and their relative sizes
- Change directories into drosophila_fasta in your shared training directory
- Print out the last 20 lines of the file **`SRR4044399_2.fastq`** to standard out
- Print your present working directory
- Wait 30 seconds before finishing

```
#!/bin/bash

#SBATCH --time=5
#SBATCH --mem=1000
#SBATCH --job-name=testing
#SBATCH --output=testing.out
#SBATCH --error=testing.err
#SBATCH --node=1
#SBATCH --ntasks=1
#SBATCH -p defq

cd /gpfs/share/training/Tonya_ATACseq
pwd
sleep 60
cd $HOME
pwd
ls -lh
cd /gpfs/share/training/drosophila_fastq
tail -n 20 SRR4044399_2.fastq
pwd
sleep 30
```

2. Give the script instructions and submit the request

- Now we can save and exit our script and it is ready to be submitted!


```
[brunettt@cubipmtest02 ~]$ sbatch ./test_brunettt.sh
Submitted batch job 233839
```

Great! SLURM confirms it has received your job request by assigning it a **job ID**. The SLURM head node will now communicate with the compute nodes to check the availability of your requested resources on the compute nodes.

Check if your job is running. If it is, print out the statistics for NTasks, MaxRSS, AveRSS

Check on the status of all jobs run in the day

```
[brunettt@cubipmtest02 ~]$ sacct
```

```
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
188          convert_s+       defq training-+          3  COMPLETED      0:0
188.batch         batch            training-+          3  COMPLETED      0:0
189          convert_s+       defq training-+          3  COMPLETED      0:0
189.batch         batch            training-+          3  COMPLETED      0:0
190          convert_s+       defq training-+          3  COMPLETED      0:0
190.batch         batch            training-+          3  COMPLETED      0:0
191            myJob.sh       defq training-+          1  COMPLETED      0:0
191.batch         batch            training-+          1  COMPLETED      0:0
192            myJob.sh       defq training-+          1  COMPLETED      0:0
192.batch         batch            training-+          1  COMPLETED      0:0
193            myJob.sh       defq training-+          1  COMPLETED      0:0
193.batch         batch            training-+          1  COMPLETED      0:0
194            myJob.sh       defq training-+          1  COMPLETED      0:0
194.batch         batch            training-+          1  COMPLETED      0:0
195            myJob.sh       defq training-+          1  COMPLETED      0:0
195.batch         batch            training-+          1  COMPLETED      0:0
196            myJob.sh       defq training-+          1  COMPLETED      0:0
196.batch         batch            training-+          1  COMPLETED      0:0
197            myJob.sh       defq training-+          1  COMPLETED      0:0
197.batch         batch            training-+          1  COMPLETED      0:0
198            myJob.sh       defq training-+          1  COMPLETED      0:0
198.batch         batch            training-+          1  COMPLETED      0:0
199            myJob.sh       defq training-+          1  COMPLETED      0:0
199.batch         batch            training-+          1  COMPLETED      0:0
200            myJob.sh       defq training-+          1  COMPLETED      0:0
200.batch         batch            training-+          1  COMPLETED      0:0
201            myJob.sh       defq training-+          1  COMPLETED      0:0
201.batch         batch            training-+          1  COMPLETED      0:0
202            myJob.sh       defq training-+          1  COMPLETED      0:0
202.batch         batch            training-+          1  COMPLETED      0:0
203            myJob.sh       defq training-+          1  COMPLETED      0:0
203.batch         batch            training-+          1  COMPLETED      0:0
204            myJob.sh       defq training-+          1  COMPLETED      0:0
204.batch         batch            training-+          1  COMPLETED      0:0
205            myJob.sh       defq training-+          1  COMPLETED      0:0
205.batch         batch            training-+          1  COMPLETED      0:0
206            myJob.sh       defq training-+          1    RUNNING      0:0
207            myJob.sh       defq training-+          1    RUNNING      0:0
208            myJob.sh       defq training-+          1    RUNNING      0:0
209            myJob.sh       defq training-+          1    RUNNING      0:0
210            myJob.sh       defq training-+          1 CANCELLED+      0:0
210.batch         batch            training-+          1  CANCELLED     0:15
211            myJob.sh       defq training-+          1    RUNNING      0:0
[brunettt@cubipmtest02 scripts]$
```

Oops I made a mistake, how do I cancel a job>


```
[brunettt@cubipmtest02 ~]$ scancel 210
```

Job id to cancel

Want to cancel everything you are running?


```
[brunettt@cubipmtest02 ~]$ scancel -u brunettt
```

your username

You can only cancel jobs that you submitted!

