Bioinformatics for Beginners, January 2025

Alexandra L Emmons Ph.D. & Joe Wu Ph.D.
BTEP/GAU/CCR/NCI/NIH - email ncibtep@mail.nih.gov
Bioinformatics Training and Education Program

Table of Contents

Home

Module 1 - Unix/Biowulf

Module 3 - Pathway Analysis

Gene ontology and pathway analysis 188

Help Sessions

Bioinformatics for Beginners: RNA-Seq

Course Description

Bioinformatics integrates biology, statistics, and computer science to develop and apply theory, methods, and tools for the collection, storage, and analysis of biological and related data. Some key application areas in bioinformatics include genomic and molecular analysis, drug discovery and development, medical diagnosis and treatment, agricultural biotechnology, and environmental monitoring. The National Cancer Institute (NCI) uses bioinformatics extensively in its research efforts to combat cancer, including research on the "origin, evolution, progression, and treatment of cancer" *(https://www.cancer.gov/research/infrastructure/bioinformatics#:~:text=NCI's%20Role%20in%20Cancer%20Bioinformatics,-NCI%20Data%20Initiatives&As%20a%20federal%20agency%2C%20NCI,sharing%2C%20analysis%2C%20a*

This course was designed to teach the basic skills needed for bioinformatics, including working on the Unix command line. This course primarily focuses on RNA-Seq analysis. All steps of the RNA-Seq workflow, from raw data to differential expression and gene ontology analysis, are covered. However, many of the skills learned are foundational to most bioinformatics analyses and can be applied to the analysis of other types of next generation sequencing experiments.

Why learn bioinformatics?

Here are a few compelling reasons to explore the world of bioinformatics:

- Analyze your data: Empower yourself to delve into your own biological data, gaining valuable insights.
- Enhancing Scientific Skills: Broaden your scientific knowledge and skills by mastering bioinformatics tools and techniques. By understanding the principles involved with data collection and analysis, you'll be better equipped to design robust experiments and interpret their results effectively.
- Career Opportunities: Open doors to exciting career paths in the rapidly growing field of bioinformatics.
- Understand the Data Landscape: Gain a deeper appreciation for how others analyze biological data, fostering collaboration and critical thinking.

This course is divided into three modules.

Module 1: Unix and Biowulf

Lessons focus on developing command line skills, getting started and working on Biowulf (the NIH HPC cluster), and downloading data from NCBI.

- Lesson 1 - What is Biowulf?
- Lesson 2 - Navigating file systems with Unix
- Lesson 3 - Useful Unix
- Lesson 4 - Working on Biowulf
- Lesson 5 - Downloading data from the SRA

Module 2: RNA-Seq Analysis

Lessons focus on RNA-Seq analysis including experimental design and best practices, quality control, trimming, alignment based methods, feature counts, differential expression analysis, and biological interpretation.

- Lesson 6 - Introduction to RNA-Seq
- Lesson 7 - Introduction to Next Generation Sequencing (NGS) Data and Quality Control
- Lesson 8 - Cleaning and Preparing Next Generation Sequencing (NGS) Data for Downstream Analysis
- Lesson 9 - Aligning Next Generation Sequencing (NGS) Data to Genome
- Lesson 10 - Quantifying Gene Expression from Bulk RNA Sequencing Data
- Lesson 11 - Visualizing Genomic Data: Preparing Files
- Lesson 12 - Visualizing Genomic Data with the Integrative Genomics Viewer
- Lesson 13: Differential Expression Analysis for Bulk RNA Sequencing: QC
- Lesson 14: Differential Expression Analysis for Bulk RNA Sequencing: The Actual Analysis

Module 3: Pathway Analysis

Lessons focus on gene ontology and pathway analysis.

- Lesson 15: Introduction to gene ontology and pathway analysis
- Lesson 16: Functional enrichment with DAVID
- Lesson 17: Pathway Analysis with Reactome

Course requirements:

Who can take this course?

There are no prerequisites to take this course. This course is open to NCI-CCR researchers interested in learning bioinformatics skills, especially those relevant to analyzing bulk RNA sequencing data.

How will we work through lesson content?

For the hands-on sessions, participants will use Biowulf student accounts. To sign up for a student account, click here *(https://docs.google.com/spreadsheets/d/1Epjzdtg9NxtJ8nm8Nti-l0l5S-5FOCsASe4hE4GXmaA/edit?usp=sharing)*. ***Student accounts are only available to course registrants.***

Lesson content and practice questions can be found in these pages.

Class documents are available at https://bioinformatics.ccr.cancer.gov/docs/bioinformatics-for-beginners-2025/ *(https://bioinformatics.ccr.cancer.gov/docs/bioinformatics-for-beginners-2025/)*.

Class data

Below are the links for the class data in case participants would like to practice outside of and after this course series. **There is no need to download these for this course as the instructors have made them available on Biowulf**. If you do not have access to Biowulf, see the below instructions.

Module 1

You can find compressed Module 1 data here. Download the data and unzip.

```
unzip module_1.zip
```

Module 2

All Module 2 data were obtained from the Griffith lab RNA sequencing tutorial *(https://rnabio.org/)* and renamed for this course series.

Reference genome can be downloaded at https://rnabio.org/module-01-inputs/0001/02/01/Reference_Genomes/ *(https://rnabio.org/module-01-inputs/0001/02/01/Reference_Genomes/)*

Annotation can be downloaded at https://rnabio.org/module-01-inputs/0001/03/01/Annotations/ *(https://rnabio.org/module-01-inputs/0001/03/01/Annotations/)*

See https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/ *(https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/)* for instructions on downloading the HBR-UHR and hcc1395 data.

Contact Us

Email ncibtep@nih.gov if you have any comments, questions, or concerns.

Module 1 - Unix/Biowulf

Lesson 1: What is Biowulf?

To fully engage with the course material and complete the hands-on exercises, we'll be leveraging the powerful NIH HPC Biowulf system.

To make the most of this powerful tool, it's essential to grasp the fundamentals of working with HPC systems, specifically Biowulf. In this lesson and others in Module 1, we will delve into the key concepts and practical skills you'll need to navigate this environment effectively.

Learning Objectives

1. Understand the components of an HPC system. How does this compare to your local desktop?
2. Learn about Biowulf, the NIH HPC cluster.
3. Learn about the command line interface and resources for learning.

Additional training materials at hpc.nih.gov

Much of the content for this presentation is from hpc.nih.gov. For more information and more detailed training documentation, see hpc.nih.gov/training/ *(https://hpc.nih.gov/training/)*.

> **Note**
>
> This lesson will be demo-based only. Hands-on lessons will not begin until Lesson 2.

What is a high performance cluster (HPC)?

> A collection of standalone computers that are networked together. They will frequently have software installed that allow the coordinated running of other software across all of these computers. This allows these networked computers to work together to accomplish computing tasks faster. --- hpc-intro (Software carpentries) *(https://carpentries-incubator.github.io/hpc-intro/files/jargon.html#p6)*

NIH HPC Systems

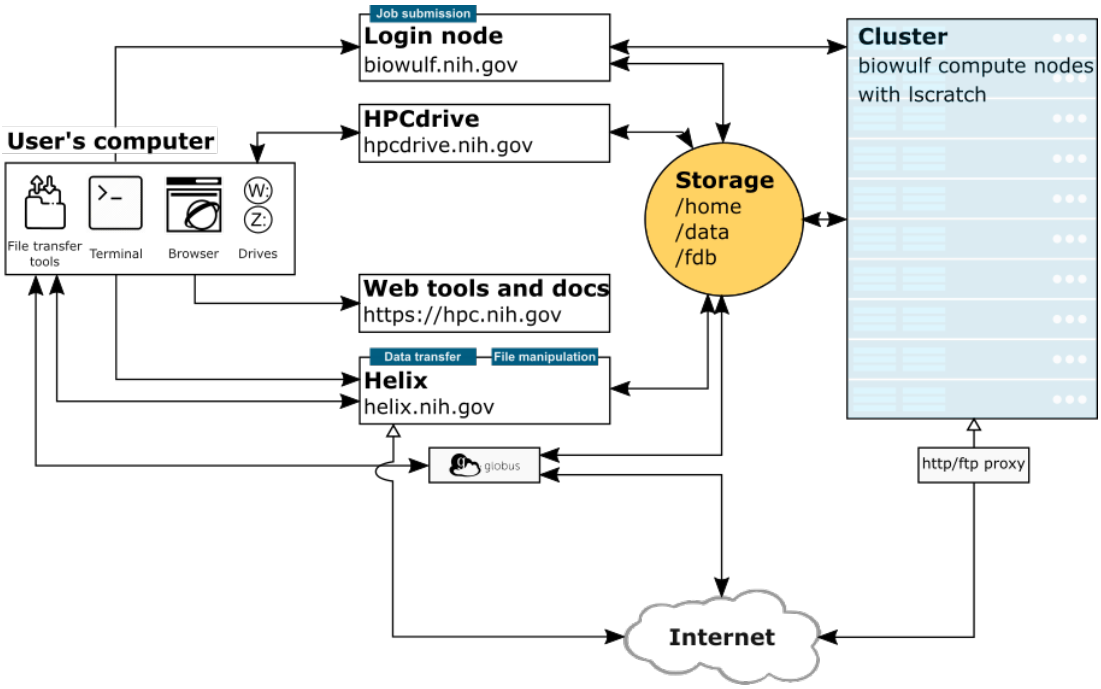


Image Credit: hpc.nih.gov/systems *(https://hpc.nih.gov/systems/)*

This diagram describes the NIH HPC Systems (Login node, Biowulf Cluster, HPC drive, and Helix), how the systems interact, and how you interact with the systems. Each "system" is described here *(https://hpc.nih.gov/systems/)*. We will also discuss these further below.

When using an HPC

- We use a **command line interface** and a Secure shell protocol (SSH) to establish a remote connection to the login node / head node.
    - Most of us are likely used to a graphical user interface (GUI), which is a point-and-click interface, meaning we use a mouse to move around and click on various display icons. We visually interact with our compute environment. However, we do not generally interact with an HPC in this way. HPCs are remote resources that require connections using slow or intermittent interfaces (over WIFI and VPNs). Because of this, it is more practical to guide functionality over the command line using plain text. *(https://carpentries-incubator.github.io/hpc-intro/11-connecting/index.html)*
    - The cluster head node (login node) distributes compute tasks (the things we want to do outside of file manipulation and editing) using a scheduling system (e.g., SLURM). We use a scheduling system because the HPC is a shared resource with hundreds of worker nodes and thousands of processors.

Note

Slurm stands for Simple Linux Utility for Resource Management.

What is Biowulf?

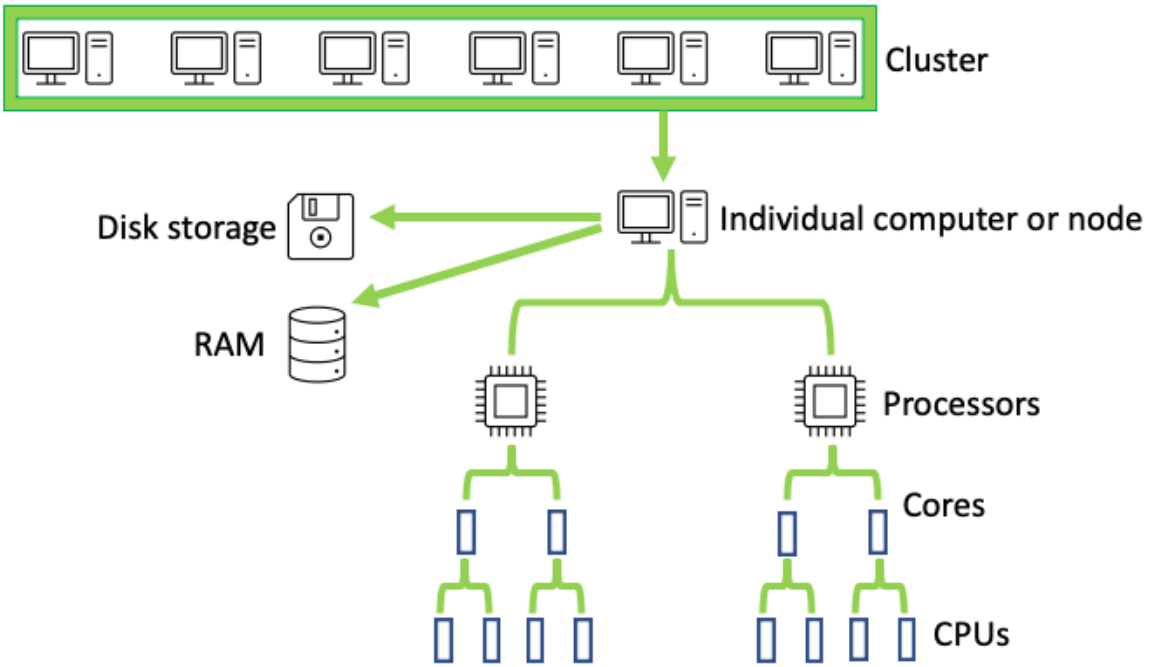
Biowulf is the National Institutes of Health's (NIH) high-performance computing (HPC) cluster. This massive Linux system boasts over 90,000 processors, allowing it to tackle enormous computational tasks by dividing jobs and running them simultaneously across multiple nodes. Biowulf comes pre-loaded with over 600 scientific software programs *(https://hpc.nih.gov/apps/)* and databases *(https://hpc.nih.gov/apps/db.php)* relevant to various fields like genomics, molecular biology, and bioinformatics. For security reasons, access to Biowulf is restricted to the NIH campus network or through a Virtual Private Network (VPN).

When should we use Biowulf?

You should use Biowulf when:

- Software is unavailable or difficult to install on your local computer and is available on Biowulf.
- You are working with large amounts of data that can be parallelized to shorten computational time AND/OR
- You are performing computational tasks that are memory intensive.

Example of High Performance Computing Structure



Essentially Biowulf is a scaled up version of your local computer.

In Biowulf, many computers make up a cluster. Each individual computer or node has disk space for storage and random access memory (RAM) for running tasks. The individual computer is composed of processors, which are further divided into cores, and cores are divided into CPUs.

Info

Information on the NIH HPC architecture and hardware here *(https://hpc.nih.gov/systems/hardware.html)*.

Getting an NIH HPC account

- If you do not already have a Biowulf account, you can obtain one by following the instructions here *(https://hpc.nih.gov/docs/accounts.html)*.
- NIH HPC accounts are available to all NIH employees and contractors listed in the NIH Enterprise Directory.
- Obtaining an account requires PI approval and a nominal fee of $40 per month.
- Accounts are renewed annually contigent upon PI approval.

Node types on the HPC

- Login node (head node)
  - Used for submitting resource intensive tasks as jobs
  - Editing and compiling code
  - File management and data transfers on a small scale
- Compute nodes (worker nodes)
  - For computational processes
  - Requires interaction with a job scheduling system (SLURM)
  - Batch jobs, sinteractive sessions
- Data transfer node (For Biowulf, this is Helix.)
  - more on this below

Info

`sinteractive` - work on biowulf compute nodes interactively; suitable for testing/debugging cpu-intensive code, Pre/post-processing of data, and using graphical applications.
`sbatch` - for submitting shell scripts via jobs, taking away any interactive component.
`swarm` - used for runnning embarassingly parallel code as independent jobs.

The Data transfer node: Helix

- Used for data transfers and file management on a large scale.
- 48 core system with 1.5 TB of main memory
- direct internet connection
- Helix should be used when
  - you are transferring >100 GB using `scp`
  - gzipping a directory containing >5K files, or > 50 GB
  - copying > 150 GB of data from one directory to another.
  - uploading or downloading data from the cloud.
- For more information on data transfers see hpc.nih.gov *(https://hpc.nih.gov/docs/transfer.html)*.

Biowulf Data Storage

Summary of file storage options

	Location	Creation	Backups	Space	Available from
/home	network (NFS)	with Biowulf/Helix account	yes	16 GB default quota	B,C,H
/lscratch (nodes)	local	created by user job	no	~850 GB shared	C
/scratch	network (NFS)	created by user	no	100 TB shared	B,H
/data	network (GPFS/NFS)	with Biowulf/Helix account	no	100 GB default quota	B,C,H

H = helix, B = biowulf login node, C = biowulf compute nodes

- You may request more space on `/data`, but this requires a legitimate justification.
- More information on data storage here *(https://hpc.nih.gov/storage/)*.

Important

Data storage on the HPC system should not be for archival purposes.

Note

Though there aren't true back-ups of your data directories, there are snapshots with a view of your home and data directories at a specific point in time. You can learn more about snapshots *(https://hpc.nih.gov/storage/backups.html)* in the HPC documentation.

- To check disk space use:

  `checkquota` - this shows the directories for which you have write access

  OR

  Look on the *user dashboard *(https://hpcnihapps.cit.nih.gov/auth/dashboard)* -> disk storage.

  *Only works on VPN

Best practices file storage

BAD	GOOD
Submitting a swarm without knowing how much data it will generate	Run a single job, sum up the output and tmp files, and figure out if you have enough space before submitting the swarm
Directory with 1 million files	Directories with < 5,000 files
100 jobs all reading the same 50 GB file over and over from /data/$USER/	Use /lscratch instead, copy the file there, and have each job access the file on local disk
100 jobs all writing and deleting large numbers of small temporary files	Use /lscratch instead, have all tmp files written to local disk
Each collaborator having a copy of data on Biowulf	Ask for a shared area and keep shared files there to minimize duplication
Use Biowulf storage for archiving	Move unused or old data back to you local system

How do I create a directory in scratch?

```
mkdir /scratch/$USER
```

Applications on Biowulf

- Bioinformatics applications and other programs are available on Biowulf via **modules**.
- View a list of available applications here *(https://hpc.nih.gov/apps/)*.

Info

Loading software as environment modules allows us to better control our computational environment and easily use a large number of programs and even different versions of the same programs. Modules alter the user's environment variables such as the execution path.

The Command Line Interface (CLI)

What is Unix?

- Unix is a proprietary operating system like Windows or MacOS (Unix based).
- There are many Unix and Unix-like operating systems, including open source Linux and its multiple distributions.
- Biowulf nodes use a Unix-like (Linux) operating system (distributions RHEL8/Rocky8).
- Biowulf requires knowledge and use of the command line interface (shell) to direct computational functionality.
- To work on the command line we need to be able to issue Unix commands to tell the computer what we want it to do.

Tip

A basic foundation of Unix is advantageous for most scientists, as many bioinformatics open-source tools are available or accessible by command line on Unix-like systems.

Accessing your local terminal or command prompt

Mac OS

Type `cmd + spacebar` and search for "terminal". Once open, right click on the app logo in the dock. Select `Options` and `Keep in Dock`.

Windows 10 or greater

> You can start an SSH session in your command prompt by executing ssh user@machine and you will be prompted to enter your password. ---Windows documentation *(https://docs.microsoft.com/en-us/windows/terminal/tutorials/ssh?source=recommendations)*

To find the Command Prompt, type `cmd` in the search box (lower left), then press `Enter` to open the highlighted Command Prompt shortcut.

Are you a Windows user and not affiliated with NIH?

If you are using a Windows operating system, Windows 10 or greater, you can use the Windows Subsystem for Linux *(https://docs.microsoft.com/en-us/windows/wsl/)* (WSL) for your computational needs.

> The Windows Subsystem for Linux (WSL) is a feature of the Windows operating system that enables you to run a Linux file system, along with Linux command-line tools and GUI apps, directly on Windows, alongside your traditional Windows desktop and apps. --- docs.microsoft.com *(https://docs.microsoft.com/en-us/windows/wsl/faq)*

To install WSL, follow the instructions here *(https://learn.microsoft.com/en-us/windows/wsl/install)*. There are multiple Linux distributions. We recommend new users install "Ubuntu".

Windows WSL is not available to NIH employees due to security policies.

How much Unix do we need to learn?

As with any language, the learning curve for Unix can be quite steep. However, to work on Biowulf you really need to understand the following:

- **Navigating the File System:** Understanding the hierarchical structure of directories, using the `cd` command to move between directories.
- **File Paths:** Learning how to specify the location of files using absolute and relative paths.
- **Basic Unix Commands:** Getting acquainted with common commands like `ls` for listing files, `mv` for moving files, `rm` for removing files, `mkdir` for creating directories, `cat` for viewing file contents, and `man` for accessing command documentation.
- **Getting help:** Discovering how to find more information about Unix commands and their usage.
- **Command Customization:** Learning how to modify the behavior of Unix commands using flags or options, such as using `ls -l` to list files with detailed information compared to the basic `ls` command.
- **Redirecting Input and Output:** Understanding standard input and output, and how to redirect the output of one command to the input of another using pipes (`|`) or redirection operators (`>`, `<`).

More on these in the next lesson.

Connecting to Biowulf

- To connect to Biowulf, we use a secure shell (SSH) protocol.
    - used to open an encrypted network connection between two machines, allowing you to send & receive text and data without having to worry about prying eyes. *(https://carpentries-incubator.github.io/hpc-intro/11-connecting/index.html)*
    - `man ssh`

Establishing a remote connection

```
ssh username@biowulf.nih.gov
```

"username" = NIH/Biowulf login username.

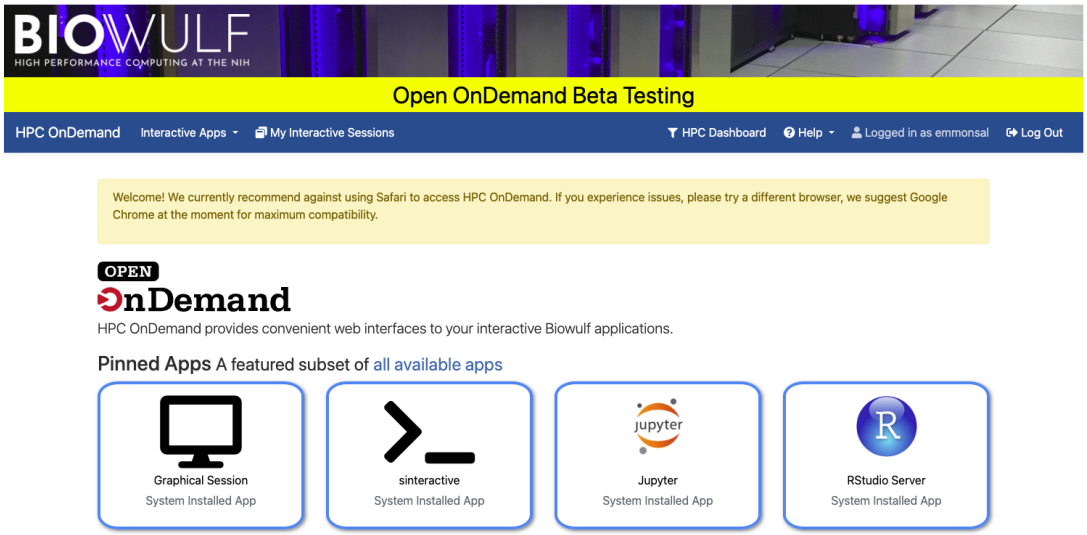
> **Note**
>
> If this is your first time logging into Biowulf, you will see a warning statement with a yes/no choice. Type "yes".

Type in your password at the prompt. **The cursor will not move as you type your password!**

HPC OnDemand

Recently, the NIH HPC Team has provided on demand access to HPC resources via web browser through integration of Open OnDemand. This integration makes working with HPC resources less intimidating for new users, as they will not have to open a terminal and remotely connect via `ssh`. Instead, navigate to your web browser (Google Chrome is preferred) and connect to NIH HPC OnDemand using https://hpcondemand.nih.gov/ *(https://hpcondemand.nih.gov/)*.

HPC OnDemand *(https://hpc.nih.gov/ondemand/index.html)* provides an online dashboard for users to easily access command line interactive sessions, graphical linux desktop environments, and interactive applications including RStudio, MATLAB, IGV, iDEP, VS Code, and Juptyer Notebook.



HPC OnDemand Dashboard

SLURM commands

You will also need to know commands specific to the Biowulf job scheduling system:

- `sbatch` submit slurm job
- `swarm` submit a swarm of commands to cluster
- `sinteractive` allocate an interactive session
- `sjobs` show brief summary of queued and running jobs
- `squeue` display status of slurm batch job
- `scancel` delete slurm jobs

We will talk about many of these in more detail in later lessons.

How to load / unload a module

- To see a list of available software in modules use

```
module avail
module avail [appname|string|regex]
module -d
```

- To load a module

```
module load appname
module load appname/version
```

- To see loaded modules

```
module list
```

- To unload modules

```
module unload appname
module purge #(unload all modules)
```

> **Note**
>
> You may also create and use your own modules.

Getting help on Biowulf: NIH HPC Documentation

The NIH HPC systems are well-documented at hpc.nih.gov *(https://hpc.nih.gov/)*.

- User guides *(https://hpc.nih.gov/docs/user_guides.html)*
- Training documentation *(https://hpc.nih.gov/training/)*
- How To *(https://hpc.nih.gov/docs/how_to.html)* docs

> **Note**
>
> **Existing safeguards make it nearly impossible for individual Biowulf users to irreparably mess up the system for others.**
>
> WORST CASE SCENARIO - You are locked out of your account pending consultation with NIH HPC staff

---

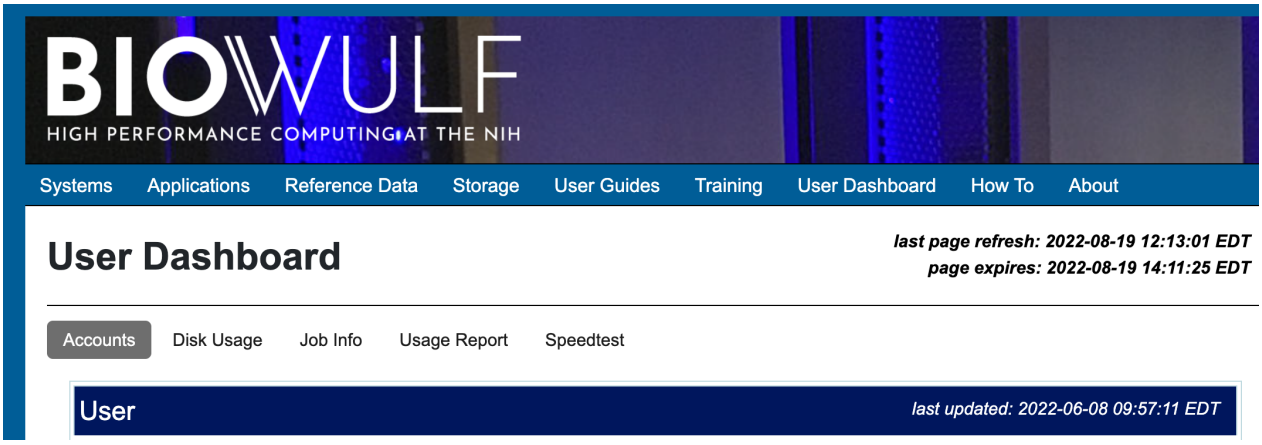
Additional HPC help

- **Contact staff@hpc.nih.gov *(mailto:staff@hpc.nih.gov)***
  The HPC team welcomes questions and is happy to offer guidance to address your concerns.

- **Monthly Zoom consult sessions**
  The HPC team offers monthly zoom consult sessions. "All problems and concerns are welcome, from scripting problems to node allocation, to strategies for a particular project, to anything that is affecting your use of the HPC systems. The Zoom details are emailed to all Biowulf users the week of the consult." *(https://hpc.nih.gov/training/)*

- **Bioinformatics Training and Education Program**
  If you experience any difficulties or challenges, especially with different bioinformatics applications, please do not hesitate to email us at BTEP *(mailto:ncibtep@nih.gov)*.

User Dashboard



- Can view disk usage and job info
- Request more disk space
- Evaluate job info for troubleshooting

Learning Unix: Classes / Courses

- Introduction to Biowulf (May – Jun, 2023) *(https://bioinformatics.ccr.cancer.gov/docs/biowulf-introduction-summer-2023/index.html)*
- Introduction to Unix on Biowulf (Jan – Feb, 2023) *(https://bioinformatics.ccr.cancer.gov/docs/unix-on-biowulf-2023/index.html)*
- Bioinformatics for Beginners: Module 1 Unix/Biowulf *(https://bioinformatics.ccr.cancer.gov/docs/b4b/Module1_Unix_Biowulf/Lesson1/)*

Additional Unix Resources:

- BashScripting_LinuxCommands from the NIH HPC team *(https://hpc.nih.gov/training/handouts/BashScripting_LinuxCommands.pdf)*
- Fosswire linux reference sheet *(https://bioinformatics.ccr.cancer.gov/docs/b4b/fosswire_reference.pdf)*

Key points

- Biowulf is the high performance computing cluster at NIH.
- To work on Biowulf, you will need to use the command line interface, which requires some knowledge of unix commands.
- When you apply for a Biowulf account you will be issued two primary storage spaces:
    1. `/home/$User` (16 GB)

2. `/data/$USER` (100 GB).

- Hundreds of pre-installed bioinformatics programs are available through the `module` system.
- Computational tasks on Biowulf should be submitted as a job (`sbatch`, `swarm`) or through an interactive session (`sinteractive`).
- Do not run computational tasks on the login node.

Lesson 2: Navigating file systems with Unix

Lesson 1 Review

- Biowulf is the high performance computing cluster at NIH.
- When you apply for a Biowulf account you will be issued two primary storage spaces: 1) `/home/$User` and 2) `/data/$USER`, with 16 GB and 100 GB of default disk space.
- Hundreds of pre-installed bioinformatics programs are available through the `module` system.
- Computational tasks on Biowulf should be submitted as a job (`sbatch`, `swarm`) or through an interactive session `sinteractive`.
- Connect to Biowulf using HPC OnDemand *(https://hpc.nih.gov/ondemand/index.html)* or `ssh`.
- Do not run computational tasks on the login node.

Lesson Objectives

In lesson 1, you learned about the NIH HPC cluster, Biowulf. Biowulf nodes use a Unix-like (Linux) operating system (distributions RHEL8/Rocky8), which requires knowledge and use of the command line interface (shell) to direct computational functionality. The purpose of today's lesson is to get you familiar with working on the command line. To this end, we will...

- Learn the basic structure of a unix command.
- Learn how to navigate your file system, including absolute vs relative directories.
- Learn unix commands related to navigating directories, creating and removing files or directories, and getting help.

A word about mistakes

YOU WILL MAKE MISTAKES...but, it is okay. We all make mistakes, and mistakes are how we learn. Remember, existing safeguards make it nearly impossible for individual Biowulf users to irreparably mess up the system for others. However, you can make your life difficult, for example, by misusing commands, ignoring existing tools, overwriting files, failing to redirect or output results, disregarding warnings, and not seeking help.

How can we overcome mistakes?

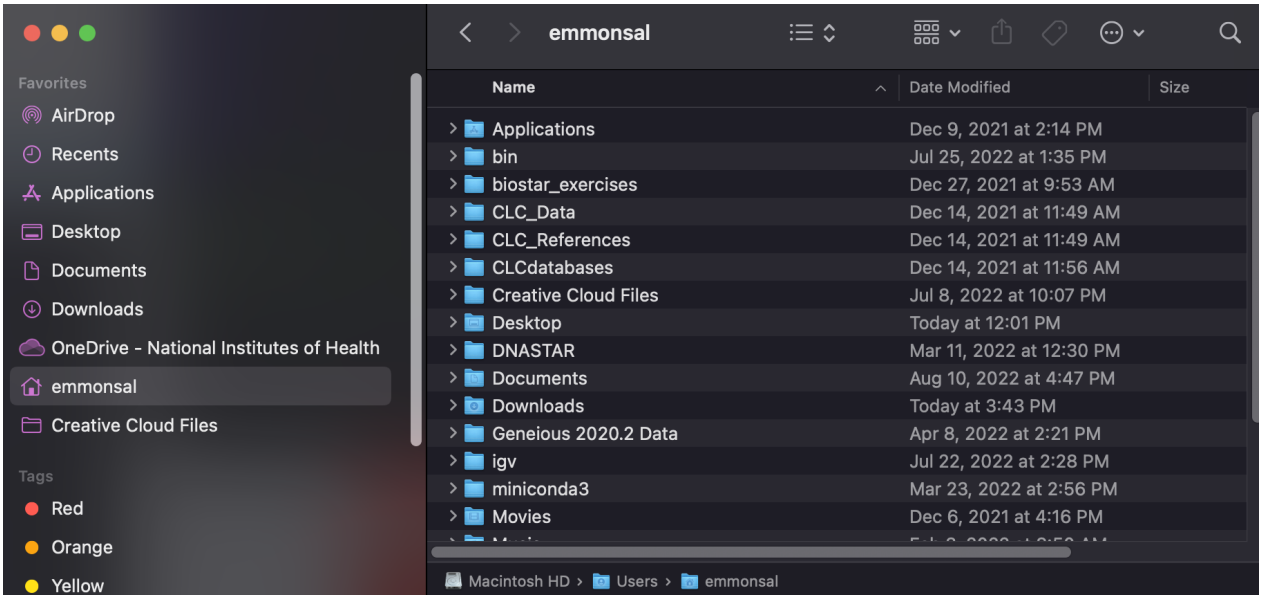
We practice. The more you use unix and bash scripting the better you will become.

You will need to learn how to troubleshoot error messages. Often this will involve googling the error in reference to the entered command. There are many forums that post help regarding specific errors (e.g., stack overflow, program repositories such as github).

File system

We manage files and directories through the operating system's file system. A directory is synonymous with a "folder", which is used to organize files, other directories, executables, etc.

On a Windows or Mac, we usually open and scroll through our directories and files using a GUI. For example, **Finder** is the default file management GUI from which we can access files or deploy programs on a macbook.



This same file system can be accessed and navigated via command line from the unix shell.

Some useful unix commands to navigate our file system and tell us some things about our files

1. `pwd` (print working directory)
2. `ls` (list)
3. `touch` (creates an empty file)
4. `nano` (basic editor for creating small text files)
5. using the `rm` command to remove files. Be careful!
6. `mkdir` (make a directory) and `rmdir` (remove a directory, must be empty of all files)
7. `cd` (change directory), by itself will take you home, `cd ..` (will take you up one directory), `cd /results_dir/exp1` (go directly to this directory)
8. `mv` (for renaming files or moving files)

9. `less` (for viewing files, or `more`)
10. `man` (for viewing the man pages when you need help on a command)
11. `cp` (copy) for copying files

Getting Started

We have already seen some unix commands relevant to Biowulf. For example, we learned about `ssh`. The ssh command is used to securely log into a remote machine and execute commands on that machine. *(https://www.tutorialspoint.com/common-ssh-commands-in-linux-with-examples#:~:text=The%20ssh%20command%20is%20used,hostname%20of%20the%20remote%20machin*

`ssh` is the command and `username@biowulf.nih.gov` is a command line argument, where `username` is the username that you wish to connect to on the remote system and `biowulf.nih.gov` is the hostname of the remote machine. For this lesson and the lessons that follow, we will use NIH HPC student accounts *(https://hpc.nih.gov/nih/student.html)* to connect to Biowulf.

More about student accounts

At the beginning of each class you must sign up for a student account. You can sign up for a student account using a Google spreadsheet, the link for which will be supplied at the beginning of each class via Webex. Click on the supplied link, and find an empty slot under the **"Name"** column. Type your name in the empty slot. The username under **"Account Username"** will now serve as your username for logging in to Biowulf.

This task will be repeated at the beginning of each lesson to allow students the option of flexible attendance.

Let's go ahead and get connected. Open a terminal and type the following:

```
ssh username@biowulf.nih.gov
```

`username` = NIH/Biowulf login username. *Remember to use the student account username here.*

Note

If this is your first time logging into Biowulf, you will see a warning statement with a yes/no choice. Type "yes".

Type in your password at the prompt. **The cursor will not move as you type your password!**

Success

We will connect to Biowulf at the beginning of each session.

We are now on the login node. Remember, you should not do work on the login node. However, you can do basic file management on the login node and edit and compile code. For now, we will stay on the login node.

Our Second Unix Command (`ls`)

Let's continue learning about the structure of linux commands using another common command, `ls`. The `ls` command "lists" the contents of the directory you are in. You may see files and other directories here.

```
ls
```

At this point, you are in your home directory, and so you will see whatever files and directories are located here. For example, if I had logged in to my Biowulf account, I would see the following:

```
Desktop  R  bin  ncbi_error_report.txt
```

However, since this is a student account, I do not see anything, as I have not yet added any files.

How can you tell the difference between a file and a directory?

We can add some additional options (flags) to our command.

```
ls -lh
```

will show permissions and indicate directories (`d`). The `-lh` are flags. `-l` refers to listing in long format, while `-h` provides human readable file sizes.

Or, many systems offset directories and files using colors (e.g., blue for directories). If you don't see colorized output, try the `-G` flag.

We can also label output by adding a marker to indicate files, directories, links, and executables using the `-F` flag.

```
ls -F
```

a terminal `/` = directory
a `@` = link
a `*` = executable

Anatomy of a command

Using `ls` as an example, we can get an idea of the overall structure of a unix command.

Image inspired by "Learn Enough Command Line to Be Dangerous"

The first thing we see is the command line prompt, usually `$` or `%`, which will vary by operating system. The prompt let's us know that the computer is waiting for a command. Next we see the actual command, in this case, `ls`, telling the computer to list the files and directories. Most commands will have various options / flags that can be included to modify the command function. We can also supply an argument, which in the case of `ls` is optional. For example, here we supplied an alternative directory from which we are interested in listing files and directories. We hit `enter` or `return` after each command, and when the command has finished running, the command prompt will reappear prompting us to enter more commands.

Where am I? (`pwd`)

```
pwd
```

`pwd` stands for "print working directory". When you run this, you should see something like this.

```
/home/username
```

where username is your name or student account. This is your home directory - where you start from when you open a terminal. This is an example of a "path". The path tells us the location of a file or directory.

Note

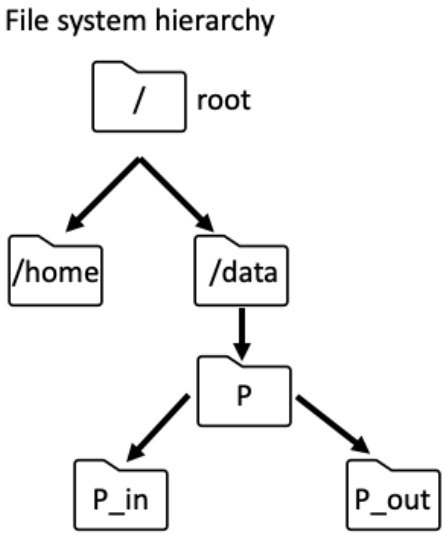
While Windows computers use a `\` as a path separator, unix systems use a `/`.

Therefore, the `pwd` command is very helpful for figuring out where you are in the directory structure. If you are getting "file not found" errors while trying to run something, it is a good idea to `pwd` and see if you are where you think you are. Type the `pwd` command and make a note of the directory you are in.

More on the home directory (~)

We see that we are in our home directory. But where is that exactly?

The file system on any computer is hierarchical. On a Unix system, the top level of the file system, or root directory, is denoted by `/`. All subdirectories on the file system branch from this root directory. See the below example.



Example of file system hierarchy structure.

In our example hierarchy, we have subdirectories `/home` and `/data`, and within `data`, we see additionally subdirectories, `P_in` and `P_out`. Only the first `/` denoted a directory (root). All other `/`s in the path serve as separating characters.

Absolute vs Relative directories

A file path that starts with the root or `/` is known as an absolute path. A path that does not start with the root directory is called a relative path. For example, in Unix, `.` is used to denote here in the present working directory and `..` is used to denote one directory back. Thus, a path that starts with `.` or `..`is a relative path. Going back to `pwd`. The output (`/home/username`) is an absolute file path. Absolute file paths will break scripts when collaborating because the likelihood that your file system matches another's is low.

We can use the `tree` command to get an idea of the structure of our home directory on Biowulf.

Getting around the Unix directory structure (`cd`)

How do we navigate this directory tree. We use `cd`, which means "change directory". Let's change directory to our data directory, which is the larger of the two allocations we are allotted on Biowulf.

```
cd /data/$USER # (1) change to your data directory
pwd #print working directory
ls #list the contents of /data/$USER
```

$USER and other environment variables

`$USER` is an example of an environment variable.

> Environment variables contain user-specific or system-wide values that either reflect simple pieces of information (your username), or lists of useful locations on the file system. --- Griffith Lab *(https://pmbio.org/module-01-setup/0001/03/01/Unix_Primer/)*

We can display these variables using `echo`.

```
echo $USER
echo $HOME
```

`$PATH` is an important environment variable.

```
echo $PATH
```

This results in a colon separated list of directories containing programs that you can run without specifying those directories each time you run the program.

You will likely need to add to your `$PATH` at some point in the future.

To do this use:

```
export PATH=$PATH:/path/to/folder
```

This change will not remain when you close the terminal. To permanently add a location to your path, add the above line to your bash shell configuration file, `~/.bashrc`.

By itself, the `cd` command takes you `home`. Let's try that, and then do a `pwd` to see where we are.

```
cd
pwd
```

We are back in our home directory.

```
/home/username
```

How can we go back to the `/data/$USER` directory? We need to give the "path" to that directory.

```
cd /data/$USER
pwd
ls
```

Check where you are with `pwd` and look at the contents of the directory with `ls`. What do you see?

> **Home shortcut**
>
> We can also use ~ as a shortcut for our home directory.
>
> ```
> ls ~
> ```

Once we create more files and directories, we can learn a bit more about the directory structure and absolute vs relative file paths.

Creating files (touch)

The `touch` command creates a file, but the file is empty, so it is not a command you will use very often, but good to know about.

```
touch file1.txt
touch file2.txt
ls
```

Now we see something like this.

```
file1.txt  file2.txt
```

The `nano` editor is a text editor useful for small files.

```
nano file2.txt
```

Let's put something in this file.

```
Unix is an operating system, just like Windows or MacOS.
Linux is a Unix like operating system;
sometimes the names are used interchangeably.
```

Nano commands for saving your file and exiting nano:

1. control O - write file (equivalent to save as)
2. File name to write: file2.txt (Hit return/enter on your keyboard to save the file with this name).
3. control X - to Exit

This brings us to our next topic which is very important!

Avoid spaces in file names and directories.

There should not be spaces in Unix file names or directories. There are many strategies that can be used to avoid spaces in file names *(https://builtin.com/articles/pascal-case-vs-camel-case#:~:text=Pascal%20case%20is%20a%20casing,capitalized%2C%20such%20as%3A%20camelCase%2* Though, consistency is key. One good method is using snake_case, in which words are separated by an _.

For example, we can use the underscore (_) where a space would go, like this, to name a directory for module 1.

```
Module_1
```

To use snake_case with file names, we may see something like this:

```
brain_rna.fastq
liver_rna.fastq
```

The first part of the file name provides info about the file, and the extension (.fastq) tells what kind of file it is. (Examples of file extensions are .csv, .txt, .fastq, .fasta and many more.)

More on file organization to come.

Understanding file extensions

It's important to understand file extensions, to know what kinds of data you are working with.

.txt are text files. These are likely but not always tab delimited.

.tsv are tab delimited files.

.csv are "comma-separated values" - good for importing into MS Excel spreadsheets

.tar.gz indicates a tarred and zipped file - so it is a compressed file

.fastq tells you that these are FASTQ files, containing sequence data and quality scores

.fasta indicates FASTA formatted sequence data, either protein or nucleotide

Removing files with `rm`

Warning

A Unix system will delete something when you ask to delete it and there is usually no way of getting it back. Be extremely careful when removing files and directories.

By adding the `-i` option, the system will ask if you're sure you want to delete. Generally speaking, when a file on a Unix system is deleted, it is gone.

```
rm -i file1.txt
```

will remove a file we created.

Creating directories (`mkdir`)

A couple things to note - this is a good time to give your directories meaningful names, which will help you keep things organized. Organization is key. I generally like to have a new directory per project, and within that directory, subdirectories separating raw data from analysis files. From there, each analysis would also get its own subdirectory. However, there are many ways to organize files and you should do whatever makes sense for your data and helps you (and others) stay organized.

Keep raw data raw

Always keep your raw data raw, and save outputs to new files. Do not overwrite raw data! Consider setting the permissions on these files to "read only". More on permissions later.

For now, let's create a directory called `Module_1`, where we can store Module 1 lesson content. To create a directory, we use `mkdir`.

```
mkdir Module_1
```

Removing directories (`rmdir`)

Directories must be completely empty of all files and other contents before you can delete them with `rmdir`. There are ways to "recursively" remove files and directories using the `-r` option of `rm`, for example (`rm -r directory`). This would remove all of the files and subdirectories in our hypothetical `directory`. Keep in mind that once these files are deleted they are gone for good. Be extremely careful with the `-r` option. As beginners it can be safer to navigate to the directory and remove content directly.

Let's take a quick second to apply some of the things we have learned and create more content to work with.

Navigate to the directory we just made (`Module_1`), and make another directory within it called `directory_to_delete`.

```
cd Module_1 #change directory
pwd #print working directory (wd)
mkdir directory_to_delete #make a new directory
ls #list the contents of wd
```

Now let's move to `directory_to_delete` and create a file, `myseq.txt`.

```
cd directory_to_delete #change directory
touch myseq.txt #Create a new file
```

Let's check the contents of our directory and see where we are located in our directory tree.

```
ls
pwd
```

To summarize what we have done:

We've moved to the `Module_1` directory, checked our directory with `pwd`, created a directory called `directory_to_delete`, and listed the contents of `Module_1`, so we can see the directory we just created. We then navigated to `directory_to_delete` using `cd`, created a file with `touch`, listed the contents with `ls`, and printed our working directory (`pwd`).

Let's move up one directory back to `Module_1`.

```
cd ..
```

Test Your Knowledge: Question 1

How could you move back to your home directory?

Answer Question 1

```
cd
pwd
```

```
/home/username
```

Test Your Knowledge: Question 2

If you changed to your home directory, how can you return to `directory_to_delete`?

Answer Question 2

We need to give the "path" to that directory.

```
cd /data/$USER/Module_1/directory_to_delete
pwd
ls
```

Getting back to removing directories (`rmdir`)

Now that we have created some directories, let's use `rmdir` to remove one.

What do you see when you try to remove this directory?

```
rmdir directory_to_delete
```

What should we do? We need to remove the contents of a directory before we can remove the directory. Here's one safe option.

```
cd directory_to_delete
ls
rm myseq.txt
ls
cd ..
ls
rmdir directory_to_delete
```

Moving and renaming files and directories, all with one command (`mv`)

The `mv` command is a handy way to rename files if you've created them with a typo or decide to use a more descriptive name. For example:

```
cd ..
mv file2.txt README.txt
ls
```

We can also move this file into a different location using the same command.

```
mv README.txt Module_1
cd Module_1
ls
```

Be careful when moving files, a mistake in the command can yield unexpected results. The `-i` interactive option will help keep you safe.

For example:

```
mkdir dir1
touch dir1/hello.txt
touch hello.txt
mv -i dir1/hello.txt hello.txt
```

Copying files (`cp`)

This is similar to `mv` but will create an actual copy of a file. You will need to specify what you are copying (the source) and where you want to make the copy (the target).

For example, let's copy a file from the BTEP teaching materials to `Module_1`.

```
cp  /data/classes/BTEP/B4B_2025/Module_1/sample.fast* .
```

Remember, the `.` is a relative path shortcut denoting our current directory, so we are copying this into our current working directory.

`*` is a wildcard, matching zero or more characters including spaces. We are using this to copy two files that differ in the last letter of the file extension. More on these in Lesson 3.

We can also copy an entire directory using the recursive flag (`cp -r`). For example, let's copy the directory `Practice_Sessions` from the BTEP teaching materials to our current directory.

```
cp -r /data/classes/BTEP/B4B_2025/Module_1/Practice_Sessions .
```

Viewing file content

There are several ways to view files. We can use the `less` command to view the contents of a file like this.

```
less sample.fasta
```

You'll need to type `q` to get out of `less` and back to the command line. Before the `less` command was available, the `more` command was commonly used to look at file content. The `less` command has more options for scrolling through files, so it is now the preferred command.

Another command for reading files is `cat`, but this will print the contents in their entirety.

Help! (`man`)

Lastly, all Unix commands have a `man` or "manual" page that describes how to use them. If you need help remembering how to use the command `ls`, you would type:

```
man ls
```

To exit `man`, again use `q`.

There are quite a few flags/options that we can use with the `ls` command, and we can learn all about them on the man page. My favorite flags for `ls` are `-l` and `-h`. We will use flags often, and you won't get far in Unix without knowing about them. Try this:

```
cd
ls -lh
```

We have already seen these flags, but as a reminder...
`-h` when used with the `-l` option, use unit suffixes (Byte, Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte) in order to reduce the number of digits to three or less using base 2 for sizes.

-l (The lowercase letter "ell".) List in long format. (See below). If the output is to a terminal, a total sum for all the file sizes is output on a line before the long listing.

Additional Resources

Software Carpentry: The Unix Shell *(https://swcarpentry.github.io/shell-novice/01-intro/index.html)*

Help Session

Practice navigating the file system and creating files. Instructions are here.

Lesson 3: Useful Unix

Lesson 2 Review

- Directories are folders that can store files, other directories, links, executables, etc.
- The file system is hierarchical with the root directory (`/`) at the top.
- Commands useful for navigating our file system and handling files include:
    - `pwd` = print working directory
    - `ls` = list contents
    - `cd` = change directory
    - `mkdir`, `rmdir` = make directory; remove directory
    - `rm` = remove file
    - `touch` = create file
    - `nano` opens text editor
- absolute file paths include the entire location from the root of the file system
- relative file paths include the location from the working directory

Learning Objectives

In this lesson, we will continue to learn tips and concepts that facilitate working on the command line, including

- Flags and command options - making programs do what they do
- Wildcards (e.g., `*`)
- Tab complete for less typing
- Accessing user history with the "up" and "down" arrows on the keyboard
- `cat`, `head`, and `tail` for reading files
- Working with file content (input, output, and append)
- Combining commands with the pipe (`|`).
- Finding information in files with `grep`
- Performing repetitive actions with Unix (e.g., `for loop`)
- File Permissions

For this lesson, you will need to connect to Biowulf.

> **Reminder: How to Connect to Biowulf**
>
> For this lesson and the lessons that follow, we will use NIH HPC student accounts *(https://hpc.nih.gov/nih/student.html)* to connect to Biowulf.
>
> Open a terminal and type the following:

```
ssh username@biowulf.nih.gov
```

`username` = NIH/Biowulf login username. *Remember to use the student account username here.*

Type in your password at the prompt. **The cursor will not move as you type your password!**

Congrats! You are now connected to the login node. Your current working directory will be your home directory.

Getting Started

To begin this lesson, let's move to our data directory `/data/$USER`.

```
cd /data/$USER
```

Let's also grab some files that we will need for this lesson from BTEP Teaching Materials.

```
cp -R /data/classes/BTEP/B4B_2025/Module_1 .
```

Did you forget what `cp` does? See Lesson 2.

Navigate to your `Module_1` directory.

```
cd ./Module_1
```

Flags and command options - making programs do what they do

In Lesson 2, I introduced the idea of command options (flags). Command options allow us to change the behavior of a command. Command options, which ultimately allow you to modify command parameters, are extremely important to obtain expected results.

Let's return to `ls` for an example. Compare the output from these commands:

```
ls
ls -S
ls -lh
```

`ls -h` (when used with `-l` option, prints file sizes in a human readable format with the unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte, Terabyte. This reduces the number of digits displayed.)