3. Collect output and pay the bill

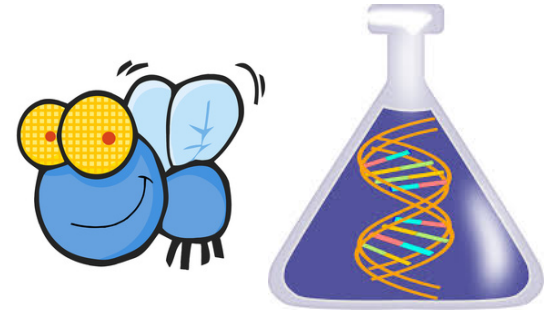
- Navigate to where your test_brunettt.sh was run
- You should 2 outputs: testing_brunettt.log and testing_brunettt.err. Open this. What do you see?

- SLURM calculates how much storage and the number of core-hours you have used during the month and will bill your speedtype at the beginning of each month

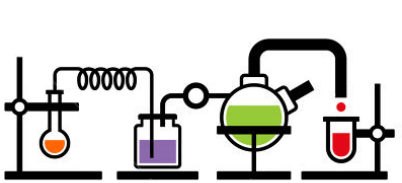
Ready to feel empowered?

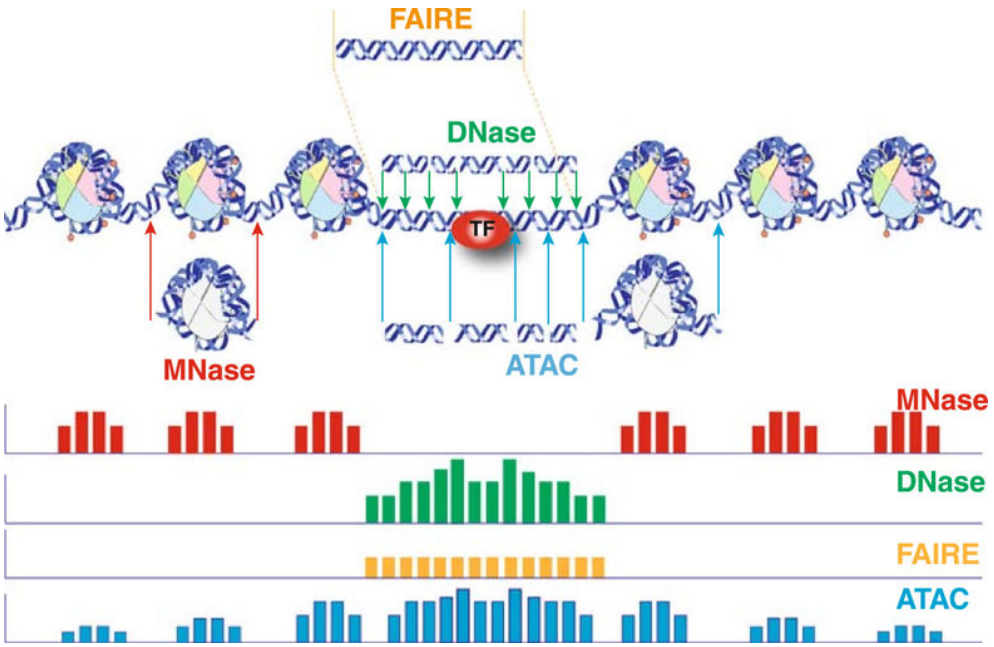
Using Your New Skills:
A bioinformatics example

One Page Overview of ATAC-seq (Diagram)



Lots of important and time-consuming wet lab magic





From Tsompana and Buck, 2014 In *Epigenetics & Chromatin*

-**A**ssay for **T**ransposase-**A**ccessible **C**hromatin using **seq**uencing

-epigentic method for surveying where chromatin may be open and for profiling nucleosomes

-Help in motif enrichment analysis for TF binding and correlation with gene expression analysis

DISCLAIMER: Although we are brushing over the wet-lab (for sake of time), it is very critical to work closely in designing these types of experiments with both wet lab and dry lab personnel and that goes for any NGS or microarray technology. Also, this is a very simplified analysis. In a real-life publication setting more thorough analytic QC would need to be performed. This example is only being used to teach Linux/HPC!!!!

Journal List › Genome Biol › v.17; 2016 › PMC5037609

BioMed Central The Open Access Publisher | Genome Biology

this article | search | submit a manuscript | register

Genome Biol. 2016; 17: 196.
Published online 2016 Sep 27. doi: 10.1186/s13059-016-1057-2

PMCID: PMC5037609
PMID: 27678375

Genome-wide identification of *Drosophila* dorso-ventral enhancers by differential histone acetylation analysis

Nina Koenecke,[#1] Jeff Johnston,[#1] Bjoern Gaertner,[1,2] Malini Natarajan,[1] and Julia Zeitlinger[1,3]

Author information ► Article notes ► Copyright and License information ► Disclaimer

This article has been cited by other articles in PMC.

Abstract

Go to:

Background

Go to:

Drosophila dorso-ventral (DV) patterning is one of the best-understood regulatory networks to date, and illustrates the fundamental role of enhancers in controlling patterning, cell fate specification, and morphogenesis during development. Histone acetylation such as H3K27ac is an excellent marker for active

Genome Biol

https://www.ncbi.nlm.nih.gov/geo/

NCBI Resources How To

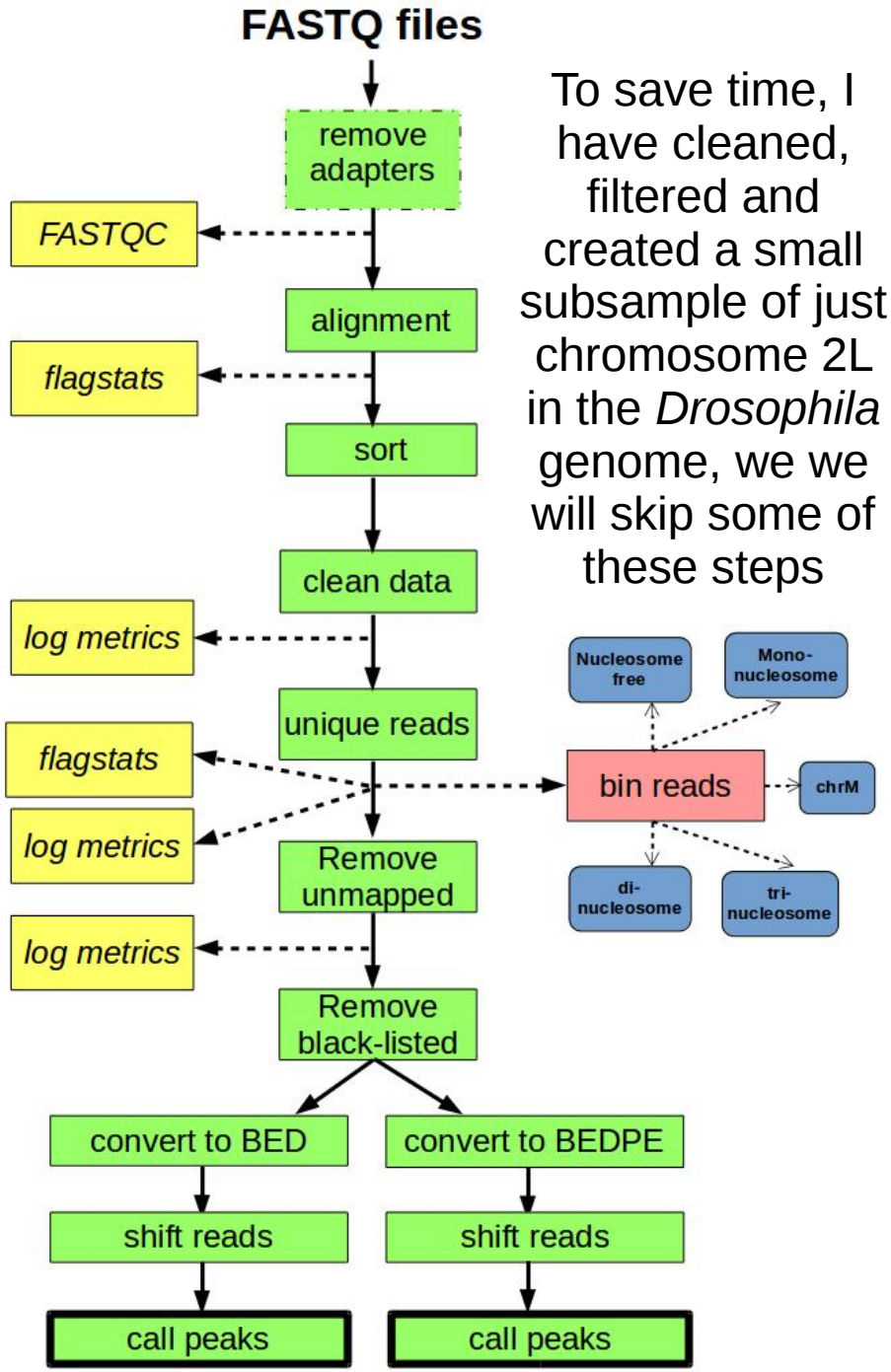
GEO Home | Documentation | Query & Browse | Email GEO

Gene Expression Omnibus

GEO is a public functional genomics data repository supporting MIAME-compliant data submissions. Array- and sequence-based data are accepted. Tools are provided to help users query and download experiments and curated gene expression profiles.

Keyword or

Getting Started	Tools	Browse Content
Overview	Search for Studies at GEO DataSets	Repository Browser
FAQ	Search for Gene Expression at GEO Profiles	DataSets:
About GEO DataSets	Search GEO Documentation	Series:
About GEO Profiles	Analyze a Study with GEO2R	Platforms:



To save time, I have cleaned, filtered and created a small subsample of just chromosome 2L in the *Drosophila* genome, we we will skip some of these steps

One of the biggest pitfalls with bioinformatics software:

BE SURE TO READ THE INSTALLATION INSTRUCTIONS AND MANUAL!