`ls -l` (list in long format). The column output is as follows:

1. directory / file type
2. Content permissions
3. Number of hard links to content
4. Owner
5. Group owner
6. Content size (bytes)
7. Last modified date / time
8. File / directory name

`ls -S` (sort from largest to smallest file size)

What do you see when combining the `-h` and `-l` flags?

There are many flags you can use with `ls`. How would we find out what they are?

```
man ls
```

Or to see a more user friendly display, google to the rescue. Google "man ls unix" and see what you get. Here's *(https://shapeshed.com/unix-ls/)* a useful, readable explanation of the "ls" command with examples.

Try combining some of the `ls` flags.

```
ls -lhS
ls -alt
```

`-a` (show hidden dot files `.`)

`-t` (sort by modification time with most recent listed first)

Flags and options add a layer of complexity to unix commands but are necessary to get the command or program to behave in the way you expect. For example, here is a command line for running "blastn" an NCBI/BLAST application. The Basic Local Alignment Search Tool (BLAST) finds regions of local similarity between sequences. *(https://hpc.nih.gov/apps/Blast.html)*

```
module load blast # Try this out by first loading the blast module
blastn -db /fdb/blastdb/nt -query seq1.fasta -out results.out
```

What's going on in this command line? First, the BLAST algorithm is specified, in this case it is `blastn`, then the `-db` flag is used to choose the database to search against (`nt` for nucleotide). The `query` flag specifies the input sequence, which is in FASTA format, and the `-out` flag specifies the name of the output file. See more about using Blast on NIH Biowulf here *(https://hpc.nih.gov/apps/Blast.html)*.

> **Note**
>
> We are not running this today. This is just an example. If you want to run this, make sure you are not on the login node (Use `sinteractive`) and be sure to allocate enough memory. See the Biowulf Blast help docs *(https://hpc.nih.gov/apps/Blast.html)*.

Use of wildcards

Wildcard characters are a handy tool when working at the command line. Wildcards "are symbols or special characters that represent other characters." *(https://www.tecmint.com/use-wildcards-to-match-filenames-in-linux/)* Want to list all your FASTA files?

Use `*`:

```
ls *.fasta
```

The * wildcard matches zero or more characters including spaces.

This example will work as long as all your FASTA files end in `.fasta`. But sometimes they don't (e.g., fas or fa). Read more about FASTA file extensions here *(https://en.wikipedia.org/wiki/FASTA_format#FASTA_file)*.

For example, to find a file ending in `.fa`, you could use the following:

```
touch file.fa #First create a file ending in fa
ls *.fa
```

or you want all the the FASTA and FASTQ files.

```
ls *.fa*
```

In addition to the asterisk (*) character, you can use other wildcards on the unix command line, not limited to the following:
? - matches a single character
{} - used for multiple matches

`[]` - specify a range of characters or numbers. For example, `[a-z]` would specify all lower case letters and `[0-9]` would mean all digits.

To see some more practical examples of using wildcards, see this article *(https://www.tecmint.com/use-wildcards-to-match-filenames-in-linux/)* from tecmint.com and this *(https://medium.com/@leedowthwaite/advanced-wildcard-patterns-most-people-dont-know-52f7fd608cb3)* from the medium.com. This second article provides a nice discussion on how wildcards differ from regular expressions.

Using tab complete for less typing

Here's a good Unix trick to know - tab complete. Start typing the name of the file or directory you want, and hit the `tab` key. The system will auto-complete the name of the file or directory if the name is unique. It may only give a partial name if there is more than one file with a similar name in the directory. You can fill in the next part of the name and then try tab-complete again.

Let's create some files to test this.

```
touch file.txt
touch file.fasta
touch file.fastq
```

Start typing...

```
less f (hit tab)
```

What do you get? How does that differ from:

```
less file (hit tab)
```

The tab complete will save you lots of typing, and also help to figure out if you are where you think you are in the directory structure.

Access your history with the "up" and "down" arrows on your keyboard

Here's another Unix trick to make your life easier. Access previous commands with the up and down arrows on your keyboard. You can scroll backwards and forwards. This helps when you've got a typo or small mistake in your command lines that you can fix without retyping the whole thing.

The `history` command

You can also search, view, and retrieve recently using commands using `history`. See this guide *(https://www.geeksforgeeks.org/history-command-in-linux-with-examples/)*.

Keyboard shortcuts

There are also a few handy keyboard shortcuts to make life on the command line easier. For example:

`ctrl c` to kill a running process
`ctrl l` to clear the screen
`ctrl a` skip to the beginning of a command
`ctrl e` skip to the end of a line

See more examples here *(https://unix.stackexchange.com/questions/255707/what-are-the-keyboard-shortcuts-for-the-command-line)*.

`cat`, `head`, and `tail`

Who says Unix programmers don't have a sense of humor? Let me introduce `cat`, `head`, and `tail`. The `cat` command (short for "concatenate") is an extremely useful command for creating new files and viewing file content. You can use it to open files for reading input and writing output. Or you can use it to copy several files into a new file. Also you can "append" the contents of a file to the end of another file.

This command reads the content of `sample.fasta` and outputs to standard output (i.e., the screen). This is not helpful for very large files, as it moves to the end of the file quickly. `Less` is a better command option for reading large files.

```
cat sample.fasta
```

You can use `cat` to combine several files into one file, such as:

```
cat seq1.fasta seq2.fasta
```

Although, this again prints to standard output, the screen. To capture that output, we need to learn how to redirect output. (Coming up next!)

In the meantime, let's take a quick look at `head` and `tail`.

`head` - prints the first 10 lines of a specified file (by default)

```
head sample.fasta
```

`tail` - prints the last 10 lines of a specified file (by default)

```
tail sample.fasta
```

You can specify how many lines you would like to see (`-n`), or you can use the default value, which is 10.

```
head -n 20 sample.fasta
```

Want to be sure of those 20 lines? Let's use `cat -n` to show the file with numbered lines and compare.

```
cat -n sample.fasta
```

What if you didn't know what `-n` did? How could you find out more about "cat"?

```
man cat
```

Working with file content (input <, output >, and append >>)

By default, commands take input from the standard input (your keyboard) and send the results to standard output (your screen). If you want to redirect the output (results) of a command to file, you need to use output redirection. Similarly, we can also redirect input, for example, instead of coming from our keyboard, it can instead come from file. This is known as input redirection. Learn more here. *(https://www.geeksforgeeks.org/input-output-redirection-in-linux/)*

< - input redirection operator
> - output redirection operator
>> - append redirection operator

Want to put the output from `cat`, `head`, or `tail` into a new file?

```
head -n 20 seq1.fasta > smaller.fasta
```

Or we could put the last 20 lines into a file with `tail`.

```
tail -n 20 seq1.fasta > smaller2.fasta
```

What if we want the first 20 lines and the last 20 lines in one file, with the first at the top and the last at the bottom? Use append, `>>` to paste the second file to the bottom of the first file. Let's try it.

```
head -n 20 sample.fasta > smaller.fasta
tail -n 20 sample.fasta >> smaller.fasta
```

Keep in mind that if you input into the same file multiple times, you are overwriting the previous contents. For example, what is the final content of our file `covid.fasta`?

```
head -n 20 seq1.fasta > covid.fasta
head -n 20 seq2.fasta > covid.fasta
head -n 20 seq3.fasta > covid.fasta
```

How many lines are now in `covid.fasta`? How can you check?

Let's try `wc`, short for word count.

```
wc covid.fasta
```

`wc` is a very useful function. Without opening a file, we can find out how many lines, words and characters are in it. Line counts are extremely useful to assess your data output.

If we created a file where we were expecting there to be 1000 lines of output? The `wc` command provides a quick way to check.

What happened to all of our content? The final results are from "seq3.fasta" only. The other two results files have been overwritten.

So, how would you get all three files into `covid.fasta`? You'll need to use append.

```
cat seq1.fasta > covid.fasta
cat seq2.fasta >> covid.fasta
cat seq3.fasta >> covid.fasta
```

OR

```
cat seq1.fasta seq2.fasta seq3.fasta > covid2.fasta
```

How could you test to see if the file has the expected number of lines?

```
wc covid.fasta
wc covid2.fasta
```

To redirect input, we would use something like this:

```
less < covid.fasta
```

Yes, you could use `less covid.fasta`. However, these commands act differently. In `less covid.fasta`, the file is directly passed to the `less`. However, with `less < covid.fasta` the content of `covid.fasta` is passed to `less`.

Combining commands with pipe (|). Where the heck is pipe anyway?

The hardest thing about using pipe (|) is finding it on your keyboard!

The pipe symbol "|" (a.k.a., vertical bar) is way over on the right hand side of your keyboard, above the backslash \.

Pipe is used to take the output from one command, and use it as input for the next command.

For example,

```
head -n 20 sample.fasta | wc
```

Using what we've learned, all together now.

Let's say you've got a very large FASTA or FASTQ file, and you want to run an analysis on it. Before working on the whole file, it can be useful to set up a smaller test file instead.

Here's one way to do it.

```
cat sample.fasta | head -n 20 > output.fasta
```

This combines several things we have learned about. The `cat` command opens the file `sample.fasta` for writing. The pipe `|` command is used to take that output and run it through the `head` command where we only want to see the first 20 lines, and we want them output > into a file called "output.fasta".

> **Note**
>
> We could simply use `head` without `cat`, but we wanted to use the pipe for an example.

Let's compare the files. How are they different?

```
ls -lh
```

and

```
less sample.fasta
less output.fasta
```

Finding information in files with grep

The `grep` utility is used to search files looking for a pattern match. It is used like this.

```
grep pattern options filename
```

As our first example we will look for restriction enzyme (EcoRI) sites in a sequence file (eco.fasta). The file has four EcoRI sites, but two of them are across the end of the line (and won't be found).

```
cat eco.fasta
grep -n GAATTC eco.fasta
```

We can modify the `eco.fasta` file to remove the line breaks (`\n`) at the ends of the lines using `tr`.

```
grep -v ">" eco.fasta | tr -d "\n" | grep GAATTC
```

`-v` : This prints out all the lines that do not match the pattern, effectively removing the header line (>) of the file.

The unix `tr` (translate) command is used for translating or deleting characters.

Usage:

```
tr [option] set1 [set2]
```

`-d` : Deletes characters in the first set from the output

So this part of the command line is finding the line breaks `\n` and removing them.

And now we can see all four of the EcoRI sites.

What if we just wanted to count the occurrence of the EcoRI sites in the sequence?

```
grep -v ">" eco.fasta | tr -d "\n" | grep GAATTC -c
```

Perhaps "1" is not what you expected. Check out `man grep`.

The `-c` option prints only a count of the lines that match a pattern. Here we are counting the entire line as one.

If we wanted to see each of the EcoRI sites listed.

```
grep -v ">" eco.fasta | tr -d "\n" | grep GAATTC -o
```

And if we want to count the number of EcoRI sites.

```
grep -v ">" eco.fasta | tr -d "\n" | grep GAATTC -o | wc
```

Let's create a word file that we can input to `grep`. We can input multiple restriction enzyme sites and search for all of them.

```
nano wordfile.txt
```

Put in the words (`GAATTC`, `TTTTT`). Now we can use that file to find lines.

```
grep -v ">" eco.fasta | tr -d "\n" | grep -f wordfile.txt -o | wc
```

We can find a list of options to be used with `grep` using `man`:

```
man grep
```

Performing repetitive actions with Unix

We can create a "for" loop to do iterative actions in Unix.

For each commands all on one line or separate lines: (`i` can be any variable name). These steps can be saved as a file, thereby creating a simple Unix script.

What does this "for loop" do?

```
for i in *.fasta; do ls $i; done
```

What do these command lines do?

```
for i in *.fasta; do echo $i; grep ">" $i; done
```

This one pulls out all the header ">" lines in the fasta files.

```
for i in seq*.fasta; do echo $i; grep ">" $i; done
```

While this one pulls out the header lines from files named seq*.fasta.

Learn more about for loops here *(https://www.geeksforgeeks.org/bash-scripting-for-loop/)*.

Permissions - when all else fails check the permissions

Permissions dictate who can access your files and directories, and what actions they can perform. If you are consistently getting error messages from your command line and you're sure you are typing it correctly, it's worthwhile checking the permissions. So what does it all mean? How do we read the permissions information?

As we have seen, `ls -l`, provides information about file types, the owner of the file, and other permissions.

For example:

```
ls -l sample.fasta
```

Here is an overview of what these permissions mean:

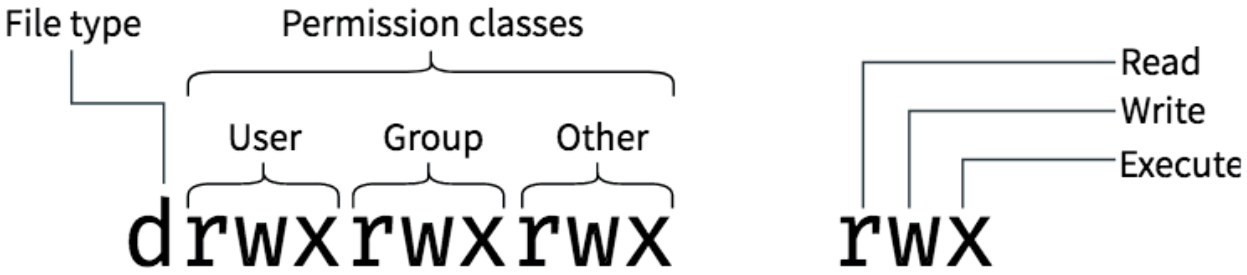


Image from booleanworld.com (https://www.booleanworld.com/introduction-linux-file-permissions/)

The first letter `d` indicates whether this is a directory or not `-` or some other special file type. The next 3 positions are the owner's/user's permissions. In this image, the owner can "read", "write" and "execute". So they can create files and directories here, read files here, and execute/run programs. The next 3 positions show the permissions for the "group". The last 3 positions shows permissions for everyone ("other").

You can modify permissions using `chmod`. Let's see this in action.

```
touch example.txt
chmod u-w example.txt
ls -l example.txt
```

Now, reassign write privilege to the user/owner

```
chmod u+w example.txt
ls -l example.txt
```

Help Session

Let's complete a Unix treasure hunt.

Lesson 4: Working on Biowulf

Lesson 3 Review

- Flags and command options
- Wildcards (`*`)
- Tab complete
- Accessing user history with the "up" and "down" arrows
- `cat`, `head`, and `tail`
- Working with file content (input, output, and append)
- Combining commands with the pipe (`|`)
- `grep`
- `for loop`
- File Permissions

Lesson Objectives

- Learn about the slurm system by working on Biowulf. Learn about batch jobs, swarms jobs, interactive sessions.
- Retrieve data from NCBI through a batch job.
- Learn how to troubleshoot failed jobs.

For this lesson, you will need to connect to Biowulf.

Working on Biowulf

Now that we are becoming more proficient at the command line, we can use these skills to do more than navigate our file system. We can actually begin working on Biowulf. Today's lesson will focus on submitting computational jobs on the Biowulf compute nodes.

Login using `ssh`

To get started, as always, we will need to log in to Biowulf. *Make sure you are on VPN.*

Open your `Terminal` if you are using a mac or the `Command prompt` if you are using a Windows machine.

```
ssh username@biowulf.nih.gov
```

`username` = NIH/Biowulf login username. *Remember to use the student account username here.*

Type in your password at the prompt. **The cursor will not move as you type your password!**

When you log in to Biowulf, you are automatically in your home directory (`/home/$USER`). This directory is very small and not suitable for large data files or analysis.

Use the `cd` command to change to your data directory (`/data`) .

```
cd /data/$USER
```

where `$USER` is an environment variable holding your username.

If you do not yet have one, create a directory to work in for this lesson and move to that directory.

```
mkdir Module_1
cd Module_1
```

Note

When working on Biowulf, you can not use any computational tools on the "login node". Instead, you need to work on a node or nodes that are sufficient for what you are doing.

To run jobs on Biowulf, you must designate them as interactive, batch or swarm. Failure to do this may result in a temporary account lockout.

What kind of work can we do on the login node?

The login node can be used for:

- Submitting resource intensive tasks as jobs
- Editing and compiling code
- File management and data transfers on a small scale

Batch Jobs

Most jobs on Biowulf should be run as batch jobs using the "sbatch" command.

```
sbatch yourscript.sh
```

Where `yourscript.sh` is a shell script containing the job commands including input, output, cpus-per-task, and other steps. Batch scripts always start with `#!/bin/bash` or similar call. The sha-bang (`#!`) tells the computer what command interpreter to use, in this case the Bourne-again shell.

For example, to submit a job checking sequence quality using `fastqc` (MORE ON THIS LATER), you may create a script named `fastqc.sh`:

```
nano fastqc.sh
```

Inside the script, you may include something like this:

```
#!/bin/bash

module load fastqc
fastqc -o output_dir -f fastq seqfile1 seqfile2 ... seqfileN
```

where `-o` names the output directory

`-f` states the format of the input file(s)

and `seqfile1 ... seqfileN` are the names of the sequence files.

> **Note**
>
> `fastqc` is a available via Biowulf's module system, and so prior to running the command, the module had to be loaded.

For more information on running batch jobs on Biowulf, please see: https://hpc.nih.gov/docs/userguide.html *(https://hpc.nih.gov/docs/userguide.html)*.

Multi-threaded jobs and `sbatch` options

In high-performance computing, multithreading lets a single program split itself into multiple "workers" that can run at the same time on different parts of the computer's brain (CPUs), speeding up complex tasks. Many bioinformatics programs use multi-threading.

For multi-threaded jobs, you will need to set `--cpus-per-task`. You can do this at the command line or from within your script.

Example at the command line:

```
sbatch --cpus-per-task=# yourscript.sh
```

In your script:

```
#!/bin/bash
#SBATCH --job-name qc
#SBATCH --mail-type BEGIN,END
#SBATCH --cpus-per-task #

module load fastqc
fastqc -o output_dir -t $SLURM_CPUS_PER_TASK -f fastq seqfile1 seqfi
```

Within the script we can use **directives** denoted by `#SBATCH` to support command line arguments such as `--cpus-per-task`. If included within the script, you will not need to call these at the command line when submitting the job. You should also pass the environment variable, `$SLURM_CPUS_PER_TASK` to the thread argument. Some other useful directives include `--job-name`, where you assign a name to the submitted job, and `--mail-type`, which you can use to direct `slurm` to send you an email when a job begins, ends, or both.

> **Tip**
>
> **The jobscript should always be the last argument of** `sbatch`**.**

To see more `sbatch` options, use

```
sbatch --help
```

Some slurm commands

Once you submit a job, you will need to interact with the `slurm` system to manage or view details about submitted jobs.

Here are some useful commands for this purpose:

Command	Purpose	Pros & Cons
squeue	Information about jobs	Lots of options, odd syntax
scancel	Delete job(s)	Only way to cancel your jobs!
sacct	Job accounting for completed jobs	Useful to get a list of jobs you submitted recently (default is 'today'), odd syntax
sjobs (Biowulf utility)	Info about running and pending jobs	Informative but slow

Courtesy of NIH HPC Team Training Documentation

Standard error and standard output

Output that you would expect to appear in the terminal (e.g., standard error and standard output) will not in batch mode. Rather, these will be written by default to `slurm######.out` in the submitting directory, where `######` represents the job id. These can be redirected using `--output=/path/to/dir/filename` and `--error=/path/to/dir/filename` on the command line or as an `#SBATCH` directive.

Partitions

Your job may be in a waiting phase ("Pending" or "PD") depending on available resources. You can specify a particular node partition using `--partition`.

Use `freen` to see what's available.

Summary of partitions

Nodes available to all users	
norm	the default partition. Restricted to single-node jobs
multinode	Intended to be used for large-scale parallel jobs. Single node jobs are not allowed. See here for detailed information.
largemem	Large memory nodes. Reserved for jobs with memory requirements that cannot fit on the norm partition. Jobs in the largemem partition must request a memory allocation of at least 350GB.
unlimited	Reserved for jobs that require more than the default 10-day walltime. Note that this is a small partition with a low CPUs-per-user limit. Only jobs that absolutely require more than 10 days runtime, that cannot be split into shorter subjobs, or that are a first-time run where the walltime is unknown, should be run on this partition.
quick	For jobs < 4 hours long. These jobs are scheduled at higher priority. They may run on the dedicated quick partition nodes, or on the buy-in nodes when they are free.
gpu	GPU nodes reserved for applications that are built for GPUs.
visual	Small number of GPU nodes reserved for jobs that require hardware accelerated graphics for data visualization.
Buy-in nodes	
ccr*	for NCI CCR users
forgo	for individual groups from NHLBI and NINDS
persist	for NIMH users

Student partition

The student accounts have their own partition for running jobs, `--partition=student`. This will need to be included on the command line or in the job script.

Walltime

The default walltime, or amount of time allocated to a job, is 2 hours on the `norm` partition. To change the walltime, use `--time=d-hh:mm:ss`.

Here are the walltimes by partition:

```
batchlim
```

	Walltime	
Partition	Default	Maximum
norm	02:00:00	10-00:00:00
multinode	08:00:00	10-00:00:00
+turbo		08:00:00
interactive	08:00:00	1-12:00:00
gpu	02:00:00	10-00:00:00
unlimited	UNLIMITED	UNLIMITED
largemem	02:00:00	10-00:00:00
visual	08:00:00	1-12:00:00
quick	02:00:00	04:00:00
persist (NIMH)	UNLIMITED	UNLIMITED
ccr (NCI_CCR)	04:00:00	10-00:00:00
forgo (lab only)	1-00:00:00	3-00:00:00

You can change the walltime after submitting a job using

```
newwall --jobid <job_id> --time <new_time_spec>
```

Submit an actual job

Let's submit a batch job. We are going to download data from the **Sequence Read Archive (SRA)**, a public repository of high throughput, short read sequencing data. We will discuss the SRA a bit more in detail in Lesson 5. For now, our goal is to simply download multiple fastq files associated with a specific BioProject on the SRA. We are interested in downloading RNAseq files associated with **BioProject PRJNA578488**, which "aimed to determine the genetic and molecular factors that dictate resistance to WNT-targeted therapy in intestinal tumor cells".

We will learn how to pull the files we are interested in directly from SRA at a later date. For now, we will use the run information stored in `sra_files_PRJNA578488.txt`.

Let's copy this file to our working directory and view using `less`.

```
cp /data/classes/BTEP/B4B_2025/Module_1/sra_files_PRJNA578488.txt .
less sra_files_PRJNA578488.txt
```

Now, let's build a script downloading a single run, `SRR10314042`, to a directory called `/data/$USER/testscript`.

```
mkdir testscript
cd testscript
```

Open the text editor `nano` and create a script named `filedownload.sh`.

```
nano filedownload.sh
```

Inside our script we will type

```
#!/bin/bash
#SBATCH --cpus-per-task=6
#SBATCH --gres=lscratch:10
#SBATCH --partition=student

#load module
module load sratoolkit

fasterq-dump -t /lscratch/$SLURM_JOB_ID SRR10314042
```

Notice our use of sbatch directives inside the script.

`fasterq-dump` assigns 6 threads by default, so we are specifying `--cpus-per-task=6`. This can be modified once we get an idea of the CPUs needed for the job. `-t` assigns the location to be used for temporary files. Here, we are using `/lscratch/$SLURM_JOB_ID`, which is created upon job submission. In combination, we need to request local scratch space allocation, which we are setting to 10 GB (`--gres=lscratch:10`). The final directive, `--partition=student` is required because we are using student accounts for this lesson. If you are with the Center for Cancer Research and using your own account, you can use `--partition=ccr`.

Remember:
Default compute allocation = 1 physical core = 2 CPUs
Default Memory Per CPU = 2 GB. Therefore, default memory allocation = 4 GB

Now, let's run the script.

```
sbatch filedownload.sh
```

Let's check our job status.

```
squeue -u $USER
```

Once the job status changes from `PD` (pending) to `R` (running), let's check the job status.

```
sjobs -u $USER
```

Some other useful job monitoring commands include `jobload`, and `jobhist`. The latter is useful for when the job completes.

When the job completes, we should have a file called `SRR10314042.fastq`.

```
ls -lth
```

Now, what if we want to download all of the runs from `sra_files_PRJNA578488.txt`. We could use a for loop, `GNU parallel`, which acts similarly to a for loop (MORE ON THIS NEXT LESSON), or we can submit multiple `fasterq-dump` jobs in a job array, one subjob per run accession.

> **Note**
>
> **There are instructions for running SRA-Toolkit on Biowulf here *(https://hpc.nih.gov/apps/sratoolkit.html)*.**

Swarm-ing on Biowulf

`Swarm` is for running a group of commands (job array) on Biowulf. `swarm` reads a list of command lines and automatically submits them to the system as sub jobs. "By default, swarm runs one command per core on a node, making optimum use of a node. Thus, a node with 16 cores will run 16 commands in parallel." *(https://hpc.nih.gov/apps/swarm.html)*. To create a `swarm` file, you can use `nano` or another text editor and put all of your command lines in a file called `file.swarm` (`file` is just a placeholder). Then you will use the `swarm` command to execute.

For example,

```
$ swarm -f file.swarm
```

Note

By default, each subjob is allocated 1.5 gb of memory and 1 core (consisting of 2 cpus) --- hpc.nih.gov *(https://hpc.nih.gov/apps/swarm.html)*

Swarm creates two output files for each command line, one each for STDOUT (file.o) and STDERR (file.e). You can look into these files with the `less` command to see any important messages.

For example,

```
$ less swarm_jobid_subjobid.o
$ less swarm_jobid_subjobid.e
```

View the `swarm` options using

```
swarm --help
```

For more information on swarm-ing on Biowulf, please see: https://hpc.nih.gov/apps/swarm.html *(https://hpc.nih.gov/apps/swarm.html)*

Let's create a swarm job

To retrieve all the files at once, you can create a swarm job.

```
nano set.swarm
```

Copy the following and paste into the swarm file.

```
#SWARM --threads-per-process 6
#SWARM --gb-per-process 4
#SWARM --gres=lscratch:20
#SWARM --module sratoolkit
#SWARM --partition=student

fasterq-dump -v -t /lscratch/$SLURM_JOB_ID SR10314043 #Add error to s
fasterq-dump -t /lscratch/$SLURM_JOB_ID SRR10314044
fasterq-dump -t /lscratch/$SLURM_JOB_ID SRR10314045
```

Here, based on our swarm directives, each subjob will request 6 cpus and 4 GB of RAM. We are also denoting local scratch space of 20 GB per each command.

There is advice for generating a swarm file using a `for loop` and `echo` in the swarm user guide *(https://hpc.nih.gov/apps/swarm.html)*. If you can help it, **do not type each of these commands.**

Something like this could generate the lines above:

```
cat ../sra_files_PRJNA578488.txt | while read line; do
    echo 'fasterq-dump -v -t /lscratch/$SLURM_JOB_ID' $line >> scrip
done
```

Fix the code

You may have noticed that the `while` loop above does not produce the same lines. The order of the files is in reverse, and there is an extra line of code.

```
fasterq-dump -v -t /lscratch/$SLURM_JOB_ID SRR10314045
fasterq-dump -v -t /lscratch/$SLURM_JOB_ID SRR10314044
fasterq-dump -v -t /lscratch/$SLURM_JOB_ID SRR10314043
fasterq-dump -v -t /lscratch/$SLURM_JOB_ID
```

How can we fix the lines written to script.sh? Use the skills you have learned and a little help from google.

Possible Solution

One solution is to sort the script using `sort` and drop the first line of the file using `tail -n +2`.

```
sort script.sh | tail -n +2 > script_mod.sh
```

Let's run our swarm file. Because we included our directives within the swarm file, the only option we need to include is `-f` for file.

```
swarm -f set.swarm
```

Running Interactive Jobs

Interactive nodes are suitable for testing/debugging cpu-intensive code, pre/post-processing of data, graphical application, and to GUI interface to application.

To start an interactive node, type `sinteractive` at the command line and press Enter/Return on your keyboard.

```
sinteractive
```

You will see something like this printed to your screen. You only need to use the `sinteractive` command once per session. If you try to start an interactive node on top of another interactive node, you will get a message asking why you want to start another node.

```
[username@biowulf ]$ sinteractive
salloc.exe: Pending job allocation 34516111
salloc.exe: job 34516111 queued and waiting for resources
salloc.exe: job 34516111 has been allocated resources
salloc.exe: Granted job allocation 34516111
salloc.exe: Waiting for resource configuration
salloc.exe: Nodes cn3317 are ready for job
srun: error: x11: no local DISPLAY defined, skipping
[username@cn3317 ]$
```

You can use many of the same options for `sinteractive` as you can with `sbatch`. *(https://hpc.nih.gov/docs/userguide.html#int)* **The default sinteractive allocation is 1 core (2 CPUs) and 3 GB (1.5 GB per CPU) of memory and a walltime of 8 hours**.

For example,

```
sinteractive  --gres=lscratch:20  --cpus-per-task=6
module load sratoolkit
fasterq-dump -t /lscratch/$SLURM_JOBID SRR2048331 -O /data/$USER/sra
```

To terminate / cancel the interactive session use

```
exit
```

Exiting from Biowulf

To disconnect the remote connection on Biowulf, use

```
exit
```

Transferring files to Biowulf

Before ending this lesson, I want to briefly discuss file transfers. At some point you will need to know how to get your data on Biowulf. The NIH HPC Team has fantastic documentation on this subject at https://hpc.nih.gov/docs/transfer.html *(https://hpc.nih.gov/docs/transfer.html)*.

Remember, Helix (helix.nih.gov) is the interactive data transfer and file management node for the NIH HPC Systems. If you are interactively transferring files to and from NIH HPC directories, you should connect to Helix:

```
ssh username@helix.nih.gov
```

where `username` is your Biowulf username.

Recommended methods for file transfers

Large scale transfers

For large scale transfers and large files use, Globus. Find instructions here. *(https://hpc.nih.gov/docs/globus/)*

Small scale transfers

For small scale transfers, use `scp` to or from Helix or drag and drop files by mounting HPC Systems directories to your local machine.

For example, to copy `Module_1` to your local machine, use:

```
scp -r username@helix.nih.gov:/data/username/Module_1 .
```

> **Note**
>
> This command and the next are issued from your local computer. The `.` in the above command means that you are copying `Module_1` into your current working directory.

To copy from your local machine to Biowulf, use:

```
scp -r ./Module_1 username@helix.nih.gov:/data/username/Module_1
```

Make sure to substitute `username` *with your Biowulf username.*

To mount the HPC directories to your local computer, follow the instructions here. *(https://hpc.nih.gov/docs/hpcdrive.html)*

Help Session

Practice the skills learned in this lesson here.

So you think you know Biowulf?

Quiz yourself using the hpc.nih.gov biowulf-quiz *(https://hpc.nih.gov/training/intro_biowulf/biowulf-quiz/)*.

Lesson 5: Downloading data from the SRA

Lesson 4 Review:

- The majority of computational tasks on Biowulf should be submitted as jobs: `sbatch` or `swarm`
- the SRA-toolkit can be used to retrieve data from the Sequence Read Archive

Learning Objectives:

1. Download data from the SRA with `fastq-dump`
    - split files into forward and reverse reads
    - download part, not all, of the data
2. Compare `fastq-dump` to `fasterq-dump`
3. Introduce `prefetch`
4. Grab SRA run info and run accession information
5. Learn to automate data retrieval with the `parallel` command
6. Learn about alternative download options

Get Started

Connect to Helix

Helix is the dedicated data management and file transfer node.

```
ssh username@helix.nih.gov
```

`username` = NIH/Biowulf login username. *Remember to use the student account username here.*

Type in your password at the prompt. **The cursor will not move as you type your password!**

Congrats! You are now connected to Helix. Your current working directory will be your home directory.

Navigate to your Module_1 directory

```

```

```
cd /data/$USER/Module_1
```

If you do not have a Module_1 directory, use the following:

```
cp -R /data/classes/BTEP/B4B_2025/Module_1 /data/$USER
cd /data/$USER/Module_1
```

Create a lesson directory

Here, we will download data from the NCBI/SRA. Let's create a directory named `sra_data` to store the data we are using today.

```
mkdir sra_data
cd sra_data
```

A brief word about sequence formats

- FASTA – commonly used text format for downstream analysis such as sequence similarity searches

```
head ../sample.fasta
```

```
>NC_012920.1 Homo sapiens mitochondrion, complete genome
GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCATGCATTTGGTATTTTCGTCTGGG
GTATGCACGCGATAGCATTGCGAGACGCTGGAGCCGGAGCACCCTATGTCGCAGTATCTGTCTTTGAT
CTGCCTCATCCTATTATTTATCGCACCTACGTTCAATATTACAGGCGAACATACTTACTAAAGTGTGT
ATTAATTAATGCTTGTAGGACATAATAATAACAATTGAATGTCTGCACAGCCACTTTCCACACAGACA
ATAACAAAAAATTTCCACCAAACCCCCCCTCCCCCGCTTCTGGCCACAGCACTTAAACACATCTCTGC
AACCCCAAAAACAAAGAACCCTAACACCAGCCTAACCAGATTTCAAATTTTATCTTTTGGCGGTATGC
TTTTAACAGTCACCCCCCAACTAACACATTATTTTCCCCTCCCACTCCCATACTACTAATCTCATCAA
CAACCCCCGCCCATCCTACCCAGCACACACACACCGCTGCTAACCCCATACCCCGAACCAACCAAACC
AAAGACACCCCCCACAGTTTATGTAGCTTACCTCCTCAAAGCAATACACTGAAAATGTTTAGACGGGC
```

- FASTQ – output format from NGS technologies with quality scores

```
head ../sample.fastq
```

```
@SRR1553607.1 1 length=202
GTTAGCGTTGTTGATCGCGACGCAACAACTGGTAAAGAATCTGGAAGAAGGATATCAGTTCAAACGCTC
AGCGAGATGATGGATATTTTTGAACGACTCATCATTCGTCACATCCCGTTCAACCTGATTGAGTCGTTC
AAAATATCCATCATCTCGCTTGAGCGTTTGAACTGATATCCTTCTTCCAGATTCTTTACCAG
+SRR1553607.1 1 length=202
BB@FFFFFHHHHHJJJJJJJJJJJJJJJJJJGHIJJJJJJJJJHHHHHFFFFFEEEEEEEEDDDDDI
DDDDDDDDDDEDDDDEDEEEDDDDDDDDDDCCCFFFFFHHHHHJJJJJJJJJJJJJJJJJJJJJJ.
JJJJJJJJJJJJJJIJJJIJJHIJJHHHFFFFFEDDEFEEEDDDDDDDDDDDDDEDDDDDD
@SRR1553607.2 2 length=202
GGTGTAAGCACAGTACTCGGCCCACATCGCCTTTGTGTTAATGAAGTTTGGGTATCAACTTTCATCCCC
ATCTTCCGTGGAAGGAGTATGTTCCGTCAATCCGTACATGTTAACCCTGGGCCTGTTACTTGGGCCTTC
CGGGTTCTAATACAAATTGACGGAACATACTCCTTCCACGGAAGATTGGGGATGAAAGTTGA
+SRR1553607.2 2 length=202
?@;BDDDDFHFHFFFGIIGHIIJJGJIGIJIIIIGDGGGHEIGJIIIGIIHJ5@FGHJJIEGGEEHHFI
FFFEEDEDCBB?CCDDD CDDDDECDCB@>C@@CFFFDFHFHHHJJJJJJIJJJJIJJJGJJIJIJJJ.
JJJJGHIJJIIIIEIJIJJIJIJGHHFFFFFFEEEDEEDDDDDDBDDDDDDDDDDDDDDDDD
@SRR1553607.3 3 length=202
GGTCATCGGCGGTCTCTGGGTCGCCGGGACCGTTGGAGAAGAAGACTCCGTCCACGCCGAGTGCCTGGA
CTCCTCGAAGGTGACCGTCGCGGGCAGCACGACCCTGCACCTGCGCGACAAGGGCGCGATGCGGATGGC
ATCTTCTCCGGACCCGCCGCTGAGGCGAGCGAGGCCGAGCTGCTCGAAACGGTCCGCTCCGC
```

What is the SRA?

The SRA (Sequence Read Archive) at NCBI is a large, public database of DNA sequencing data. The repository holds "short reads" generated by high-throughput next-generation sequencing, usually less than 1,000 bp.

We will download data from the SRA using the command line package `sratoolkit`. We will also explore how you can download these data from NCBI via a web browser.

Helpful documentation for the SRA:
1. SRA handbook at NCBI *(https://www.ncbi.nlm.nih.gov/books/NBK47528/)*
2. SRA Toolkit Documentation (NCBI) *(https://github.com/ncbi/sra-tools/wiki)*

SRA terms to know

BioProject (Prefix: PRJNA, example: PRJNA257197 *(https://www.ncbi.nlm.nih.gov/bioproject/PRJNA257197/)*): provides a description of the research project associated with the sequencing study. It includes information such as the study's abstract, links to publications (if available), and links to project data, including experiments, PubMed, and GEO datasets.

BioSample (Prefix: SAMN): contains sample-specific details. It could include details such as disease information, sex, response to drug, race, and other relevant attributes.

Sample Accession (Prefix: SRS): A Sample is an object imported from BioSample, which contains metadata describing the physical sample upon which a sequencing experiment was performed.
Experiment (Prefix: SRX): an object that contains metadata describing the library preparation, the sequencing technique, instrument used, and other experiment-specific details.
Run (Prefix: SRR): A Run is an object that contains the actual sequencing data for a particular sequencing experiment. Experiments may have multiple Runs. -- Decoding SRA Accessions: A Primer on Navigating NCBI's Sequence Read Archive *(https://gauravsh.com/decoding_sra_accessions/#:~:text=SRA%20Experiment%20Accession%20(SRX%23)%3A,physical%20sample%20imported%20*

Using `fastq-dump`

`fastq-dump` and `fasterq-dump` are modules in the SRA Toolkit and can be used to download FASTQ-formatted data. Both download the data in SRA format and convert it to FASTQ format.

If working on a high performance computing system such as Biowulf, either submit a script to cluster, use an interactive session, or connect to helix. Make sure to do `module load sratoolkit` prior to issuing either the `fastq-dump` or `fasterq-dump` commands. Alternatively, install the SRA Tool kit on local computer (see https://github.com/ncbi/sra-tools/wiki/01.-Downloading-SRA-Toolkit *(https://github.com/ncbi/sra-tools/wiki/01.-Downloading-SRA-Toolkit)*).

```
module load sratoolkit
```

```
fastq-dump SRR1553607
```

creates the file:

```
SRR1553607.fastq
```

Check the file to make sure it is in fastq format. How would you do this? What is FASTQ format?

What are "spots"?

Spots are a legacy term referring to locations on the flow cell for Illumina sequencers. All of the bases for a single location constitute the spot including technical reads (e.g., adapters, primers, barcodes, etc.) and biological reads (forward, reverse). In general, you can think of a spot as you do a read. For more information on spots, see the linked discussion *(https://*

www.biostars.org/p/12047/) on Biostars. When downloading "spots", always split the spots into the original files using:

```
--split-files
```

For paired-end reads, we need to separate the data into two different files like this:

```
fastq-dump --split-files SRR1553607
```

which creates

```
SRR1553607_1.fastq
SRR1553607_2.fastq
```

Split 3

There is an additional option `--split-3` that will split the reads into forward and reverse files and a third file with unmatched reads. Since many bioinformatic programs require matching paired end reads, **this is the preferred option. **

`fastq-dump` first downloads the data in SRA format, then converts it to FASTQ. If we want to work with a subset of the data, for example the first 10,000 reads, we can use `-X`:

```
fastq-dump --split-3 -X 10000 SRR1553607 --outdir SRR1553607_subset
```

We have additionally included `--outdir` to save our sub-setted data to a different directory, so that we do not overwrite our previous downloads.

To see other `fastq-dump` options, use

```
fastq-dump --help
```

Other options of interest:
`--gzip` *or* `bzip2` *allows you to compress the reads*
`--skip-technical` allows you to only download biological reads

To generate an XML report on the data that shows us the "spot" or read count, and the count of bases "base_count", including the size of the data file:

```
sra-stat --xml --quick SRR1553607
```

which yields this data in XML format:

```
<Run accession="SRR1553607" spot_count="203445" base_count="41095890"
  <Member member_name="A" spot_count="203445" base_count="41095890" b
  <Size value="25452196" units="bytes"/>
  <AlignInfo>
  </AlignInfo>
  <QualityCount>
    <Quality value="2" count="3523771"/>
    <Quality value="5" count="40006"/>
    <Quality value="6" count="30160"/>
    <Quality value="7" count="79961"/>
    <Quality value="8" count="80987"/>
    <Quality value="9" count="21904"/>
    <Quality value="10" count="63700"/>
    <Quality value="11" count="23695"/>
    <Quality value="12" count="42479"/>
    <Quality value="13" count="24910"/>
    <Quality value="14" count="22036"/>
    <Quality value="15" count="135747"/>
    <Quality value="16" count="56174"/>
    <Quality value="17" count="81514"/>
    <Quality value="18" count="123393"/>
    <Quality value="19" count="79256"/>
    <Quality value="20" count="269807"/>
    <Quality value="21" count="68295"/>
    <Quality value="22" count="125196"/>
    <Quality value="23" count="267733"/>
    <Quality value="24" count="287688"/>
    <Quality value="25" count="296721"/>
    <Quality value="26" count="338062"/>
    <Quality value="27" count="507890"/>
    <Quality value="28" count="226423"/>
    <Quality value="29" count="569612"/>
    <Quality value="30" count="813014"/>
    <Quality value="31" count="1309505"/>
    <Quality value="32" count="812951"/>
    <Quality value="33" count="2398417"/>
    <Quality value="34" count="2324453"/>
    <Quality value="35" count="10399039"/>
    <Quality value="36" count="1250313"/>
    <Quality value="37" count="2681177"/>
    <Quality value="38" count="1210457"/>
    <Quality value="39" count="2744683"/>
    <Quality value="40" count="2268539"/>
    <Quality value="41" count="5496222"/>
```

```
  </QualityCount>
  <Databases>
    <Database>
      <Table name="SEQUENCE">
        <Statistics source="meta">
          <Rows count="203445"/>
          <Elements count="41095890"/>
        </Statistics>
      </Table>
    </Database>
  </Databases>
</Run>
```

where we can see the spot count, the base count, and the number of bases corresponding to quality scores.

```
spot_count="203445" base_count="41095890"
```

Side bar: (XML) is Extensible Markup Language, a document format that is both human and machine-readable. XML aims for: "simplicity, generality and usability across the Internet".

Using `seqkit stat`

This tool cannot be used on Helix.

We can get similar information from our downloaded files using a program called `seqkit` *(https://bioinf.shenwei.me/seqkit/)*. If you get a chance, grab an interactive session on Biowulf and check out seqkit.

```
module load seqkit
```

First, let's see what options are available?

```
seqkit --help
```

We can see that `seqkit stats` provides "simple statistics of FASTA/Q files". Let's try it out.

```
seqkit stat SRR1553607_1.fastq
```

`fasterq-dump`

We have already seen `fasterq-dump` in action (See Lesson 4). `fasterq-dump` is faster than `fastq-dump` because it uses multi-threading (default `--threads 6`). `--split-3` and `--`

`skip-technical` are defaults with `fasterq-dump`, so you do not need to worry about specifying how you want the files to be split.

However, you can not grab a subset of the file like you can with `fastq-dump` nor can you compress the file during download, so `fasterq-dump` is not necessarily a replacement for `fastq-dump`.

Using `prefetch`

Both `fastq-dump` and `fasterq-dump` are faster when following `prefetch` *(https://github.com/ncbi/sra-tools/wiki/08.-prefetch-and-fasterq-dump)*, and `fasterq-dump` paired with `prefetch` is the fastest way to pull the files from the SRA. There are alternative options, which we will mention later.

`prefetch` will first download all of the necessary files to your working directory. Runs are downloaded in the SRA format (compressed).

```
mkdir prefetch
cd prefetch
prefetch SRR1553607
ls -l
```

`fasterq-dump` will then convert the files to the `fastq` format.

```
fasterq-dump  -p   -t /scratch/$USER  SRR1553607
```

What does `-p` do? How can we find out?

> **Note**
>
> We are not on a computational node, so we will use `scratch` instead of `lscratch`.

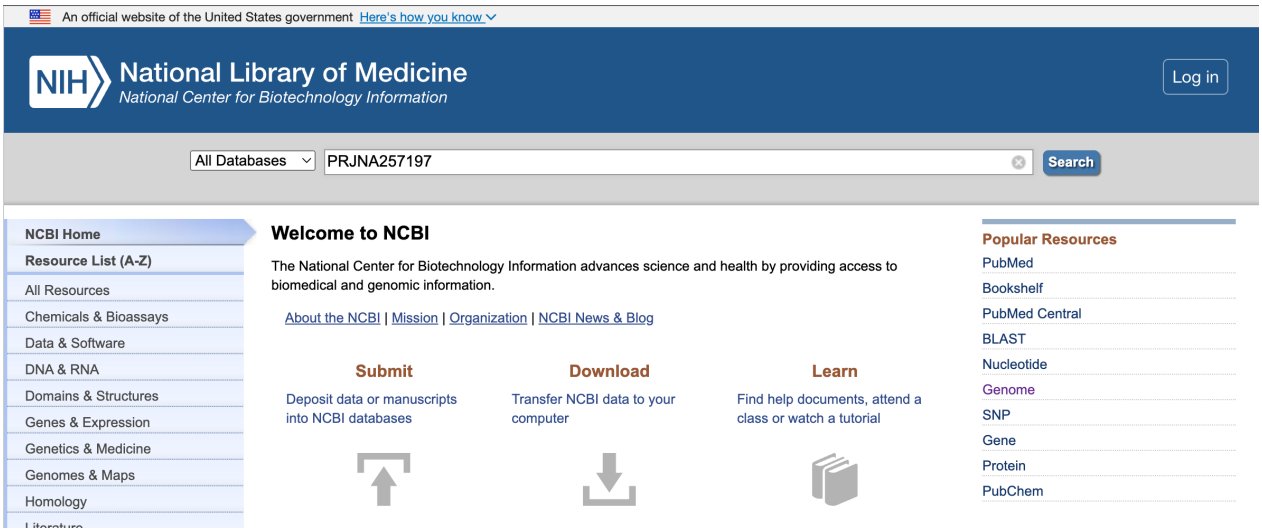
Navigating NCBI SRA

Where can we get an accession list or run information for a particular project?

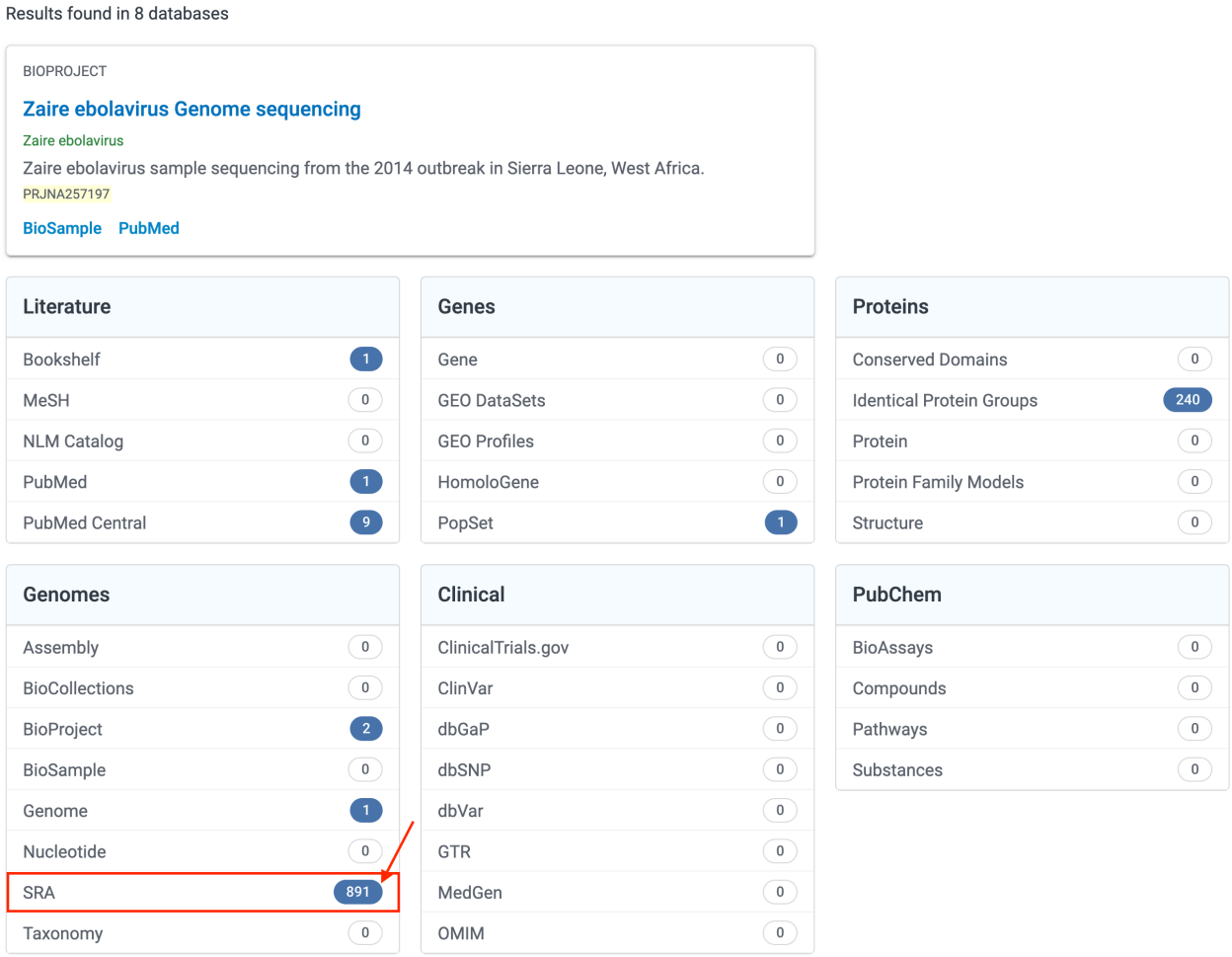
Navigating the NCBI website can be challenging. From a user perspective, nothing is as straight forward as you would expect. For the sequence read archive (SRA), there are fortunately some options. There are the convuluted e-utilites, which can grab run information without navigating to the NCBI website, or there is the SRA Run Selector *(https://www.ncbi.nlm.nih.gov/Traces/study/)*. The Run Selector is nice because you can filter the results

for a given BioProject and obtain only the accessions that interest you in a semi user-friendly format. We can search directly from the SRA Run Selector, or simply go to the main NCBI website and begin our search from there. Let's see the latter example, as this is likely the primary way you will attempt to find data in the future.