It is very important to truly understand how what arguments and parameters are available in an algorithm and what it means to run the default value. Often times you will have to adjust the algorithm use to fit your experimental design and needs! **SUPER IMPORTANT**

Check sequencing quality

Software: *fastqc*

Location: $HOME/software

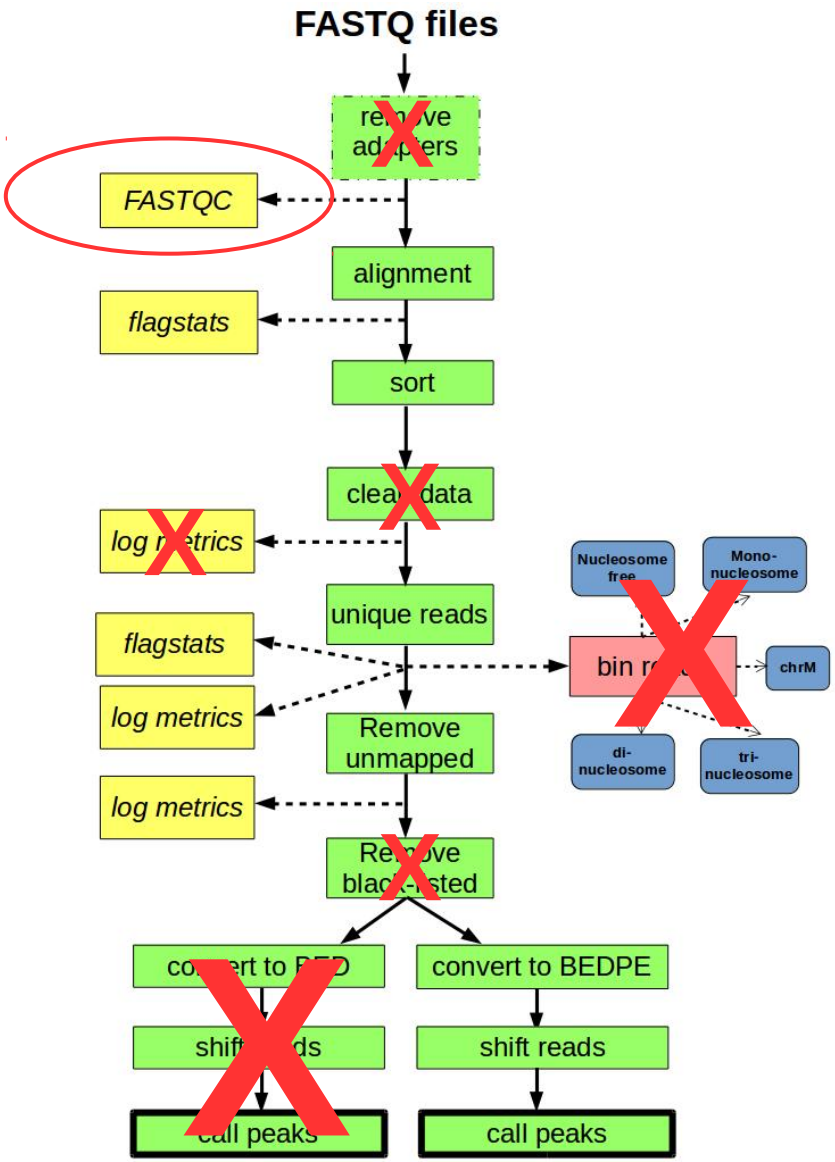
GOAL: check raw sequencing quality

1. Go to $HOME/scripts and open check_seq_qual.sh

2. Modify this script to run the fastqc software and then submit it using SLURM

3. Check to see when you job finishes and check your error logs and log files to make sure the run was successful

What did it output?



Viewing Results

- Rosalind and Wilkins do not have graphics cards, therefore, to view the results we should need to transfer them to our local computer.

  Hint: /gpfs/transfer/your_username

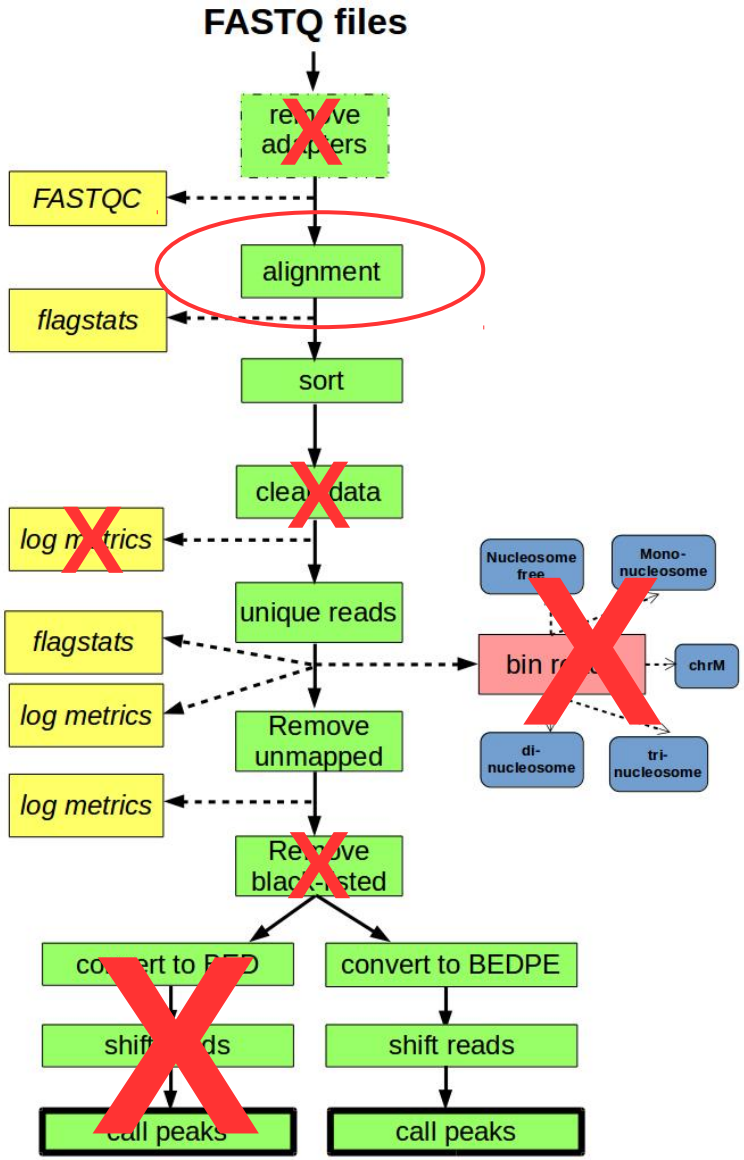
Align reads to reference genome

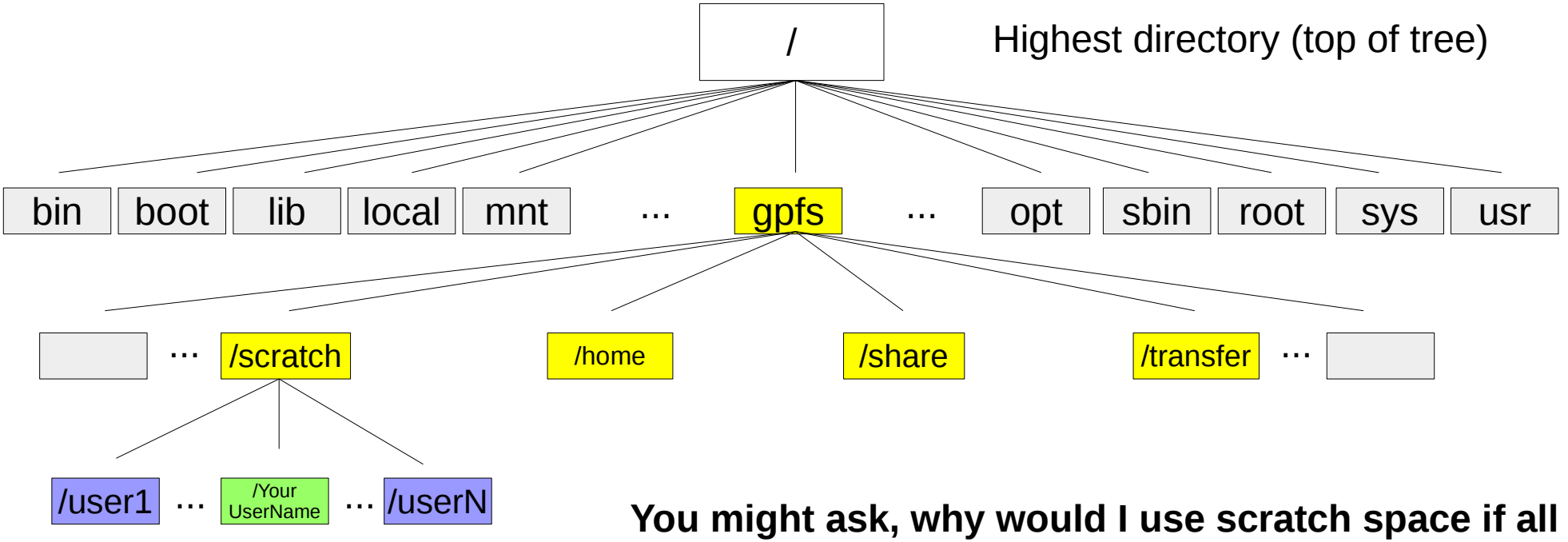
Software: *bwa*

Location: $HOME/software

Goal: take your raw reads in the fastq and map them to the reference genome

The alignment step creates large intermediate files (.sam and .sai files) that we will eventually delete...**however, all the files we have in our project share are charged 0.02GB/month...can we make this more efficient?**





What should be stored in /gpfs/scratch/your_username?

- Intermediate files generated from programs that you don't necessarily need

LOTS OF SPACE!