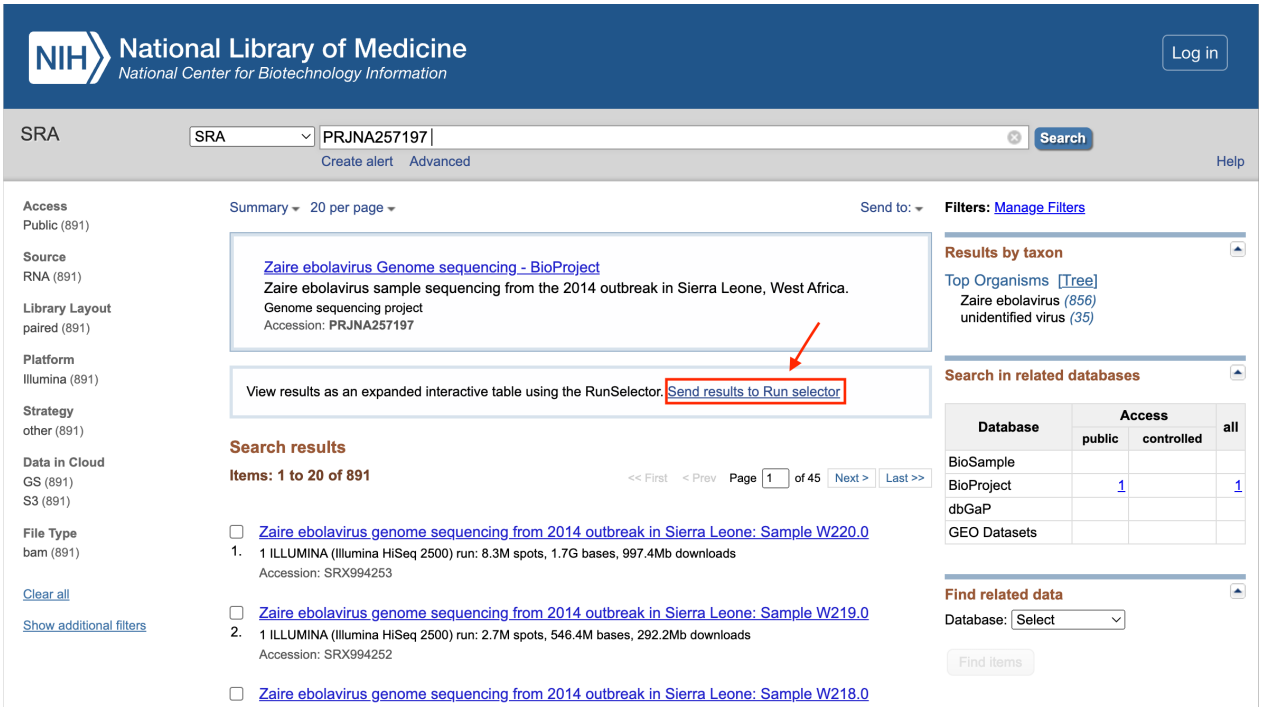
Step 1: Start from the NCBI homepage *(https://www.ncbi.nlm.nih.gov/)*. Type the BioProject ID (`PRJNA257197`) into the search field.



Step 2: In the search results, select the entries next to the SRA.



Step 3: From the SRA results, select "Send results to Run Selector".



Step 4: From there, you can download the metadata or accession list to your local computer. You could also download the fastq files directly to local.

Copy and paste the accession list into a file using `nano`. Save to `runaccessions.txt`. Now use `head` to grab the first few results.

First, let's create a new directory to work in.

```
mkdir run_selector
cd run_selector
nano runaccessions.txt
```

```
head -n 3 runaccessions.txt > runids.txt
```

E-utilities

Using e-utilities to programmatically grab the run info

`esearch` and `efetch`, can be used to query and pull information from Entrez *(https://www.ncbi.nlm.nih.gov/Web/Search/entrezfs.html)*. They aren't the easiest to understand or use, but you can use them to get the same run info that we grabbed using the Run Selector with the following one liner:

```
module load edirect
esearch -db sra -query PRJNA257197 | efetch -format runinfo > runinfo.csv
```

Here we provide the BioProject ID to `esearch`, which queries the SRA. That info can be piped to `efetch`, which will pull down the run info in comma separated format, which we can save to file using >.

Then we can use a combo of `cat`, `cut`, `grep`, and `head` to grab only the accession info we are interested in:

```
cat runinfo.csv | cut -f 1 -d ',' | grep SRR | head > runids.txt
```

Here, we opened the file with `cat`, piped it to the `cut` command, piped it to `grep` for only the accession IDs (skipping the column header), and get only the first few results.

So what is `cut`? It is a unix command, that is used to cut out selected portions of the file (`man cut` for more info). The `-f` option specifies which field to cut (the first field), and `-d` tells the program which delimiter to use. We know this is a comma-separate value file, so we do `-d ','` to specify the comma.

Using the "parallel" tool for automating downloads

Now that we have accession numbers to work with, let's use `parallel` and `fastq-dump` to download the data.

GNU parallel is a shell tool for executing jobs in parallel using one or more computers. *(https://www.gnu.org/software/parallel/man.html)* It can serve as a nice replacement of the `for loop`; though, its behavior is different. Check out this post from the Biostars forum, Tool: Gnu Parallel -

Parallelize Serial Command Line Programs Without Changing Them *(https://www.biostars.org/p/63816/)*, and this from OMGenomics Lab, Using GNU parallel painlessly *(https://omgenomics.com/blog/parallel)*.

Let's see how this works.

```
mkdir parallel
cd parallel
cat ../runids.txt | parallel -j 1 fastq-dump -X 10000 --split-files
```

This gives us six files, two files for each run and 3 runs. It takes the place of the multiple `fastq-dump` commands we would need to use.

What's up with the curly braces `{}`?

The curly braces are default behavior for `parallel`. This is the default replacement string; when empty, it appends inputs. So the following gets the same result.

```
cat ../runids.txt | parallel fastq-dump -X 10000 --split-files
```

More on `parallel`

The real power of `parallel` comes when using something between the brackets such as `{.}`, `{/}`, and `{//}`, like this...

```
nano filelist
```

Include the following:

```
/dir/subdir/file.txt
/other/list.csv
/raw/seqs.fastq
```

Save the file -> `Ctrl O`, `Yes`, `Ctrl X`.

```
cat filelist | parallel echo {} "without extension" {.}
```

/dir/subdir/file.txt without extension /dir/subdir/file
/other/list.csv without extension /other/list
/raw/seqs.fastq without extension /raw/seqs

(This replaces the extension.)

```
cat filelist | parallel echo {} "without path" {/}
```

/dir/subdir/file.txt without path file.txt
/other/list.csv without path list.csv
/raw/seqs.fastq without path seqs.fastq

(This removes the path.)

```
cat filelist | parallel echo {} "with root" {//}
```

/dir/subdir/file.txt with root /dir/subdir
/other/list.csv with root /other
/raw/seqs.fastq with root /raw

(This keeps only the path.)

Note

`parallel` will default to 1 concurrent job per available CPU. On a shared system like helix with 48 CPUs, where users do run fast(er)q-dump, that can cause an overload. Even when submitting a job on Biowulf, you may not want to run as many concurrent jobs as there are allocated CPUs (e.g. if you ran fasterq-dump with 2 threads and you have 12 CPUs allocated --jobs would be 6). Therefore, it is good practice to always specify the `-j` flag, which assigns the number of jobs for the `parallel` command.

Using `swarm`

We can also take advantage of swarm for running jobs in parallel, as we did in the previous lesson.

Note

This should be run from Biowulf, not Helix.

```
cd ..
mkdir swarm
cd swarm
```

We can generate a script using:

```
cat ../runids.txt | while read file; do
    echo prefetch $file >> download.sh
    echo 'fasterq-dump -t /lscratch/$SLURM_JOB_ID' $file >> download
done
```

Edit to include `#SBATCH` directives.

```
nano download.sh
```

At the top of the file add the following:

```
#SWARM --threads-per-process 6
#SWARM --gb-per-process 4
#SWARM --gres=lscratch:10
#SWARM --module sratoolkit
#SWARM --partition=student
```

Submit the script.

```
swarm -f download.sh
```

European Nucleotide Archive (ENA)

Everything in the SRA is also in the ENA (See PRJNA257197 on ENA *(https://www.ebi.ac.uk/ena/browser/view/PRJNA257197?show=reads)*). Files in the ENA share the same naming convention as the SRA but are stored directly as gzipped fastq files and bam files. You can easily use `wget` or `curl` to pull files directly from the ENA without using the `sratoolkit`. Check out the ENA training modules *(https://ena-docs.readthedocs.io/en/latest/retrieval/file-download.html#)* for more information on downloading data from the ENA, including examples.

Using `curl` and `wget`

The curl command is used to retrieve data from web sites. A similar command is `wget`. The Unix system you are working with may have either `curl` or `wget` installed. To see which is active on your system, just type the command at the command line.

```
wget
curl
```

Let's see how we can use `wget` to grab a couple of files from the ENA.

We use `wget` followed by the path to the file:

```
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR155/006/SRR1553426/SRR1553426_2.fast
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR155/006/SRR1553426/SRR1553426_1.fast
```

The `-nc` option refers to "no clobber", which will not overwrite existing files.

To see additional options, use

```
man wget
```

These files are compressed. To decompress a file use `gunzip`.

```
gunzip SRR1553426_2.fastq.gz
```

In the case of `.zip` you can use `unzip`, and with tar.gz, you can use `tar -xvf`.
When in doubt, use google.

Note

Many programs accept compressed fastq files, so rarely will you need to decompress a fastq file.

There is also a fantastic web service called `SRA Explorer` *(https://sra-explorer.info/#)* that can be used to easily generate code for file downloads from the SRA, if you are looking to download a maximum of 500 samples. Use caution, the SRA-explorer is not associated with the `ENA` or `NCBI`, and program support / updates cannot be guaranteed.

Help Session

Practice the skills learned in this lesson here.

Module 2 - Introduction to RNA Sequencing

Lesson 7: Introduction to Next Generation Sequencing (NGS) Data and Quality Control

Lesson 6 Review

Lesson 6 introduced the basics of RNA sequencing, including experimental considerations and basic ideas behind data analysis. In lessons 7 through 14 participants will get hands-on experience with RNA sequencing analysis.

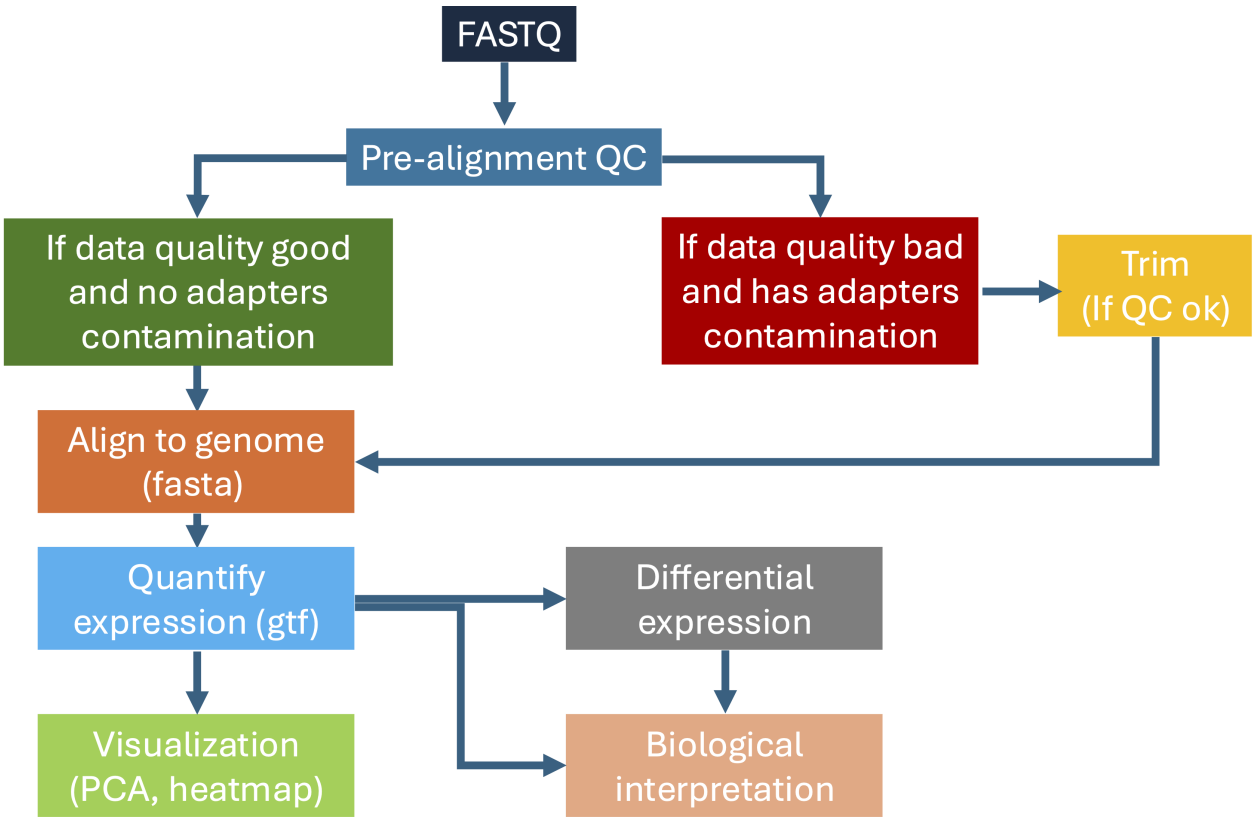
Learning Objectives

After this class, participants will be able to:

- Describe the format in which NGS data is stored.
- Know why sequence quality score is important and how this is calculated.
- Run quality assessment on NGS data and interpret results.

Basic RNA Sequencing Analysis Workflow

The diagram below shows a basic workflow for RNA sequencing analysis, which starts with FASTQ files as input. Quality check is performed on the FASTQ files ensure that sequencing quality is good and that there are no contaminations such as those arising from adapter read through *(https://knowledge.illumina.com/software/general/software-general-reference_material-list/000002905)*. If there are no issues arising from quality check, perform alignment to determine where in the genome each sequence comes from. If issue(s) arise from quality check, then perform trimming (either to remove low quality reads or contamination such as adapters) and perform QC to ensure the trimmed sequences are good prior to alignment. Quantification of gene expression can be performed on the aligned data. From the expression quantification, the researcher can construct visualizations (ie. PCA and heatmap) as well as perform differential expression analysis and biological interpretation.



Note

A reference genome in the form of a `fasta` (or `.fa`) file is needed for the alignment step while a `gtf` file describing location of genomic features (ie. genes, transcripts, exons, and introns) is needed for the quantification step.

Example Data

The data used are located in `/data/classes/BTEP/hcc1395_b4b` on Biowulf. This data was obtained from rnabio.org *(https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/)* and essentially a transcriptomic profiling study comparing between tumor and normal HCC1395 breast cancer cells. There a 3 normal and 3 tumor samples for each condition.

Sign onto Biowulf

Before following along in these exercises, please sign onto Biowulf using the `ssh` construct below where "user" is the class participant's assigned Biowulf student account ID.

```
ssh user@biowulf.nih.gov
```

Request an Interactive Session

Request an interactive session to perform compute intensive tasks on Biowulf by issuing the `sinteractive` command construct below where:

- `--mem`: enables the user to request the desired amount of RAM *(https://en.wikipedia.org/wiki/Random-access_memory)* for analysis (12gb requested in this example).
- `--gres`: enables the user to request generic compute resources such as local temporary storage space or `lscratch` (5gb requested in this example).

```
sinteractive --mem=12gb --gres=lscratch:5
```

Copy the Data to User's `/data` folder

The data is located in the `hcc1395_b4b` folder of `/data/classes/BTEP`. Copy this to the participants' own `/data` directory in Biowulf prior to proceeding using the `cp` command with the `-r` option for copying folders. The directory path where the data resides is `/data/classes/BTEP/hcc1395_b4b` and the destination to which it needs to copied is `/data/user/` and name the copied folder `hcc1395_b4b`. Be sure to change "user" to the participant's assigned Biowulf student ID.

```
cp -r /data/classes/BTEP/hcc1395_b4b /data/user/hcc1395_b4b
```

Use the `ls` command to list the content in the participant's Biowulf `/data` folder to make sure the copy was successful.

```
ls
```

```
hcc1395_b4b
```

FASTQ Files for `hcc1395_b4b`

Change into the `hcc1395_b4b folder` in the participant's Biowulf `/data` directory.

```
cd hcc1395_b4b
```

Next, use `ls` with the `-1` option to list the contents in the `reads` directory one item per line.

```
ls -1 reads
```

The `reads` folder contains the FASTQ or `.fq` files for the hcc1395 data. There are 12 of these, two for each sample as this study generated paired end sequences.

```
hcc1395_normal_rep1_R1.fq
hcc1395_normal_rep1_R2.fq
hcc1395_normal_rep2_R1.fq
hcc1395_normal_rep2_R2.fq
hcc1395_normal_rep3_R1.fq
hcc1395_normal_rep3_R2.fq
hcc1395_tumor_rep1_R1.fq
hcc1395_tumor_rep1_R2.fq
hcc1395_tumor_rep2_R1.fq
hcc1395_tumor_rep2_R2.fq
hcc1395_tumor_rep3_R1.fq
hcc1395_tumor_rep3_R2.fq
```

Definition

In paired end sequencing, a template or unknown nucleotide fragment is sequenced from both ends, thus generating two FASTQ files per sample. A gap between the two reads is usually present in this sequencing protocol.
Image source: https://www.ebi.ac.uk/training/online/courses/functional-genomics-ii-common-technologies-and-data-analysis-methods/rna-sequencing/performing-a-rna-seq-experiment/design-considerations/ *(https://www.ebi.ac.uk/training/online/courses/functional-genomics-ii-common-technologies-and-data-analysis-methods/rna-sequencing/performing-a-rna-seq-experiment/design-considerations/).*

Overview of a FASTQ File

Each FASTQ file is composed of many sequences. The tool `seqkit` and its `stats` function can be used to get statistics the hcc1395 FASTQ files.

Change in the `reads` folder for this.

```
cd reads
```

Load `seqkit`.

```
module load seqkit
```

```
[+] Loading seqkit  2.7.0
```

Run `seqkit stats` for all of the hcc1395 FASTQ files in one go by using `*` before the `.fq` extention. `*` acts as a wildcard.

```
seqkit stats *.fq
```

Query of the stats for the FASTQ files generates the results below, which informs of things such as the number of sequences (or reads) in a FASTQ file. For the hcc1395_normal_rep1 biological replicates, both files in the pair (R1 and R2) have 331,958 sequences. The average length of the sequences in these files is 151 bases. Pairs in paired end sequencing should have the same number of sequences because the reads were generated from the same template.

```
file                        format  type  num_seqs     sum_len  min_le
hcc1395_normal_rep1_R1.fq   FASTQ   DNA    331,958  50,125,658      15
hcc1395_normal_rep1_R2.fq   FASTQ   DNA    331,958  50,125,658      15
hcc1395_normal_rep2_R1.fq   FASTQ   DNA    331,958  50,125,658      15
hcc1395_normal_rep2_R2.fq   FASTQ   DNA    331,958  50,125,658      15
hcc1395_normal_rep3_R1.fq   FASTQ   DNA    331,956  50,125,356      15
hcc1395_normal_rep3_R2.fq   FASTQ   DNA    331,956  50,125,356      15
hcc1395_tumor_rep1_R1.fq    FASTQ   DNA    390,607  58,981,657      15
hcc1395_tumor_rep1_R2.fq    FASTQ   DNA    390,607  58,981,657      15
hcc1395_tumor_rep2_R1.fq    FASTQ   DNA    390,607  58,981,657      15
hcc1395_tumor_rep2_R2.fq    FASTQ   DNA    390,607  58,981,657      15
hcc1395_tumor_rep3_R1.fq    FASTQ   DNA    390,607  58,981,657      15
hcc1395_tumor_rep3_R2.fq    FASTQ   DNA    390,607  58,981,657      15
```

Recall from the `seqkit stat` results shown above that each FASTQ file contains many sequencing reads. The record for each sequencing read is composed of four lines and these include the following.

- The first line is the sequencing read metadata (ie. instrument used, location on the flow cell where the read was derived, and whether the read is the first or second of a pair in paired end sequencing) - in this example,

- "/1" at the end of the metadata line in the reads from `hcc1395_normal_rep1_R1.fq` denotes the first read of a pair
- "/2" at the end of the metadata line in the reads from `hcc1395_normal_rep1_R2.fq` denotes the second read of a pair
- The second line is the biological sequence
- The third line is a "+"
- The fourth line contains the quality score of each of the bases in the sequencing read. The quality score reflects the likelihood that a base at a particular location along the read is wrong.

Below, `head -4` is used to show the first sequencing reads of `hcc1395_normal_rep1_R1.fq` and `hcc1395_normal_rep1_R2.fq`. `-4` in the `head` command tells it to only retrieve the first 4 lines in a file.

```
head -4 hcc1395_normal_rep1_R1.fq
```

```
@K00193:38:H3MYFBBXX:4:1101:10003:44458/1
TTCCTTATGAAACAGGAAGAGTCCCTGGGCCCAGGCCTGGCCCACGGTTGTCAAGGCACATCATTGCCA
+
AAFFFKKKKKKKKKKKKKKKKKKKKKKKKFKKFKKKKF<AAKKKKKKKKKKKKKKKKFKKKFKKKKKKI
```

```
head -4 hcc1395_normal_rep1_R2.fq
```

```
@K00193:38:H3MYFBBXX:4:1101:10003:44458/2
ATGGAACTCCCTACTGACCTCTGAGAACTGGAAACGAGTTTGTACAGAAGTCAGAACTTTGGGTTGGGA
+
A<FFFKKKKKKKKKKKKKKKKKKKKKKFKAKKKKKKK7FFKKKKKKKKK7FKKFKKKKKKKKKKKKKI
```

Click here *(http://www.drive5.com/usearch/manual/quality_score.html)* and here *(https://training.galaxyproject.org/training-material/topics/sequence-analysis/tutorials/quality-control/slides-plain.html)* to learn more about the quality scores. The image below shows a table that translates the characters you see in the quality score lines to a numeric score.

ASCII_BASE=33 Illumina, Ion Torrent, PacBio and Sanger

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	33 !	11	0.07943	44 ,	22	0.00631	55 7	33	0.00050	66 B
1	0.79433	34 "	12	0.06310	45 -	23	0.00501	56 8	34	0.00040	67 C
2	0.63096	35 #	13	0.05012	46 .	24	0.00398	57 9	35	0.00032	68 D
3	0.50119	36 $	14	0.03981	47 /	25	0.00316	58 :	36	0.00025	69 E
4	0.39811	37 %	15	0.03162	48 0	26	0.00251	59 ;	37	0.00020	70 F
5	0.31623	38 &	16	0.02512	49 1	27	0.00200	60 <	38	0.00016	71 G
6	0.25119	39 '	17	0.01995	50 2	28	0.00158	61 =	39	0.00013	72 H
7	0.19953	40 (	18	0.01585	51 3	29	0.00126	62 >	40	0.00010	73 I
8	0.15849	41)	19	0.01259	52 4	30	0.00100	63 ?	41	0.00008	74 J
9	0.12589	42 *	20	0.01000	53 5	31	0.00079	64 @	42	0.00006	75 K
10	0.10000	43 +	21	0.00794	54 6	32	0.00063	65 A			

Note

$$quality\ score = -10\times log10(error\ probability)$$ $$error\ probability=10^{\frac{quality\ score}{-10}}$$

Quality Assessment on the `hcc1395` FASTQ Files

Go back to the `hcc1395_b4b` folder.

```
cd /data/users/hcc1395_b4b
```

Make a folder called `pre_alignment_qc_raw`.

```
mkdir pre_alignment_qc_raw
```

Load `fastqc`, which is a tool used to perform quality check on Illumina sequencing data.

```
module load fastqc
```

```
[+] Loading singularity  4.1.5  on cn0047
[+] Loading fastqc  0.12.1
```

Change into the `pre_alignment_qc_raw` folder.

```
cd  pre_alignment_qc_raw
```

Next, run `fastqc` using the following command construct where:

- `../reads/*.fq` tells `fastqc` to look one directory back `../` and then in the `reads` folder and perform QC on all of the `.fq` files. `*` is used as a wildcard.

- `-o` is an option in `fastqc` that allows for specification of output folder. In this case, the output folder should be `pre_alignment_qc_raw`. Since the current directory is `pre_alignment_qc_raw`, the '.' can be used to denote here, in the current folder.

```
fastqc ../reads/*.fq -o .
```

List the contents for `pre_alignment_qc_raw` after QC completes. FASTQC generates a html report and provides the results in a zip file for each file. The html report can be viewed on a local web browser.

```
ls
```

```
hcc1395_normal_rep1_R1_fastqc.html  hcc1395_tumor_rep1_R1_fastqc.html
hcc1395_normal_rep1_R1_fastqc.zip   hcc1395_tumor_rep1_R1_fastqc.zip
hcc1395_normal_rep1_R2_fastqc.html  hcc1395_tumor_rep1_R2_fastqc.html
hcc1395_normal_rep1_R2_fastqc.zip   hcc1395_tumor_rep1_R2_fastqc.zip
hcc1395_normal_rep2_R1_fastqc.html  hcc1395_tumor_rep2_R1_fastqc.html
hcc1395_normal_rep2_R1_fastqc.zip   hcc1395_tumor_rep2_R1_fastqc.zip
hcc1395_normal_rep2_R2_fastqc.html  hcc1395_tumor_rep2_R2_fastqc.html
hcc1395_normal_rep2_R2_fastqc.zip   hcc1395_tumor_rep2_R2_fastqc.zip
hcc1395_normal_rep3_R1_fastqc.html  hcc1395_tumor_rep3_R1_fastqc.html
hcc1395_normal_rep3_R1_fastqc.zip   hcc1395_tumor_rep3_R1_fastqc.zip
hcc1395_normal_rep3_R2_fastqc.html  hcc1395_tumor_rep3_R2_fastqc.html
hcc1395_normal_rep3_R2_fastqc.zip   hcc1395_tumor_rep3_R2_fastqc.zip
```

Copy to `hcc1395_normal_rep1_R1_fastqc.html` to local computer to learn how to interpret a FASTQC report. To do this open another terminal (Mac) or command prompt (Windows) and change into the local `Downloads` folder.

```
cd Downloads
```

In the `scp` construct below, change user to the participant's assigned Biowulf student ID and can be broken down as follows.

- `user@helix.nih.gov`: connects the user to `helix`, which is the HPC partition used for interactive data transfer.
- `:` separates the `helix` login part from the path of the file or folder to transfer, which is `/data/user/hcc1395_b4b/pre_alignment_qc_raw/hcc1395_tumor_rep1_R1_fastqc.html`.
- `.`: specifies for `scp` to download to the current directory, which should be the local `Downloads` folder.

Enter the user's Biowulf password when prompted and the download should commence.

```
scp user@helix.nih.gov:/data/user/hcc1395_b4b/pre_alignment_qc_raw/h
```

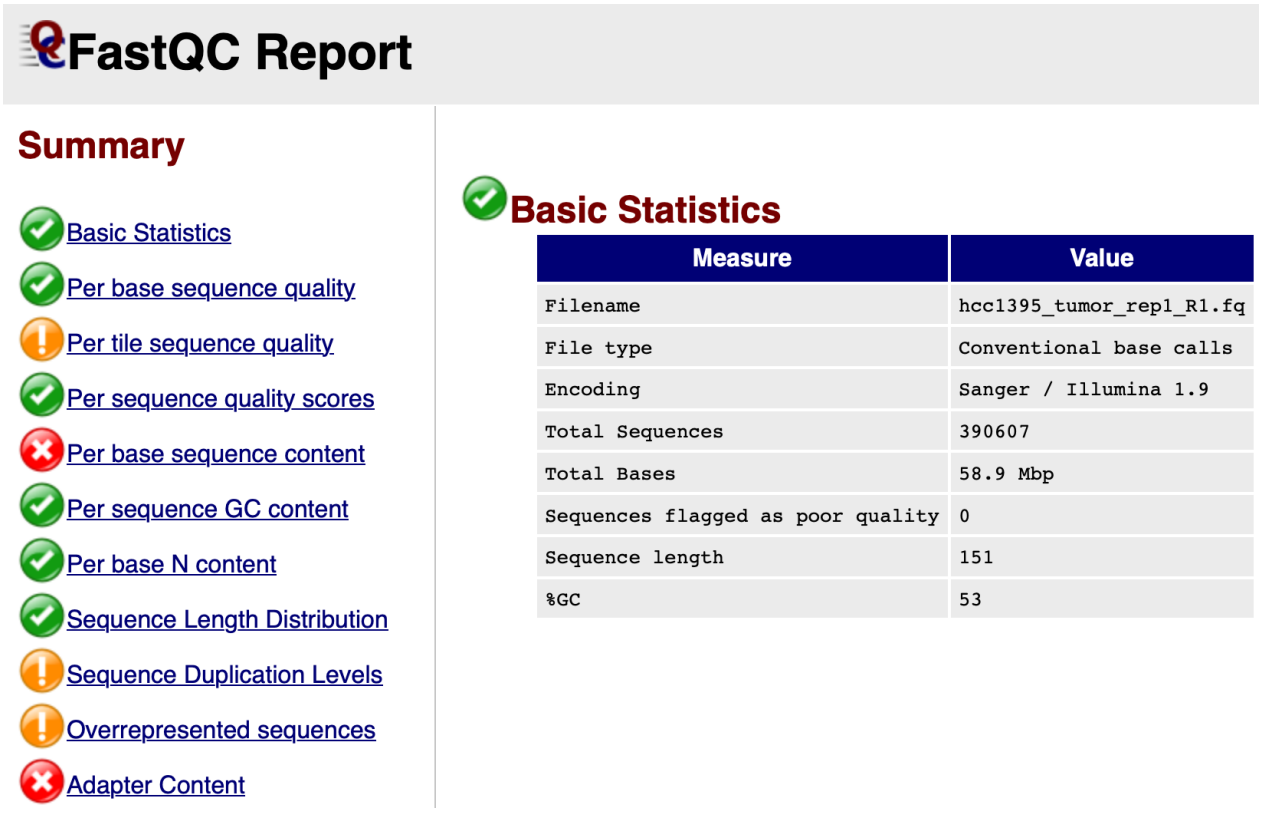
Interpreting FASTQC Report

Report Navigator and Summary

The first item is the navigation panel that enables scientists to quickly link to different portions of the FASTQC report. The QC modules that passed are indicated by a green circle with a check. Those with warnings are denoted by a yellow cirlces with a "!". Failed modules are indicated by a red circle with "x".

Next there is a summary statistics table, which includes the following information.

- Name of the FASTQ file in which the report was generated for (hcc1395_tumor_rep1_R1.fq in this case)
- The number of sequences in the file.
- Number of sequences flagged with poor quality.
- The sequence length (ie. how many bases are in each sequencing read of the FASTQ file).

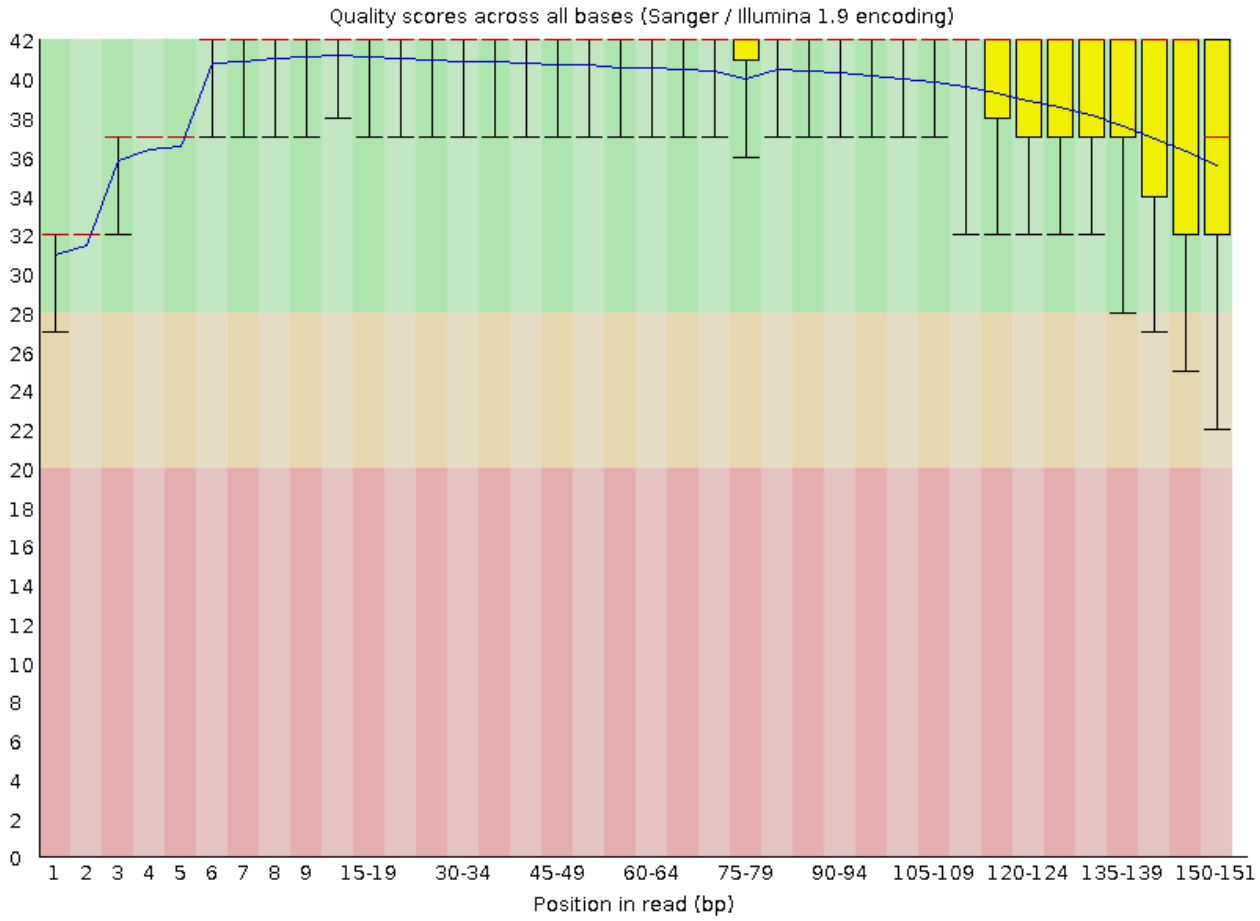


Per Base Sequence Quality

Next, there is the "Per base sequence quality" plot.

This chart plots the error likelihood at each base position averaged over all sequences in a FASTQ file.

- On the vertical axis are the quality scores that are in row 4 of the sequencing reads in a FASTQ file. These quality scores represent error probabilities, where:
    - 10 corresponds to 10% error (1/10),
    - 20 corresponds to 1% error (1/100),
    - 30 corresponds to 0.1% error (1/1000) and
    - 40 corresponds to one error every 10,000 measurements (1/10,000) that is an error rate of 0.01%
- The three colored bands (green, yellow, red) illustrate the typical labels assigned to these measure:
    - reliable (28-40, green)
    - less reliable (20-28, yellow)
    - error prone (1-20, red)
- The yellow boxes contain 50% of the data, the whiskers indicate the 75% outliers.
- The red line inside the yellow boxes is the median quality score for that base.
- The blue line is the average quality score at a particular base.



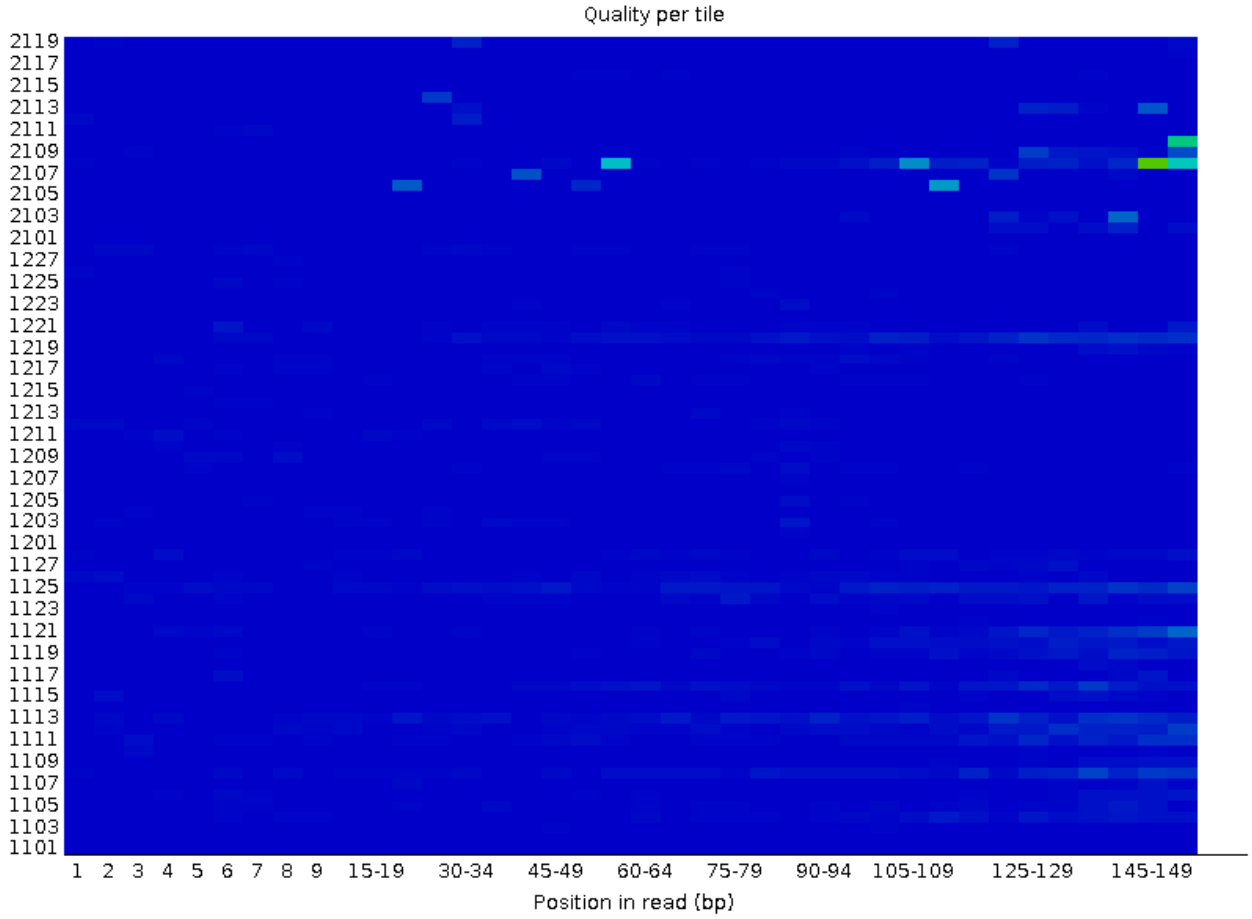
Per Tile Sequence Quality

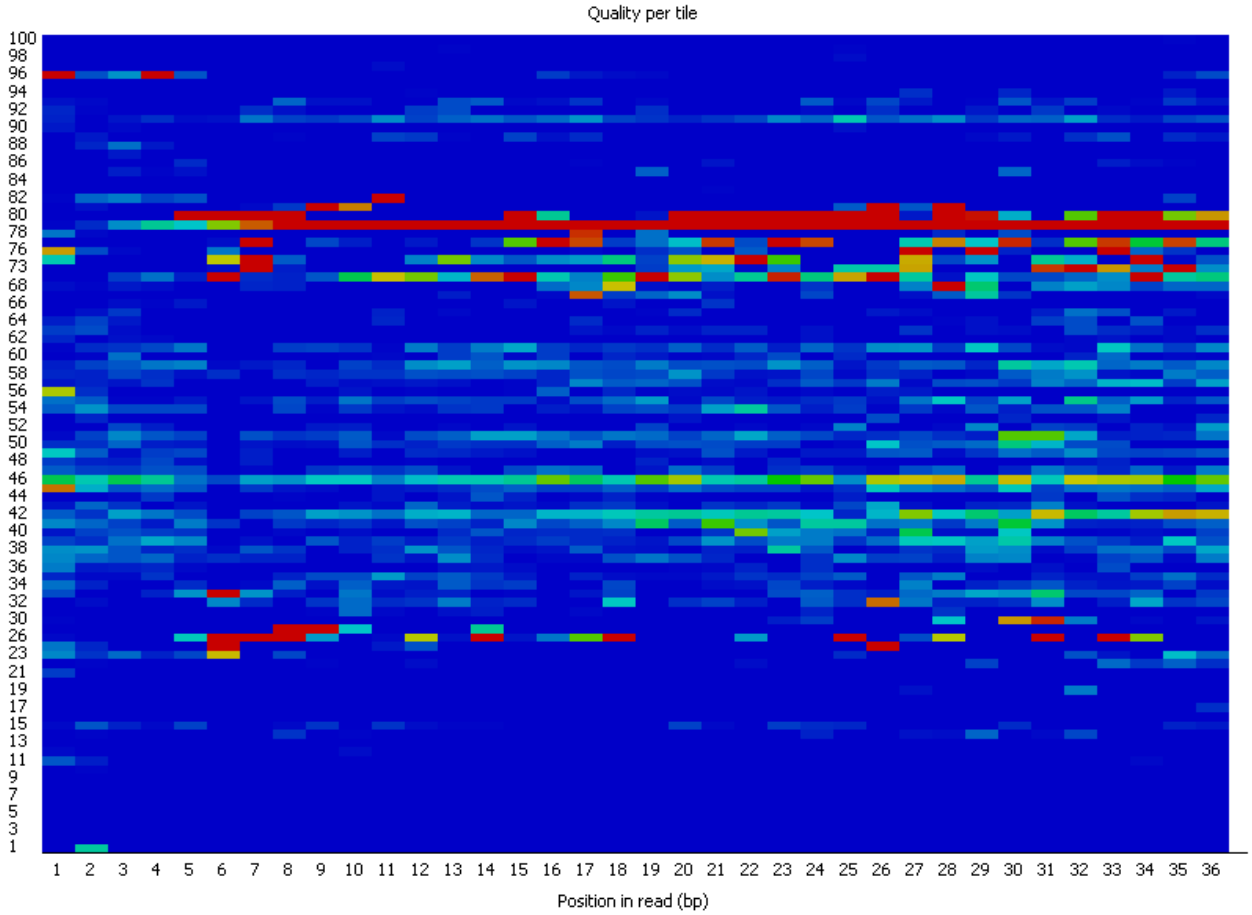
Next, there is the "Per tile sequence quality" plot. This graph informs whether something is wrong with a tile in the flow cell. Along the horizontal axis are the base positions. The vertical axis represents the tile number in which the read came from.

> This plot compares the *average quality score of a tile* to the *average quality score of all tiles at a particular base position*. -- https://sequencing.qcfail.com/articles/position-specific-failures-of-flowcells/ *(https://sequencing.qcfail.com/articles/position-specific-failures-of-flowcells/)*

The interpretation for this plot is as follows

- Colder colors indicate that the average quality score at a tile is at or above the average quality score of all tiles at a particular base position. So a plot that is entirely blue is good.
- Hotter colors indicate that the average quality score at a tile is below the average quality score of all tiles at a particular base position. So a plot with red indicates a part of the flow cell has problems.

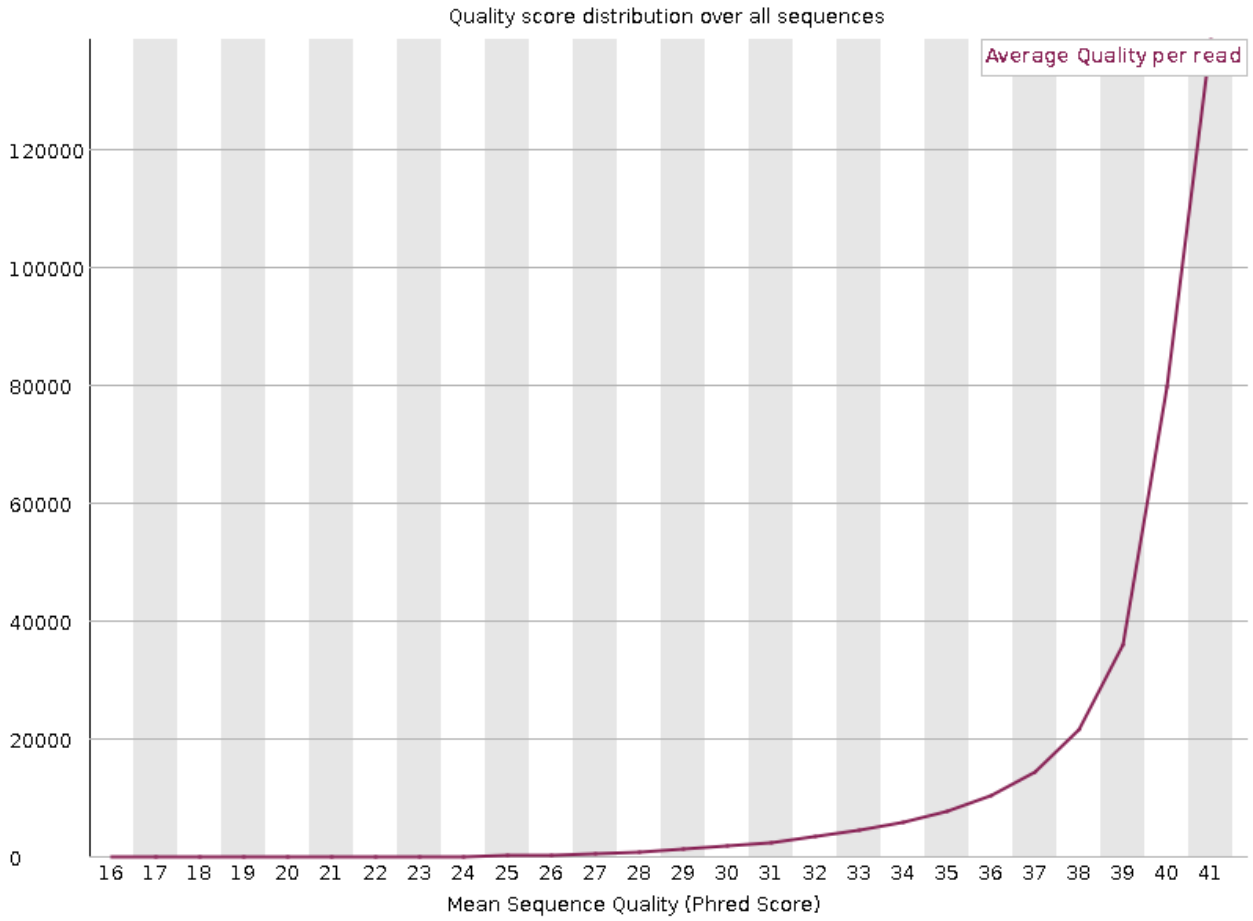

Quality per tile
2119
2117
2115
2113
2111
2109
2107
2105
2103
2101
1227
1225
1223
1221
1219
1217
1215
1213
1211
1209
1207
1205
1203
1201
1127
1125
1123
1121
1119
1117
1115
1113
1111
1109
1107
1105
1103
1101
1 2 3 4 5 6 7 8 9 15-19 30-34 45-49 60-64 75-79 90-94 105-109 125-129 145-149
Position in read (bp)



Bad per tile quality. Source: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/12%20Per%20Tile%20Sequence%20Quality.html *(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/12%20Per%20Tile%20Sequence%20Quality.html)*

Per Sequence Quality Scores

Next, the distribution of the sequence quality scores is shown in the "Per sequence quality scores" plot. This graph can reveal whether the quality of a subset of sequences is poor. Here the sequences have good quality, where most reads have quality that clusters at around 37 or above (~0.02% error likelihood).

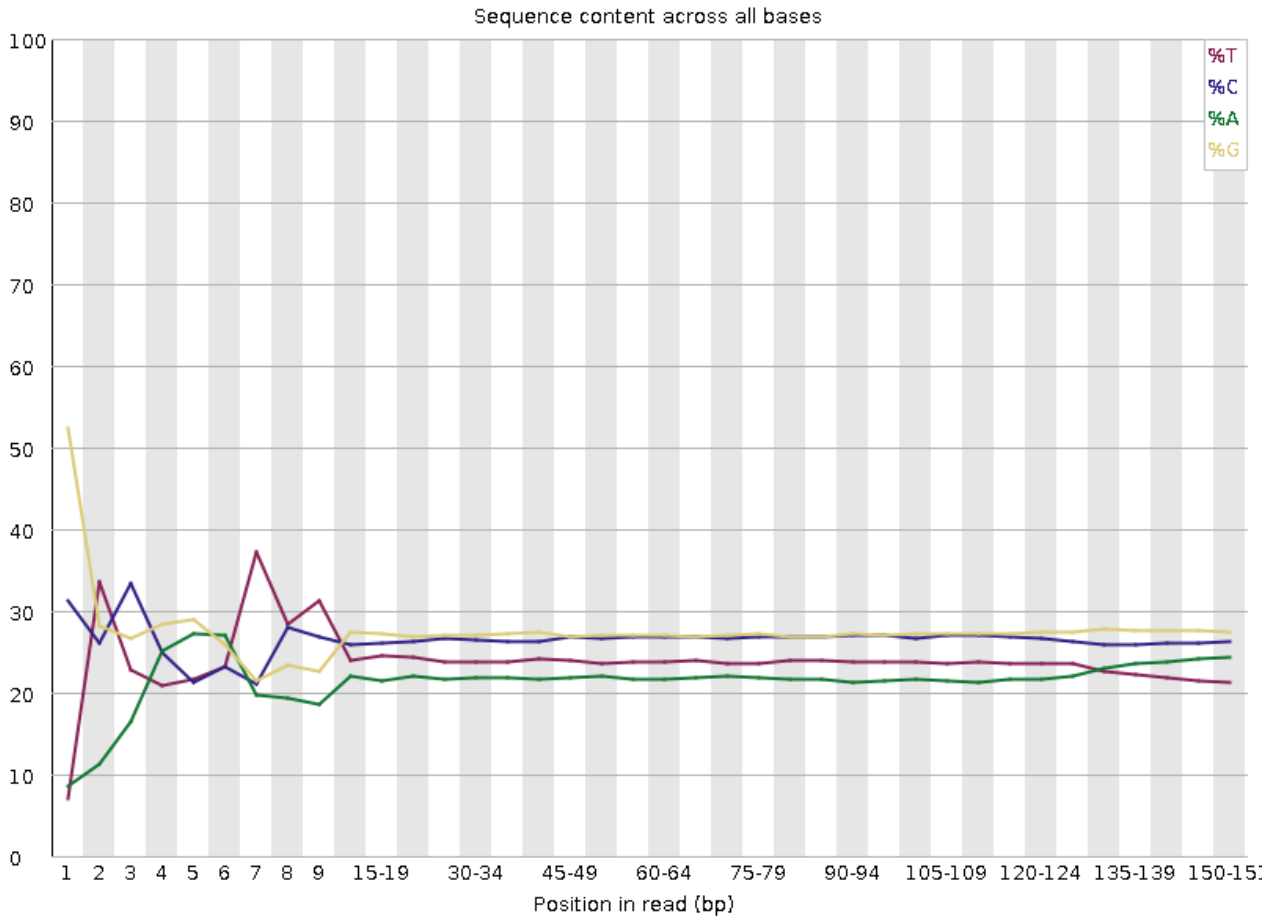


Per Base Sequence Content

The next figure shows "Per base sequence content", which is essentially the sequence make up along the bases of reads in the FASTQ file. If a library is random, then the percent composition of each nucleotide base (A,T,C,G) should be the same (~25%).

This module fails for `hcc1395_tumor_rep1_R1.fq` because the difference between the percentage of A and the percentage of T is larger than 20 at a particular location **OR** the difference between the percentage of C and the percentage of G is larger than 20 at a particular location -- Babraham bioinformatics Per base sequence content *(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/4%20Per%20Base%20Sequence%20Content.html)*.