You might ask, why would I use scratch space if all my output can go directly into my project shared folder?

$$ COST $$

- Jobs write output faster to scratch, so time savings and money savings
- If you are backing up data, scratch is not backed up so you are not paying for monthly intermediate file storage

```
[brunettt@cubipmtest02 ~]$ $SCRATCH
```

= sudo required

= limited access

= full access*

= no access w/o permission

Align reads to reference genome

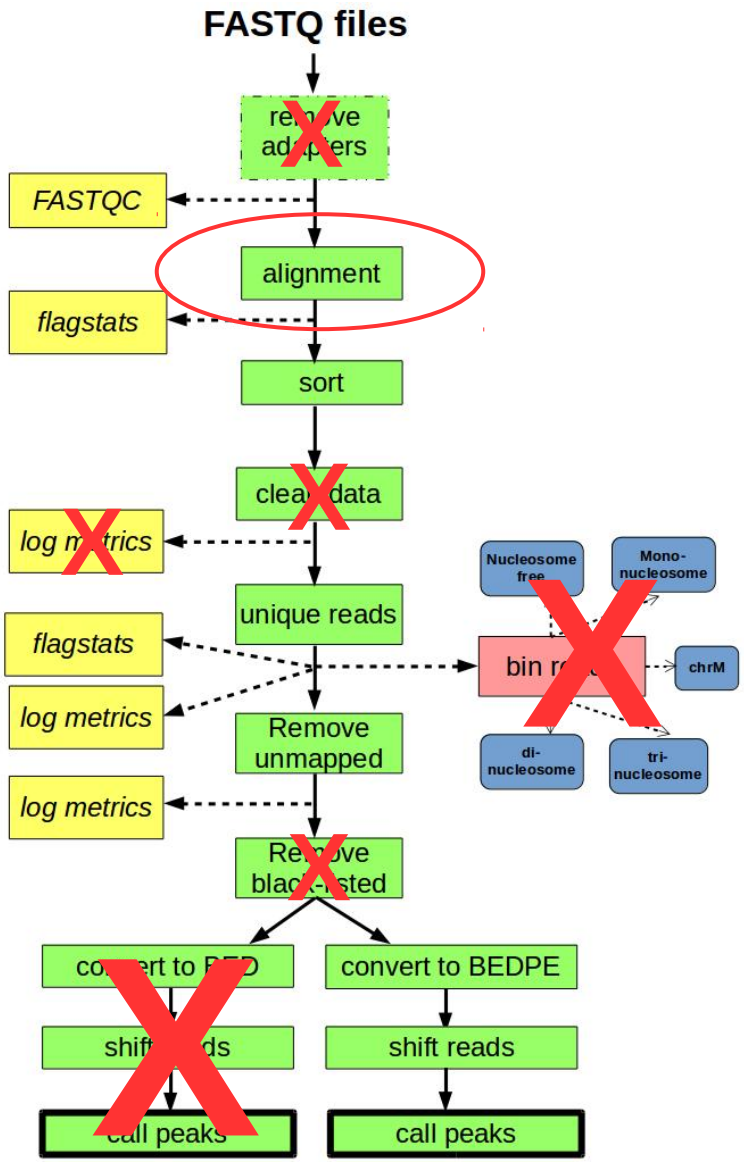
Software: *bwa*

Location: $HOME/software

Goal: take your raw reads in the fastq and map them to the reference genome

1. Go to $HOME/scripts and open alignment.sh

Let's take a look at some new things we have implemented in the batch script:



Create and set variable that make your code more readable; you can reference them by using the $

ntasks is now set to 2, meaning we can use 2 threads as long as the software we are using supports threading which it does! This means it could potentially cut run time in half!

```
  GNU nano 2.3.1                                File: alignment.sh

#!/bin/bash

#SBATCH -p defq
#SBATCH --ntasks=2
#SBATCH --job-name=sub
#SBATCH --error=alignment.err
#SBATCH --output=alignment.log


#TO DO -- THIS SCRIPT WILL NOT RUN UNLESS YOU PERFORM ALL THE FOLLOWING:
#add SBATCH header to give it max time of 10 minutes
#add SBATCH header to give it max memory of 10GB
#change SBATCH --error and --output so that alignment.err and alignment.log get redi
#change your job-name to inclue your name somewhere
#put the full path to your bwa program that you installed within the single quotes o


myBwaPath=''
refFile='/gpfs/share/training/drosophila_ref_files/GCF_000001215.4_Release_6_plus_IS
read1='/gpfs/share/training/drosophila_fastq/SRR4044399_PE_chr2L_subsample_read1.fas
read2='/gpfs/share/training/drosophila_fastq/SRR4044399_PE_chr2L_subsample_read2.fas


#INTRODUCTION TO $SCRATCH

time $myBwaPath aln -t 2 $refFile $read1 > $SCRATCH/SRR4044399_1.sai

time $myBwaPath aln -t 2 $refFile $read2 > $SCRATCH/SRR4044399_2.sai
```