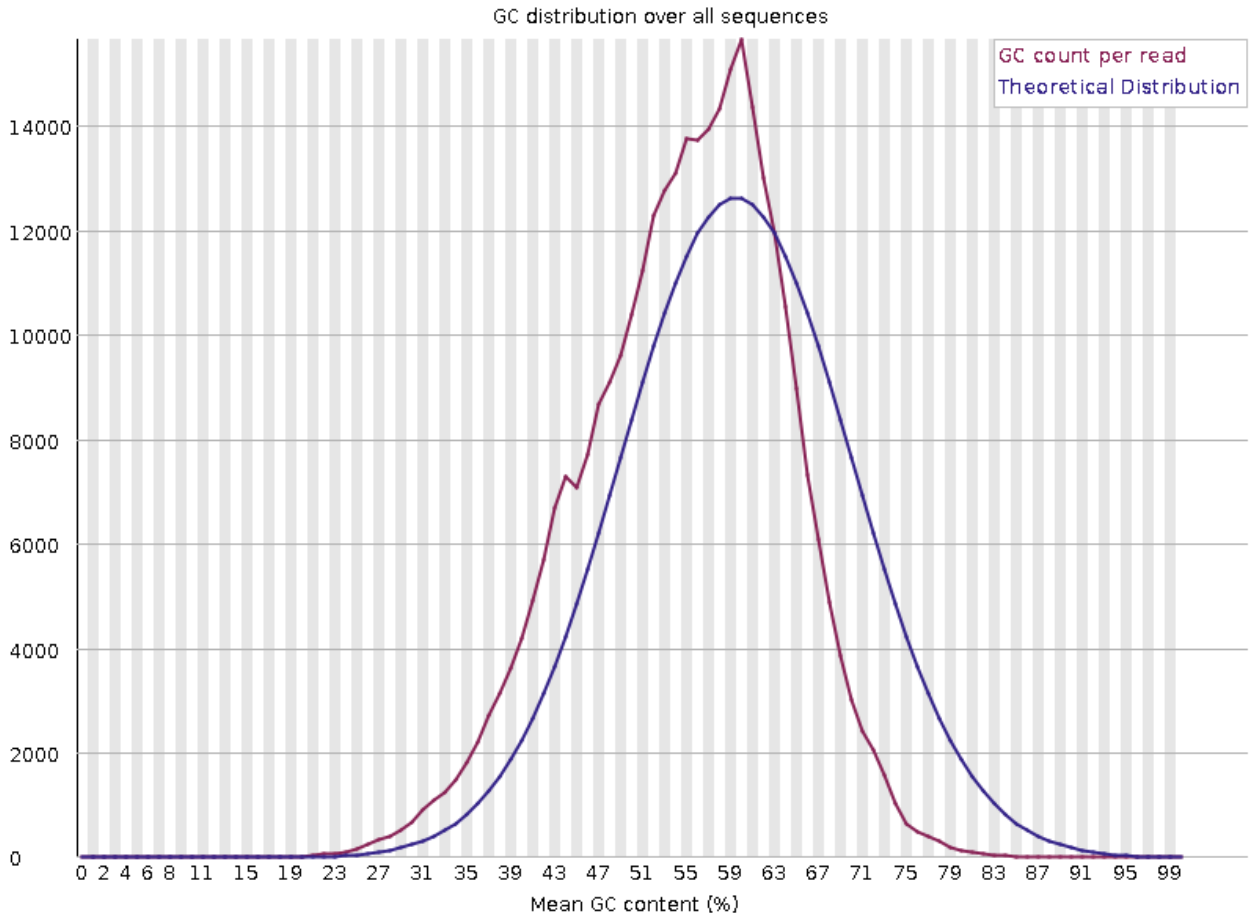
In `hcc1395_tumor_rep1_R1.fq`, it looks like the difference between the percent composition of A and T at base position 2 is causing the failure. Unfortunately, this type of uneveness in base distribution at the beginning of a read is observed in RNA sequencing due to priming with random hexamers during the library preparation stage.



Per Sequence GC Content

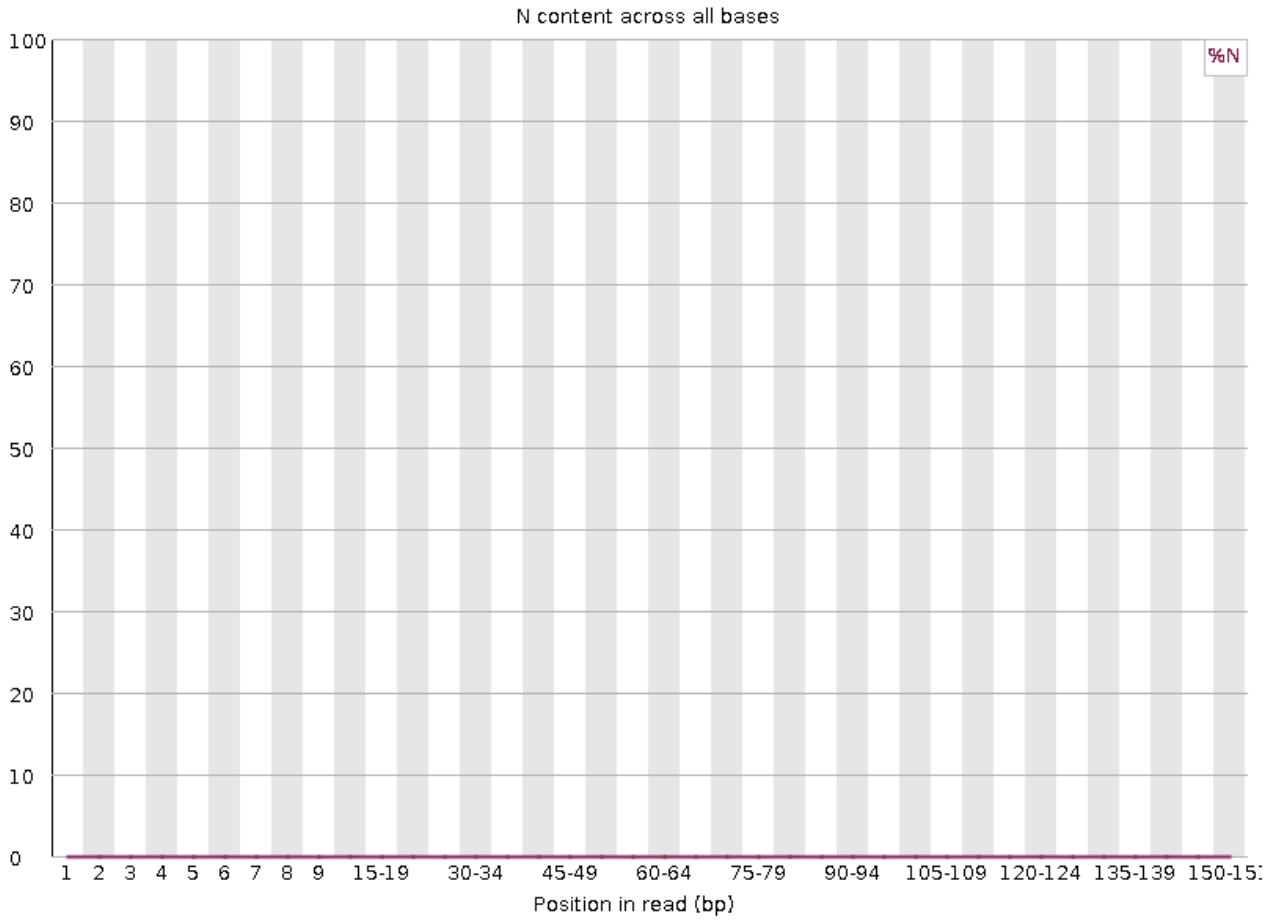
Next, the GC content across each sequence compared to a normal distribution in what is shown in the "Per sequence GC content" plot. The GC content in `hcc1395_tumor_rep1_R1.fq` is off from the normal theoretical distribution.

The peak of this theoretical distribution is an estimate of the GC content of the underlying genome. Deviation from of the GC content from the theoretical distribution could be caused by contamination or sequencing bias. -- Babraham bioinformatics Per base sequence content *(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/5%20Per%20Sequence%20GC%20Content.html)*.

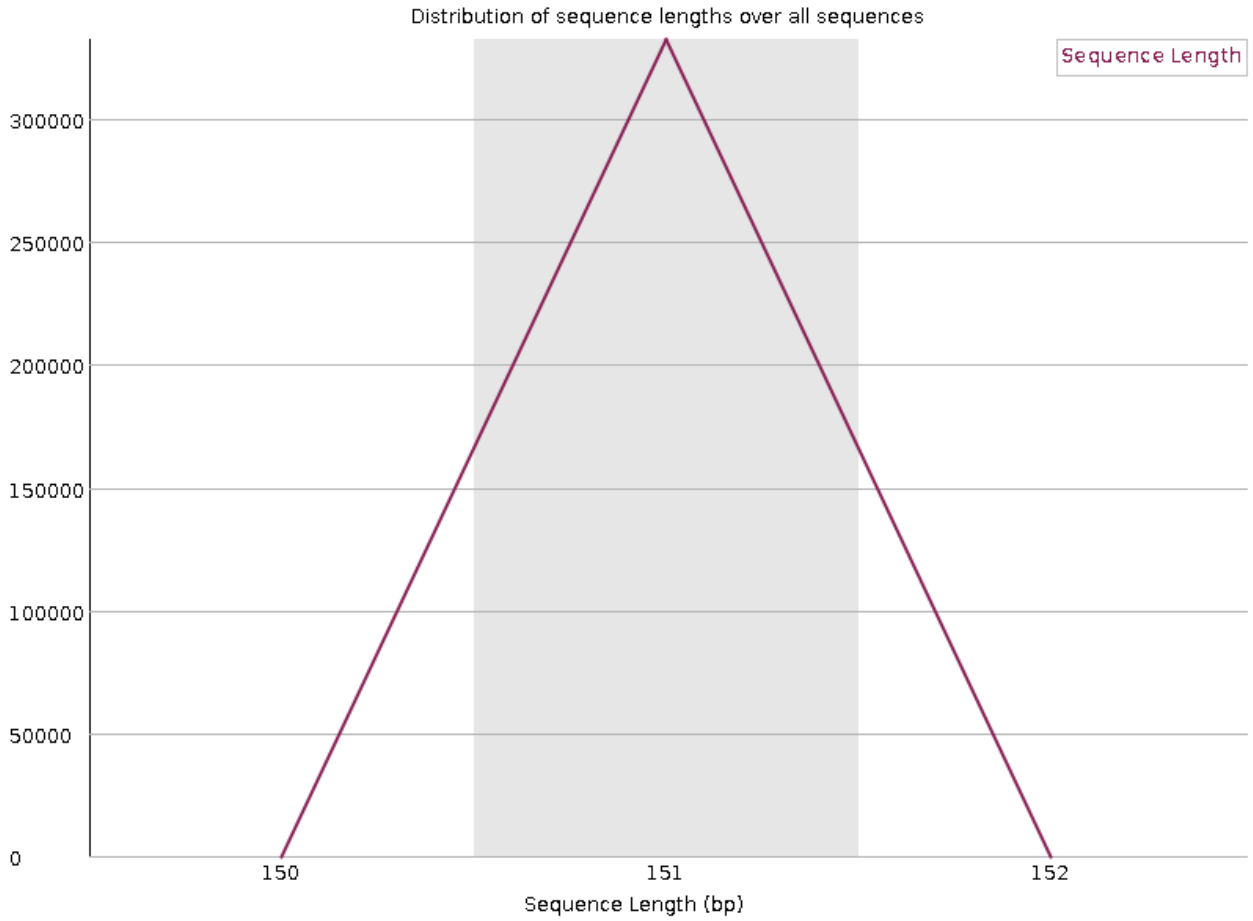


Per Base N Content

Whether there are any unknown bases in the sequencing reads is shown in next plot, which is the "Per base n content".



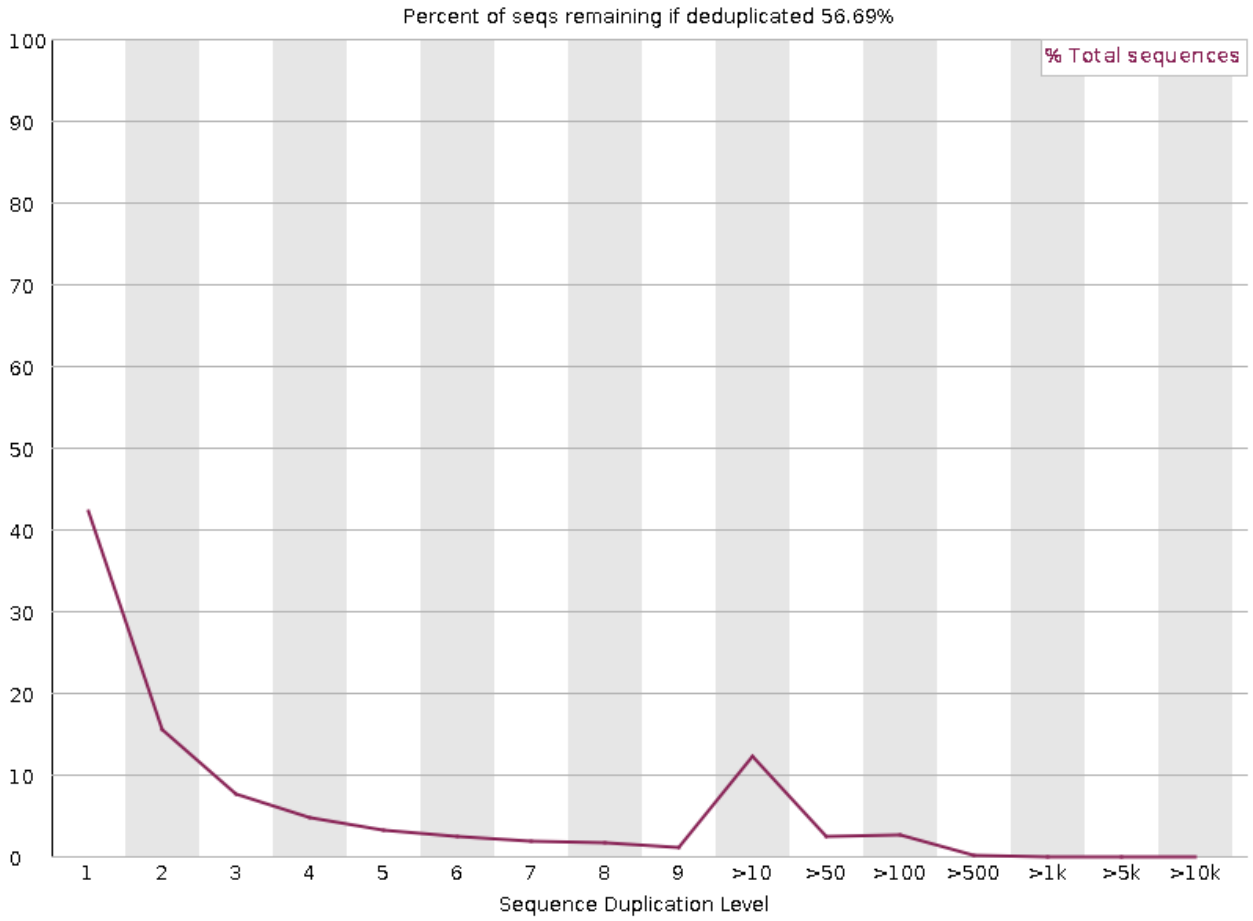
The sequence length distribution of `hcc1395_tumor_rep1_R1.fq` in what is known as the "Sequence Length Distribution" plot. All sequences have 151 bases as trimming has not been performed.



Sequence Duplication Level

The plot below shows the sequence duplication levels. High levels of duplication may indicate an enrichment bias such as over-amplification in the PCR step. However, in RNA sequencing, duplicate sequences could be biologically meaningful as these could be derived from multiple copies of the same transcript.

> See here *(https://dnatech.ucdavis.edu/faqs/should-i-remove-pcr-duplicates-from-my-rna-seq-data#:~:text=The%20short%20and%20generalized%20answer,are%20analyzed%20without%20duplic*
> for advice regarding handling duplicates in RNA sequencing.



Overrepresented Sequences

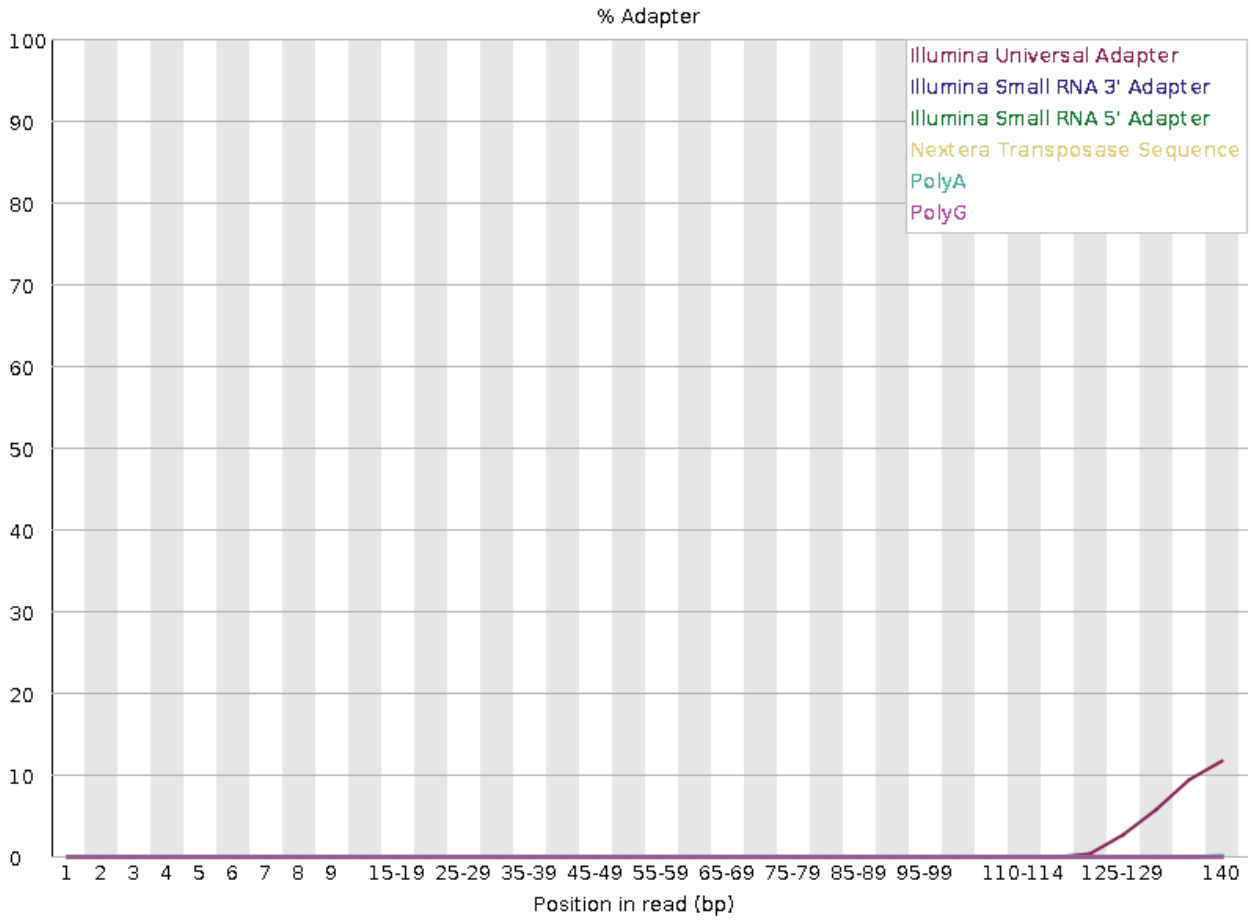
The overrepresented sequences graph is shown next. Presence of overrepresented sequences may indicate enrichment artifact or the sequencing library is not very diverse. On the other hand, overrepresented sequences could have biological significance.

Overrepresented sequences

Sequence	Count	Percentage	Possible Source
CTGGAGTCTTGGAAGCTTGACTACCCTACGTTCTCCTACAAATGGACCTT	1544	0.39528221460444896	No Hit
CTACGTTCTCCTACAAATGGACCTTGAGAGCTTGTTTGGAGGTTCTAGCA	654	0.1674317152534389	No Hit
GGGAGCAGAGGCAGCTGCCTTTATTGGGGGCCATGGGCTGGCTGATTTCA	597	0.1528390428230933	No Hit
GTGGAAAATAACTTTTATTGAGACCCCACCAACTGCAAAATCTGTTCCTG	594	0.15207100743202245	No Hit

Adapter Contamination

If there is adapter contamination in the sequences, then this will be shown in the next plot.



Conclusions from `hcc1395_normal_rep1_R1.fastq` Quality Assessment

In general, FASTQC shows that the file `hcc1395_normal_rep1_R1.fastq` have good quality sequences (less than 1% error likelihood). The concern is the presence of Illuminal universal adapters.

Merging Multiple FASTQC Reports into One

Reviewing multiple FASTQC reports is cumbersome but there is an application called MultiQC that allows scientists to combine many FASTQC reports into one.

To use MultiQC on Biowulf (or other HPC systems), first load it using the following.

```
module load multiqc
```

The `--filename` option allows users to enter a base name (ie. file name without an extension) for the output MultiQC report. Here, the base name `pre_alignment_qc_raw` will be used.

```
multiqc --filename pre_alignment_qc_raw .
```

```
ls
```

MultiQC generates a folder that contains the QC data should the researcher choose to interrogate these programmatically. It provides a report in the html format as well, which can be downloaded to local computer and viewed in a web browser.

```
pre_alignment_qc_raw_data
pre_alignment_qc_raw.html
```

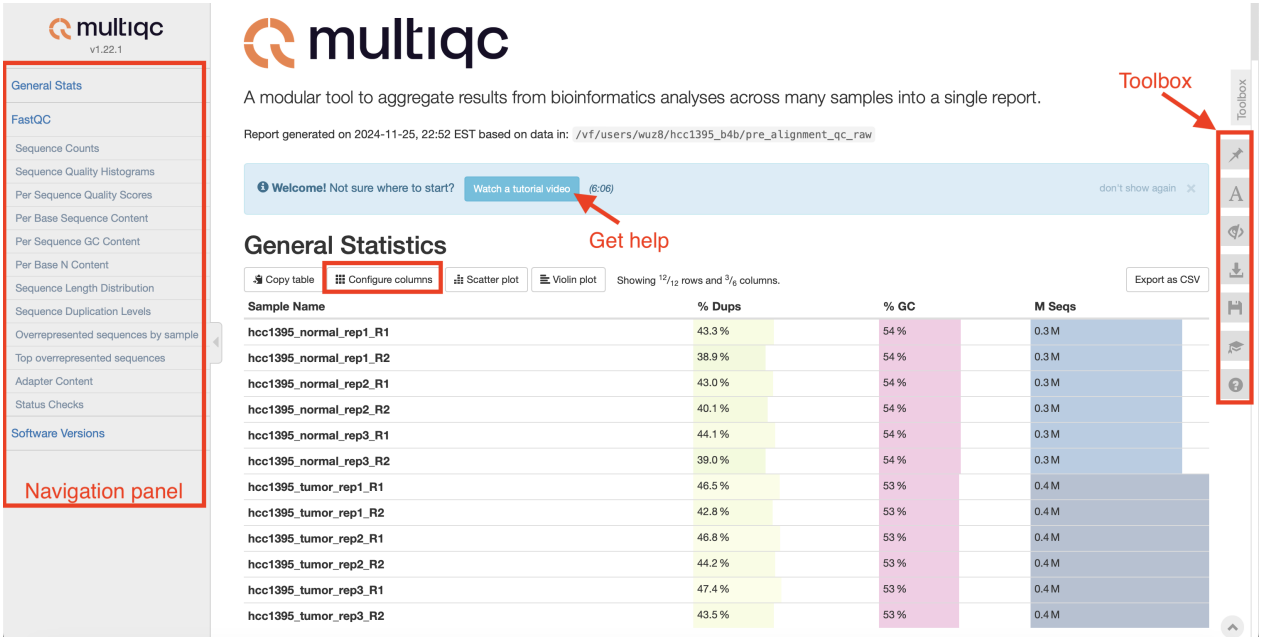
Take a look at the MultiQC html report by copying it to the Downloads folder on local computer using `scp`.

```
scp user8@helix.nih.gov:/data/user/hcc1395_b4b/pre_alignment_qc_raw/
```

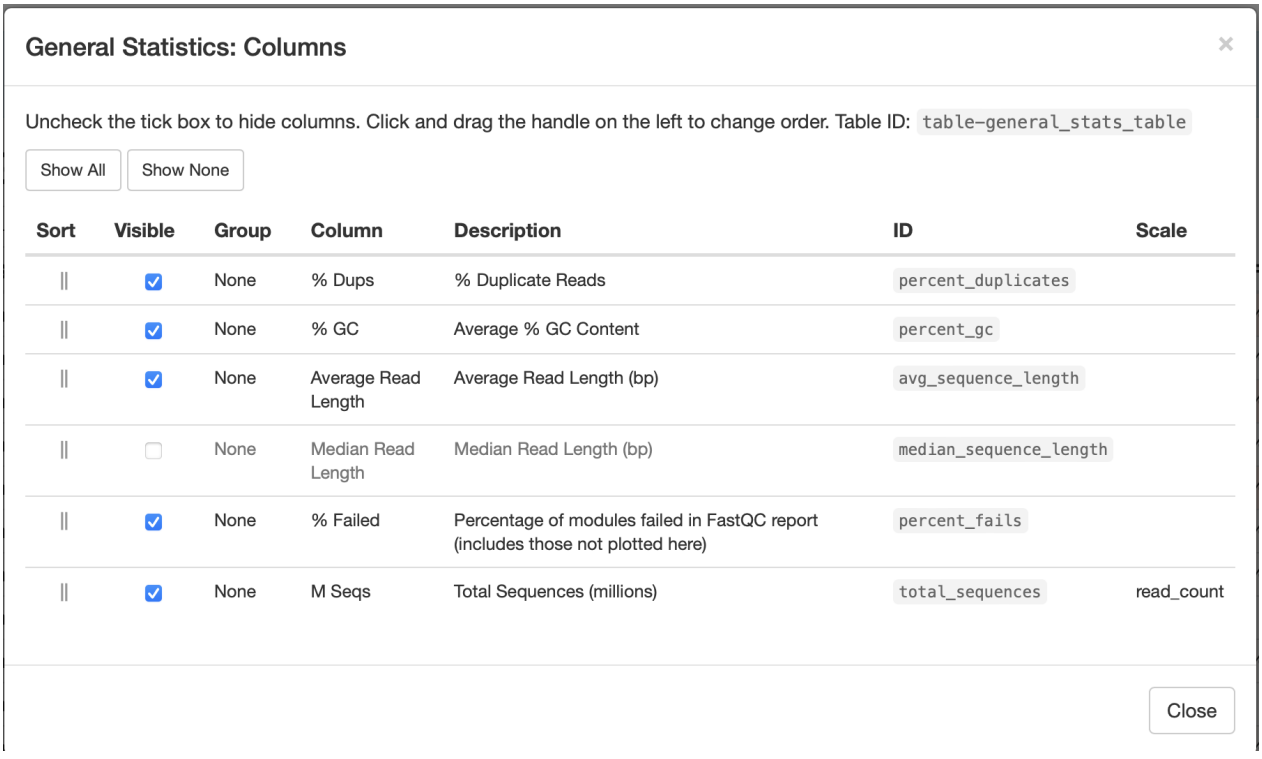
MultiQC Navigation Panel and Summary Statisics

Upon opening the MultiQC report, the user will be greeted with a navigation panel (similar to that found in the individual FASTQC reports) that allows users to quickly move to different sections of the report. A link to get help for using the MultiQC report is available as well. To the right of the report page, there is a tool box that allows the researcher to do things like highlighting different samples in different colors for better visualization, rename samples, and export each of the individual QC plots as an image for inclusion in presentations and/or publications. MultiQC reports are interactive.

The General Statistics table shows some basic statistics about the samples, including percentage of duplicate reads, GC content, and number of sequences (reported as million of sequences in the M Seqs column).



Click on the Configure Columns tab to choose which columns to include in the General Statistics table and the plot button to visualize the data in graphical format.



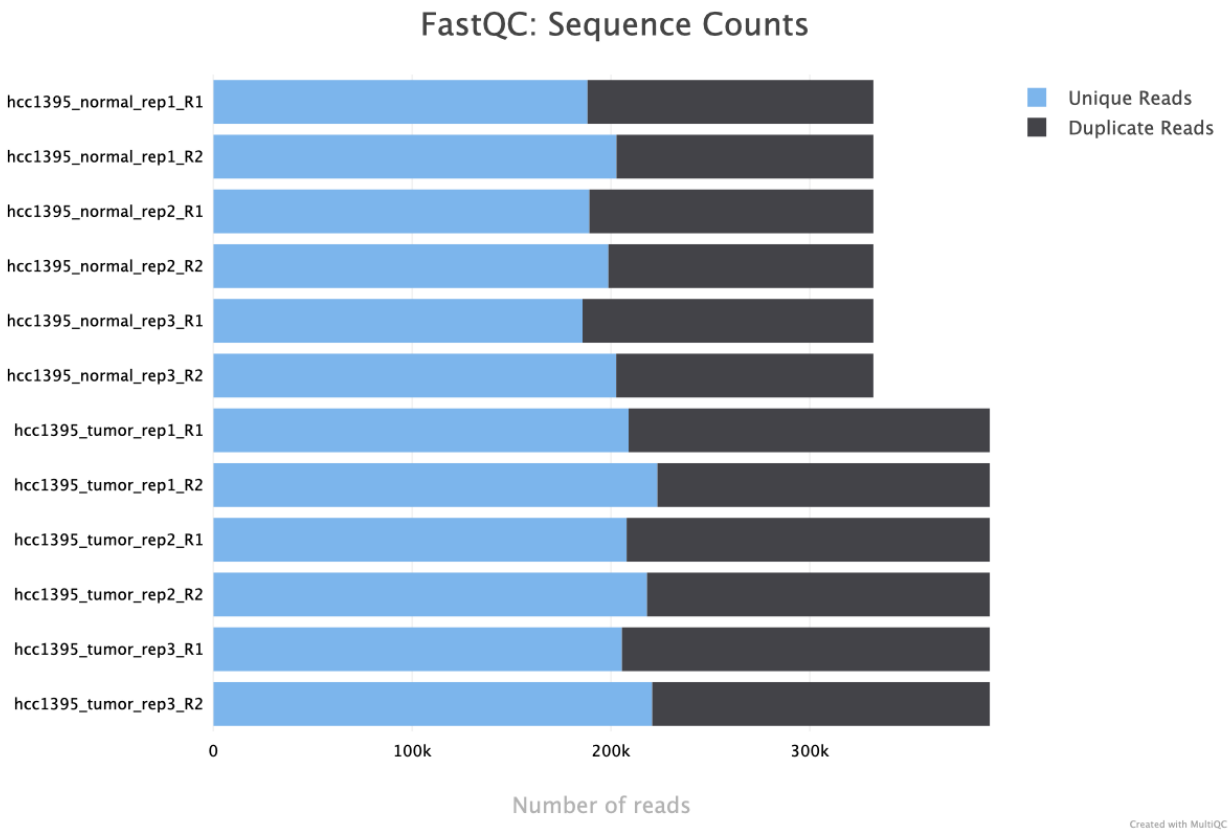
Altering the way the total number of sequences is reported

The number of sequences could be reported as the actual number rather than in millions. Users can alter this by issuing the `--config` option in `multiqc` and specifying a custom report configuration `yaml` file. The content for this configuration file is shown below and can be explained as follows. * `read_count_multiplier`: set to 1 to report the actual sequence number. * `read_count_prefix`: set to "" to remove the units at the end of the number of sequences. The default is "M" for milion. * Under `table_columns_name` change the `total_sequences` column heading to "# Sequences" or any name that reflects the fact the actual number of sequences is being reported.

```
read_count_multiplier: 1
read_count_prefix: ""
table_columns_name:
  FastQC:
    total_sequences: "# Sequences"
```

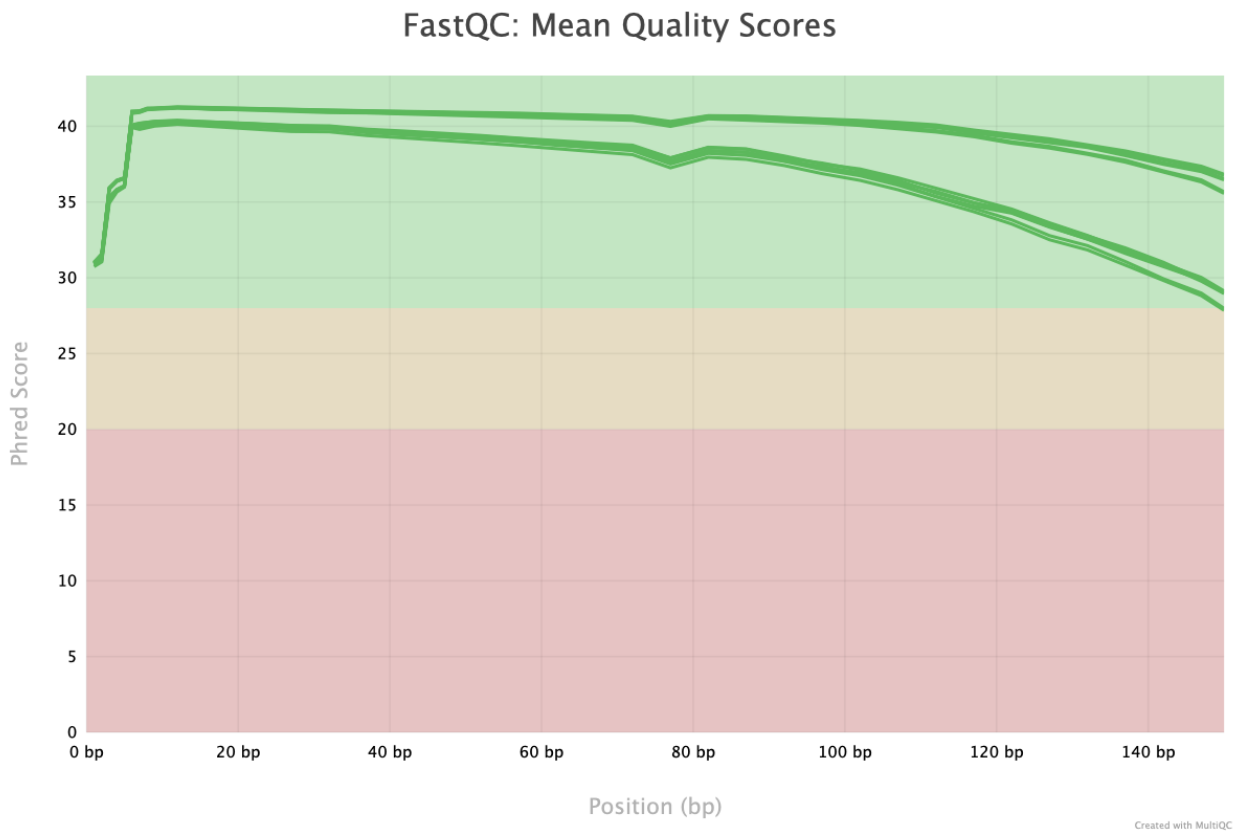
MultiQC Sequence Count

The Sequence Count plot shows the break down of unique and duplicate reads for each FASTQ file. Again, duplication suggests some sort of enrichment bias or in the case of RNA sequencing, it could be that the duplicates are biologically meaningful due to multiple copies of the same trancript being expressed.

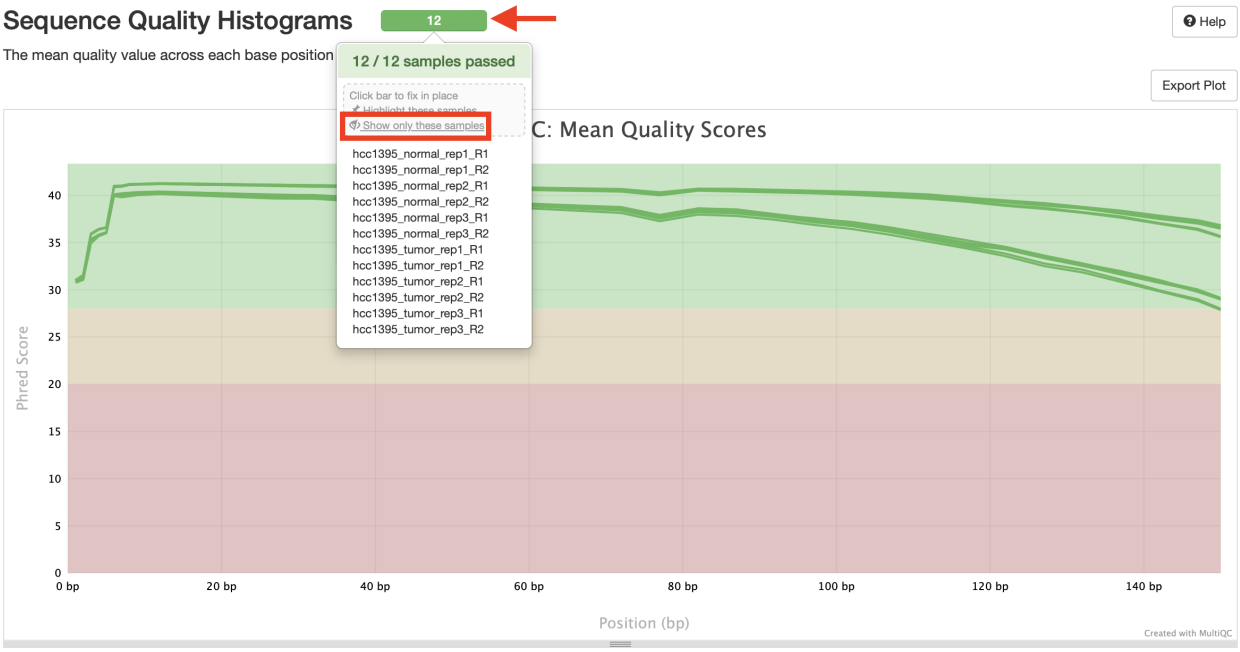


MultiQC Mean Quality Score

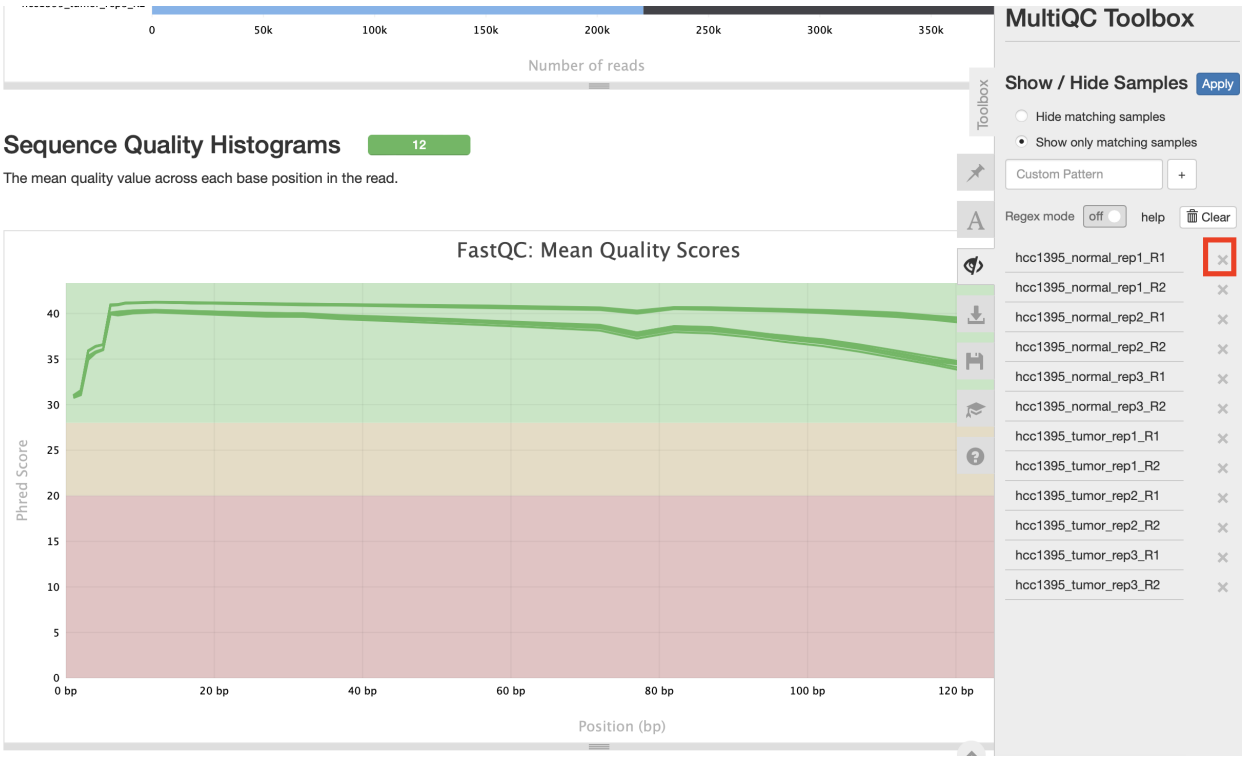
The average quality score of the sequencing reads in FASTQ files along each base position is shown in the figure below. Regarding the boxes at the top of the QC plots, green means QC passed while orange and red indicate warning and failed, respectively.



Click on the green rectangle labeled with the number of files to filter out samples to see in the plot.

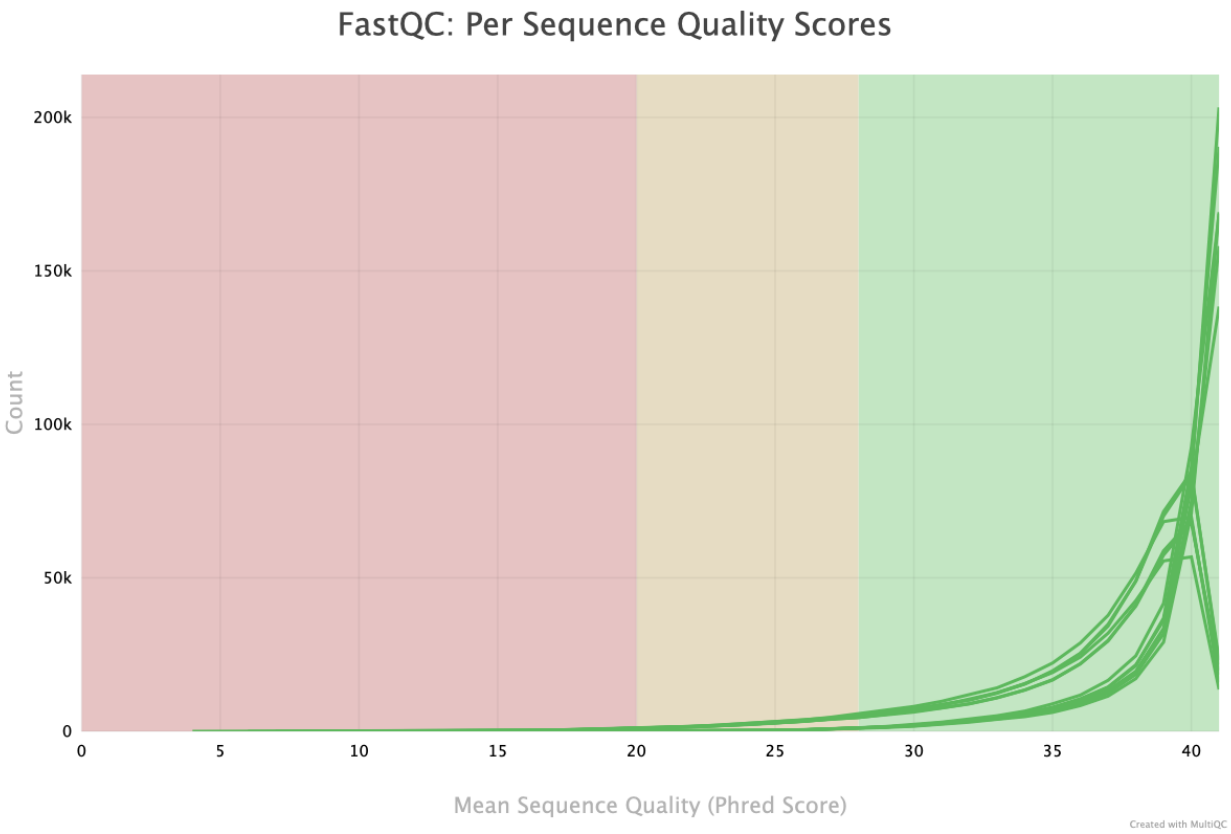


Upon clicking on the green rectangled labeled with the number of files, a tool box will appear and users can click on the "x" next to each file to deselect and remove it from the visualization.



MultiQC Quality Score Distribution

The quality score distributions of each of the FASTQ files are plotted in the next figure.

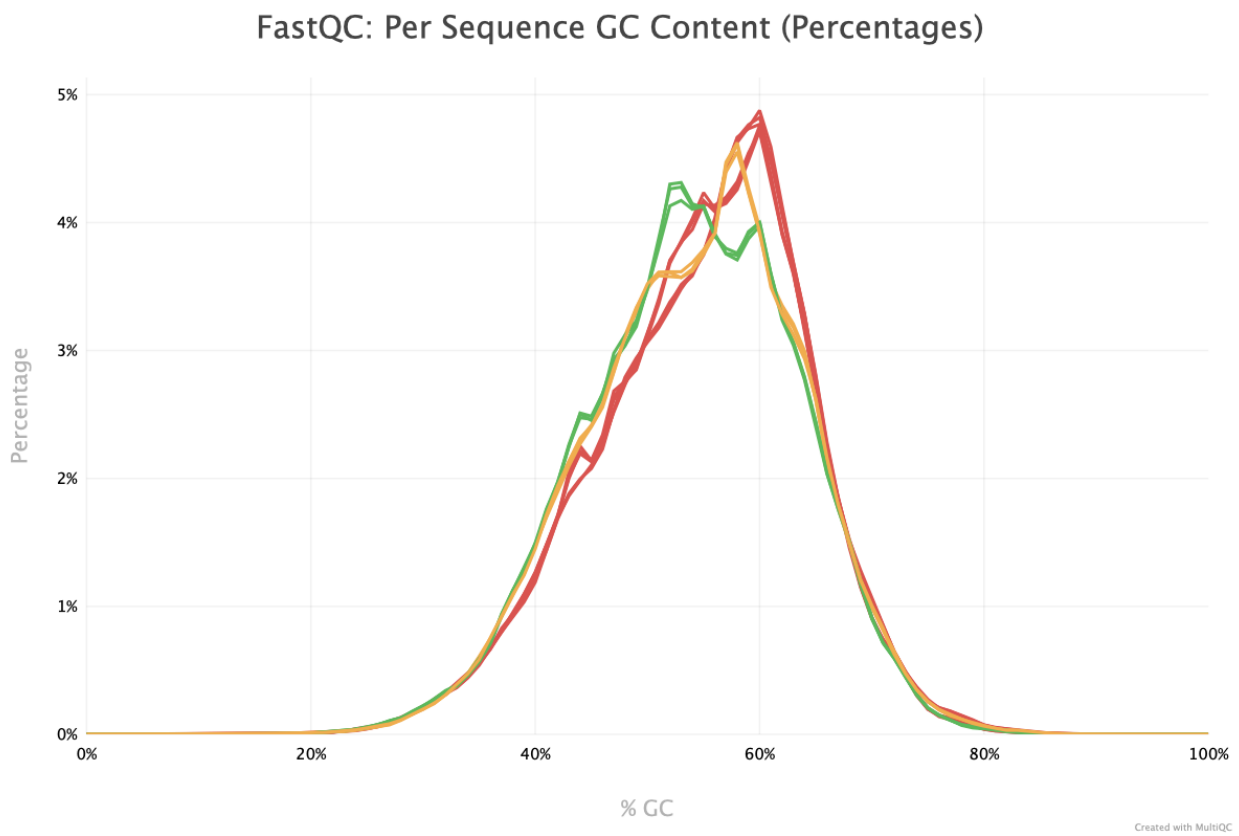


MultiQC Per Base Sequence Content

An interactive heatmap of the percent composition of each nucleotide base (A,T,C,G) along the bases (horizontal axis) for each of the FASTQ files (vertical axis) is presented next. Hover over a tile to see the corresponding numbers for a given sample. Click on any row in this heatmap to get the base composition plot for just that sample.

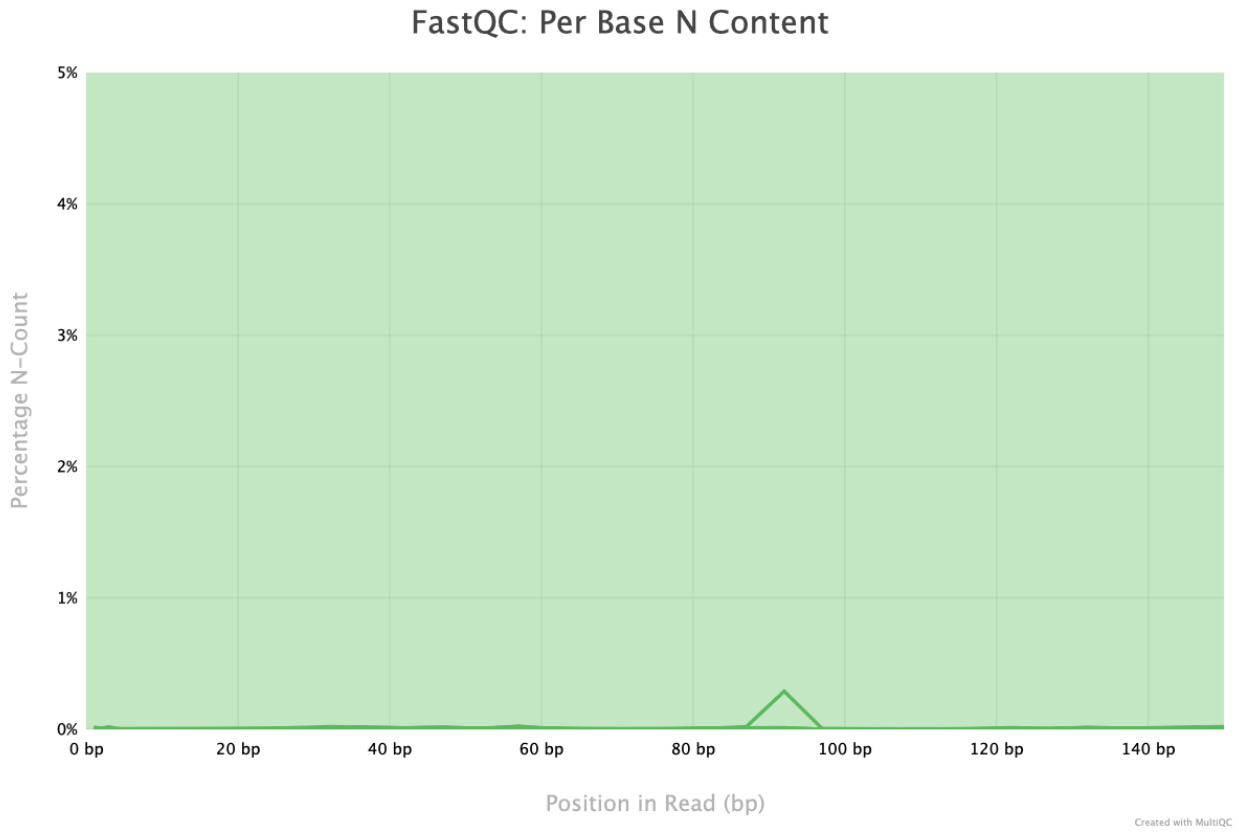
MultiQC GC Distribution

The GC distribution for each of the FASTQ files is shown in figure below.



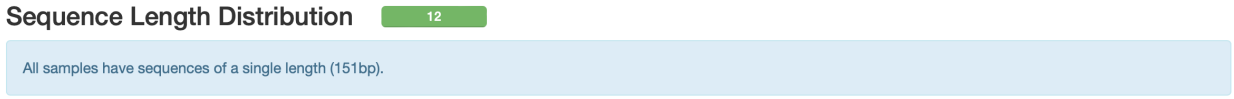
MultiQC Per Base N Content

The percentage of unknown bases (or N) per FASTQ file is available as the next figure.



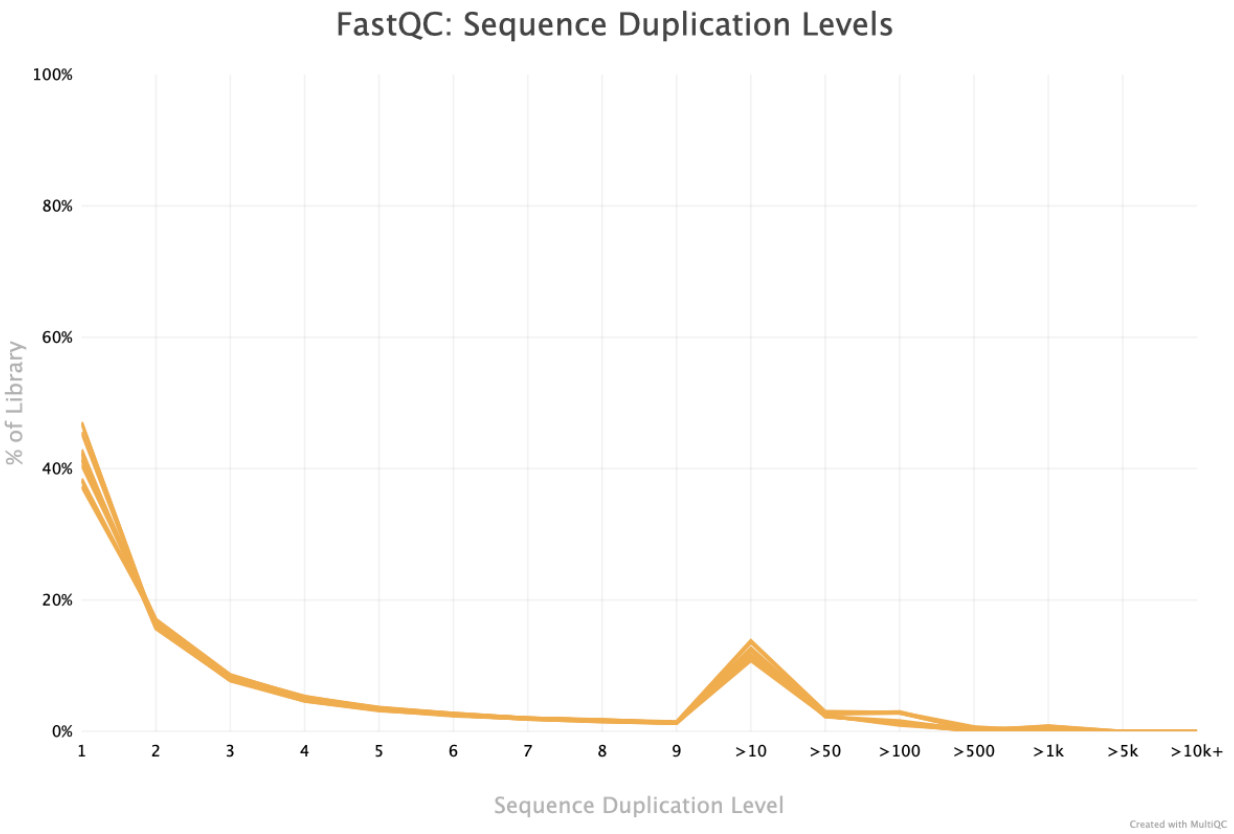
MultiQC Sequence Length Distribution

As shown below, all FASTQ files for the hcc13995 study have sequences that contain 151 bases.



MultiQC Duplication Level

The percentage of sequences with the duplication level shown on the horizontal axis.

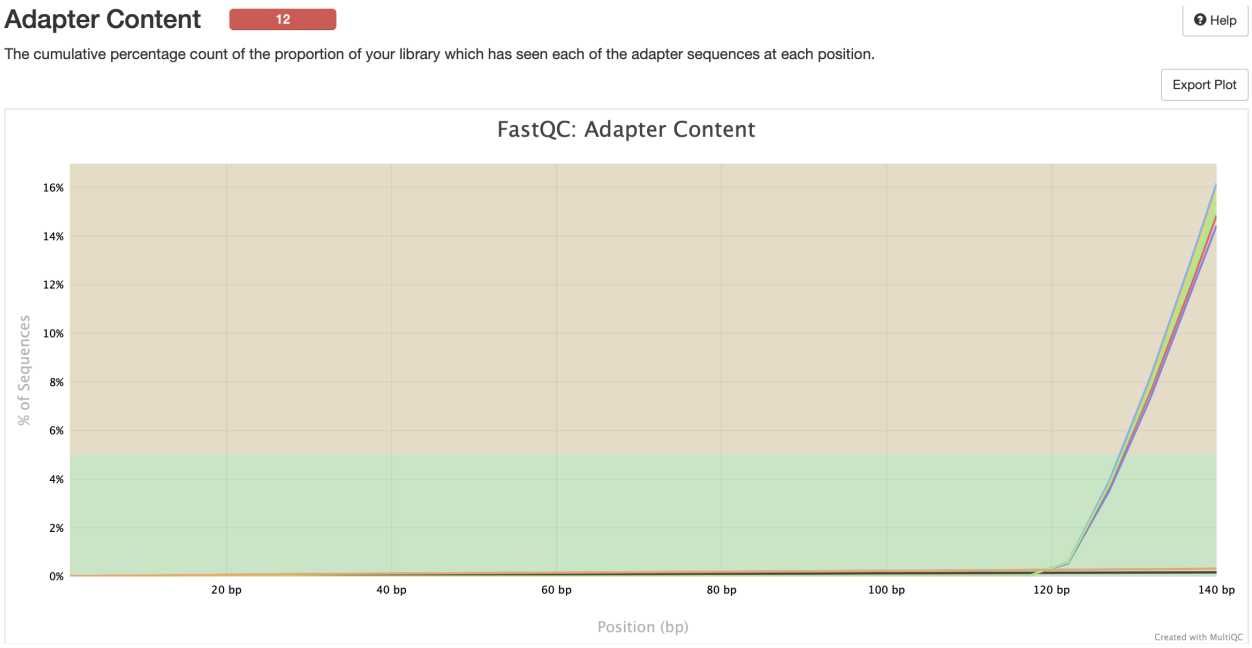


MultiQC Adapter Contamination

The sequences in the hcc1395 data have adapter contamination. Mousing over the area where the adpater content starts to increase suggests Illumina Universal Adapter.

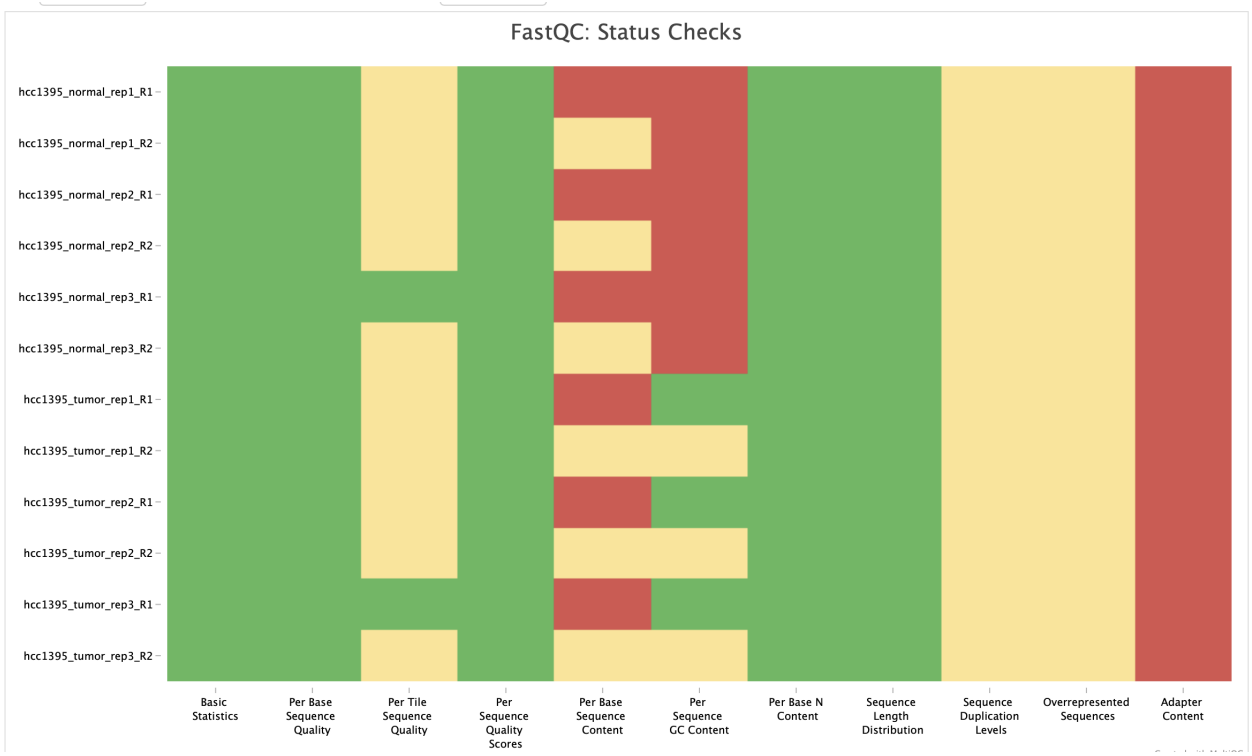
Note

FASTQC will issue a warning if any sequence is present in more than 5% of all reads and failure if any sequence is present in more than 10% of all reads. -- FASTQC manual *(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/10%20Adapter%20Content.html)*



MultiQC Status Heatmap

Finally, a heatmap showing the per file QC modules that have passed, warn, or failed is provided.



View FASTQC and MultiQC HTML Reports

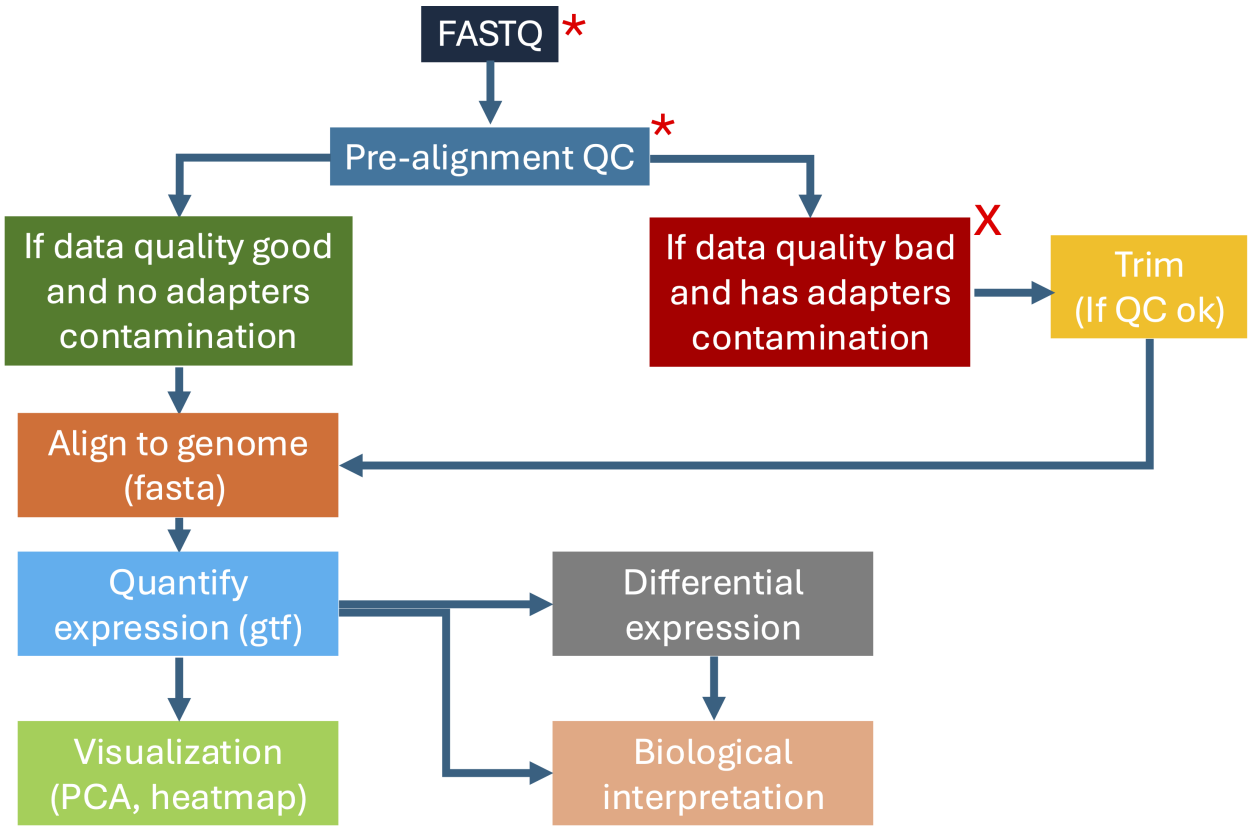
Untrimmed FASTQ Files

- hcc1395_normal_rep1_R1
- hcc1395_normal_rep1_R2
- hcc1395_normal_rep2_R1
- hcc1395_normal_rep2_R2
- hcc1395_normal_rep3_R1
- hcc1395_normal_rep3_R2
- hcc1395_tumor_rep1_R1
- hcc1395_tumor_rep1_R2
- hcc1395_tumor_rep2_R1
- hcc1395_tumor_rep2_R2
- hcc1395_tumor_rep3_R1
- hcc1395_tumor_rep3_R2
- MultiQC

Lesson 8: Cleaning and Preparing Next Generation Sequencing (NGS) Data for Downstream Analysis

Lesson 7 Review

Lesson 7 introduced the FASTQ file, which is the format used to store Next Generation Sequencing (NGS) data. Participants also learned about assessing quality of the sequences in FASTQ files using the tool FASTQC. This step is essential as it will inform whether sequencing is of high quality and if there are contamination such as adapters in the sequences. QC results revealed that while sequencing data quality is good, there are adapters in the sequences. Adapters will interfere with the mapping stage where algorithms are used to determine where in the genome the sequences came from. Thus, the next step is to trim away the adapters and perform QC again on the trimmed data to make sure the contamination has been removed.



Learning objectives

At the end of this session, participants will be able to use the tool Trimmomatic *(http://www.usadellab.org/cms/?page=trimmomatic)* to remove adapters from the NGS data.

Sign onto Biowulf and Request an Interactive Session

Before getting started, sign onto Biowulf and request an interactive session. In the `ssh` command below, replace user with the participant's assigned Biowulf student ID.

```
ssh user@biowulf.nih.gov
```

Change into `/data/user/hcc1395_b4b`.

```
cd /data/user/hcc1395_b4b
```

Next, request an interactive session with 12 gb of RAM and 10 gb of local temporary storage. The option `--cpus-per-task` is used to request 6 CPUs on Biowulf in the `sinteractive` command below in addition to the 12 gb of memory and 10 gb of local temporary storage.

```
sinteractive  --cpus-per-task 6 --mem=12gb --gres=lscratch:10
```

Adapter Trimming with Trimmomatic

Load Trimmomatics

Trimmomatic is a tool that can remove both low quality sequence and adapters. The first step to using Trimmomatic on Biowulf is to load it.

```
module load trimmomatic
```

> **Tip**
>
> Other tools used for trimming include bbduk *(https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/bbduk-guide/)* and Cutadapt *(https://cutadapt.readthedocs.io/en/stable/)*. Both are capable of quality and adapter trimming.

Make a New Folder to Stored the Trimmed Reads

```
mkdir trimmed_reads
```

Stay in the `/data/user/hcc1395_b4b` folder for these exercises.

Run Trimmomatic

To run remove adapters for all FASTQ files in one go, the `parallel` command will be introduced. This command enables the analyst to run multiple tasks in parallel such as trimming of high throughput sequencing data. The command construct is broken down below.

- `cat` is used to print the hcc1395 samples ids stored in the file `hcc1395_sample_ids.txt`. Rather than printing to terminal, `|` is used to send the output from `cat` to `parallel`.
    - The Trimmomatic construct is enclosed in double quotes within the `parallel` command. The components are as follows.
        - `-j`: enables users to specify how many jobs to run in parallel (6 in this case since there are 6 samples).
        - `java -jar $TRIMMOJAR`: when running on Trimmomatics on Biowulf, start with this. It essentially runs the Trimmomatic Java archive located in the folder pointed to by environmental variable `$TRIMMOJAR`.
        - `PE`: this next piece tells Trimmomatic to expect paired end sequencing.
        - `phred33`: this indicates the quality score encoding, which is used by modern Illumina sequencers.
        - The input FASTQ files are specified next. Because the input FASTQ files are in the `reads` folder, this must be included in the path. `{}` in the file path acts as a place holder to store the sample ids sent by `cat`. All users have to do is to add the `_R1` and `_R2` part.
        - The outputs are specified next. These will be written to the `trimmed_reads` folder. Trimmed versions will be labeled with "_trimmed". The files labeled "unpaired" will store sequences where only one of the pair met the trimming threshold.
        - The adapter file (see `references/illumina_multiplex.fa`) is specified after the `ILLUMINACLIP` argument followed by some parameters to help Trimmomatic decide whether a portion of a sequence matches an adapter and continue with trimming. See the Trimmomatic manual to learn more about options and parameters *(http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf)*
        - The `MINLEN` argument allows users to specify a sequence length threshold after trimming. If the length of the sequence post trimming is shorter than this number than it will be discarded. The threshhold is set to 25 bases here. Short sequences may also interfere with alignment as they could be aligned to multiple spots in a genome.

```
cat hcc1395_sample_ids.txt | parallel -j 6 "java -jar $TRIMMOJAR PE ·
```

Note

When running the `parallel` command, users will see the message below regarding citation and donation to the creators. To turn off the citation, use `parallel --citation`.

Academic tradition requires you to cite works you base your article on. If you use programs that use GNU Parallel to process data for an article in a scientific publication, please cite:

Tange, O. (2024, December 22). GNU Parallel 20241222 ('Bashar'). Zenodo. https://doi.org/10.5281/zenodo.14550073

This helps funding further development; AND IT WON'T COST YOU A CENT. If you pay 10000 EUR you should feel free to use GNU Parallel without citing.

More about funding GNU Parallel and the citation notice: https://www.gnu.org/software/parallel/parallel_design.html#citation-notice

To silence this citation notice: run 'parallel --citation' once.

Come on: You have run parallel 166 times. Isn't it about time you run 'parallel --citation' once to silence the citation notice?

Run QC on Trimmed FASTQ Files

Make a directory `pre_alignment_qc_trimmed` to store the QC results for the adapter trimmed FASTQ files.

```
mkdir pre_alignment_qc_trimmed
```

Then load FASTQC and MultiQC.

```
module load fastqc
```

```
module load multiqc
```

In the `fastqc` construct below, specify the path to the trimmed FASTQ files, which are located in the folder `trimmed_reads`. Then use `*trimmed*.fq` to get FASTQC to check all of the trimmed FASTQ files (`*` is used as a wild card). Write the results into the `pre_alignment_qc_trimmed` folder using the `-o` option.

```
fastqc trimmed_reads/*trimmed*.fq -o pre_alignment_qc_trimmed/
```

Next, combine the FASTQC reports for the trimmed data into one using MultiQC. In the `multiqc` command below, specify the path the FASTQC reports (`pre_alignment_qc_trimmed`) and then use `--filename` option to assign a base name to

the report (ie. `hcc1395_multiqc`, which will write a the report to the file `hcc1395_multiqc.html`).

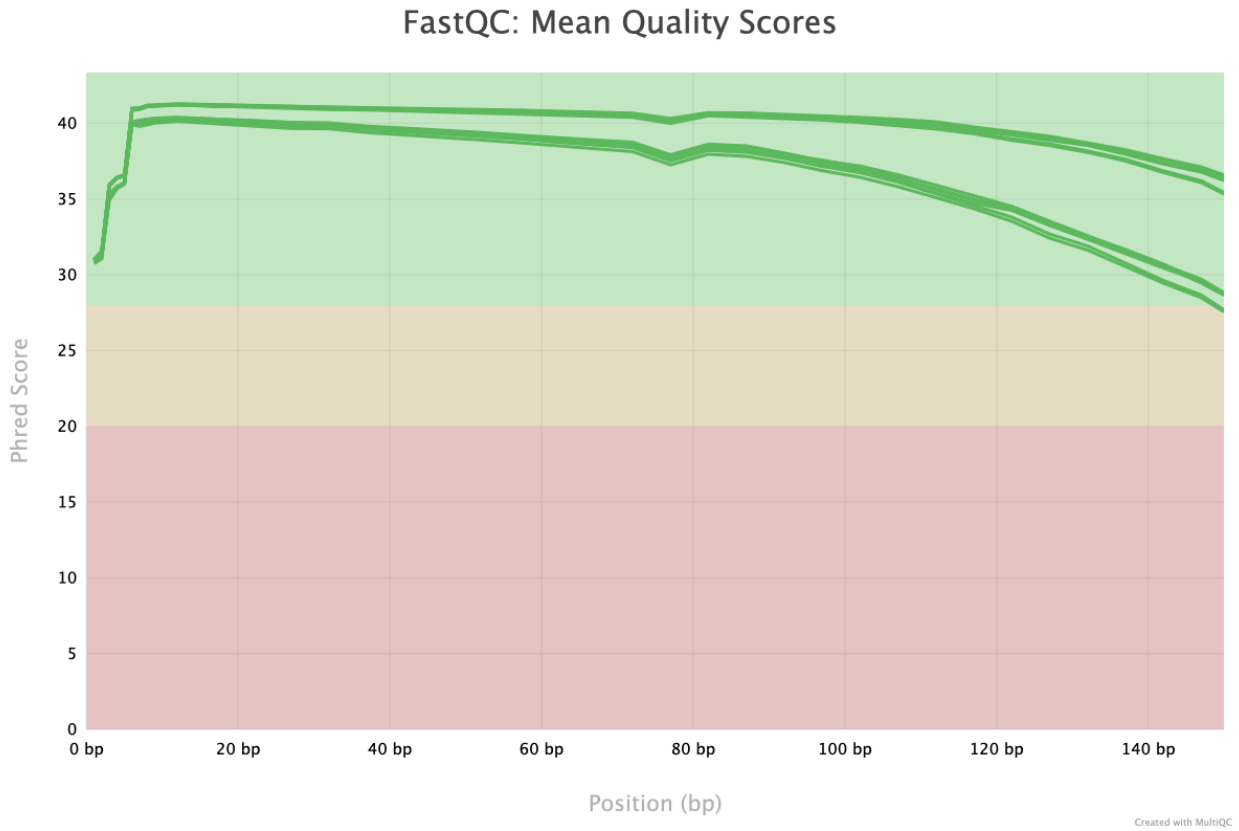
```
multiqc pre_alignment_qc_trimmed/ --filename hcc1395_multiqc
```

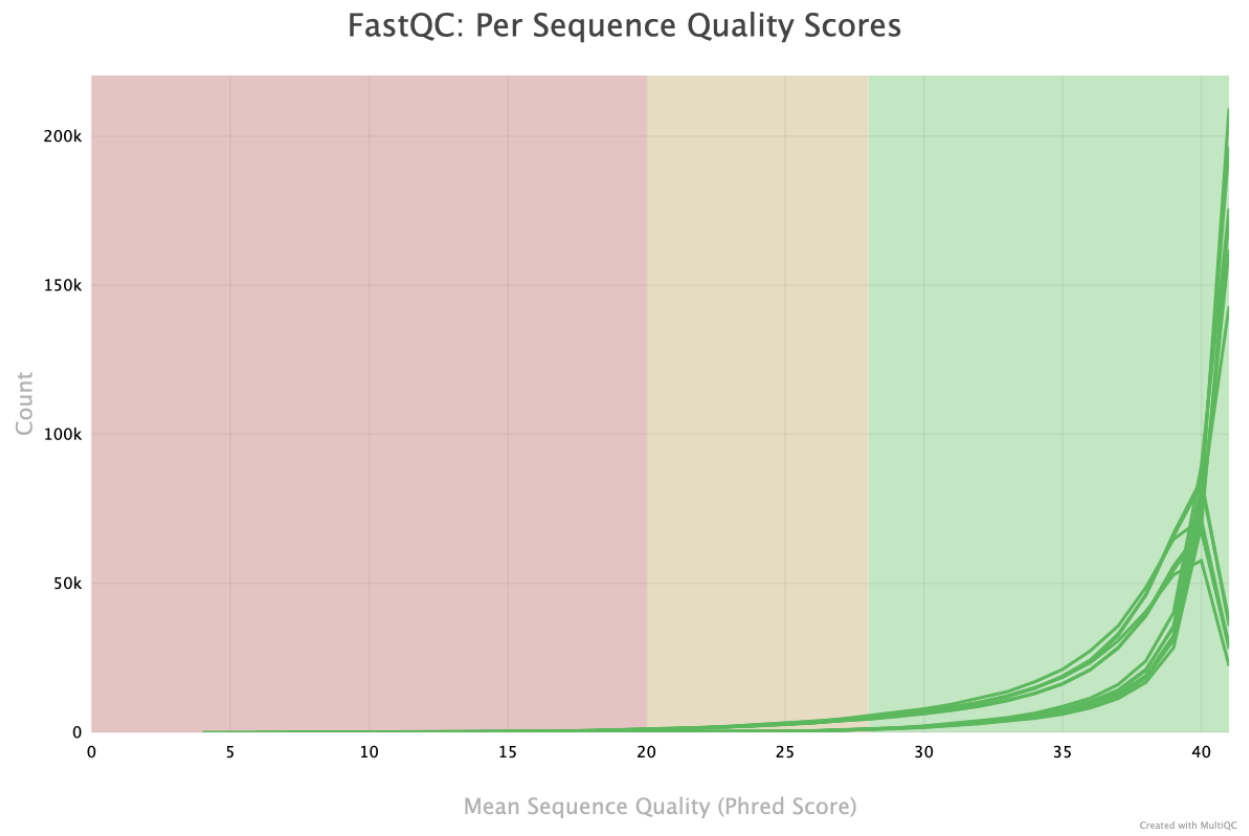
Then use `scp` to copy `hcc1395_multiqc.html` to local `Downloads` folder.

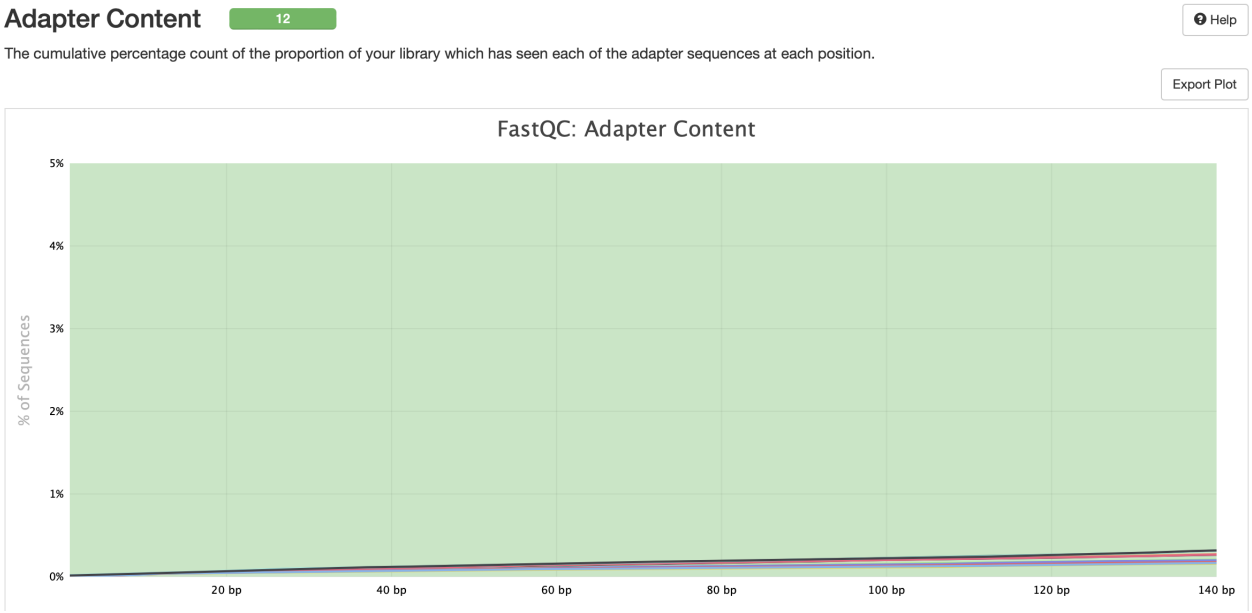
```
scp user@helix.nih.gov:/data/user/hcc1395_b4b/hcc1395_multiqc.html .
```

Adapter Trimming Conclusion

Adapter trimming did not influence the quality scores of the FASTQ files but sequences were cleaned of adapter contamination.







View FASTQC and MultiQC HTML Reports

Adapter Trimmed FASTQ Files

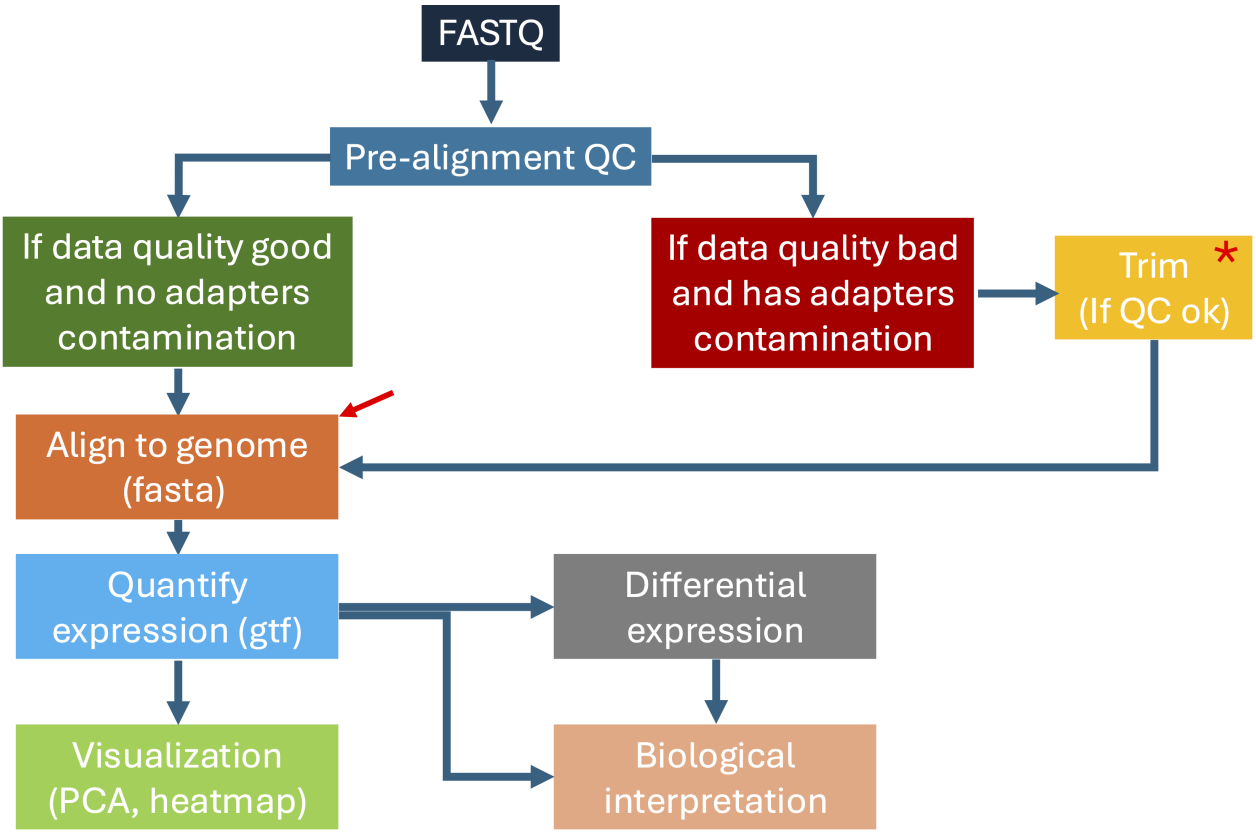
- hcc1395_normal_rep1_R1
- hcc1395_normal_rep1_R2
- hcc1395_normal_rep2_R1

- hcc1395_normal_rep2_R2
- hcc1395_normal_rep3_R1
- hcc1395_normal_rep3_R2
- hcc1395_tumor_rep1_R1
- hcc1395_tumor_rep1_R2
- hcc1395_tumor_rep2_R1
- hcc1395_tumor_rep2_R2
- hcc1395_tumor_rep3_R1
- hcc1395_tumor_rep3_R2
- MultiQC

Lesson 9: Aligning Next Generation Sequencing (NGS) Data to Genome

Lesson 8 Review

In Lesson 8, participants were introduced to the tool Trimmomatic, which can be used to remove low quality reads and adapter contamination from sequencing data. For the hcc1395 dataset, only adapters had to be removed. Subsequent QC of the trimmed data revealed that adapter contamination was removed. Thus, the next step will determine where in the genome each of the sequences in the trimmed FASTQ files for hcc1395 came from in a process known as the alignment or mapping.



Learning objectives

After this lesson, participants will be able to:

- Describe a reference genome.
- Understand why a splice aware aligner is needed in RNA sequencing.
- Perform alignment using the tool HISAT2 *(http://daehwankimlab.github.io/hisat2/)*.

Sign onto Biowulf

Prior to getting started, sign onto Biowulf.

```
ssh user@biowulf.nih.gov
```

Next, change into the `/data/user/hcc1395_b4b` folder.

```
cd /data/user/hcc1395_b4b
```

Finally, request an interactive session with the following resources. The option `--cpus-per-task` is used to request 6 CPUs on Biowulf in the `sinteractive` command below in addition to the 12 gb of memory and 10 gb of local temporary storage.

```
sinteractive --cpus-per-task 6 --mem=12gb --gres=lscratch:10
```

The Reference Genome

The reference genome is a completely assembled sequence (ie. the order in which the nucleotides are arranged is known). For high throughput sequencing, the known sequence help scientists find out where in the genome each of the reads came from. The reference genome in a way acts like a template that researchers can follow to reconstruct the genome of the unknown. In other words, think of the reference genome as a picture of the completed puzzle that helps guide the assembly of the actual puzzle, by allowing the overlap of pieces to see where they fit in the completed version. The file `22.fa` contains the reference genome for human chromosome 22.

> **Tip**
>
> Biowulf staff has downloaded and made the genome as well as annotations available on the cluster (see https://hpc.nih.gov/refdb/index.php *(https://hpc.nih.gov/refdb/index.php)*) so users generally will not need to download anything on their own.

Change into the `references` folder in `/data/user/hcc1395_b4b` for this example. Since `/data/user/hcc1395_b4b` should be the current working directory, just do the following.

```
cd references
```

Use the `head` command to view the first 10 lines of `22.fa`.

```

```

```
head 22.fa
```

A FASTA (.fa) file starts with a header line that contains a ">" followed by the name or some annotation of the sequence (ie. chr22). The first 10 lines indicate unknown sequences (denoted by N's).

```
>chr22
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

Note

The header line in a FASTA file can provide more information, depending on how the sequence was curated. As an example, in the human ADSL transcript nucleotide sequence *(https://www.ncbi.nlm.nih.gov/nuccore/NM_001363840.3?report=fasta)*, the header line provides the accession number or sequence ID (NM_001363840.3), species in which the sequence was derived (Homo sapiens), name of the gene (ADSL) where the transcript originated, and that this is a mRNA sequence.

How many lines does `22.fa` contain? To find out, use the `wc` command with `-l` option. The `-l` options tells `wc` to count the number of lines in a file.

```
wc -l 22.fa
```

The output of `wc -l` is the line number count followed by the name of the input file (ie. `22.fa`). The result below indicates that `22.fa` had 846976 lines.

```
846976 22.fa
```

But are there actual sequences in `22.fa` (ie. A, T, C, or, G)? To find out, use the `grep` command with the following options for pattern searching.

- `-n`: instructs `grep` to retrieve the line number of the file where the pattern is found.
- `-E`: treats the pattern as an extended regular expression *(https://www.google.com/search?client=safari&rls=en&q=what+is+an+extended+regular+expression&ie=UTF-8&oe=UTF-8)*.

- Within the single quotes is the search pattern A, T, C, or G where each is separated by "|".
- The `grep` output is sent via `|` (pipe) to the `head` command to print the first 10 lines of the results.

```
grep -n -E 'A|T|C|G' 22.fa | head
```

The results of the above grep command shows that the first line with actual nucleotide sequences is 175168.

```
175168:NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNGAATTCTTGTGTTTATATAA
175169:TAAGATGTCCTATAATTTCTGTTTGGAATATAAAATCAGCAACTAATATGTATTTTCAAA
175170:GCATTATAAATACAGAGTGCTAAGTTACTTCACTGTGAAATGTAGTCATATAAAGAACAT
175171:AATAATTATACTGGATTATTTTTAAATGGGCTGTCTAACATTATATTAAAAGGTTTCATC
175172:AGTAATTCATTATATCAAAATGCTCCAGGCCAGGCGTGGTGGCTTATGCCTGTAATCCCA
175173:GCACTTTGGGAGGTCGAAGTGGGCGGATCACTTGAGGTCAGGAGTTGGAGACTAGCCTGG
175174:CCAACATGATGAAACCCCGTCTCTAATAATAATAATAAAAAAAAATTAGCTGGGTGTGGT
175175:GGTGGGCAACTGTAATCTCAGCTAATCAGGAGGCTGAGGCAGAAGAATTGCTTGAACGTG
175176:GAAGACAGAGTTTACAGTGTGCCAAGATCACACCACCCTACTCCAACTTGGGTGACAGAG
175177:CAAGACTCAGTCTCAAGGAAAAAAAAAAGCTCGAAAAATGTTTGCTTATTTTGGTAAAAT
```

How big is the `22.fa` reference genome (ie. how many bases)? To answer this, use the tool `seqkit` and it's `stats` feature.

Why is the size of the genome important?

Prior to the sequencing experiment, the size of the genome will help us determine the number of reads needed to achieve a certain coverage (https://www.illumina.com/documents/products/technotes/technote_coverage_calculation.pdf *(https://www.illumina.com/documents/products/technotes/technote_coverage_calculation.pdf)*).

After the experiment, we could use the size of the genome along with other information to decide the computing resources (ie. time and memory) needed for analysis. Chromosome 22 is the second smallest in humans *(https://medlineplus.gov/genetics/chromosome/22/)*, so it would be faster and require less compute power to align to this than the entire human genome, thus this dataset was chosen for the class.

```
module load seqkit
```

```
seqkit stat 22.fa
```

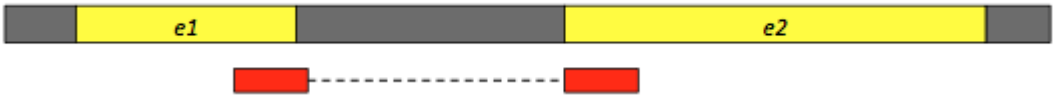
```
file   format  type  num_seqs     sum_len     min_len     avg_len
22.fa  FASTA   DNA          1  50,818,468  50,818,468  50,818,468  5(
```

Go back up one folder to `/data/user/hcc1395_b4b`.

```
cd ..
```

Aligning hcc1395 Sequencing Data to Reference Genome

The tool HISAT2 will be used to align the trimmed hcc1395 FASTQ files to the human chromosome 22 reference. RNA sequencing analyses require the use of splice aware aligners in order to map sequences that span across exons.



A sequencing read (red fragments) aligning two exons (e1 and e2). Modified from: https://training.galaxyproject.org/training-material/topics/transcriptomics/tutorials/rb-rnaseq/tutorial.html *(https://training.galaxyproject.org/training-material/topics/transcriptomics/tutorials/rb-rnaseq/tutorial.html)*

Build an Index

Users will need to build an index of the reference genome prior to aligning using HISAT2. This step essentially breaks down the reference into manageable pieces so that the alignment process is made more efficient.

Load HISAT2.

```
module load hisat2
```

```
 This is HISAT2 v 2.2.1, built for NCBI NGS 3.0.0
[+] Loading samtools 1.21  ...
```

Stay in `/data/user/hcc1395_b4b` for this exercise.

To build the index for HISAT2, use the `hisat2-build` command where the arguments are:

- Path to the reference genome FASTA (or .fa) file: the reference genome file in this case is `22.fa` and it is located in the `references` folder so the path starting from `/data/user/hcc1395_b4b` is `references/22.fa`.
- The folder to write the indices: in this case, write the indices to the `reference` folder. Thus, the path starting from `/data/user/hcc1395_b4b` is `references/22`. Note that an extension is not needed as HISAT2 will append `.ht` to the index files.

```
hisat2-build references/22.fa references/22
```

After indexing of `22.fa` has been completed, use `ls` with the `-1` option to view the contents of the `reference` folder one item per line and confirm that 8 index files with `.ht` extension were generated in this step.

```
ls -1 references/
```

```
22.1.ht2
22.2.ht2
22.3.ht2
22.4.ht2
22.5.ht2
22.6.ht2
22.7.ht2
22.8.ht2
```

Alignment

Make a new folder `hcc1395_hisat2` to store the HISAT2 alignment results.

```
mkdir hcc1395_hisat2
```

Stay in `/data/user/hcc1395_b4b` for this exercise.

To perform alignment for all samples at once, the `parallel` command will be used. Below is the explanation of the components.

- `cat hcc1395_sample_ids.txt`: print the hcc1395 sample ids and `|` will send the output to `parallel` rather than printing to screen.

- The HISAT2 alignment command is enclosed within double quotes within `parallel`. The alignment starts with the command `hisat2` and arguments and options below are include:
- `-j`: enables users to specify how many jobs to run in parallel (6 in this case since there are 6 samples).
- `-x`: prompts user to enter the path to the reference genome indice (ie. `references/22`, note that the `ht2` extension is not needed)
- `-1`: prompts for the first read in the pair for paired end sequencing.
    - The data is in the `trimmed_reads` folder.
    - `{}` is used as a place holder for the sample ids sent by `cat`.
- `-2`: prompts for the second read in the pair for paired end sequencing.
- `-S`: prompts the output SAM file path. The alignment output will be stored in the folder `hcc1395_hisat2`.
- `--summary-file`: prompts for the text (txt) file that summarizes alignment statistics. The summary will be stored in the folder `hcc1395_hisat2`.

```
cat hcc1395_sample_ids.txt | parallel -j 6 "hisat2 -x references/22 ·
```

After alignment completes, list the contents of hcc1395_hisat2 to see what outputs were generated.

```
ls -1 hcc1395_hisat2
```

The folder `hcc1395_hisat2` contains the aligned data in `.sam` format and alignment summary text files (ie. those labeled with `summary.txt`).

```
hcc1395_normal_rep1_hisat2_summary.txt
hcc1395_normal_rep1.sam
hcc1395_normal_rep2_hisat2_summary.txt
hcc1395_normal_rep2.sam
hcc1395_normal_rep3_hisat2_summary.txt
hcc1395_normal_rep3.sam
hcc1395_tumor_rep1_hisat2_summary.txt
hcc1395_tumor_rep1.sam
hcc1395_tumor_rep2_hisat2_summary.txt
hcc1395_tumor_rep2.sam
hcc1395_tumor_rep3_hisat2_summary.txt
hcc1395_tumor_rep3.sam
```

Change into the `hcc1395_hisat2` directory.

```

```

```
cd hcc1395_hisat2
```

Use `cat` to take a look at the alignment summary for hcc1395_normal_rep1.

```
cat hcc1395_normal_rep1_hisat2_summary.txt
```

```
331945 reads; of these:
  331945 (100.00%) were paired; of these:
    43449 (13.09%) aligned concordantly 0 times
    283794 (85.49%) aligned concordantly exactly 1 time
    4702 (1.42%) aligned concordantly >1 times
    ----
    43449 pairs aligned concordantly 0 times; of these:
      10336 (23.79%) aligned discordantly 1 time
    ----
    33113 pairs aligned 0 times concordantly or discordantly; of thes
      66226 mates make up the pairs; of these:
        35020 (52.88%) aligned 0 times
        30758 (46.44%) aligned exactly 1 time
        448 (0.68%) aligned >1 times
94.73% overall alignment rate
```

Users will see the alignment statistics shown above and reports are generated for all samples. The first line of the HISAT2 alignment statistics says of 331945 reads (100.00%) were paired. Recall from FASTQC that read 1 and read 2 FASTQ files for hcc1395_normal_rep1 each has 331945 reads after trimming. So the first line in the HISAT2 alignment statistics is saying that out of all the reads from read 1 and read 2 FASTQ files of hcc1395_normal_rep1, there are 331945 pairs. This means there are 331945x2 or 663890 reads for the hcc1395_normal_rep1 sample.

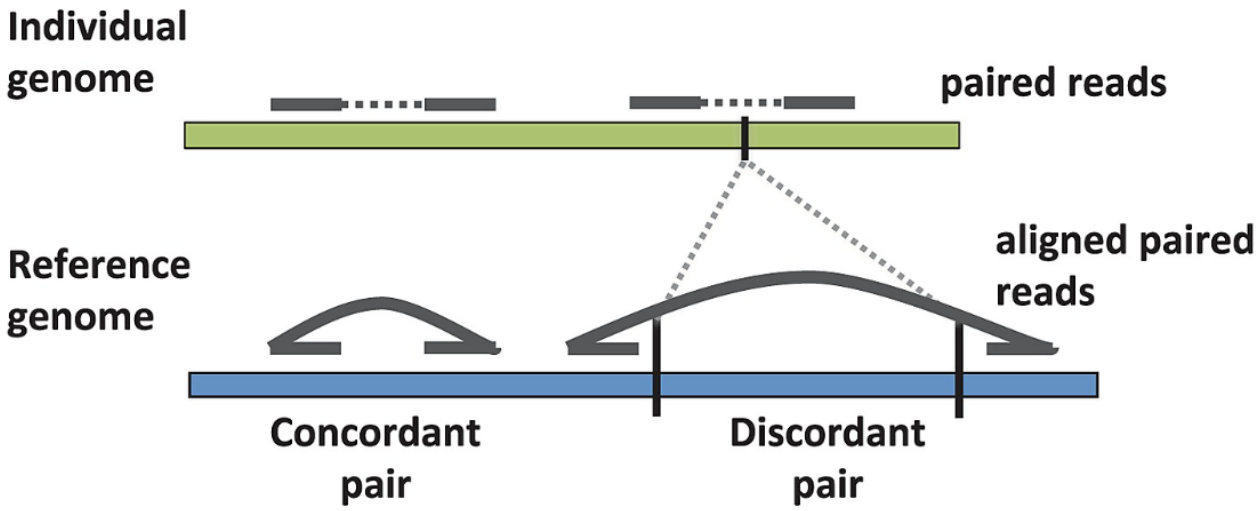
Basic Statistics

Measure	Value
Filename	hcc1395_normal_rep1_trimmed_R1.fq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	331945
Total Bases	49.2 Mbp
Sequences flagged as poor quality	0
Sequence length	34-151
%GC	54

Basic Statistics

Measure	Value
Filename	hcc1395_normal_rep1_trimmed_R2.fq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	331945
Total Bases	49.2 Mbp
Sequences flagged as poor quality	0
Sequence length	25-151
%GC	54

Following the first line of the HISAT2 alignment statistics, users will see some terminologies like concordant or discordant mapped reads. See the figure below for a visual explanation.



Source: Benjamin J. Raphael, Chapter 6: Structural Variation and Medical Genomics, PLOS Computational Biology, December 27, 2012 *(https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002821)*

- A concordant read pair is defined as those that align in an expected manner where the reads are oriented towards one another and the distance between the outer edges is within expected ranges.
- A discordant read pair is defined as those that do not align as expected such as where the distance between the outer edges is smaller or larger than expected.

For the hcc1395_normal_rep1 sample, there are:

- 283794 pairs (567588 reads) that aligned concordantly once
- 4702 pairs (9404 reads) that aligned concordantly more than once - these are mapped to multiple parts of the genome and likely cause by duplicated or similiar regions in the genome
- 10336 paris (20672 reads) that aligned discordantly once
- 30758 reads that did align (neither concordantly or discordantly) once
- 448 reads that aligned (neither concordantly or discordantly) more than once

Sum up the number of reads that mapped in the above break down and divide by the total number of reads in the two FASTQ files for the hcc1395_normal_rep1 sample to get an overall alignment rate of 94.73%.

SAM File

Aligned sequencing data are stored in the form of SAM files *(https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://samtools.github.io/hts-specs/SAMv1.pdf&ved=2ahUKEwiS2sSX8ImKAxWuF1kFHbE1NSYQFnoECBQQAQ&usg=AOvVaw0MUgu2LimUqu*
The SAM file is human readable so users can use `cat`, `less`, and `head` to view its content in the terminal.

SAM Header

SAM files always start off with metadata information and these lines start with @. SAMTOOLS *(http://www.htslib.org)* can be used to view and manipulate SAM files.

```
module load samtools
```

To view the header for hcc1395_normal_rep1.sam, use the following `samtools` command construct, where:

- `head`: this argument is used to view SAM file header.
- `hcc1395_normal_rep1.sam`: SAM file in which the header should be viewed.