The time function

Create and set variable that make your code more readable; you can reference them by using the $

ntasks is now set to 2, meaning we can use 2 threads as long as the software we are using supports threading which it does! This means it could potentially cut run time in half!

```
  GNU nano 2.3.1                                File: alignment.sh

#!/bin/bash

#SBATCH -p defq
#SBATCH --ntasks=2
#SBATCH --job-name=sub
#SBATCH --error=alignment.err
#SBATCH --output=alignment.log


#TO DO -- THIS SCRIPT WILL NOT RUN UNLESS YOU PERFORM ALL THE FOLLOWING:
#add SBATCH header to give it max time of 10 minutes
#add SBATCH header to give it max memory of 10GB
#change SBATCH --error and --output so that alignment.err and alignment.log get redi
#change your job-name to inclue your name somewhere
#put the full path to your bwa program that you installed within the single quotes o


myBwaPath=''
refFile='/gpfs/share/training/drosophila_ref_files/GCF_000001215.4_Release_6_plus_IS
read1='/gpfs/share/training/drosophila_fastq/SRR4044399_PE_chr2L_subsample_read1.fas
read2='/gpfs/share/training/drosophila_fastq/SRR4044399_PE_chr2L_subsample_read2.fas


#INTRODUCTION TO $SCRATCH

time $myBwaPath aln -t 2 $refFile $read1 > $SCRATCH/SRR4044399_1.sai

time $myBwaPath aln -t 2 $refFile $read2 > $SCRATCH/SRR4044399_2.sai
```

The time function

Checking error and log outputs

```
[brunettt@cubipmtest02 ~]$ less alignment.err
```

```
[main] CMD: /homelink/brunettt/software/bwa-master/bwa aln -t 2 /gpfs/share/bru
T_genomic.fna /gpfs/share/brunettt/drosophila_fastq/SRR4044399_PE_chr2L_subsamp
[main] Real time: 17.777 sec; CPU: 35.101 sec

real    0m17.784s
user    0m34.986s
sys     0m0.120s
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 34.74 sec
[bwa_aln_core] write to the disk... 0.02 sec
[bwa_aln_core] 211506 sequences have been processed.
[main] Version: 0.7.17-r1194-dirty
[main] CMD: /homelink/brunettt/software/bwa-master/bwa aln -t 2 /gpfs/share/bru
T_genomic.fna /gpfs/share/brunettt/drosophila_fastq/SRR4044399_PE_chr2L_subsamp
[main] Real time: 17.656 sec; CPU: 34.964 sec

real    0m17.662s
user    0m34.827s
sys     0m0.143s
[bwa_sai2sam_pe_core] convert to sequence coordinate...
[infer_isize] (25, 50, 75) percentile: (52, 54, 57)
```

1st time function

2nd time function

Real is the time it literally took according to a wall clock to run the program

User is the time you are charged for (core-hours) remember we used 2 threads to run our program so it shows the time it would take if it ran on 1 core