```
samtools head hcc1395_normal_rep1.sam
```

- @HD includes
- SAM file format information (in this case version 1.0 as indicated by VN:1.0)
- Whether the alignment file has been sorted (in this case no as indicated by SO:unsorted)
- @SQ provides reference information
- SN denotes reference sequence name, which is chr22
- LN is the reference length in bases, which is 50818468
- @PG provides information about the program that was used to generate the alignment
- ID is the program record identifier
- PN is the program name (HISAT2 and version 2.2.1 was used as indicated next to VN)
- CL is the command line call used to generate the alignment

After the metadata information, each SAM file contains 11 fields.

SAM File Information

```
samtools view hcc1395_normal_rep1.sam | head -1 | column -t | less -S
```

```
K00193:38:H3MYFBBXX:4:1101:10003:44458  99  chr22  31282436  60  151M
```

1. QNAME or query template name, which is essentially the sequencing read that was mapped onto a reference genome. Use the `grep` to search for any template in the SAM file and users will retrieve a sequence in the corresponding FASTQ file.
2. The second column is a FLAG that provides detail on mapping. See https://broadinstitute.github.io/picard/explain-flags.html *(https://broadinstitute.github.io/picard/*

explain-flags.html) to learn about these FLAGS. For instance, the first alignment in the hcc1395_normal_rep1 SAM file has a FLAG of 99, which indicates that:

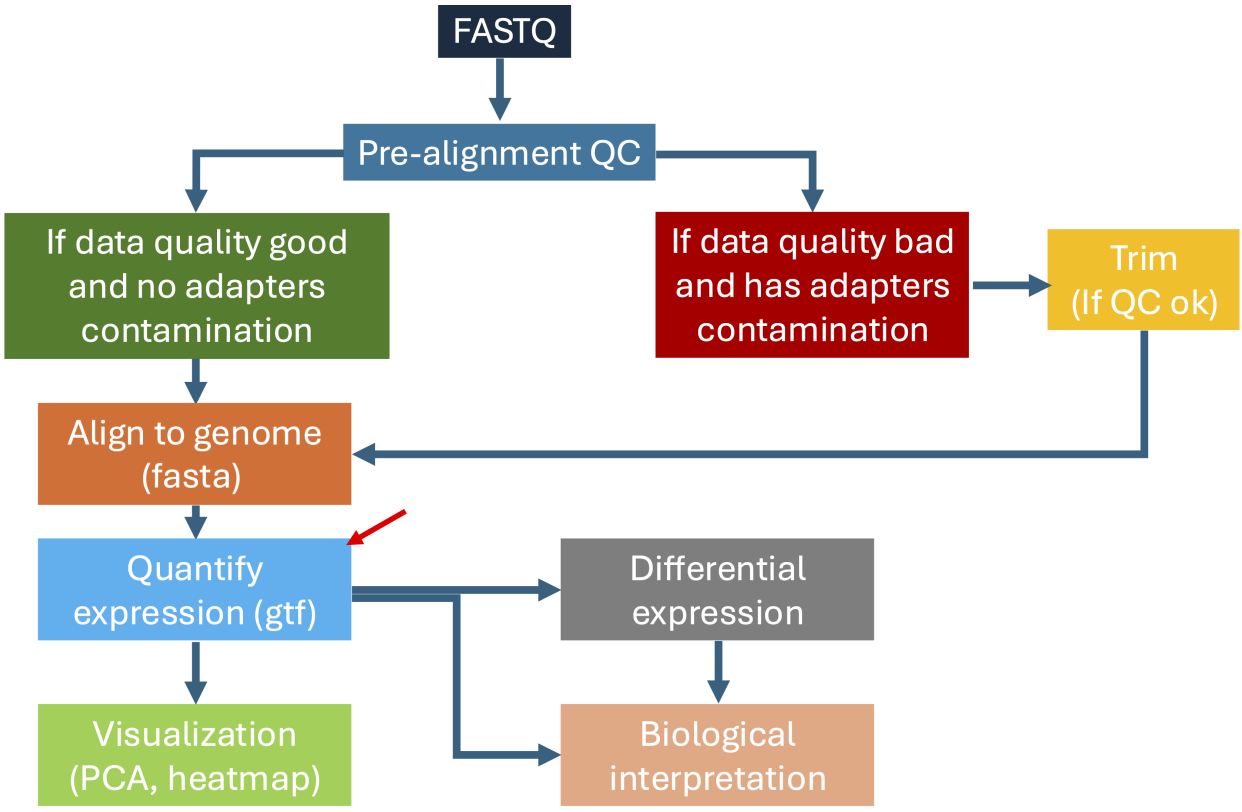
    1. The read is properly paired.
    2. Read is mapped in proper pair.
    3. The mate is in the reverse strand.
    4. The read is the first in a pair.

3. Column three contains the name of our reference genome (ie. `chr22`)
4. Column four provides the left most genomic coordinate where the sequencing read maps.
5. The mapping quality (MAPQ) is provided in the fifth column (the higher the number, the less likely that the mapping is due to error); a value of 255 in this column means that the mapping quality is not available.
6. Column six presents the CIGAR string, which informs about the match/mismatch, insertion/deletion, etc. of the alignment.
7. Column seven is the reference sequence name of the primary alignment of the NEXT read in the template. A "=" will appear if this name is the same as the current (which is expected for paired reads)
8. The alignment position of the next read in the template is provided in column eight. When this is not available, the value is set to 0.
9. Column nine provides the template length (TLEN), which should reflect the DNA fragment that was size selected for during library preparation.
10. The tenth column is just the sequencing read (some are written as the reverse complement so be cautious. The FLAGS in column two will tell whether the sequence is reverse complemented).
11. The eleventh column is the Phred quality scores of the sequencing read.
12. For definition of optional fields, see https://samtools.github.io/hts-specs/.

Lesson 10: Quantifying Gene Expression from bulk RNA Sequencing Data

Lesson 9 Review

In Lesson 9, participants learned to align the hcc1395 sequencing data to reference genome in order to determine where in the genome each of the sequences came from. The next step is to quantify at each gene, how many sequences mapped to it, which can be used to approximate expression.



Learning objectives

After this lesson, participants will be able to:

- Convert SAM files to machine readable BAM.
- Quantify gene expression from alignment results.

Sign onto Biowulf

Before getting started, sign onto Biowulf. Remember to replace `user` with the participant's assigned Biowulf student ID.

```
ssh user@biowulf.nih.gov
```

Then, change into the `/data/user/hcc1395_b4b` folder.

```
cd /data/user/hcc1395_b4b
```

Next, request an interactive session with 6 CPUs, 12 gb of RAM or memory and 10 gb of local temporary storage.

```
sinteractive --cpus-per-task 6 --mem=12gb --gres=lscratch:10
```

Converting SAM to BAM

SAM files are human readable. Many bioinformatics programs take the binary version of SAM files known as BAM for input. Below, SAMTOOLS will be used to convert the hcc1395 SAM alignment outputs to BAM.

```
module load samtools
```

Change into the `hcc1395_hisat2` folder from `/data/user/hcc1395_b4b` and list the contents.

```
cd hcc1395_hisat2
```

```
ls
```

The following SAM alignment files should be present.

```
hcc1395_normal_rep1.sam
hcc1395_normal_rep2.sam
hcc1395_normal_rep3.sam
hcc1395_tumor_rep1.sam
```

```
hcc1395_tumor_rep2.sam
hcc1395_tumor_rep3.sam
```

The command construct below can be used to convert the SAM files to BAM and it can be explained as follows:

- `cat` is used to print the hcc1395 sample IDs to the terminal. The sample IDs are stored in a text file called `hcc1395_sample_ids.txt` one folder up, so it's path can be specified as `../hcc1395_sample_ids.txt`. The output from this `cat` command is sent via pipe (`|`) to `parallel`. The `samtools` command used to convert SAM to BAM is enclosed in parentheses. Within `samtools`:
- `sort` will sort the alignment output by coordinate (see http://www.htslib.org/doc/samtools-sort.html *(http://www.htslib.org/doc/samtools-sort.html)* for more information).
- `-o`: prompts for the output file name, which has the `.bam` extension and prepended by the sample ID (specified using `{}` as a place holder) sent by `cat`.
- `{}.sam` is the input SAM file. Again, `{}` is used as a place holder for the sample IDs sent by `cat`.

```
cat ../hcc1395_sample_ids.txt | parallel -j 6 "samtools sort -o {}.ba
```

The next step is to index the BAM files. Again, `cat` and `parallel` will be used to performing the indexing step for all BAM files at once. The `index` subcommand of `samtools` is used for indexing. The option `-b` states create an index with extension `.bai`. The input BAM file follows the `-b` option and the output `.bai` file name is specified next.

```
cat ../hcc1395_sample_ids.txt | parallel -j 6 "samtools index -b {}.b
```

Quantifying Gene Expression

Go back to `/data/user/hcc1395_b4b`.

```
cd /data/user/hcc1395_b4b
```

Create a directory called `hcc1395_featurecounts`.

```
mkdir hcc1395_featurecounts
```

Go back into `/data/user/hcc1395_b4b/hcc1395_hisat2` for this exercise.

The module `featureCounts` *(http://bioinformatics.oxfordjournals.org/cgi/reprint/btt656?ijkey=ZzPz96t2lqzAH6F&keytype=ref)* from the `subread` *(https://subread.sourceforge.net/)* package will be used to quantify gene expression. So before getting started do the following.

```
module load subread
```

In the `featureCounts` command below:

- `-p`: specifies the presence of paired end data.
- `--countReadPairs`: specifies to count both reads in a pair.
- `-a`: prompt for path the `gtf` annotation file, which is locate one directory up from the current directory `hcc1395_hisat2` (specified as `../`) and in the `references` folder with file name `22.gtf`, so the path would be `../references/22.gtf`.
- `-g`: indicates to report expression by gene name as reported in the `gtf` file.
- `-o`: prompts for path to write the output, which is one directory up from the current of `hcc1395_hisat2` and in `hcc1395_featurecounts` with assigned file name `hcc1395_gene_expression.txt`. So the path for the output is `../hcc1395_featurecounts/hcc1395_gene_expression.txt`.
- `*.bam`: tells `featureCounts` to count all of the `bam` alignment output files in `hcc1395_hisat2`.

```
featureCounts -p --countReadPairs -a ../references/22.gtf -g gene_nar
```

After `featureCounts` has completed, change into the `/data/user/hcc1395_b4b/hcc1395_featurecounts` folder, which resides one folder up in `/data/user/hcc1395_b4b`.

```
cd ../hcc1395_featurecounts
```

`head -1` reveals that the first line in the `featureCounts` gene expression table is the program and command line call used to perform the quantification. This can be removed in order to generate a gene by sample expression matrix.

```
head -1 hcc1395_gene_expression.txt
```

```
# Program:featureCounts v2.0.6; Command:"featureCounts" "-p" "--count
```

To remove the header in the gene expression table `hcc1395_gene_expression.txt`, use `sed` where the option `1d` indicates to delete the first line in the file (ie. the header information). Then > will be use to write the output to `hcc1395_gene_expression_no_header.txt`.

```
sed '1d' hcc1395_gene_expression.txt > hcc1395_gene_expression_no_hea
```

Use `head -1` to view the first line in `hcc1395_gene_expression_no_header.txt`. The expression table contains the Geneid (gene names) in the first column, followed by chromosome coordinates and then on column for each sample with the estimated expression.

```
head -1 hcc1395_gene_expression_no_header.txt
```

```
Geneid  Chr    Start     End       Strand  Length  hcc1395_normal_rep
U2      chr22  10736171  10736283  -       113     0                 0
```

The chromosomal information in columns 2 thru 6 can be removed using the `cut` command below where `-f` prompts for the columns to extract (ie. gene IDs in column 1 and sample expression estimates in columns 7 thru 12) with the construct being `-f1,7-12`. The file to extract these columns from (ie. `hcc1395_gene_expression_no_header.txt`) is specified next. Then `|` is used to send the output of `cut` to `tr` which will swap out the tab ('\t') that separate the columns with comma essentially turning the gene expression table in to a comma separate value (CSV) file saved under the name `hcc1395_gene_expression.csv` using >.

```
cut -f1,7-12 hcc1395_gene_expression_no_header.txt | tr '\t' ',' > hc
```

The `column` and `head` commands can be used to view the first two lines of `hcc1395_gene_expression.csv` in a way that the columns are nicely aligned. In the `column` command `-t` indicates to create a table while `-s` prompts for specification of column separator (ie. comma for CSV files). The file in which the content is to be displayed is `hcc1395_gene_expression.csv`. The output of `column` is sent to `head` with the `-2` option to print the first two lines. The output indicates that this table has the Geneid (gene names) in the first column and the subsequent columns contain the expression estimates for each gene for each sample.

```
column -t -s ',' hcc1395_gene_expression.csv | head -2
```

```
Geneid              hcc1395_normal_rep1.bam  hcc1395_normal_rep2.bam
U2                  0                        0
```

Lesson 11: Visualizing Genomic Data: Preparing Files

Quick Review

The hcc1395 sequencing data were aligned to reference genome in Lesson 9. This enables the researcher to determine where in the genome each sequence from the Next Generation Sequencing experiment came from.

Learning objectives

Following this lesson, participants will know how to prepare sequencing alignment results for visual inspection using the Integrative Genomics Viewer *(https://igv.org)*.

Signing onto Biowulf

Before getting started, sign onto Biowulf.

```
ssh user@biowulf.nih.gov
```

Next, change into the `/data/user/hcc1395_b4b` directory.

```
cd /data/user/hcc1395_b4b
```

Finally, request an interactive session with 6 CPUs, 12 gb of RAM or memory and 10 gb of local temporary storage.

```
sinteractive --cpus-per-task 6 --mem=12gb --gres=lscratch:10
```

Preparing Alignment Output for Visualization Using IGV

This exercise will create bigWig files from the hcc1395 BAM alignment output. bigWig files have pre-computed sequencing depth information and will enable researchers to see areas in the genome where there are sequences mapped to it and subsequently zoom in on these regions to further interrogate.

Step 1: Convert BAM to bedGraph

The first step for generating bigWig files is to convert the BAM alignment results to bedGraph (with extension `bg`) files that contains sequencing depth along genomic regions (ie. how many sequences mapped to a genomic region).

> **Definitions**
>
> BED file: this is also known as Browser Extensible Format and contains three columns, which are chromosome, start coordinate and end coordinate -- https://genome.ucsc.edu/FAQ/FAQformat.html#format1 *(https://genome.ucsc.edu/FAQ/FAQformat.html#format1)*
>
> bedGraph: this has the same first three columns as the BED file but also has a fourth column that could be anything such as sequencing coverage -- https://genome.ucsc.edu/goldenPath/help/bedgraph.html *(https://genome.ucsc.edu/goldenPath/help/bedgraph.html)*

Example BED file is shown below -- source: https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html *(https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html)*

```
Chromosome  Start End
chr1        10    20
chr1        20    30
chr2        0     500
```

Example bedGraph is shown below -- source: https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html *(https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html)*

```
Chromosome  Start   End     Depth
chr1        554304  554309  5
chr1        554309  554313  6
chr1        554313  554314  1
chr1        554315  554316  6
chr1        554316  554317  5
chr1        554317  554318  1
chr1        554318  554319  2
chr1        554319  554321  6
chr1        554321  554323  1
chr1        554323  554334  7
```

The application `bedtools` *(https://bedtools.readthedocs.io/en/latest/index.html)* will be used to convert the hcc1395 BAM alignment outputs to bedGraph format. `bedtools` can be used for a range of tasks including compiling information on genomic intervals. The `genomecov` function

of `bedtools`, which calculates depth over a genomic range with the following options will be used:

- `-ibam`: prompts users to provide the name of the BAM input file(s).
- `-split`: when applied to RNA sequencing data, does not count reads that map to introns (see this post on why we get reads that map to introns in RNA sequencing *(https://www.biostars.org/p/42890/)*).
- `-bg`: reports sequencing depth along a genomic interval rather than at each nucleotide position (Figure 1 shows the ways that `bedtools` can report coverage information - where some options report coverage along an interval, some report at each base position. `bedtools` also enables the fine tuning of coverage along splice junctions are reported using the `split` option).

To use `bedtools` on Biowulf, do:

```
module load bedtools
```

To generate bedGraph file for all of the hcc1395 BAM alignments in one go, the command construct below can be used.

- `cat`: used to print the modified sample IDs from the hcc1395 study to the terminal (recall that these new sample IDs not have "hcc1395" at the beginning). The output is sent to `parallel` using `|` or pipe.
- `-j 6` in the `parallel` command tells it to perform 6 tasks simultaneously.
- The `bedtools genomecov` construct is enclosed in double quotes with `parallel`:
- The input BAM files are located in the folder `hcc1395_hisat2` and `{}` is used as a place holder for the sample ID sent by `cat`. Thus, the path to the BAM files can be written as `hcc1395_hisat2/{}.bam`.
- > will write the output to the `hcc1395_hisat2` folder into a file with sample name sent by `cat` and `.bg` extension. Thus, the path to the output is `hcc1395_hisat2/{}.bg`.

```
cat hcc1395_sample_ids.txt | parallel -j 6 "bedtools genomecov -ibam
```

Change into the `hcc1395_hisat2` folder after the bedgraph files have been made and use the `ls` command to ensure that that `.bg` files are there.

```
cd hcc1395_hisat2
```

Next, use `head` to view the first 10 lines of `hcc1395_normal_rep1.bg`.

```
head hcc1395_normal_rep1.bg
```

The last column in the `.bg` file is the sequencing depth.

```
chr22   10564456    10564607    1
chr22   10617976    10618127    1
chr22   10684437    10684486    1
chr22   10684486    10684588    2
chr22   10684588    10684637    1
chr22   10712579    10712719    2
chr22   10712719    10712728    1
chr22   10839950    10840096    2
chr22   10865165    10865298    6
chr22   10940293    10940443    1
```

Change back to the `/data/user/hcc1395_b4b`.

```
cd /data/user/hcc1395_b4b
```

Step 2: Create an Index for the Genome

To view genomic alignment output using IGV, an index for the reference needs to be created. `samtools` can be used to create the index. Recall that reference genome is located in the `references` folder and saved as `22.fa`.

```
module load samtools
```

```
samtools faidx references/22.fa
```

```
ls -1 references
```

```
22.1.ht2
22.2.ht2
22.3.ht2
22.4.ht2
22.5.ht2
22.6.ht2
22.7.ht2
22.8.ht2
22.fa
22.fa.fai
```

```
22.gtf
illumina_multiplex.fa
```

Step 3: Creating bigWig Files

The final step is to create the bigWig files using the application `bedGraphToBigWig` from UCSC tools *(https://hpc.nih.gov/apps/Genome_Browser.html)*.

First, load `ucsc` tools into the Biowulf working environment.

```
module load ucsc
```

All of the hcc1395 `.bg` files can be converted to bigWig (`.bw` extension) at once using the command construct below. The first argument in `bedGraphToBigWig` is the path to the `.bg` files, which reside in the folder `hcc1395_hisat2` (thus the path in the command is `hcc1395_hisat2/{}.bg` where `{}` is a place holder for the sample ID sent by `cat`). The second argument is the path to the reference index (ie. `22.fa.fai` residing in the folder `references`, thus the path as constructed from the `hcc1395_b4b` folder is `references/22.fa.fai`). The final argument is the path to the bigWig file outputs, which will be written to the folder `hcc1395_hisat2` and will have `.bw` extension. The for the output from the `hcc1395_b4b` folder will be `hcc1395_hisat2/{}.bw`.

```
cat hcc1395_sample_ids.txt | parallel -j 6 "bedGraphToBigWig hcc1395_
```

List the contents of `hcc1395_hisat2` to confirm that the `.bw` files are written.

Lesson 12: Visualizing Genomic Data with the Integrative Genomics Viewer

Lesson 11 Reviews

Participants learned how to prepare files (ie. bigWig) for visualizing genomic alignment results in Lesson 12.

Learning Objectives

After this lesson, participants will:

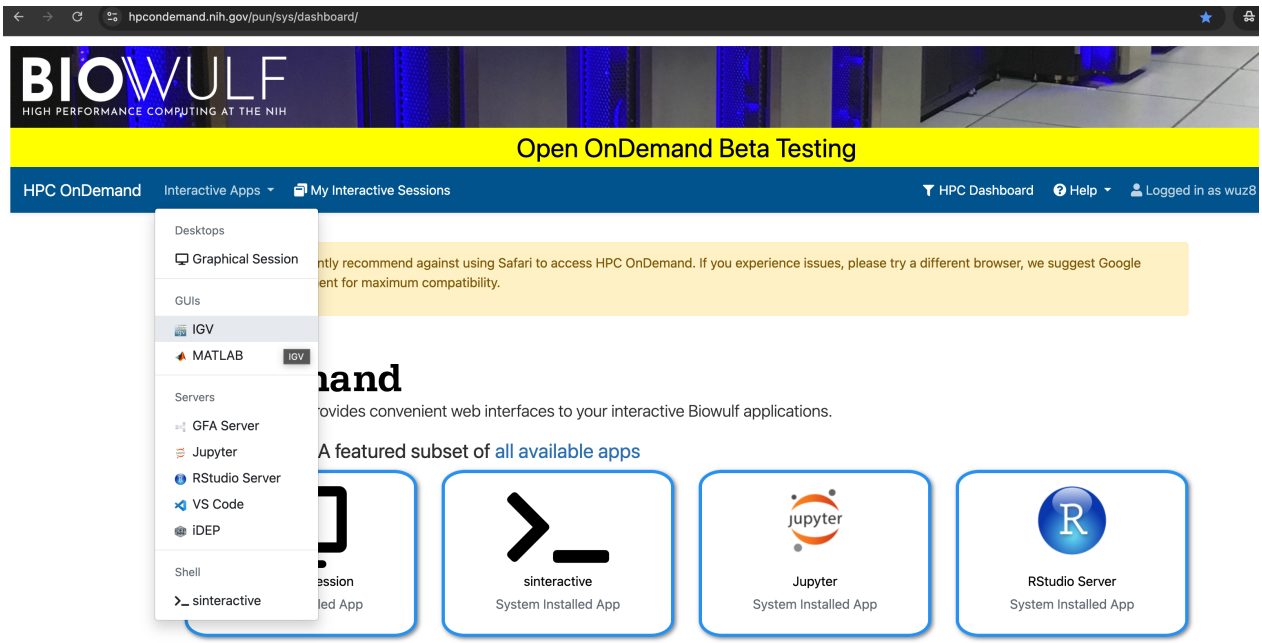
- Have a high level understanding of the Integrative Genomics Viewer (IGV) *(https://igv.org)* used for visualizing genomic data.
- Know the difference between visualizing genomic information from a bigWig file and BAM file.
- Understand the difference in output obtained from a splice aware versus a non-splice aware sequence aligner.

This class is demo only and not hands-on!

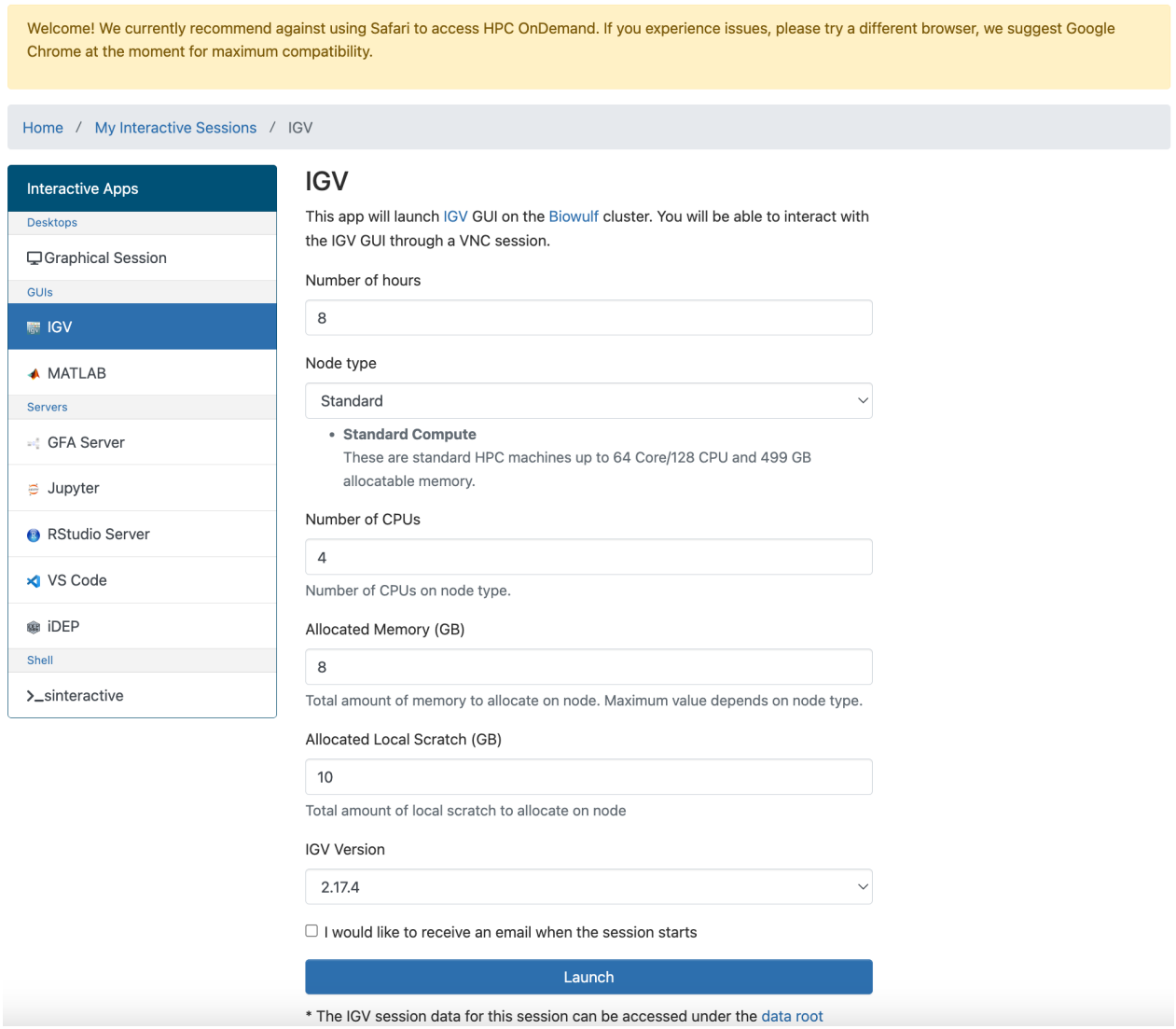
Launch IGV From HPC OnDemand

> **Note**
>
> Users sign onto HPC OnDemand using NIH credentials.

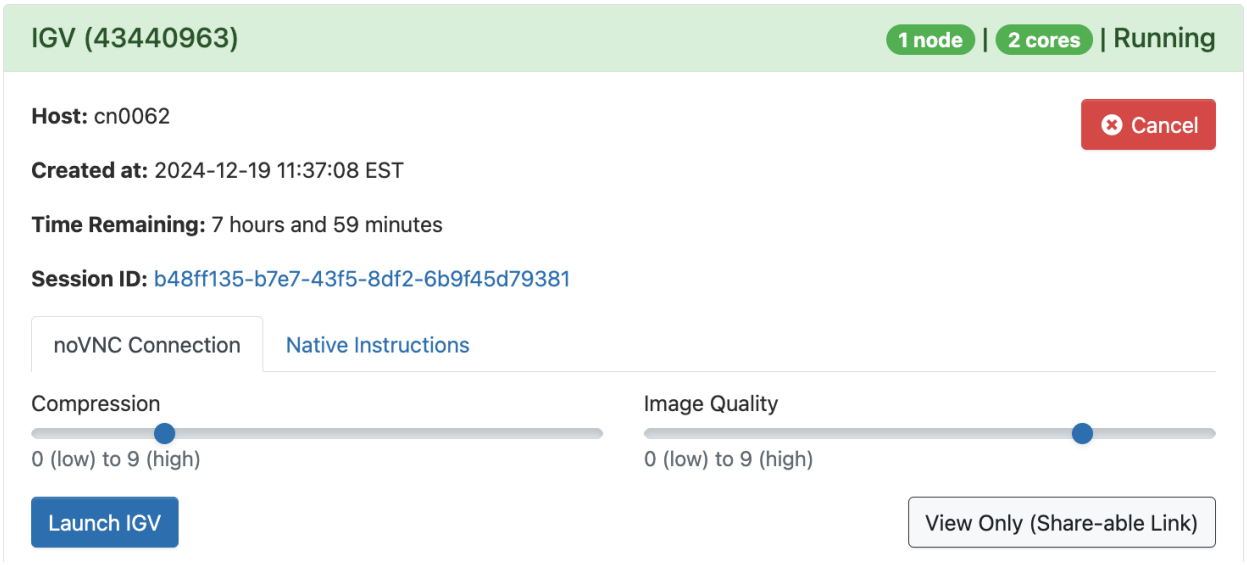
Launch the IGV session on NIH HPC OnDemand *(https://hpcondemand.nih.gov/pun/sys/dashboard/)* by clicking on "Interactive Apps" and then choosing "IGV".



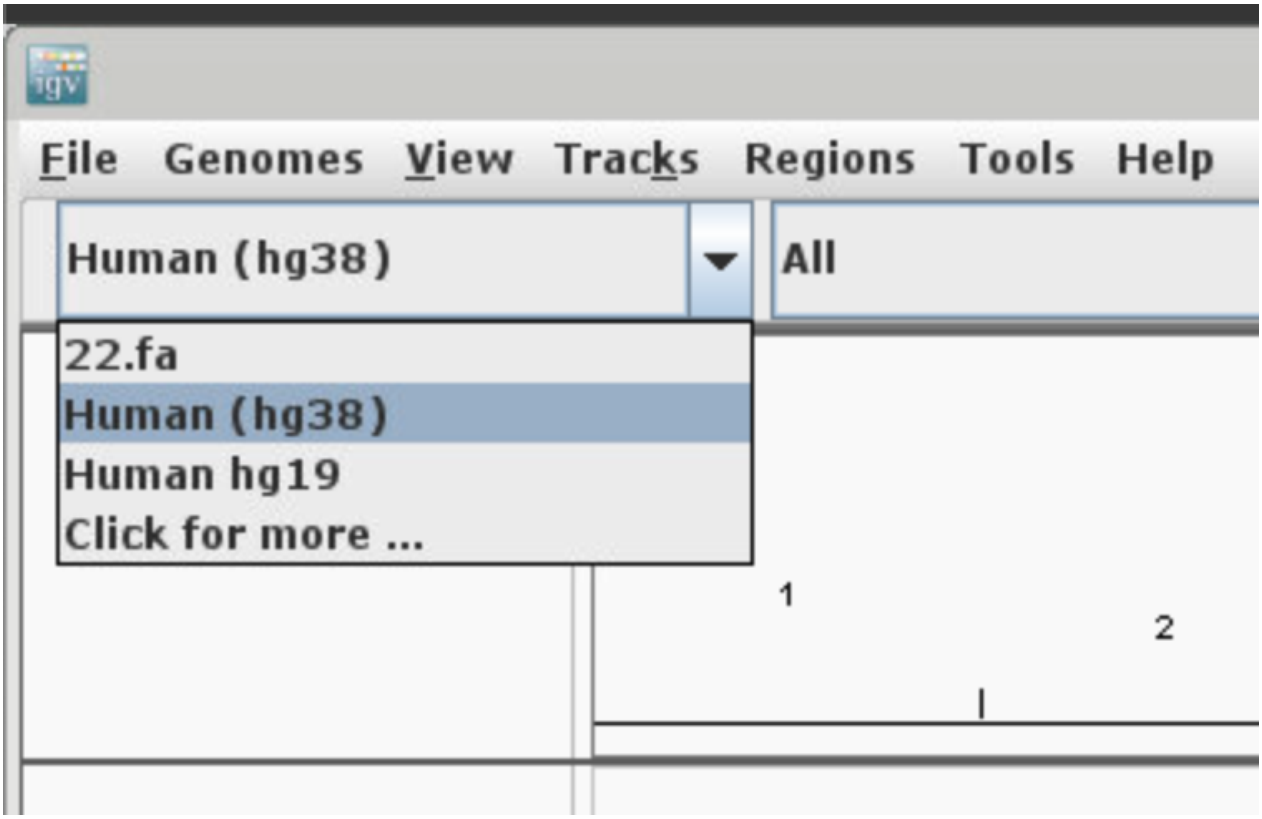
In the subsequent page, users will be able to select compute resources for the IGV session. Starting an application on HPC OnDemand will consume one of the two interactive sessions. Click on the "Launch" button when ready.



Once the IGV session's compute resources have been allocated, click on "Launch IGV" to get started.

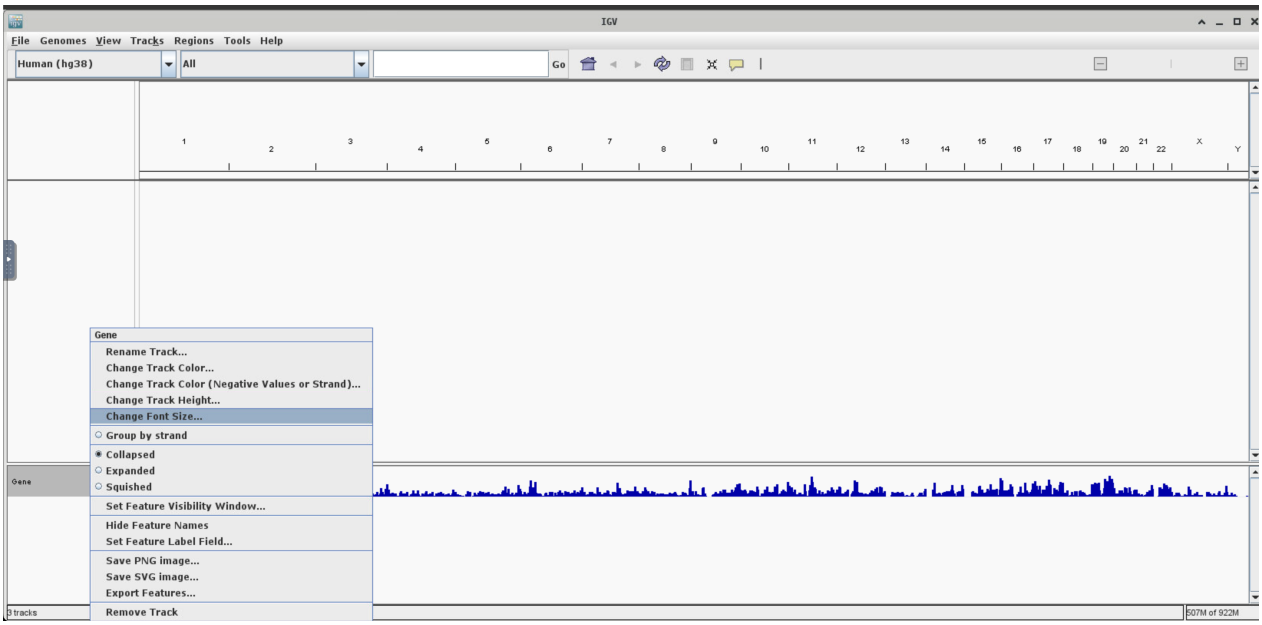


Select "Human (hg38)" as the reference in the genome selection drop down menu.



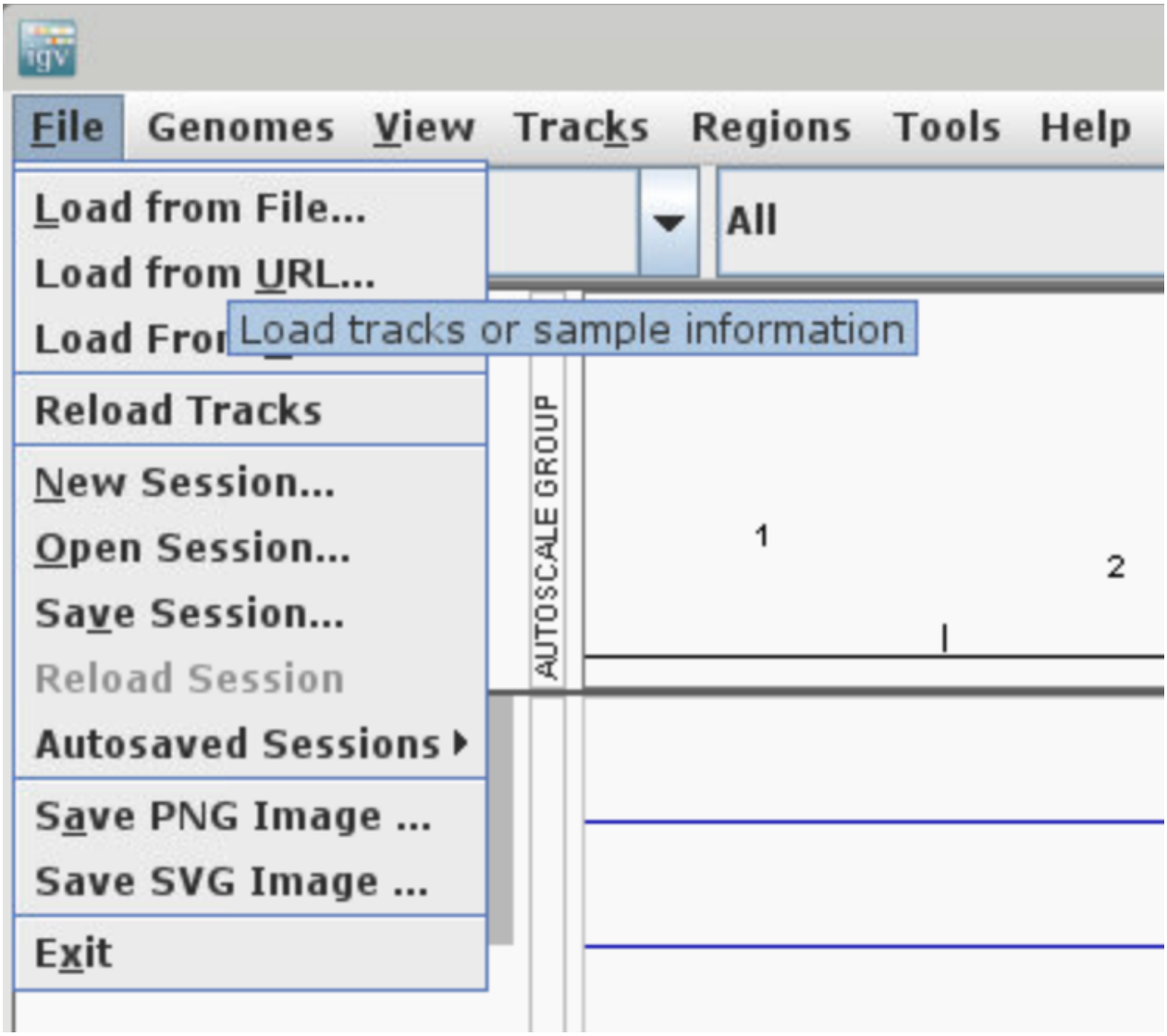
A track showing the genes for hg38 will appears. Right click on this track to see the configuration options, which include:

- Track color and font size.
- How densely the data should be displayed on the track (ie. collapsed, expanded, or squished).

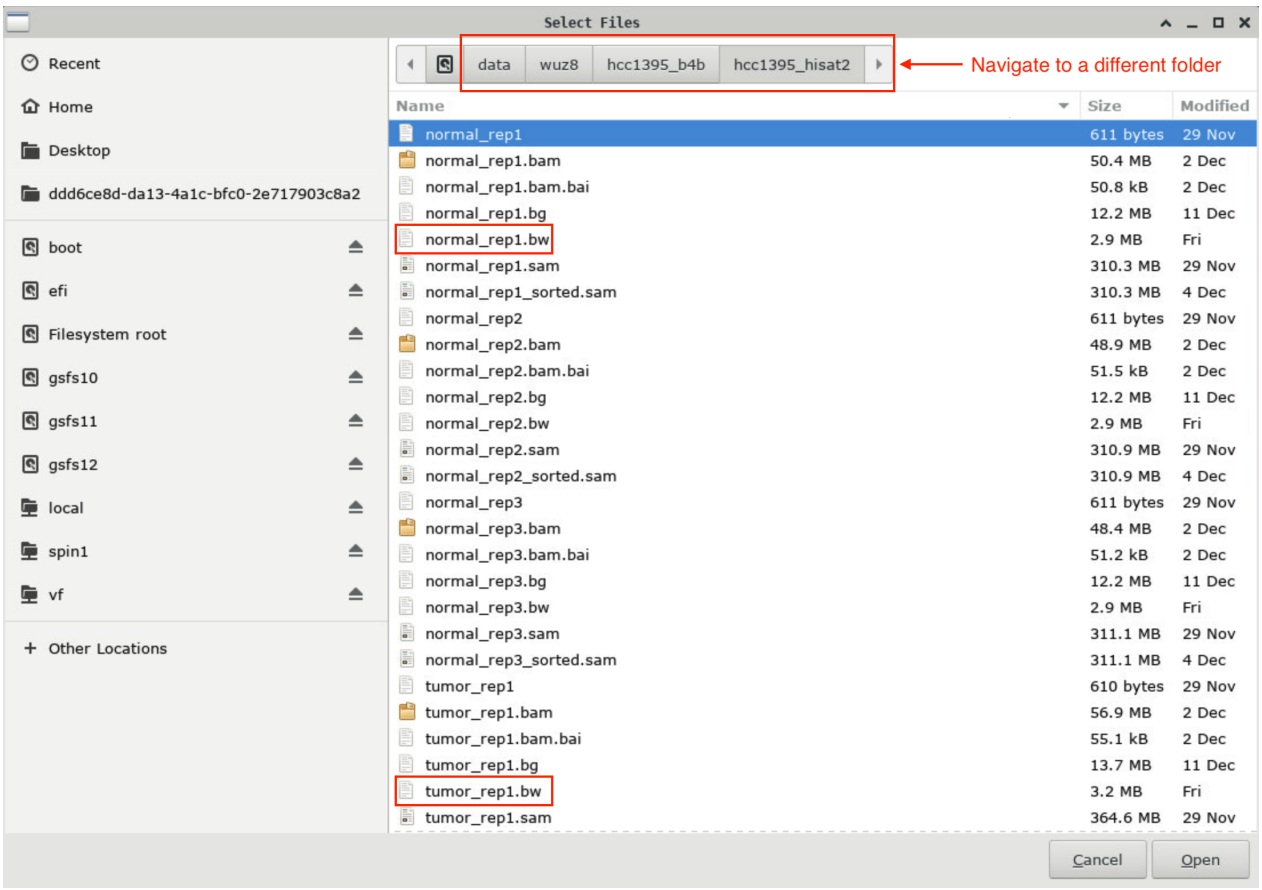


Viewing Coverage with bigWig Files

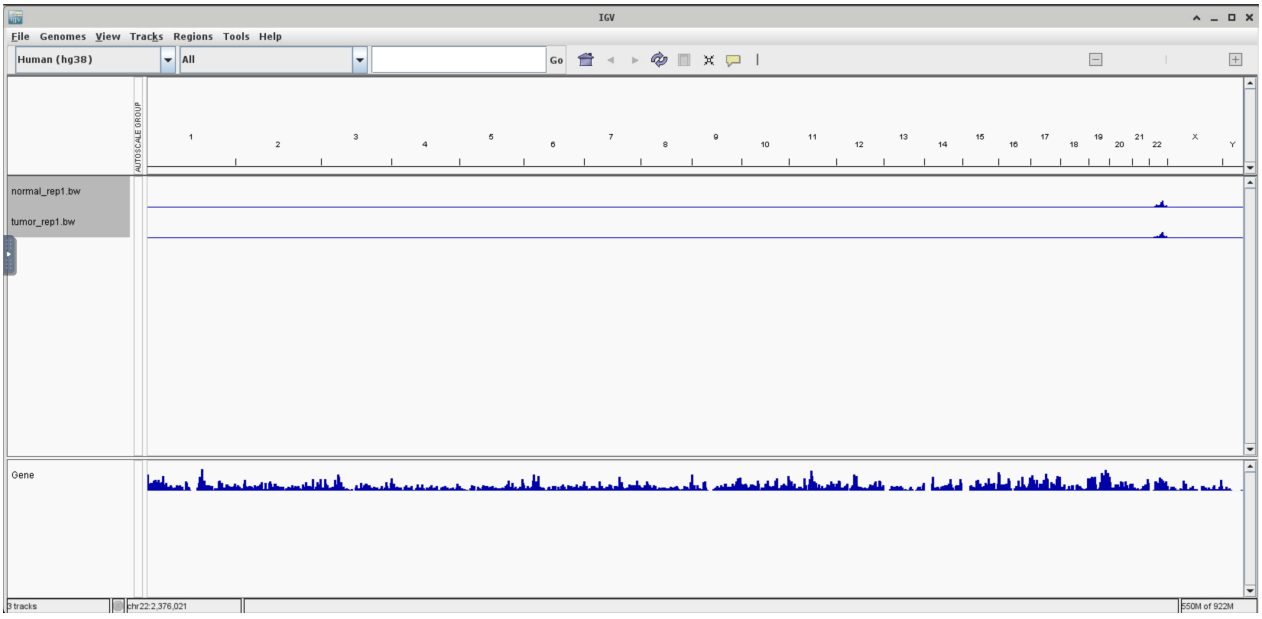
To load genomic data tracks, select "File" in the IGV menu bar. User can load from file, URL, or server. In this case, "Load from File" will be used to select alignment bigWig files from the `/data/user/hcc1395_b4b/hcc1395_hisat2` folder.

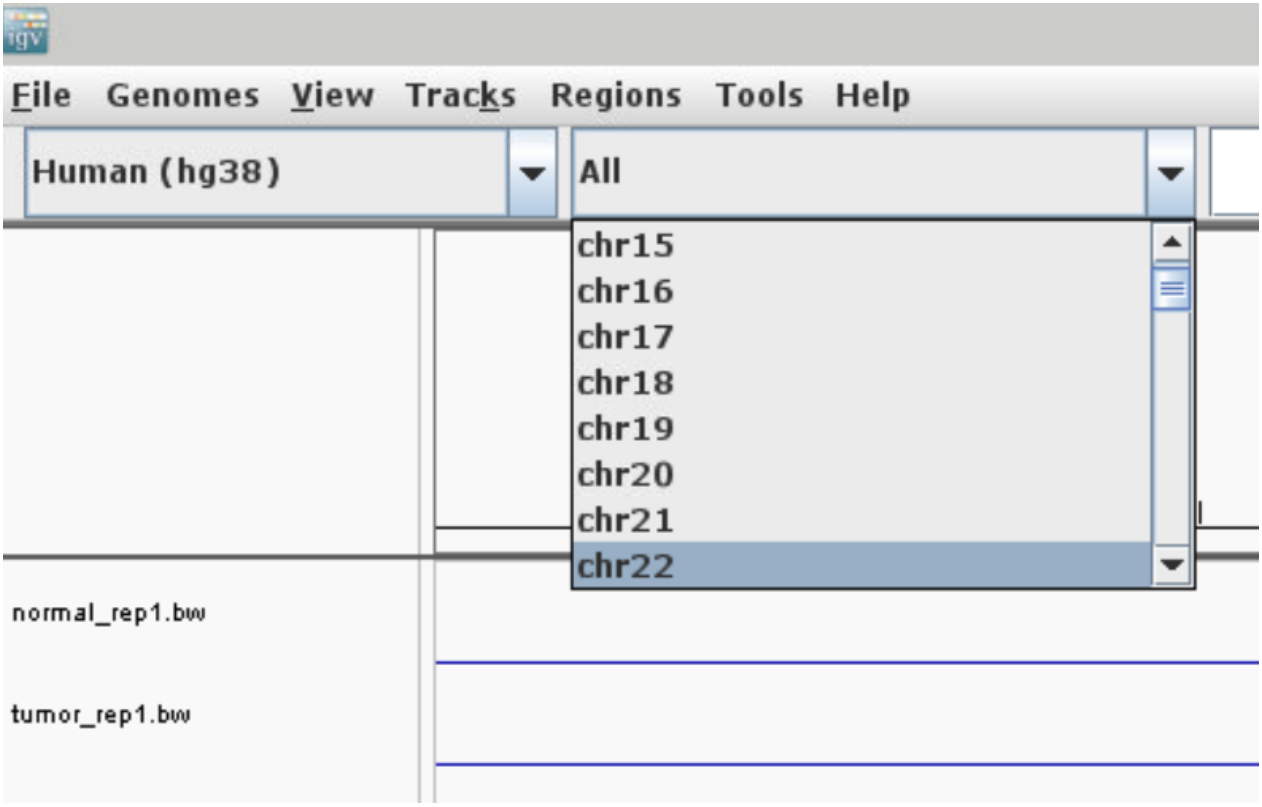


In the file explorer, choose the bigWig files `normal_rep1.bw` and `tumor_rep1.bw` (select multiple files by holding down the control key). Users can navigate to a different folder in the `/data/users` folder using the navigation tool at the top of the file selection window.



The bigWig files show pre-computed alignment coverage and it is clear that the only location where sequences have aligned is chromosome 22 as indicated by the peaks. Either click on the "22" above the peak or select from the chromosome selection drop down menu.

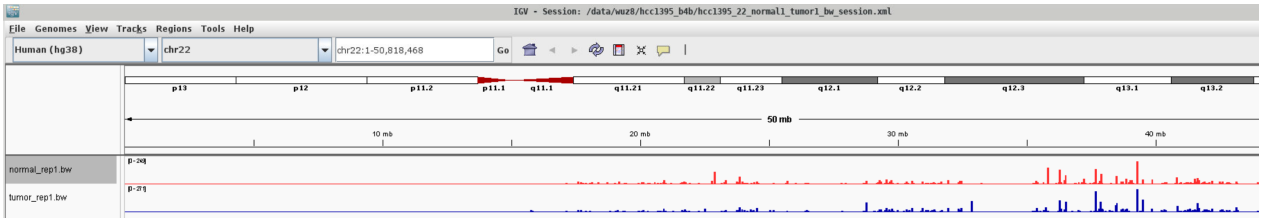




After filtering to only chromosome 22, the coverage data along with the genes on this chromosome are apparent.

Next, right click on the track labeled "normal_rep1.bw" and change the color to help distinguish is from the tumor sample (ie. tumor_rep1.bw).

Change the "normal_rep1.bw" track to red.

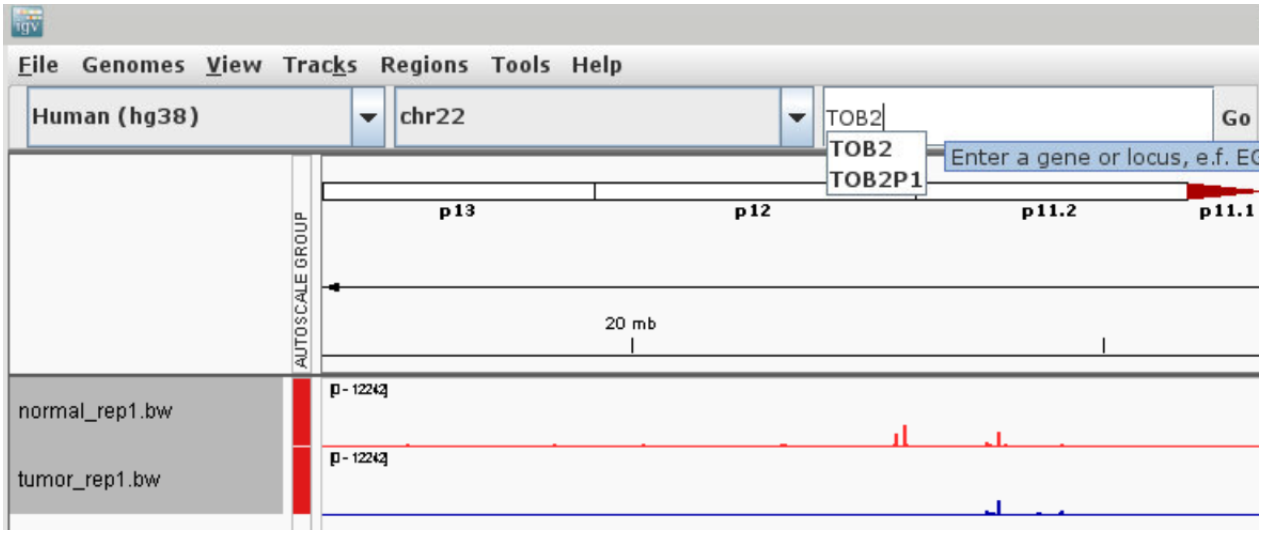


Next, select both the normal and tumor bigWig tracks, right click and select "none" for the Windowing function and Group auto scale to put the data ranges shown on the tracks on the same scale.

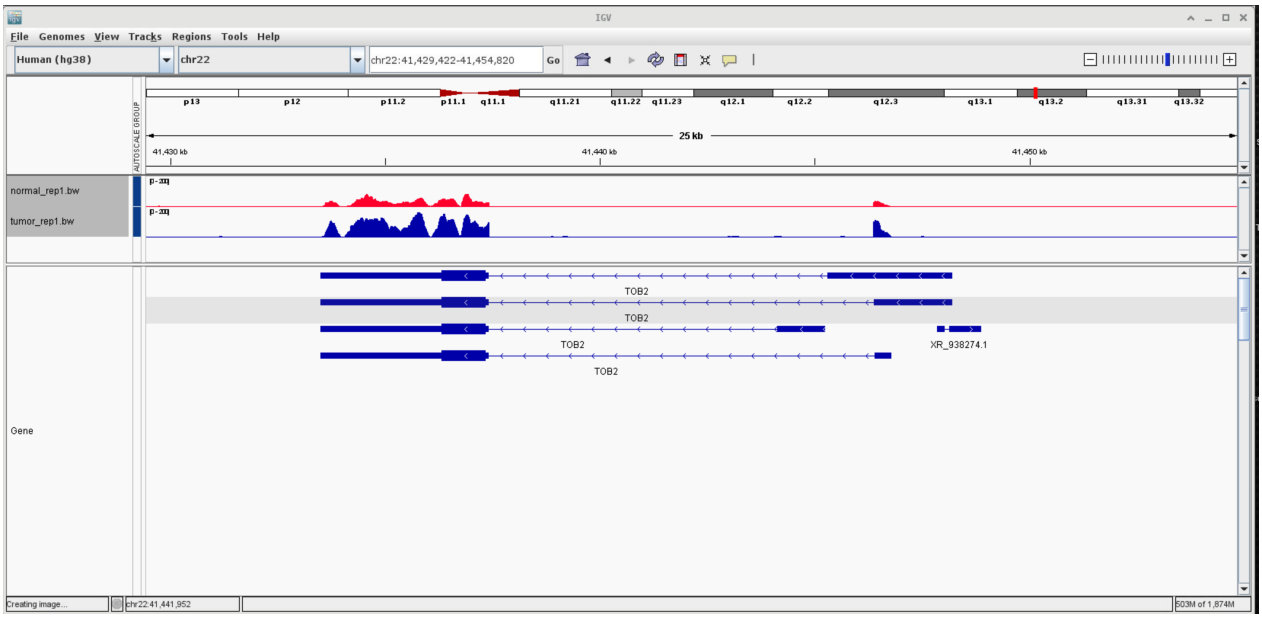
Note

Regarding the Windowing function: "When the view is zoomed out, each pixel on the screen may represent a genomic region that encompasses multiple numeric values in the data. The windowing function specifies which of the multiple values to display. To set the function, select one of the options in the Windowing Function section of the track right-click pop-up menu. The available options will depend on the file type, but most include: Minimum, Mean, Maximum, and None. By default, the function is set to Mean. The None option will display all the values, rather than combining them into one value, which can be useful for tracks displayed as points". -- IGV *(https://igv.org/doc/desktop/#UserGuide/tracks/quantitative_data/#windowing-function)*

Regarding group autoscale: "When comparing several sets of tracks, it is helpful to scale them on the same axis using the “Group Autoscale” option." -- https://eclipsebio.com/eblogs/how-to-use-igv-1/ *(https://eclipsebio.com/eblogs/how-to-use-igv-1/)*

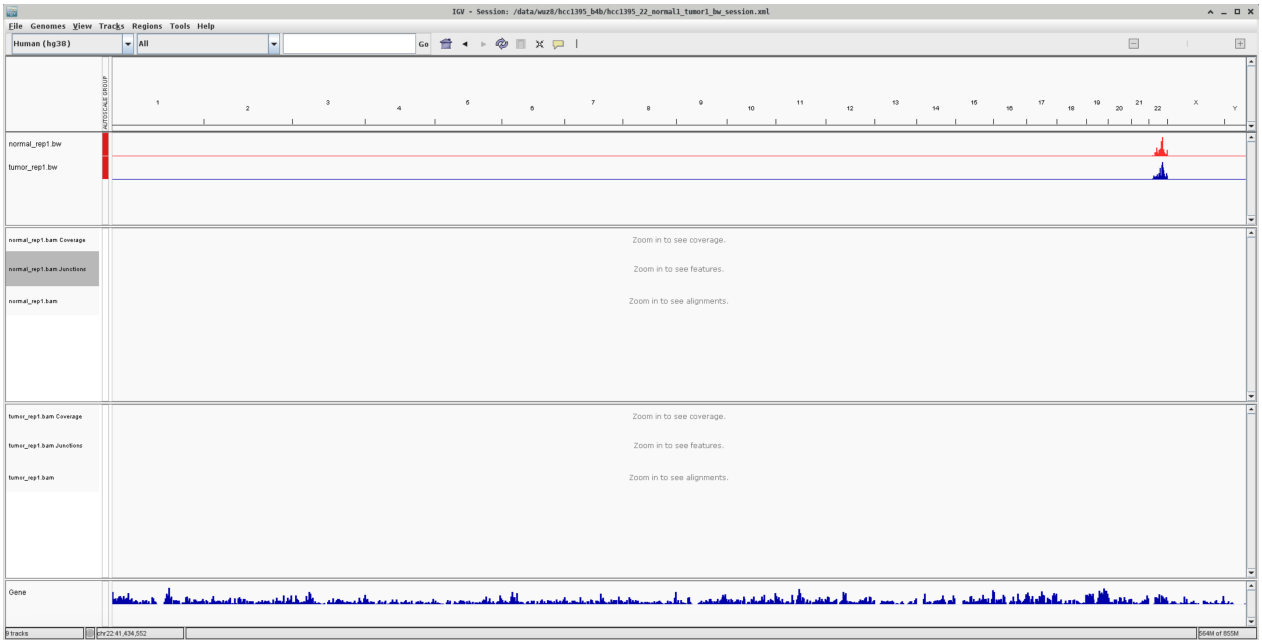


Search for the gene TOB2 *(https://www.genecards.org/cgi-bin/carddisp.pl?gene=TOB2)*. From this IGV view, it appears that TOB2 is expressed higher in the "tumor_rep1" as compared to the "normal_rep1" sample due to more reads aligning to TOB2 in "tumor_rep1". In the expanded view of the gene track, transcript isoforms are shown. From the IGV image below, which one of the TOB2 transcripts is likely expressed?



Viewing BAM files in IGV

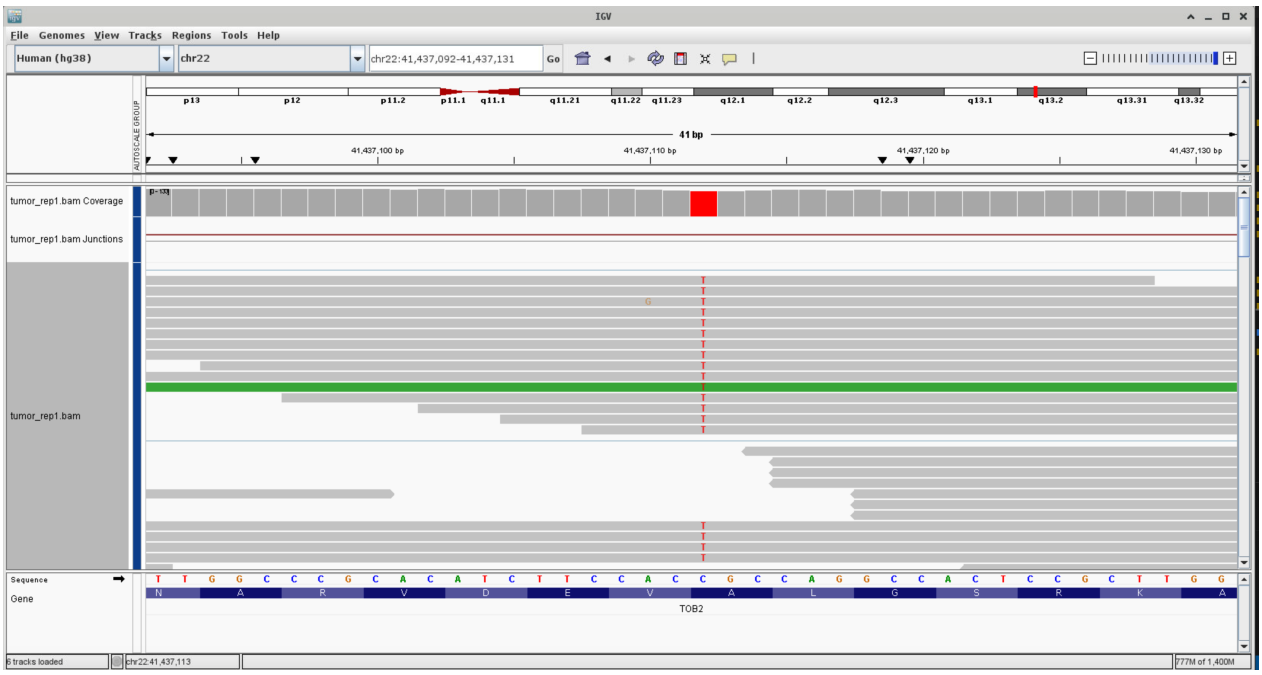
For this exercise, click on the chromosome selection drop down and choose "All". Then load `normal_rep1.bam` and `tumor_rep1.bam` to the tracks. Unlike the bigWig files, which shows pre-calculated coverage in IGV as soon as they are loaded, BAM files requires users to zoom in to a specific location in order to see the coverage and alignment information.



Upon zooming into TOB2, users will see that the BAM files contains more information than the bigWig file. These include:

- Coverage information (also shown in the bigWigs).
- The actual alignments.
- Splice junctions (note that the parts of sequencing reads that span across exons are connected by solid lines in the alignment track).

Zoom in a bit and a potential single nucleotide variant is apparent. Where the sequence contains a T, the reference contains a C. Users can also view insertions/deletions in IGV when looking at BAM files.

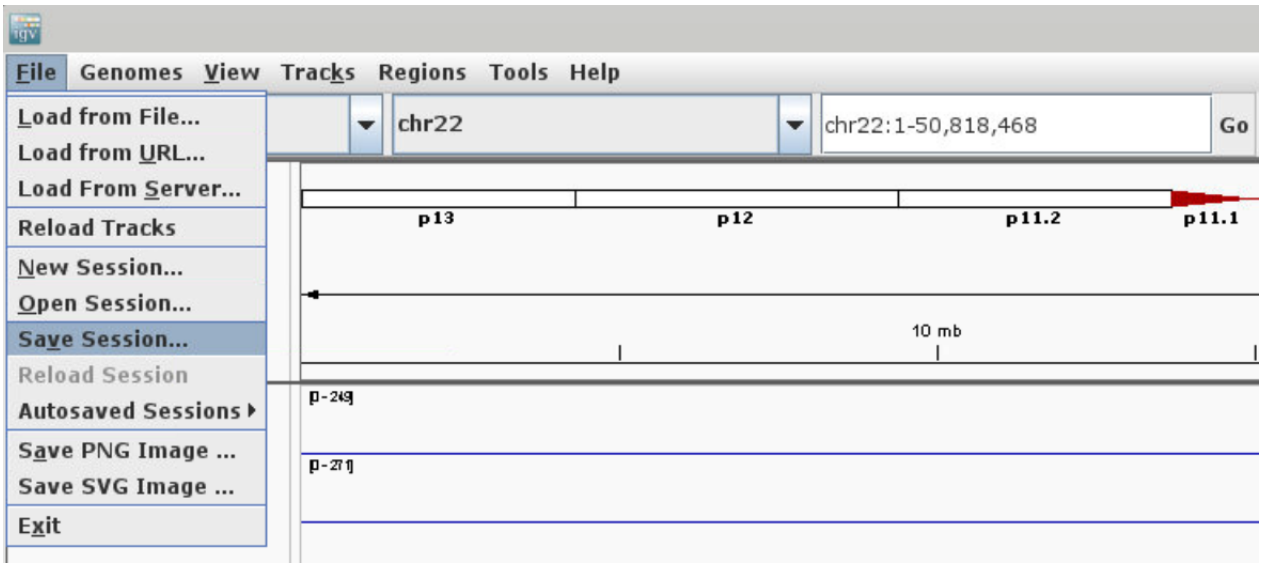


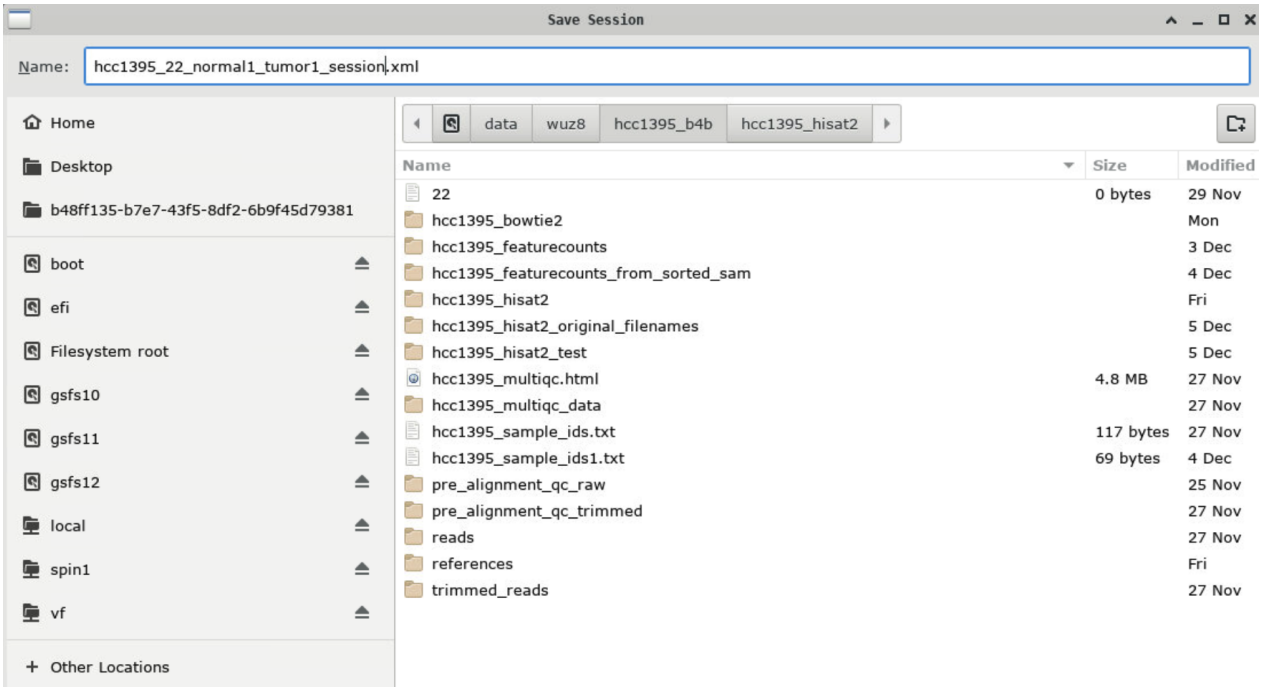
Difference Between HISAT2 and BOWTIE2 Alignment Results

Right click on the `normal_rep1.bw` and `tumor_rep1.bw` tracks and remove them. Also remove the `normal_rep1.bam` track. Then, load the `tumor_rep1_bowtie2.bam` file into IGV. Note that the coverage and alignment information are available in the BOWTIE2 BAM outputs. However, because BOWTIE2 is not splice aware, the junction track is missing. Further, reads that map across exons are not connected by a line when the hcc1395 data is aligned to genome using BOWTIE2.

Saving IGV Session

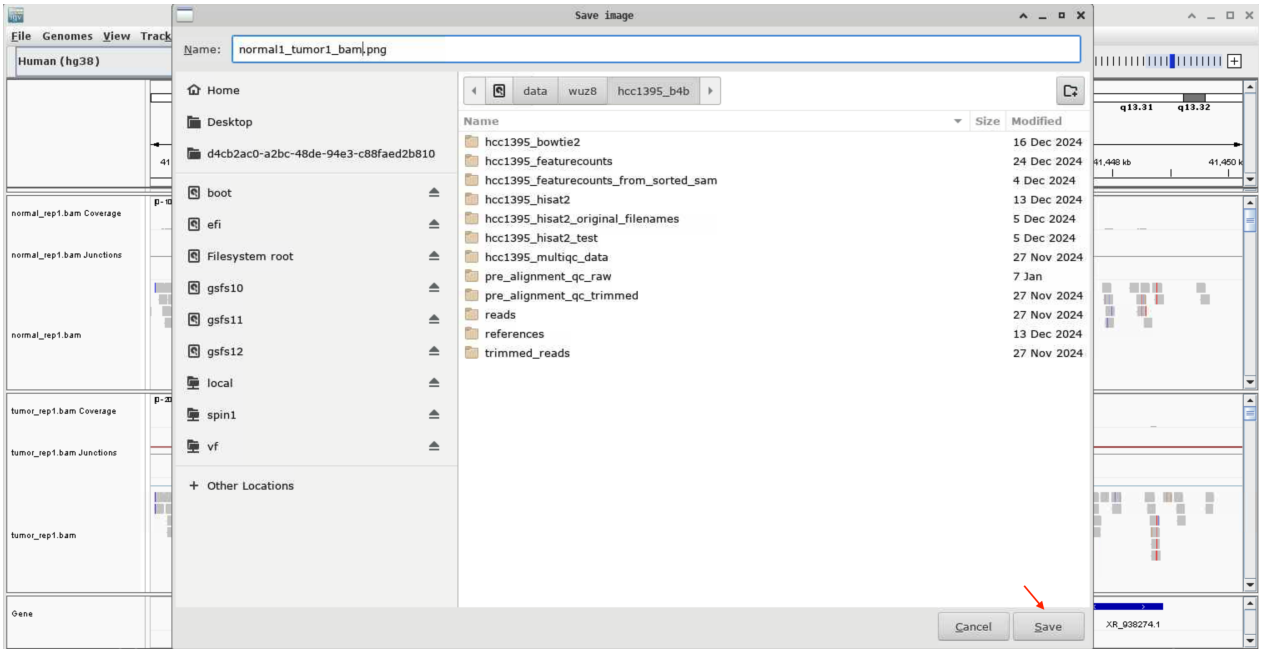
Users can save the IGV session and open it at a later time point to resume work. To do this, select File from the menu and then "Save Session". Subsequently, select the folder to save the session and provide a file name. The session is saved as a `.xml` file.



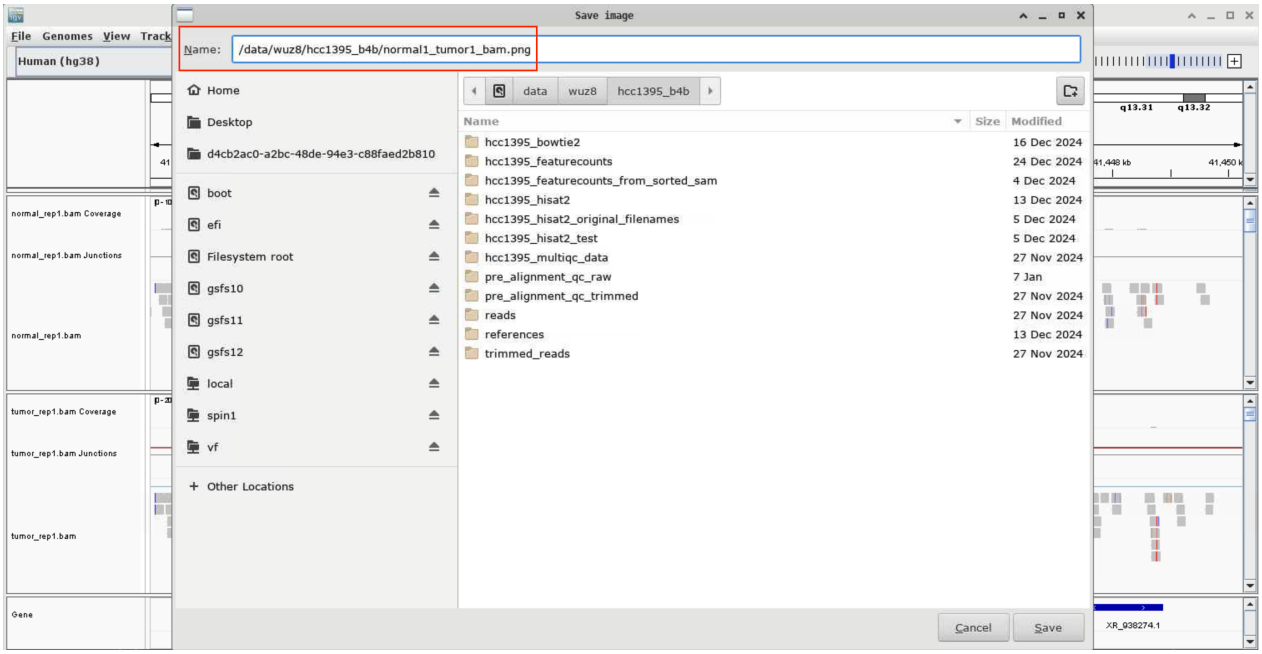


Saving IGV Image

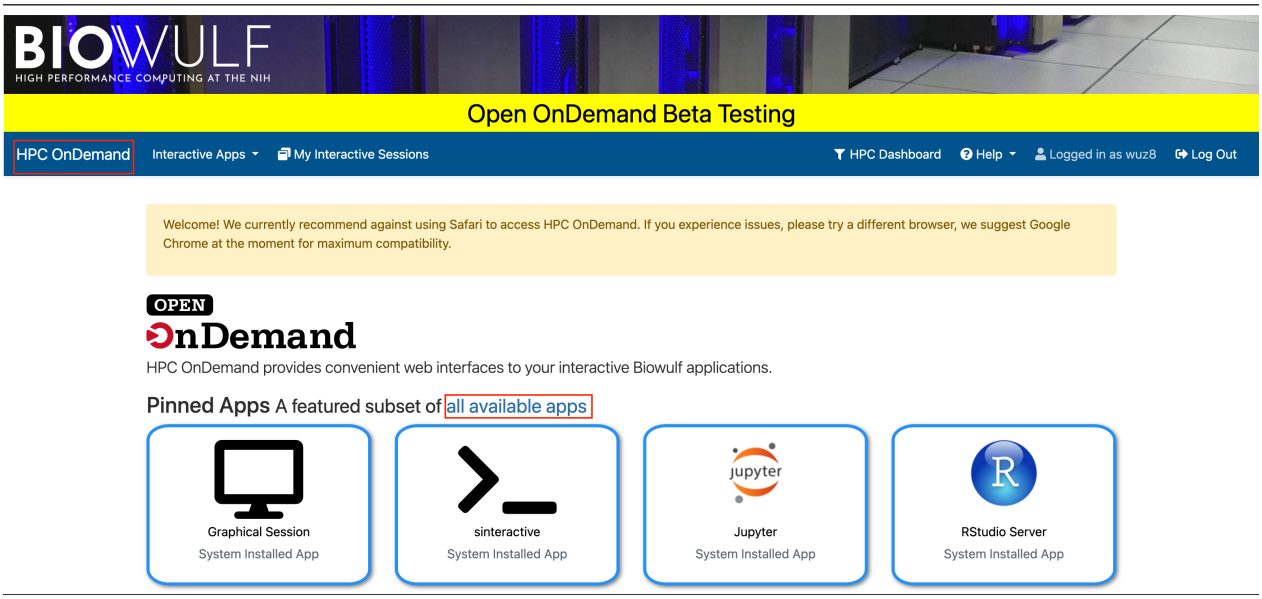
To save an IGV view as a snapshot image, goto File and Select Save PNG Image or Save SVG Image. PNG will be used here. In the dialogue box, enter the file name for the images (in this case it is `normal1_tumor1_bam.png`).



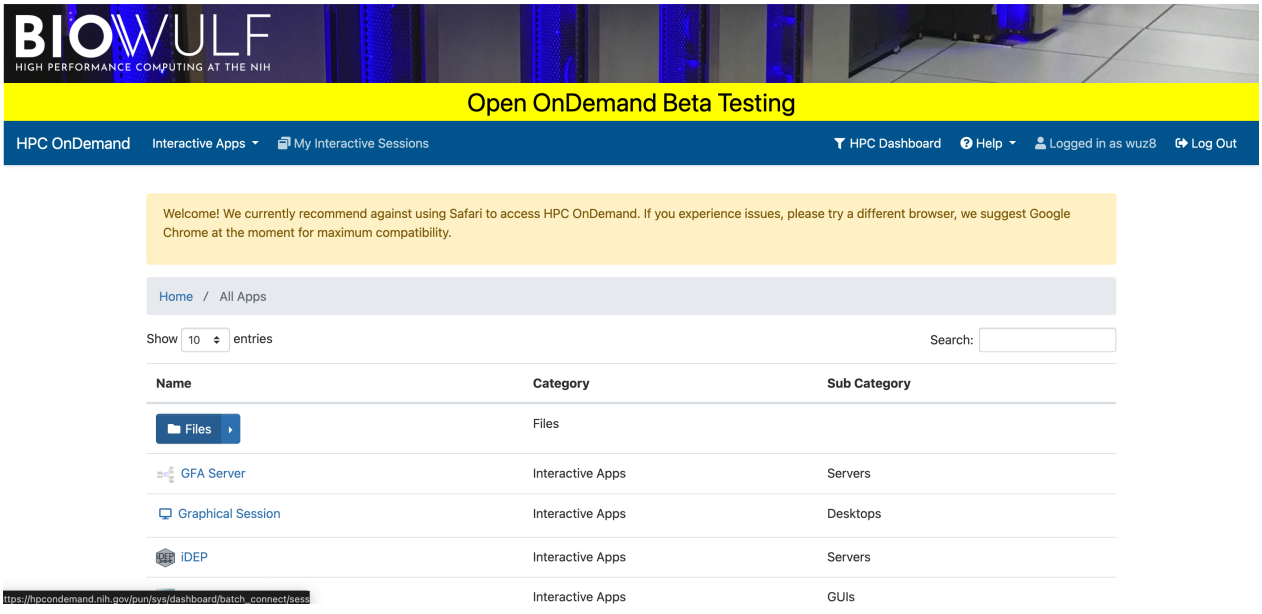
If not in the directory in which the images should be saved, just start typing the destination directory path and enter the desired file name in the address bar.



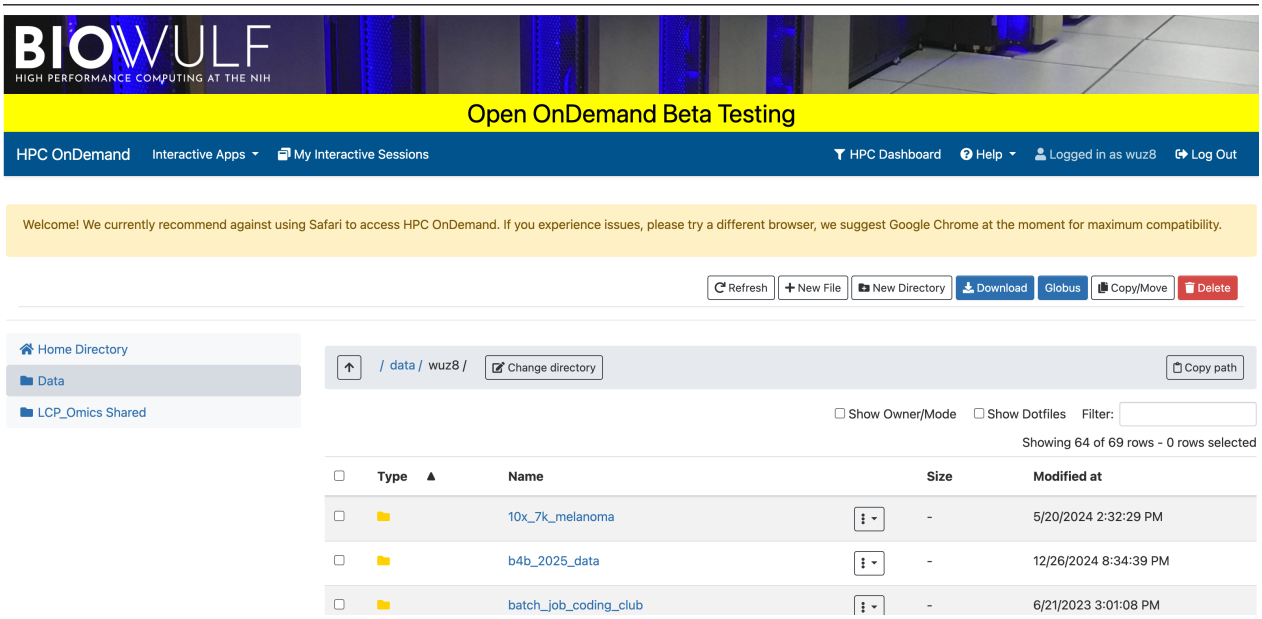
After the image has been saved, click on "HPC OnDemand" tab in the HPC OnDemand page. Then select "all available applications."



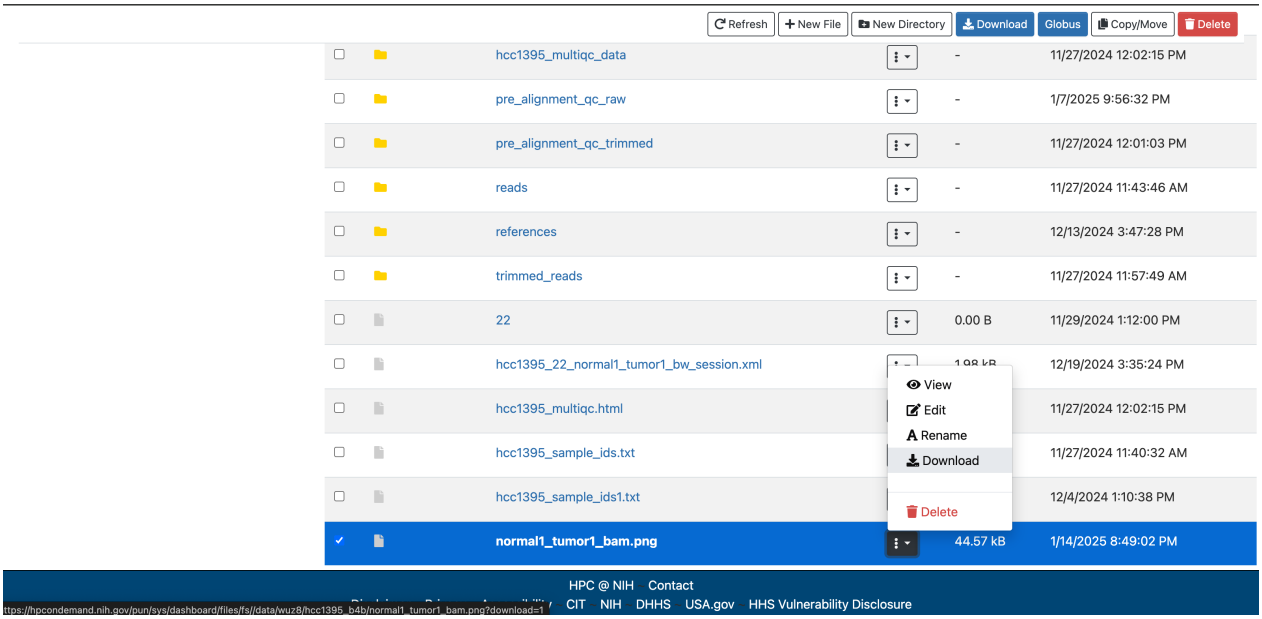
Click on "File" in the next screen.



Then choose the Data directory.



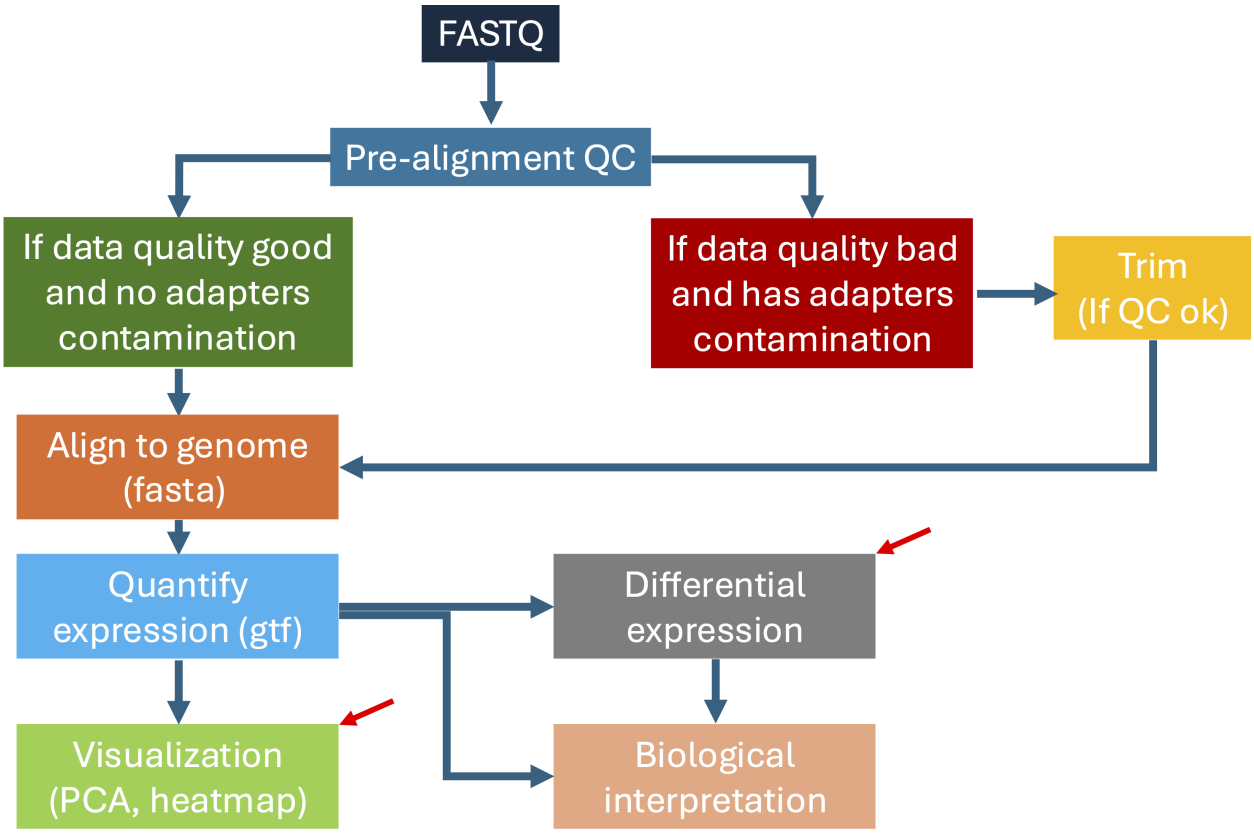
Scroll to the folder containing the IGV image, and select to download from the corresponding drop down menu.



Lesson 13: Differential Expression Analysis for Bulk RNA Sequencing: QC

Quick Review

Previously, in this Introduction to Bulk RNA Sequencing Analysis course series, participants learned to align data to reference to genome and estimate gene expression. The next 2 classes will address differential expression (DE) analysis where scientists will determine genes whose expression are statistically significantly changed between biological conditions.



Learning Objectives

After this class, participants will be able to:

- Describe rationale for filtering low expression genes and filtering these out from the gene expression estimates table.
- Perform quality control measures such as Principal Components Analysis on expression estimates and provide rationale for each QC measure.

This class will not introduce the R programming language but participants will run R scripts from the command line.

Sign onto Biowulf

Before getting started using the participant's assigned Biowulf student account ID.

```
ssh user@biowulf.nih.gov
```

Next, change into the `/data/user/hcc1395_b4b` folder.

```
cd /data/user/hcc1395_b4b
```

Request an interactive session with 12 gb of RAM, and 10 gb of local temporary storage.

```
sinteractive --mem=12gb --gres=lscratch:10
```

Load R

```
module load R
```

Background on R Scripts

The R scripts used for this class are in the folder `b4b_scripts`. Stay in `/data/user/hcc1395_b4b` and list the contents of `b4b_scripts` to learn what is available. **These scripts were generated by BTEP and inspired by the Biostars Handbook's *(https://www.biostarhandbook.com/)***.

```
ls b4b_scripts
```

There are 3 R scripts.

- `deg.R` will calculate differential gene expression using DESeq2.
- `filter_expression.R` will filter low expressing genes from the expression estimates.
- `quality_check.R` enables QC of raw expression estimates.

```
deg.R  filter_expression.R  quality_check.R
```

Warning

These R scripts are for class and demonstration purposes. Do not take these and blindly use to analyze data as modification will be needed to meet the challenges of each study.

This session will use `filter_expression.R` and `quality_check.R`. To run R scripts the terminal, users will start with the `RScript` command follow by the script name and then arguments.

Stay in `/data/user/hcc1395_b4b` and create a folder called `hcc1395_deg` to store the differential expression analysis results.

```
mkdir hcc1395_deg
```

Note

This course will not teach participants how to script in R.

Filter Low Expressing Genes

It's common practice to filter low expression genes as these may represent noise. In these exercises, the criteria below will be used to determine which genes will be removed from the expression estimates stored in the folder `hcc1395_featurecounts` as file `hcc1395_gene_expression.csv`.

- If a gene has 0 expression across all samples, then it is removed.
- If a 2 out of 3 samples in a treatment group has a gene expression of 0, then it is removed (ie. if a gene for a particular condition has 2 sample with 0 expression and 1 sample with expression of 10, then it is removed).

Filtering criteria is up to the researcher. Other methods include filtering based on the variance of expression of a gene across samples.

Issuing one of the following will pull the arguments needed for `filter_expression.R`.

```
Rscript filter_expression.R
```

or

```
Rscript filter_expression.R -h
```

or

```

```

```
Rscript filter_expression.R --help
```

Regardless of method, users will see the following tips on how to run the R scripts, in this case `filter_expression.R`.

The first line is a description of the script followed by usage. The required arguments are listed as well.

```
This R script filters low expressing genes from RNA sequencing data.
Usage: Rscript filter_expression.R arguments

Use Rscript filter_expression.R -h or Rscript filter_expression.R --h

The following arguments are required.

Argument 1: File name of expression results in CSV format.
    *The first column should contain gene identifiers followed by co
Argument 2: The integer number of samples in each group that needs t
Argument 3: File name of a phenotypes table in CSV format. This file
    *Column - this contains the sample column headings from the expr
    *Sample - these are the shortened sample names and will used to r
    *Treatment - these are the biological conditions to which the sar
Argument 4: Study name! This will be prepended to the outputs.
Argument 5: Output directory! Outputs will be written here.
```

Stay in the `/data/user/hcc1395_b4b` folder for the exercises below.

To run `filter_expression.R` from the `/data/user/hcc1395_b4b` folder, use the following command construct where:

- `b4b_scripts/filter_expression.R` is the path for the `filtered_expression.R` script since it is being run from `/data/user/hcc1395_b4b`.
- `hcc1395_featurecounts/hcc1395_gene_expression.csv` is the path for the gene expression table as referenced from `/data/user/hcc1395_b4b`. The expression table is stored in the folder `hcc1395_featurecounts` and file `hcc1395_gene_expression.csv`
- `hcc1395_phenotypes.csv` is the path to the phenotypes table as referenced from `/data/user/hcc1395_b4b`.
- The study name will be `hcc1395` and this will be prepended to all output files.
- The output folder is `hcc1395_deg` as referenced from `/data/user/hcc1395_b4b`.

```
Rscript b4b_scripts/filter_expression.R hcc1395_featurecounts/hcc139
```

List the contents of `hcc1395_deg` from `/data/user/hcc1395_b4b` to check that a filterd expression CSV file is present.

```
ls hcc1395_deg
```

```
hcc1395_gene_expression_filtered.csv
```

Use `wc -l` to compare number of lines in the unfiltered expression table versus the filtered.

```
wc -l hcc1395_featurecounts/hcc1395_gene_expression.csv
```

```
1336
```

```
wc -l hcc1395_deg/hcc1395_gene_expression_filtered.csv
```

```
593
```

The unfiltered expression table has 1336 lines while the filtered table has 593 lines. Thus, 743 genes were removed as a result of filtering based on the criteria set. The command `expr` can be used to perform arithmetic operations in Unix command line and the construct below subtracts 593 (number of lines in the filtered expression table) from the number lines in the unfiltered expression table (1336).

```
expr 1336 - 593
```

Quality Check

Below are the quality checks that will be performed on the filtered gene expression table.

- Principal Components Analysis (PCA): This transforms high dimensional data such as those derived from RNA sequencing so that researchers can see how study variables cluster together. The result of PCA is that the original data is projected onto a set of perpendicular axes where each axis accounts for a percentage of variance in the data. To learn the math behind PCA, see https://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf *(https://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf)*. Given that this example dataset has normal and tumor samples, it is expected the samples under each condition would separate along the principal

component axis that accounts for the most variance in the dataset. Thus, indicating it is the biology that differentiates normal and tumor samples.

- Box and density plots enable scientists to visualize the distribution of expression data and check for outliers. Ideally, the expression distribution for all samples should be roughly the same for differential expression analysis (see h3abionet *(https://h3abionet.github.io/H3ABionet-SOPs/RNA-Seq-4-1.html#:~:text=Whatever%20the%20shape%20of%20the,and%20extreme%20group/batch%20effects.)*). These two visuals are also useful to determine whether a specific normalization method worked well.
- Distance plot shows the distance between samples. It is expected that samples within condition will be closer to each other (distance wise) than those samples from other conditions.

To run `quality_check.R`, use the construct below. Remember, to run a R script from the command line, start with the command `Rscript`, followed by:

- Script name, which is `quality_check.R`. However, because `quality_check.R` is in the folder `b4b_script`, it is referenced using `b4b_script/quality_check.R` from the current folder of `/data/user/hcc1395_b4b`.
- The name of expression table. Here, the filtered expression table `hcc1395_gene_expression_filtered.csv` in `hcc1395_deg` will be used. As referenced from `/data/user/hcc1395_b4b`, the path would be `hcc1395_deg/hcc1395_gene_expression_filtered.csv`.
- The study name (ie. `hcc1395`), which will be prepended to all output files.
- Directory in which to write the output (ie. `hcc1395_deg`).

```
Rscript b4b_scripts/quality_check.R hcc1395_deg/hcc1395_gene_express
```

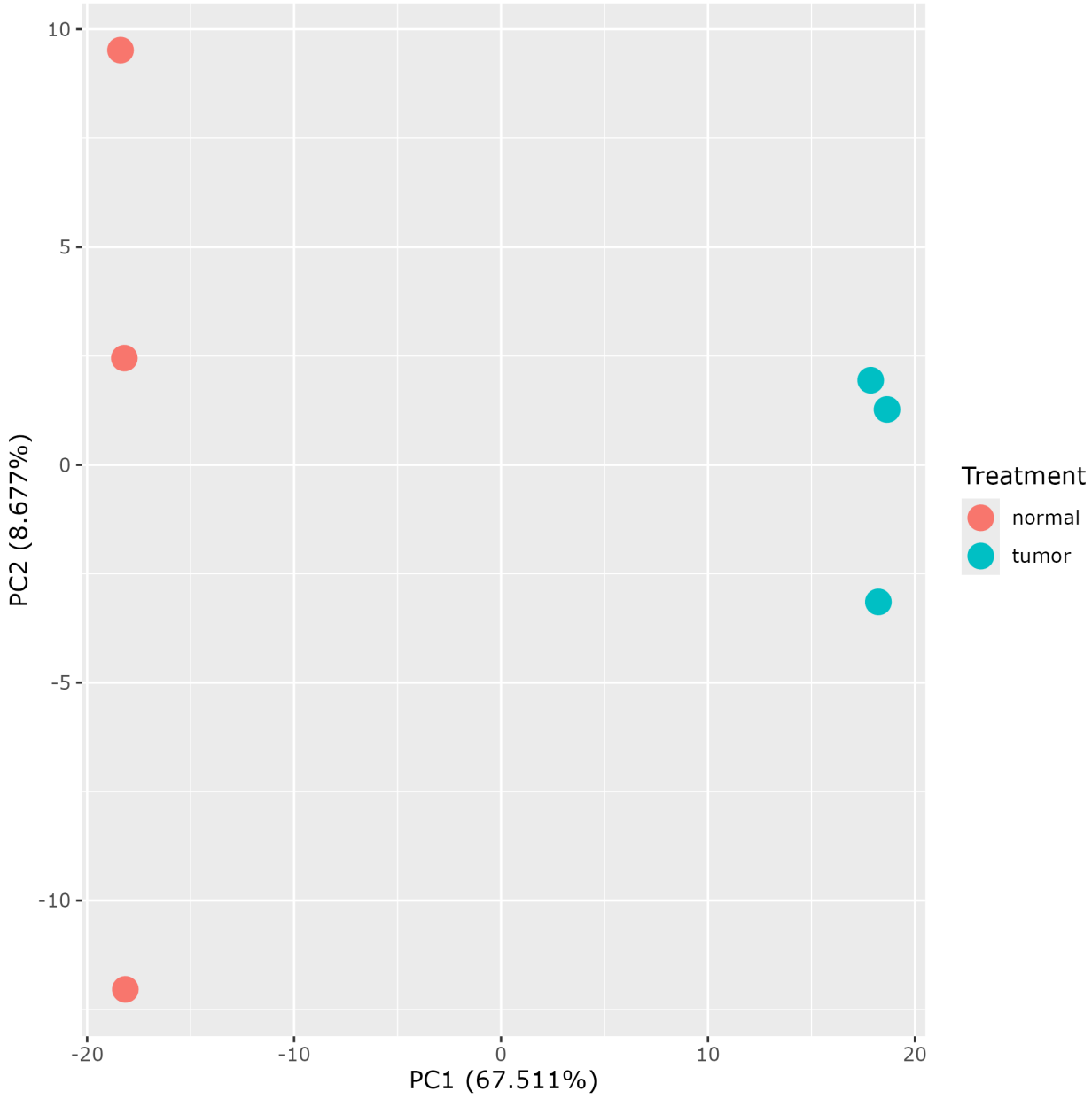
```
ls hcc1395_deg
```

```
hcc1395_box.png      hcc1395_distance.png                  hcc1395_me
hcc1395_density.png  hcc1395_gene_expression_filtered.csv  hcc1395_pc
```

Principal Components Results

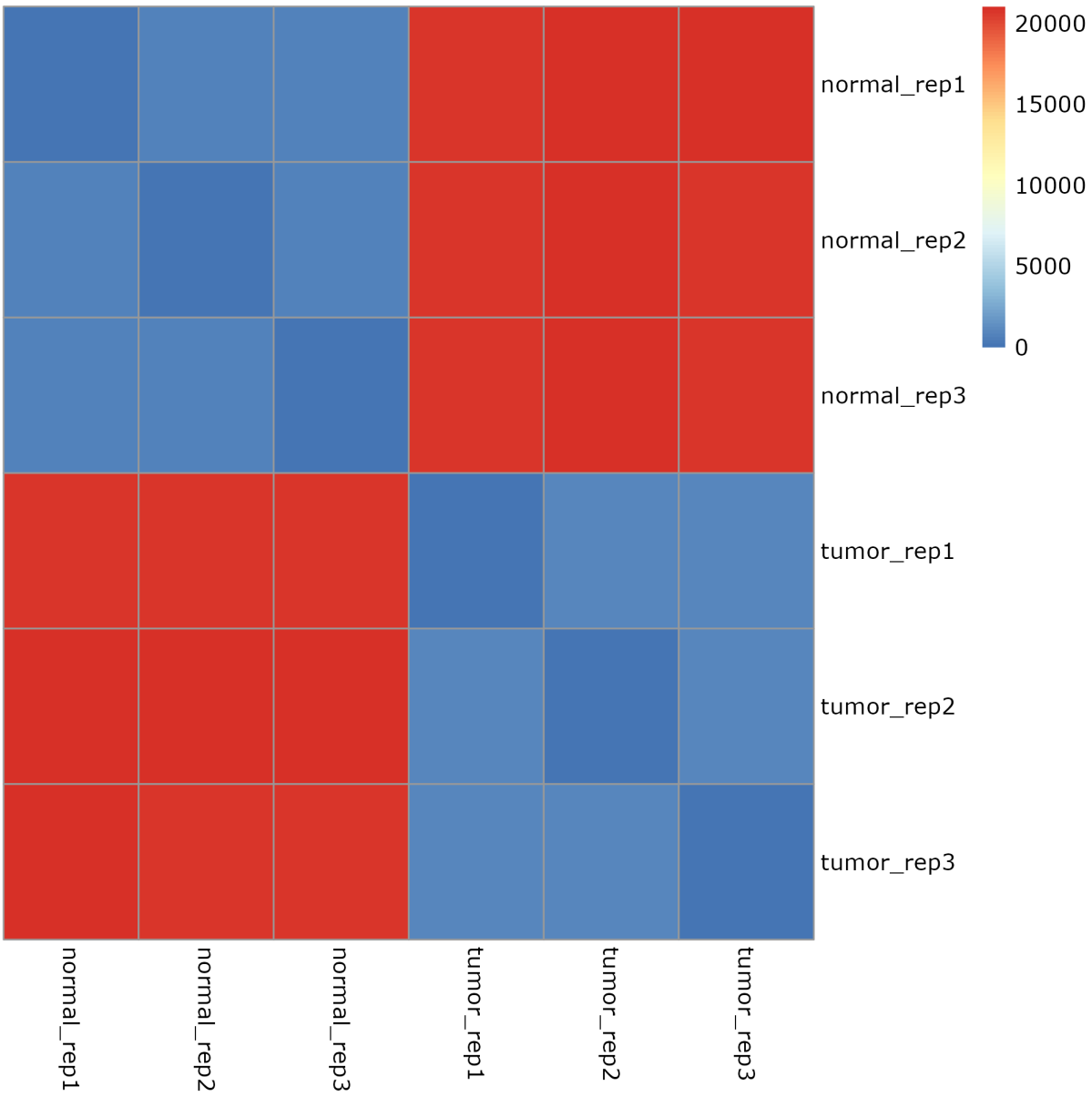
The Principal Components plot below graphs the sample along the PC1 and PC2 axes, which account for the highest and second highest variances in the data, respectively. PC1 explains 67.5% of variance while PC2 explains 8.7%. It is clear that the normal and tumor samples separate along PC1, which indicates that it is the biology that is differentiating these samples between these groups. Within group samples separation along PC2 could be due to differences

between samples of same group or maybe batch effect (although batch information is not available).



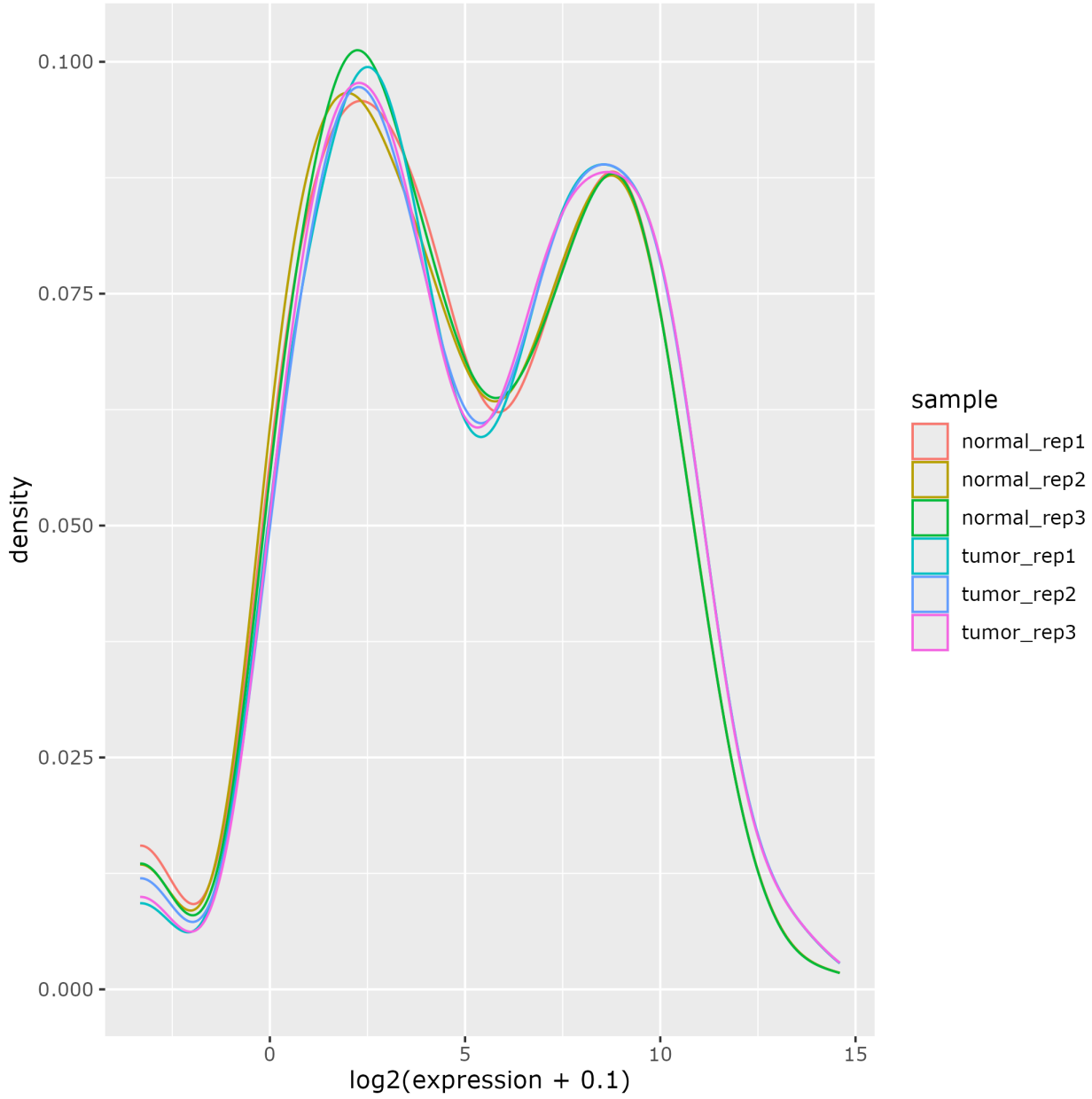
Distance Plot Results

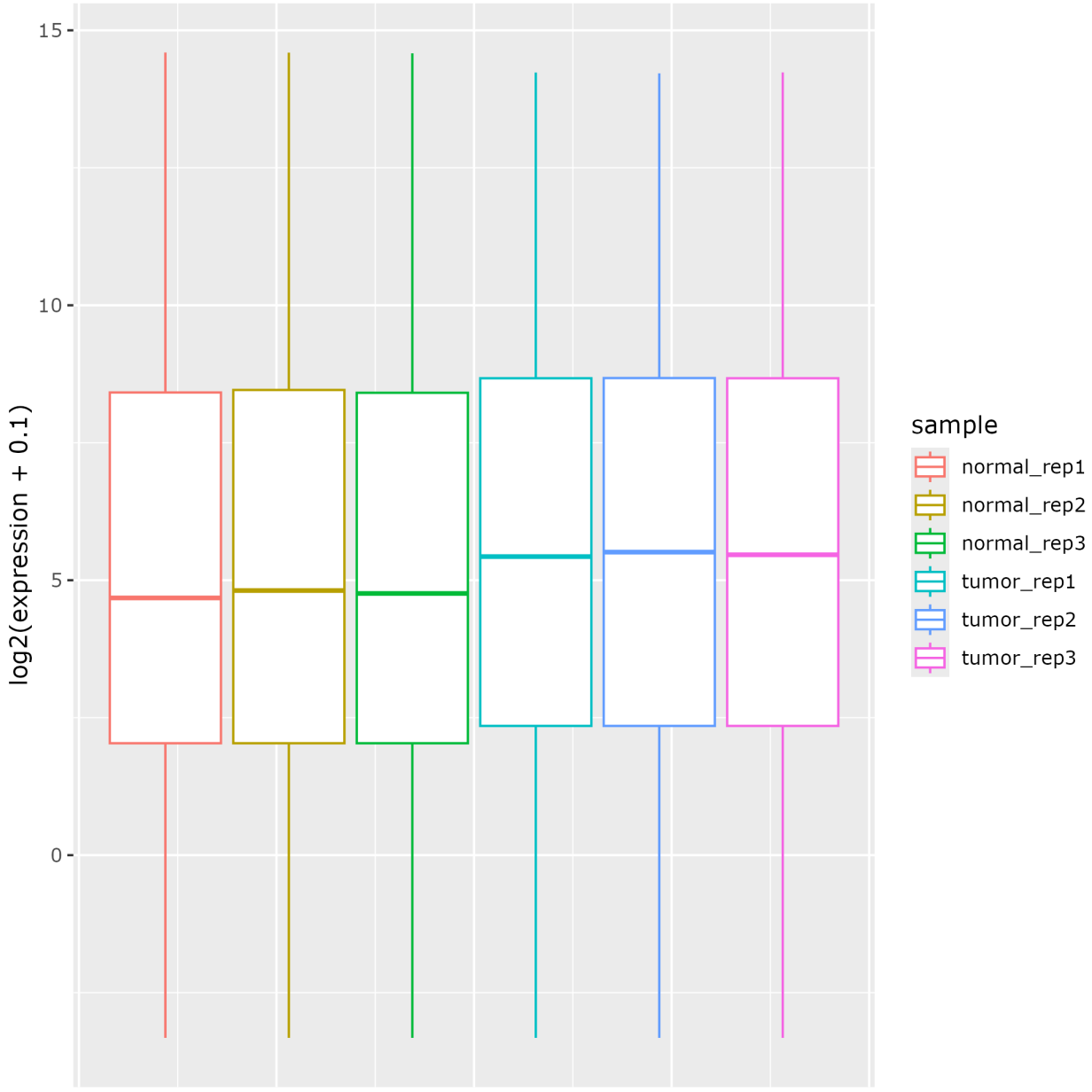
The distance plot is also a good tool for determining if the samples within groups are clustered to together. The legend scale indicates the distance and in this plot (the smaller the number the smaller the distance). It is expected that samples within the same group be closer to each other as compared to samples from a different biological condition. This QC step passes for the filtered gene expression table.



Expression Distribution

Even without normalization, the filtered expression distribution are very similar between the samples with the median and 25th to 75th percentiles higher in the tumor samples.


0.100
0.075
0.050
0.025
0.000
density
0
5
10
15
log2(expression + 0.1)
sample
normal_rep1
normal_rep2
normal_rep3
tumor_rep1
tumor_rep2
tumor_rep3


15
10
5
0
log2(expression + 0.1)
sample
normal_rep1
normal_rep2
normal_rep3
tumor_rep1
tumor_rep2
tumor_rep3

Lesson 14: Differential Expression Analysis for Bulk RNA Sequencing: The Actual Analysis

Lesson 13 Review

In the previous class, participants filtered gene expression results for the hcc1395 data to remove those genes whose expression is 0 across all samples and also those do not have enough replicates in a biological condition with greater than 0 expression. Subsequent QC of the filtered gene expression showed that the normal samples clustered together while tumor samples clustered with each other (PCA and distance plot), which indicates that biology is driving the underlying differences between the samples in this study.

Learning Objectives

After this lesson, participants will be able to:

- Describe on a high level how RNA sequencing data is modeled.
- Understand the importance of normalization in differential expression analysis.
- Interpret differential expression analysis results.

Sign onto Biowulf

Before getting started using the participant's assigned Biowulf student account ID.