Check alignment statistics, extract unique read, remove unmapped

Software: *samtools*

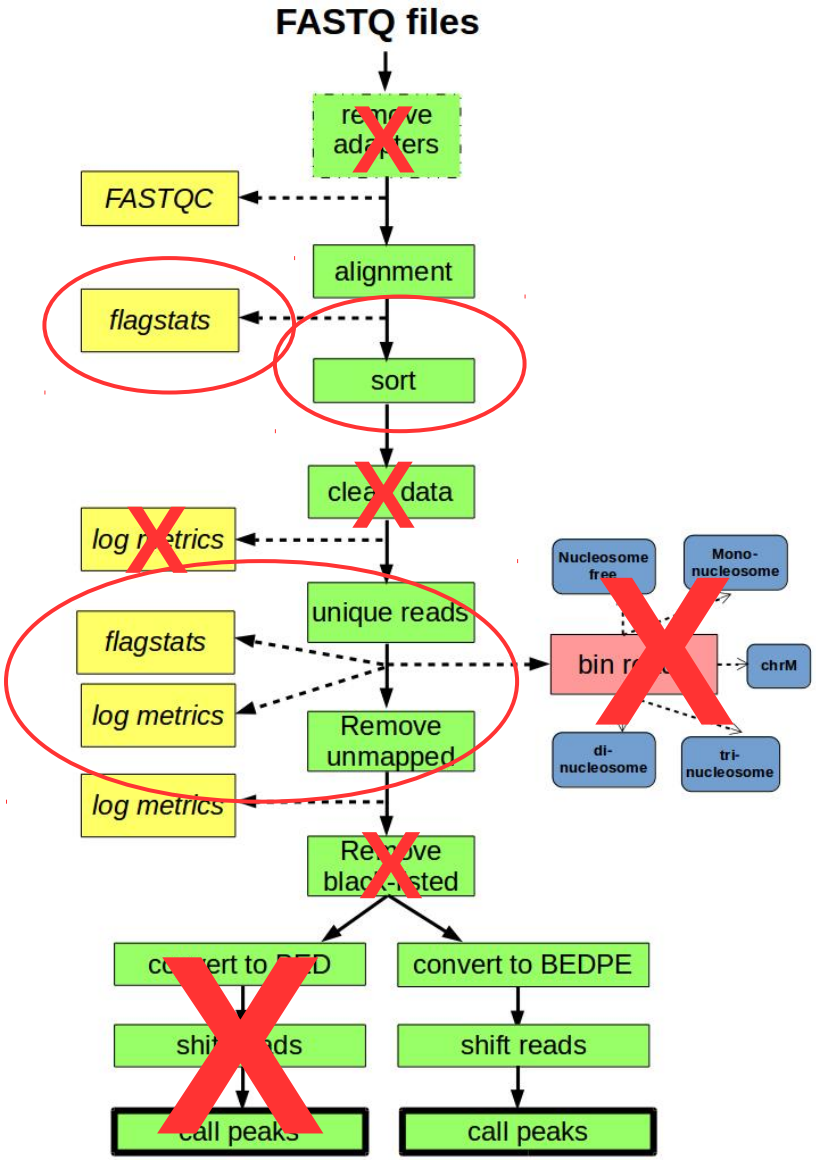
Location: $HOME/software

1. Go to $HOME/scripts and open script/samtools_manipulations.sh

2. Modify this script to run the samtools software and then submit it using SLURM

3. Check to see when you job finishes and check your error logs and log files to make sure the run was successful

What did it output?



Final cleaning and conversion to bedpe format

Software: *samtools/bedtools*

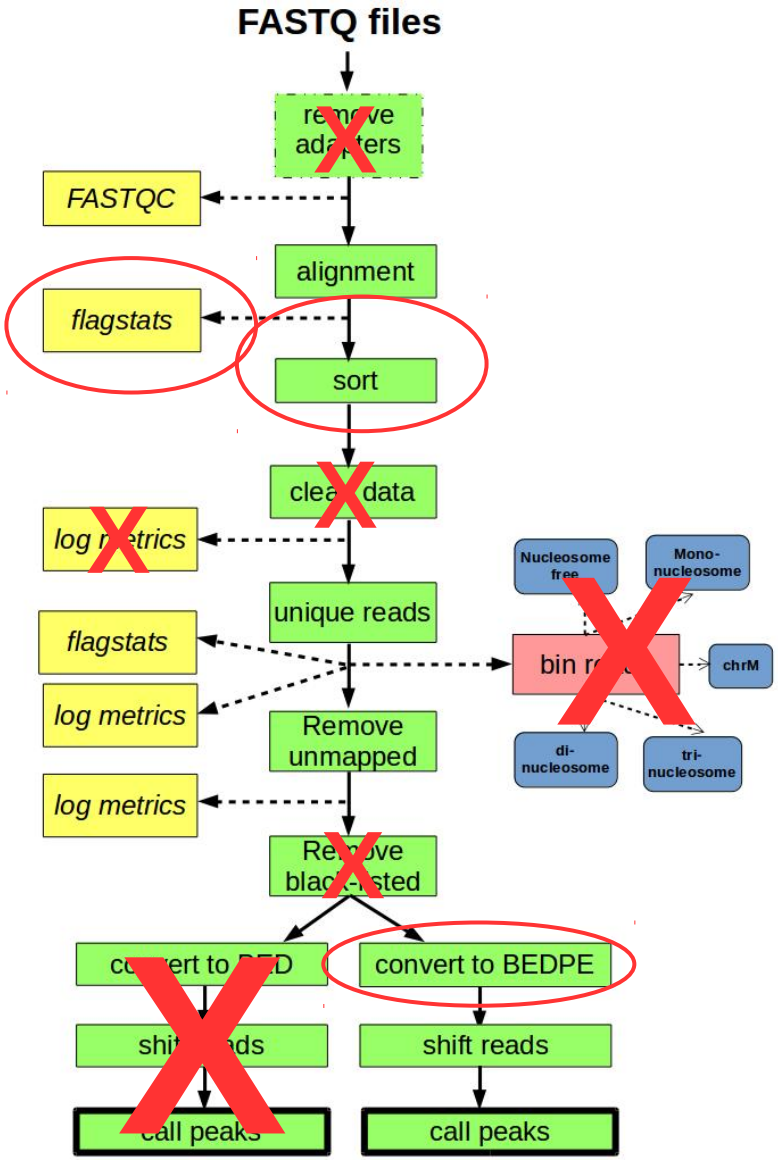
Location: $HOME/software

1. Go to $HOME/scripts and open script/convert_to_bedpe.sh

2. Modify this script to run the samtools and bedtools software and then submit it using SLURM

3. Check to see when you job finishes and check your error logs and log files to make sure the run was successful

What did it output?



We hope you enjoyed this mini-workshop and we hope it will entice you to join the Rosalind community!

Please feel free to contact our team if you have any questions!

CCPM-Rosalind@ucdenver.edu

www.ucdenver.edu/Rosalind