```
ssh user@biowulf.nih.gov
```

Next, change into the `/data/user/hcc1395_b4b` folder.

```
cd /data/user/hcc1395_b4b
```

Request an interactive session with 12 gb of RAM, and 10 gb of local temporary storage.

```
sinteractive --mem=12gb --gres=lscratch:10
```

Load R

```
module load R
```

Run `deg.R`

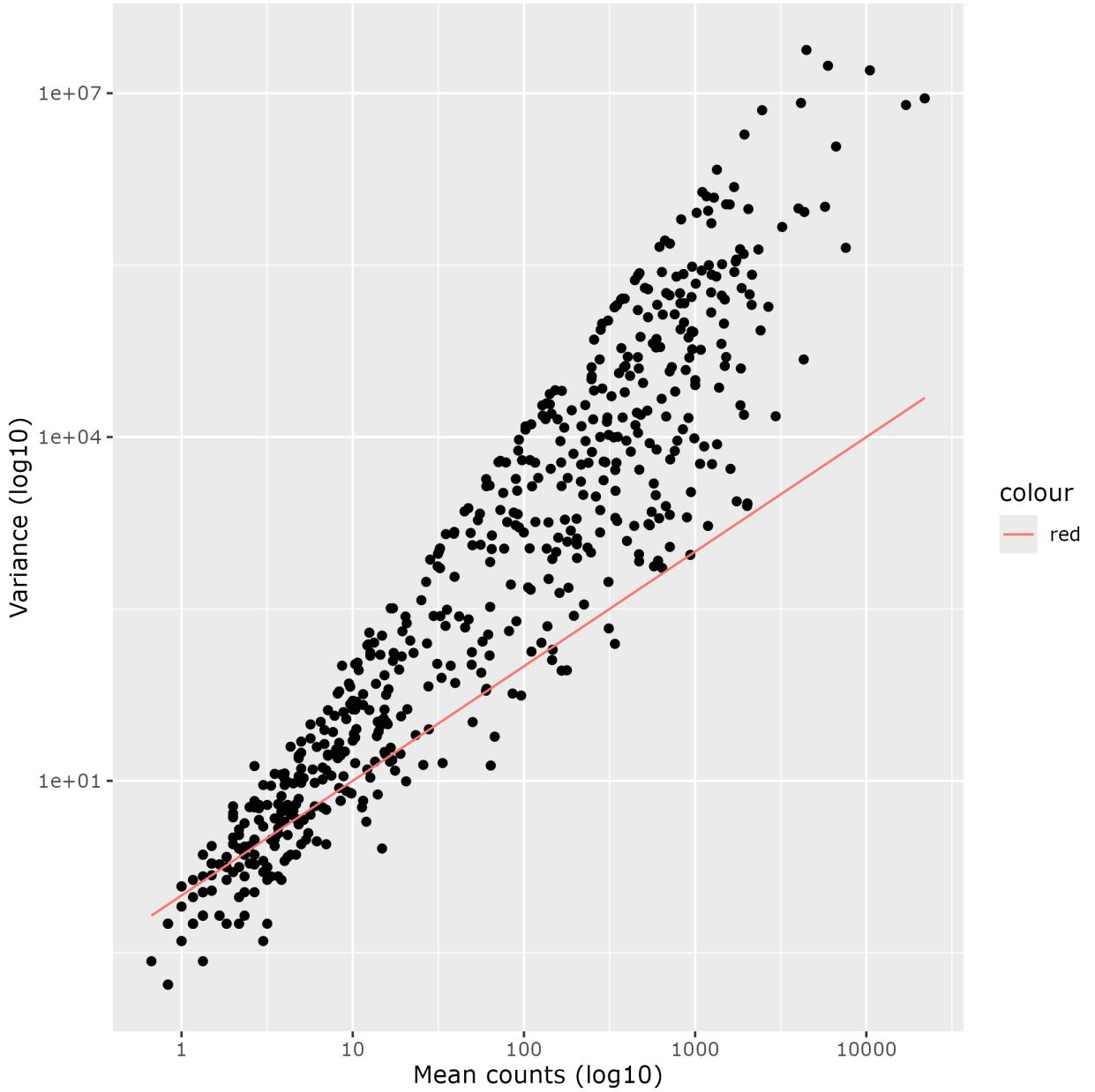
The R script for differential expression analysis, `deg.R` is stored in the `b4b_scripts` folder. It will generate differential expression results (using DESeq2), normalized expression, as well as visualizations.

A Bit of Background on the Negative Binomial Distribution

DESeq2 and another popular differential expression analysis package edgeR model expression data generated from RNA sequencing using the negative binomial distribtution *(https://en.wikipedia.org/wiki/Negative_binomial_distribution)*.

> "We assume that the number of reads in sample j that are assigned to gene i can be modeled by a negative binomial (NB) distribution" -- DESeq2 paper *(https://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-10-r106)* "We assume the data can be summarized into a table of counts, with rows corresponding to genes (or tags or exons or transcripts) and columns to samples. For RNA-seq experiments, these may be counts at the exon, transcript or gene-level. We model the data as negative binomial (NB) distributed" -- edgeR paper *(https://academic.oup.com/bioinformatics/article/26/1/139/182458?login=true)*

The rationale for modeling RNA sequencing expression data using negative binomial is that the assumption that mean equals variance under the Poisson distribution *(https://en.wikipedia.org/wiki/Poisson_distribution)* is not valid as variance is greater than the mean (see mean expression versus variance plot for the hcc1395 data below).



Note

The negative binomial distribution accommodates the unequal mean-variance relationship by modeling the expression data such that a both a mean and a variance parameter has to be estimated. It can be thought of that differential expression packages such as DESeq2 creates a model of and performs statistical testing for expected true value of gene expression. Statistical testing is not done directly on the input data.

Normalization

Normalization in RNA sequencing is important as this should place expression data for all samples under the same distribution. This process also removes technical variations due to sequencing depth (ie. more sequences generated from one sample as compared to another), RNA composition (ie. normal and tumor sample may differ in their composition of RNA), gene length, etc. For comparison between conditions, sequencing depth and RNA composition

should be accounted for. See here *(https://hbctraining.github.io/DGE_workshop/lessons/02_DGE_count_normalization.html)* for a summary of normalization methods and when it is appropriate to use them. DESeq2 uses median ratio normalization (see DESeq2 paper *(https://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-10-r106)*) which normalizes for sequencing depth and RNA composition.

To learn about the math behind median ratio normalization view the Stat Quest on DESeq2 normalization *(https://youtu.be/UFB993xufUU?si=rY2f_iP1BHE9p7Mu)*.

> **Note**
>
> Underneath the hood, given the raw expression table, DESeq2 will model the mean and variance of expression for each gene as well as perform normalization and calculate differential expression.

Perform Differential Expression Analysis

The script `deg.R` in `b4b_script` will be used perform differential expression analysis on the hcc1395 data. This script will use DESeq2 and takes the following as input.

- Filtered gene expression table, which is located in the folder `hcc1395_deg` and stored in the file `hcc1395_gene_expression_filtered.csv` so the path constructed from the current working directory of `hcc1395_b4b` is `hcc1395_deg/hcc1395_gene_expression_filtered.csv`.
- The phenotype information, which is located in the current working directory of `hcc1395_b4b` and stored in file `hcc1395_phenotypes.csv`.
- Experimental factor of interest for the differential expression analysis. Here, it is Treatment and the goal is to find whether there are genes whose expressions differ between normal and tumor samples.
- Study name: all output will have this prepended (in this case, the study name is `hcc1395`).
- Output directory for storing the results (ie. `hcc1395_deg`).

```
Rscript b4b_scripts/deg.R hcc1395_deg/hcc1395_gene_expression_filtere
```

As DESeq2 is running, the following messages will appear and informs the user that it is performing the normalization, estimating gene expression variance, fitting the expression data to a generalized linear model and performing statistical testing.

```
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
```

```
final dispersion estimates
fitting model and testing
```

Interpreting Differential Expression Analysis Results

Change into `hcc1395_deg` from `hcc1395_b4b`.

```
cd hcc1395_deg
```

List the contents. Use the `-1` option to view directory contents 1 item per line.

```
ls -1
```

The files generated from `deg.R` are:

- `hcc1395_deg.csv`: contains the differential expression analysis results for each gene. This table is used to generate the following plots:
  - `hcc1395_volcano.png`
- `hcc1395_filter_deg.csv`: contains the results for genes whose
  - log2 Fold Change is either greater than or equal to 4.5 with adjusted p-value less than 0.01.
  - log2 Fold Change is either less than or equal to -4.5 with adjusted p-value less than 0.01.
- Normalized gene expression are stored in `hcc1395_normalized_counts.csv` and is used to generate quality control plots post-normalization. The plots include:
  - `hcc1395_normalized_pca.png`
  - `hcc1395_normalized_distance.png`
  - `hcc1395_normalized_box.png`
  - `hcc1395_normalized_density.png`
- Normalized expression for genes appearing in the filtered differential analysis results (ie. `hcc1395_filtered_normalized_expression.csv`), which is used to generate the expression heatmap (ie. `hcc1395_expression_heatmap.png`).

```
hcc1395_box.png
hcc1395_deg.csv
hcc1395_density.png
hcc1395_distance.png
hcc1395_expression_heatmap.png
hcc1395_filter_deg.csv
hcc1395_filtered_normalized_expression.csv
hcc1395_gene_expression_filtered.csv
```

```
hcc1395_mean_variance.png
hcc1395_normalized_box.png
hcc1395_normalized_counts.csv
hcc1395_normalized_density.png
hcc1395_normalized_distance.png
hcc1395_normalized_pca.png
hcc1395_pca.png
hcc1395_volcano.png
```

Take a look at differential expression analysis results stored in `hcc1395_deg.csv` using `column`, where:

- `-t` tells the `column` command create a table.
- `-s` prompts for the field separator (comma in this case).
- `|` is used to send the output of `column` to `head` and to view the first 20 lines include the `-20` option in `head`.

```
column -t -s ',' hcc1395_deg.csv | head -20
```

Resize Terminal or Command Prompt window if content from the differential expression analysis table does not fit the entire screen.s

The columns in `hcc1395_deg.csv` can be explained as follows.

- Geneid: gene identifier.
- baseMean: mean expression across all samples for the corresponding gene.
- log2FoldChange: this is a metric for gene expression change between conditions. In the results below, this is calculated by taking the ratio between mean expression of the gene from the tumor samples and that for the normal samples and then taking the log2 of this ratio. A log2 fold change of 1 indicates an expression ratio between tumor and normal of 2, while a log2 fold change of -1 indicates an expression ratio between tumor and normal of 1/2.
- lfcSE: the standard error of the log2 fold change.
- pvalue: the uncorrected p-value of the likelihood of observing the effect of the size expression change (or larger) by chance alone. This p-value is not corrected for multiple comparisons.
- padj: Multiple comparison adjusted p-value.
- normalMean: average expression for a gene in the normal samples.
- tumorMean: average expression for a gene in the tumor samples.

Differential Expression Results

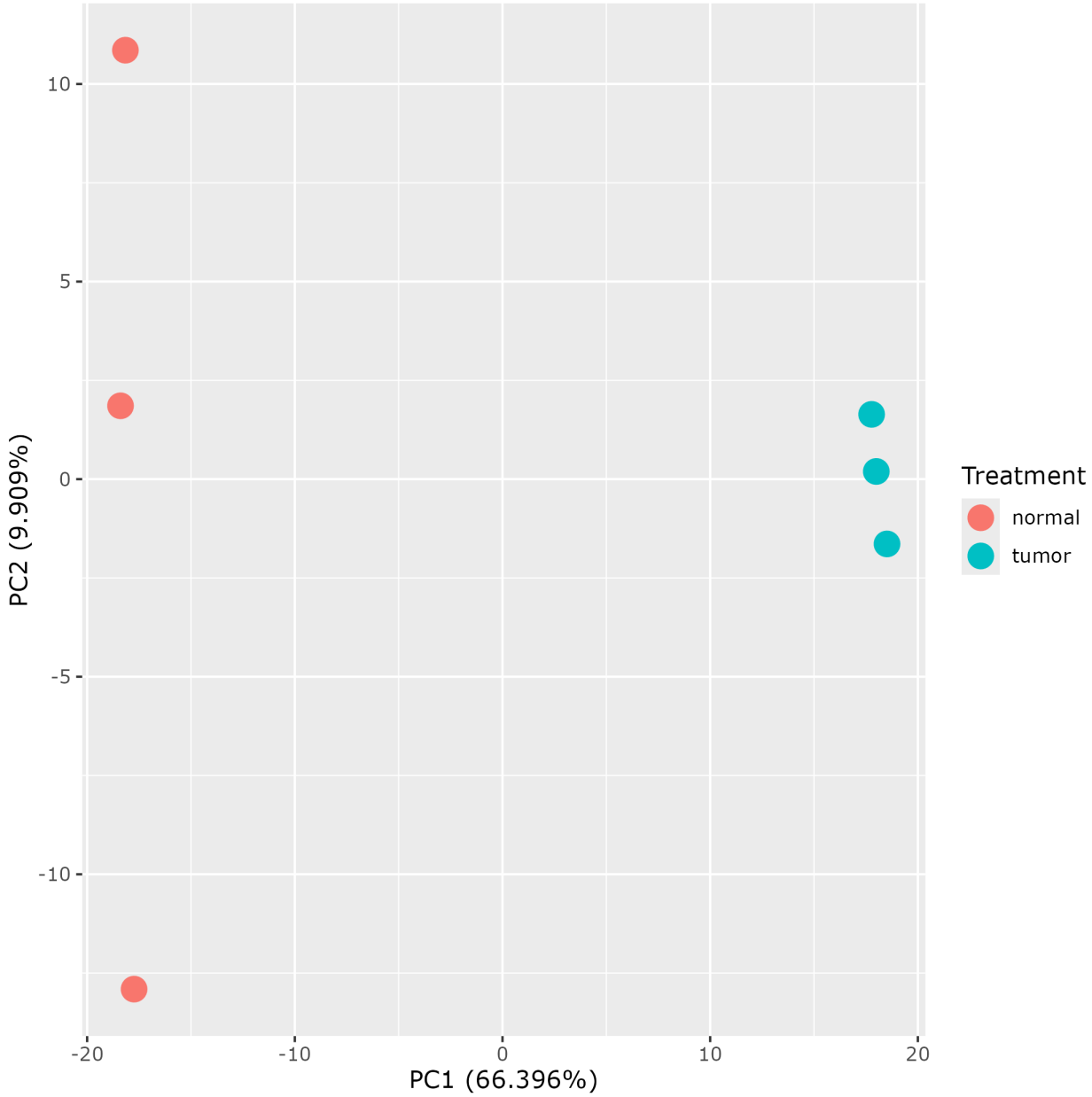
```
Geneid           baseMean   log2FoldChange  lfcSE  stat     pvalue     padj
LA16c-3G11.7     1.405      -1.878          1.6    -1.174   0.241      NA
```

DUXAP8	556.167	8.763	0.548	15.985	1.63e-57	1.1e-56
LL22NC03-N64E9.1	65.516	6.891	0.849	8.118	4.74e-16	1.5e-15
KCNMB3P1	6.339	-1.173	0.723	-1.623	0.105	0.15
XKR3	2.034	-0.778	1.26	-0.618	0.537	0.636
AC006548.28	2.586	0.453	1.047	0.432	0.666	0.746
CECR7	85.621	0.976	0.193	5.046	4.5e-07	1.05e-06
AC006946.16	28.073	1.322	0.335	3.945	7.99e-05	0.00016
AC006946.17	2.14	-0.062	1.128	-0.055	0.956	0.967
IL17RA	281.737	-0.694	0.103	-6.728	1.72e-11	4.79e-11
CECR6	2.253	0.569	1.155	0.493	0.622	0.711
AC006946.15	1.415	1.518	1.534	0.989	0.322	NA
CECR5	401.063	-1.79	0.09	-19.786	3.9e-87	4.09e-86
CECR2	3.109	0.173	0.935	0.185	0.854	0.89
SLC25A18	4.281	-0.062	0.806	-0.077	0.939	0.958
ATP6V1E1	1012.665	-0.723	0.054	-13.426	4.26e-41	2.3e-40
BCL2L13	1133.492	-0.493	0.05	-9.784	1.32e-22	5.16e-22
BID	713.738	-2.16	0.072	-29.844	1.04e-195	2.84e-194
LINC00528	38.191	-5.972	0.757	-7.887	3.09e-15	9.58e-15

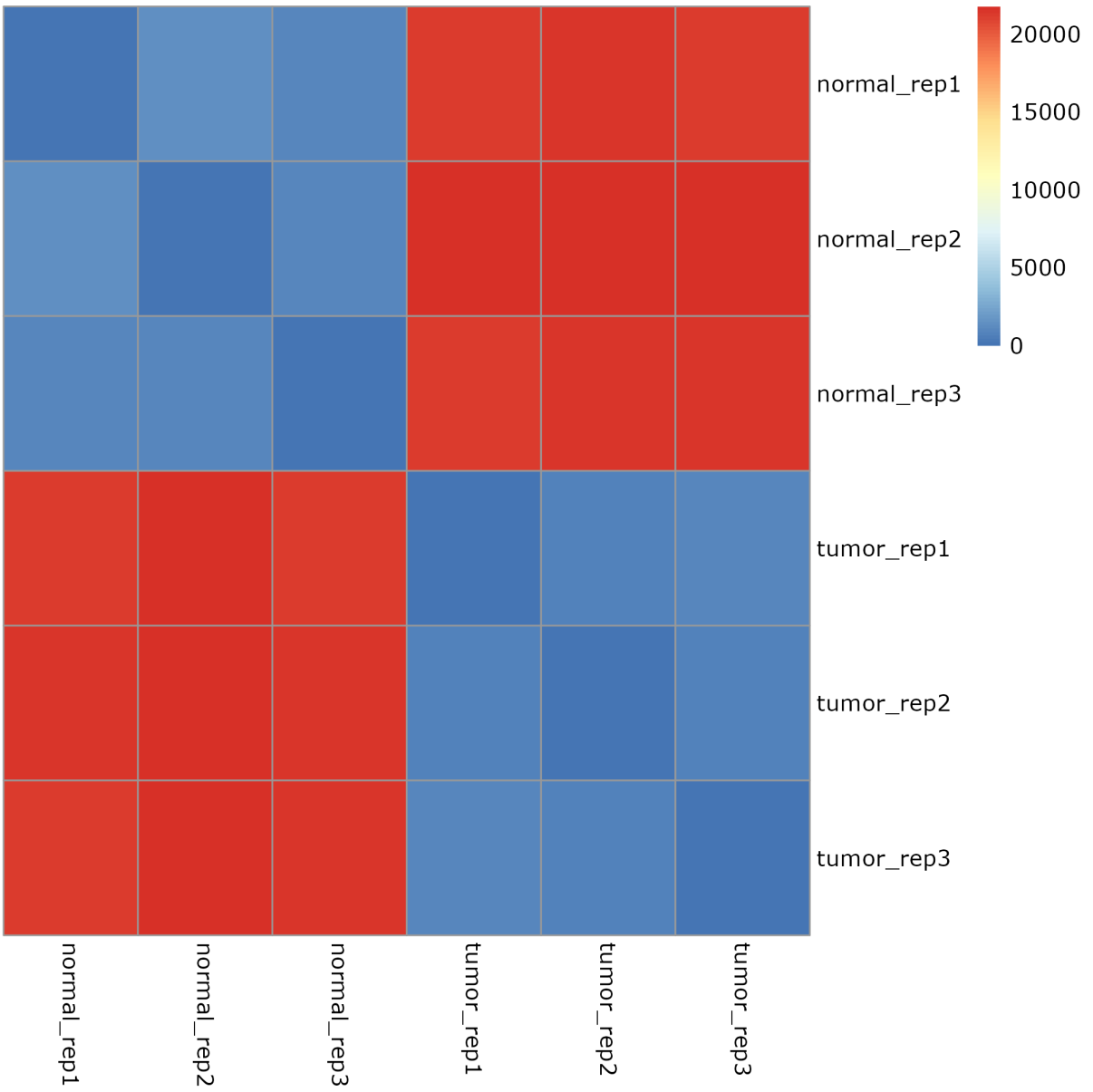
QC on Normalized Expression

It is a good idea to run QC on the normalized expression results to ensure that this step did not negatively alter the clustering of the samples and distribution of the gene expression.

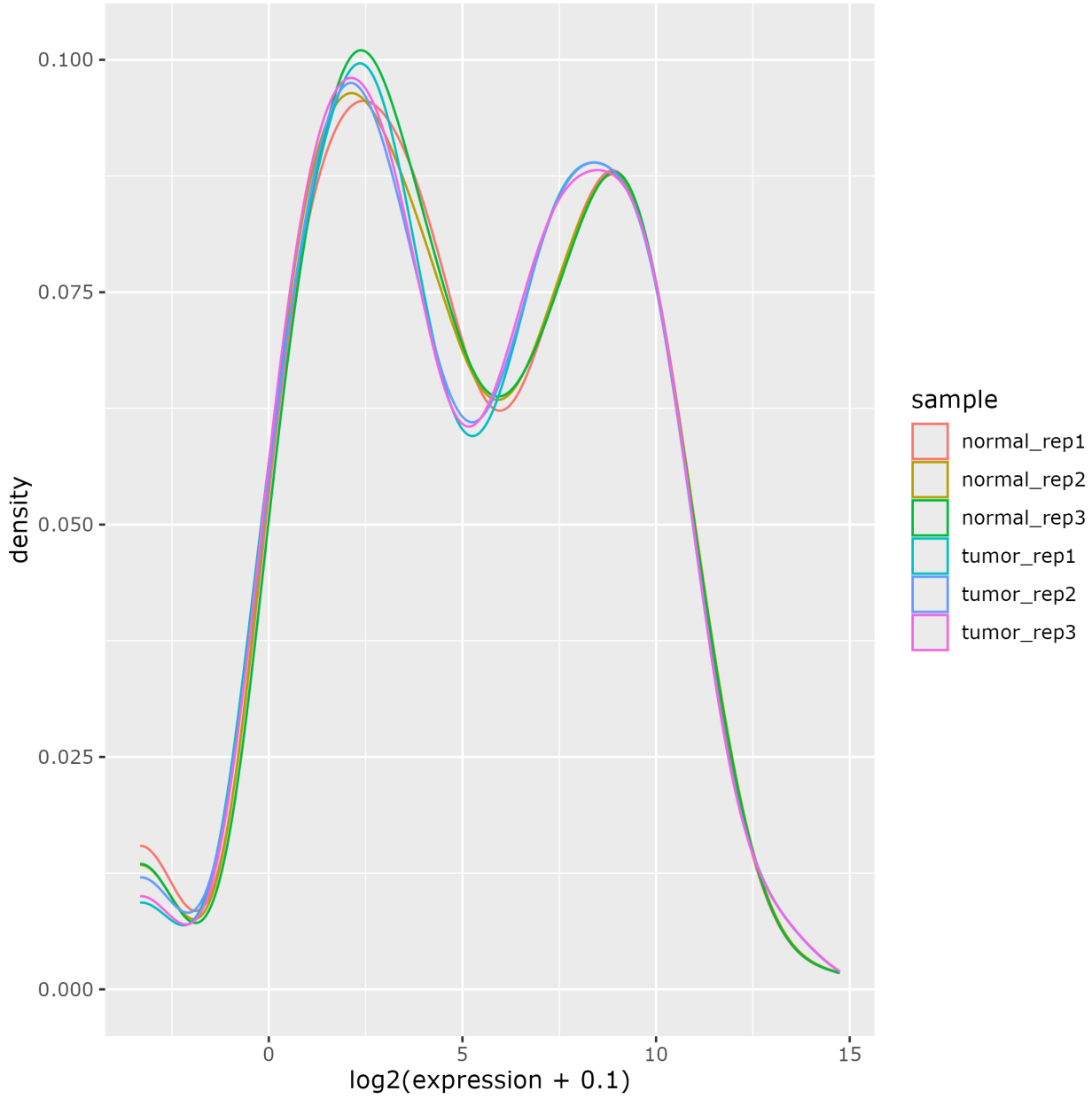
Normalization of the expression data brought the tumor samples closer together on the PCA plot.



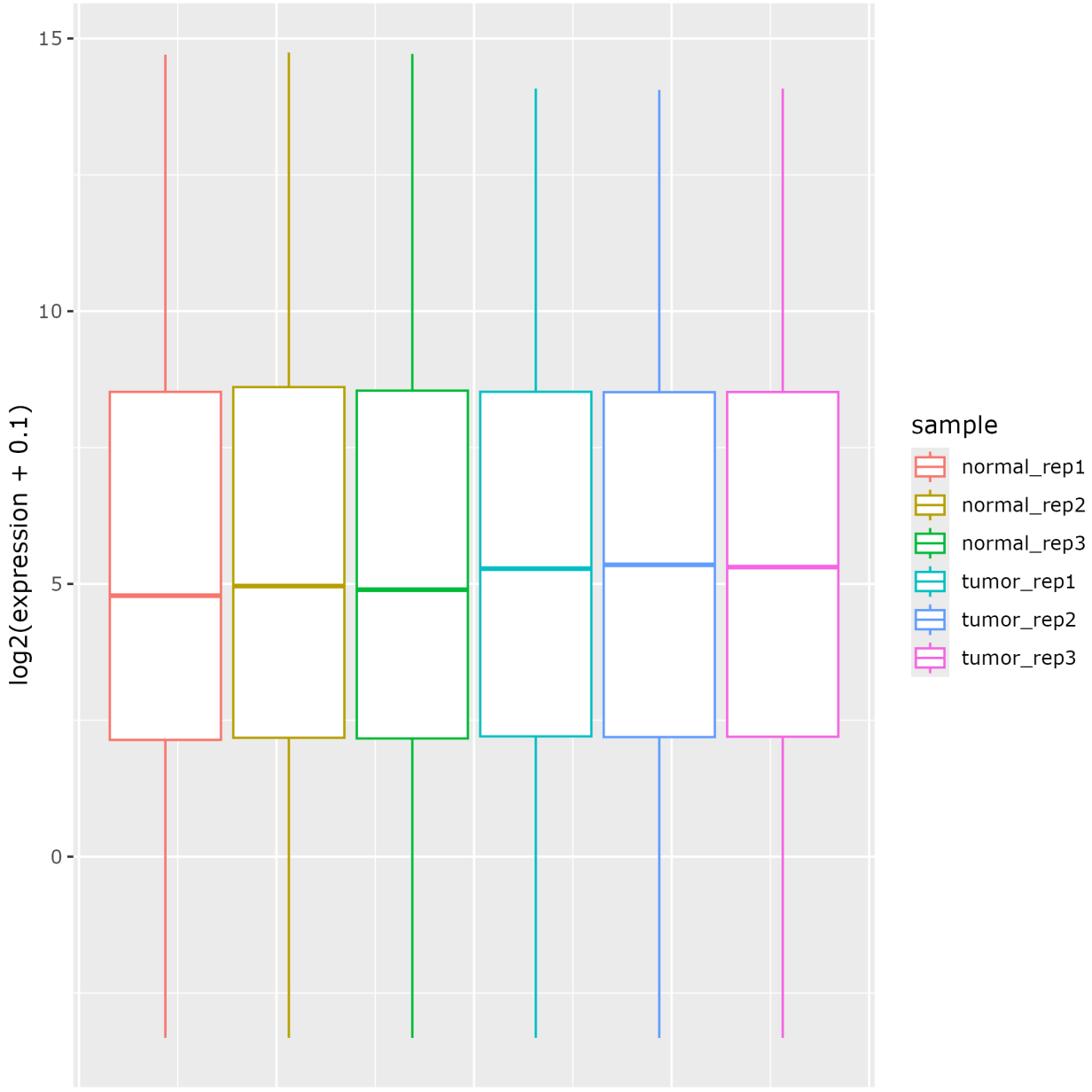
Normal samples were still closer together distance-wise as compared to the tumor samples according to the distance matrix.



Normalizing the filtered expression made the extremes (low and high expression) of the density plot overlap.



The 25th to 75th percentiles as well as median of gene expression of all samples are now more similar to each other after normalization.

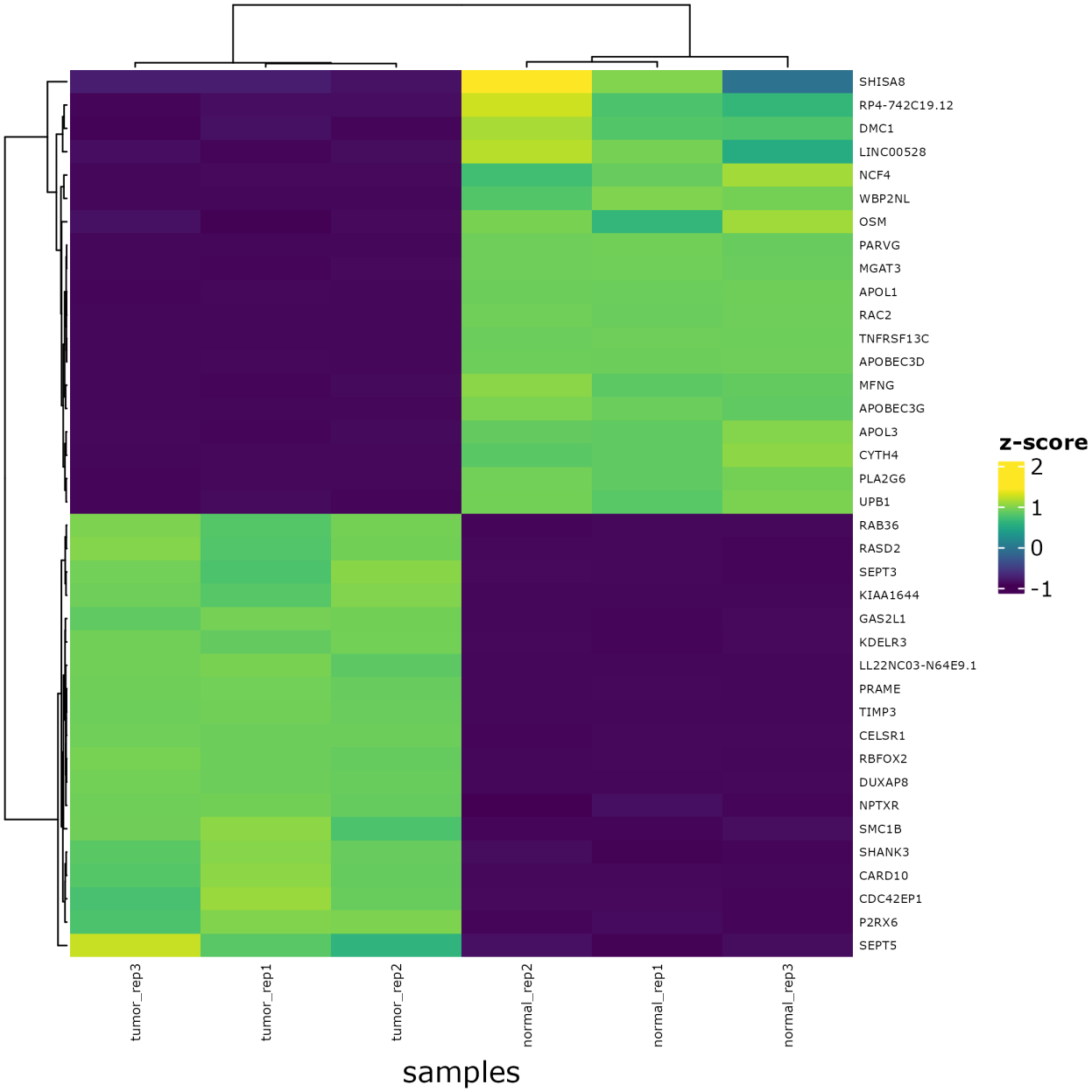


Visualizations

Expression Heatmap

A heatmap depicts numerical data on a color scale and is often used to visualize gene expression. The plot below is constructed using the normalized expression table that contains a subset of genes that were determined to be significantly changed between the normal and tumor samples according to the criteria below:

- log2 Fold Change is either greater than or equal to 2 with adjusted p-value less than 0.01
- log2 Fold Change is either less than or equal to 2 with adjusted p-value less than 0.01

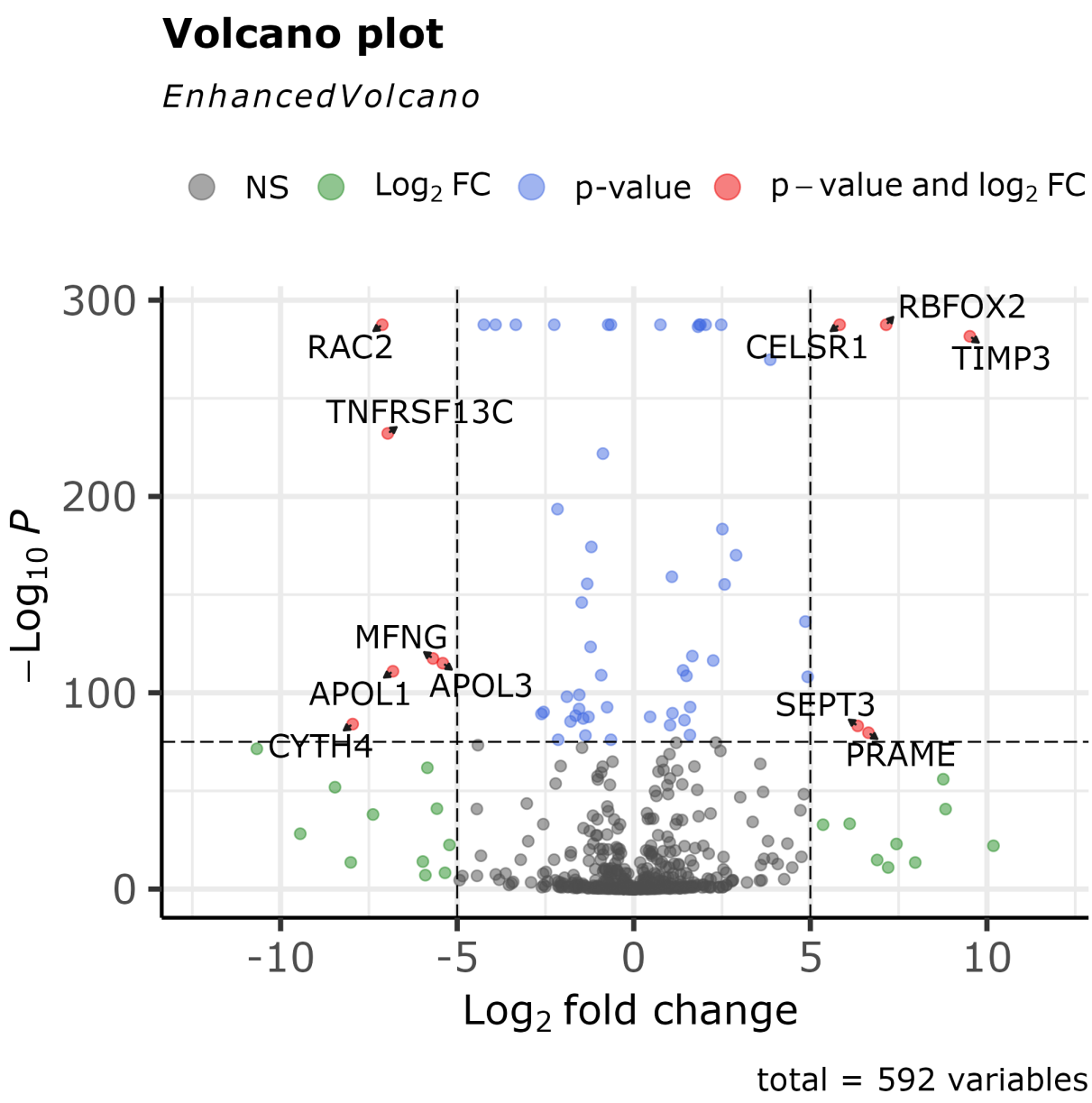


Note

The log2 fold change and p-value cutoff for statistically significant gene expression change is determined by the researcher.

Volcano Plot

A volcano plot helps scientists visualize change and statistical confidence of change. In RNA sequencing, this is log2 Fold Change plotted on the horizontal axis and -1og10 of the p-value on the vertical axis (adjusted p-value was used here).


Volcano plot
EnhancedVolcano
NS
Log2 FC
p-value
p – value and log2 FC
RAC2
TNFRSF13C
CELSR1
RBFOX2
TIMP3
MFNG
APOL1
APOL3
CYTH4
SEPT3
PRAME
300
200
100
0
−Log10 P
-10
-5
0
5
10
Log2 fold change
total = 592 variables

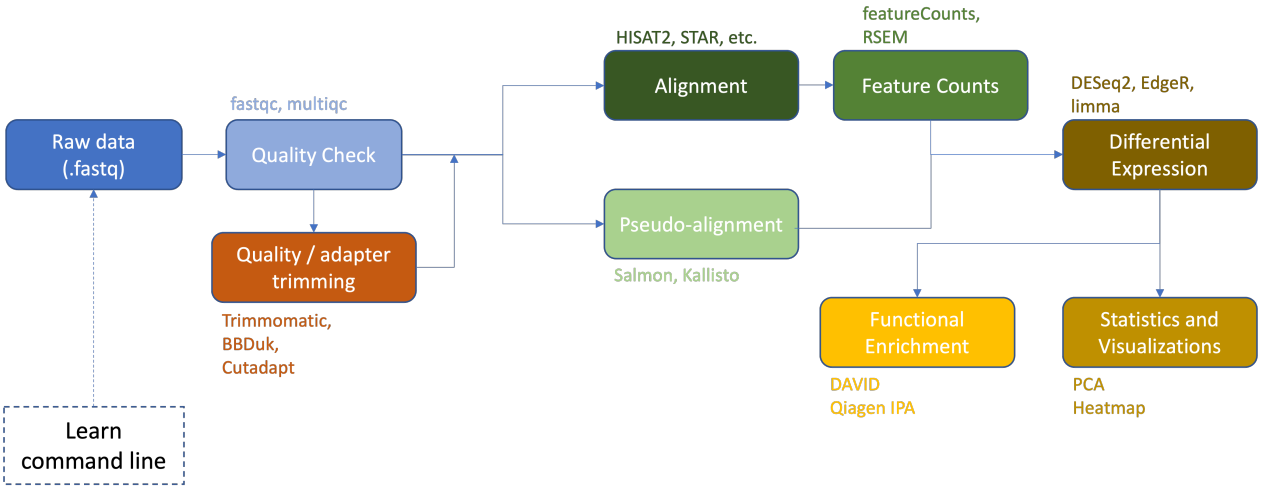
Module 3 - Pathway Analysis

Gene ontology and pathway analysis

Objectives

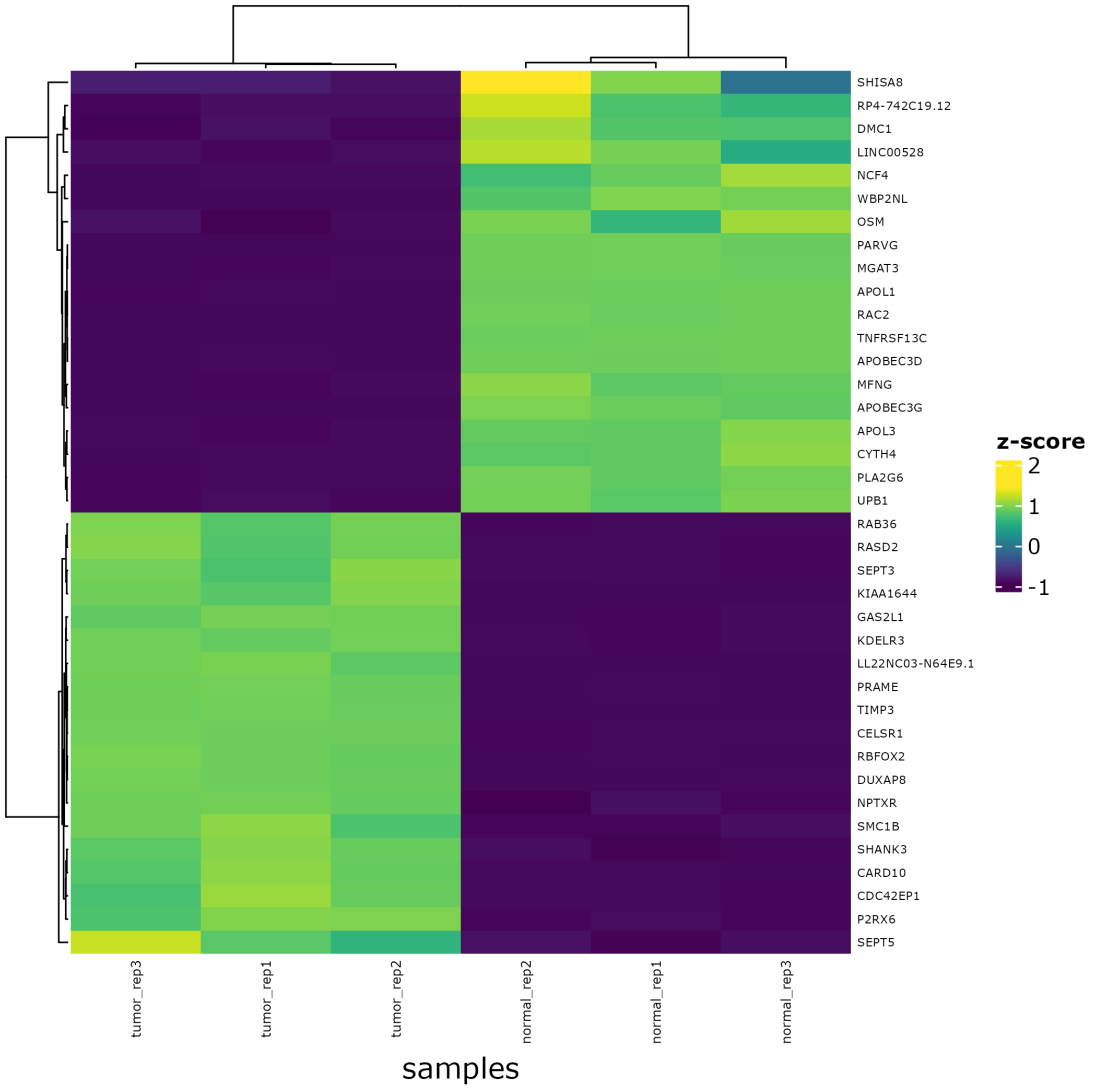
1. Determine potential next steps following differential expression analysis.
2. Tour geneontology.org and understand the three main ontologies.
3. Learn about different methods and tools related to functional enrichment and pathway analysis.
4. Get familiar with databases commonly used by popular functional enrichment tools.

Where have we been and where are we going?



Thus far we have:

1. Downloaded raw RNA-Seq data (.fastq files).
2. Examined raw data quality using `fastqc` and `multiqc` .
3. Performed adapter and quality trimming using `Trimmomatic`.
4. Aligned the raw sequences to a reference genome (human chromosome 22 from the GRCh38 version of the human reference genome) using `HISAT2`.
5. Viewed and compared alignments using IGV.
6. Generated a gene count matrix using `featureCounts`.
7. Performed differential expression analysis (`DESeq2`).
8. Generated a heatmap of differentially expressed genes.



Heatmap of differentially expressed genes (Normal vs Tumor).

You now have a potentially large list of differentially expressed genes. Now what? If you are like most biologists, you are interested in understanding these genes within their biological context.

To do that, we can examine gene ontology and perform some type of functional enrichment analysis or pathway analysis.

These types of analyses exploit the use of gene sets, and not all gene sets represent a pathway. Gene sets are collections of genes "formed on the basis of shared biological or functional properties as defined by a reference knowledge base. Knowledge bases are database collections of molecular knowledge which may include molecular interactions, regulation, molecular product(s) and even phenotype associations" Mathur et al. 2018 *(https://biodatamining.biomedcentral.com/articles/10.1186/s13040-018-0166-8)*.

Whereas, a pathway is not a simple list of genes but rather includes an interaction component usually related to a specific mechanism, process, etc.

What is gene ontology?

Many of the tools used to understand functional enrichment will use sets of GO terms, examining GO enrichment. What do we mean by GO?

> The Gene Ontology (GO) provides a framework and set of concepts for describing the functions of gene products from all organisms. --- https://www.ebi.ac.uk/ols/ontologies/go *(https://www.ebi.ac.uk/ols/ontologies/go)*.

This is manually curated by team members of the GO consortium.

There are two parts to the gene ontology: (Check out https://www.youtube.com/watch?v=6Am2VMbyTm4 *(https://www.youtube.com/watch?v=6Am2VMbyTm4)* for a more detailed overview)

1. the ontology (the GO terms and their hierarchical relationship) - form a directed, acyclic graph structure (nodes = GO terms, edges = relationships)
2. the annotations (the annotated genes linked to various GO terms)

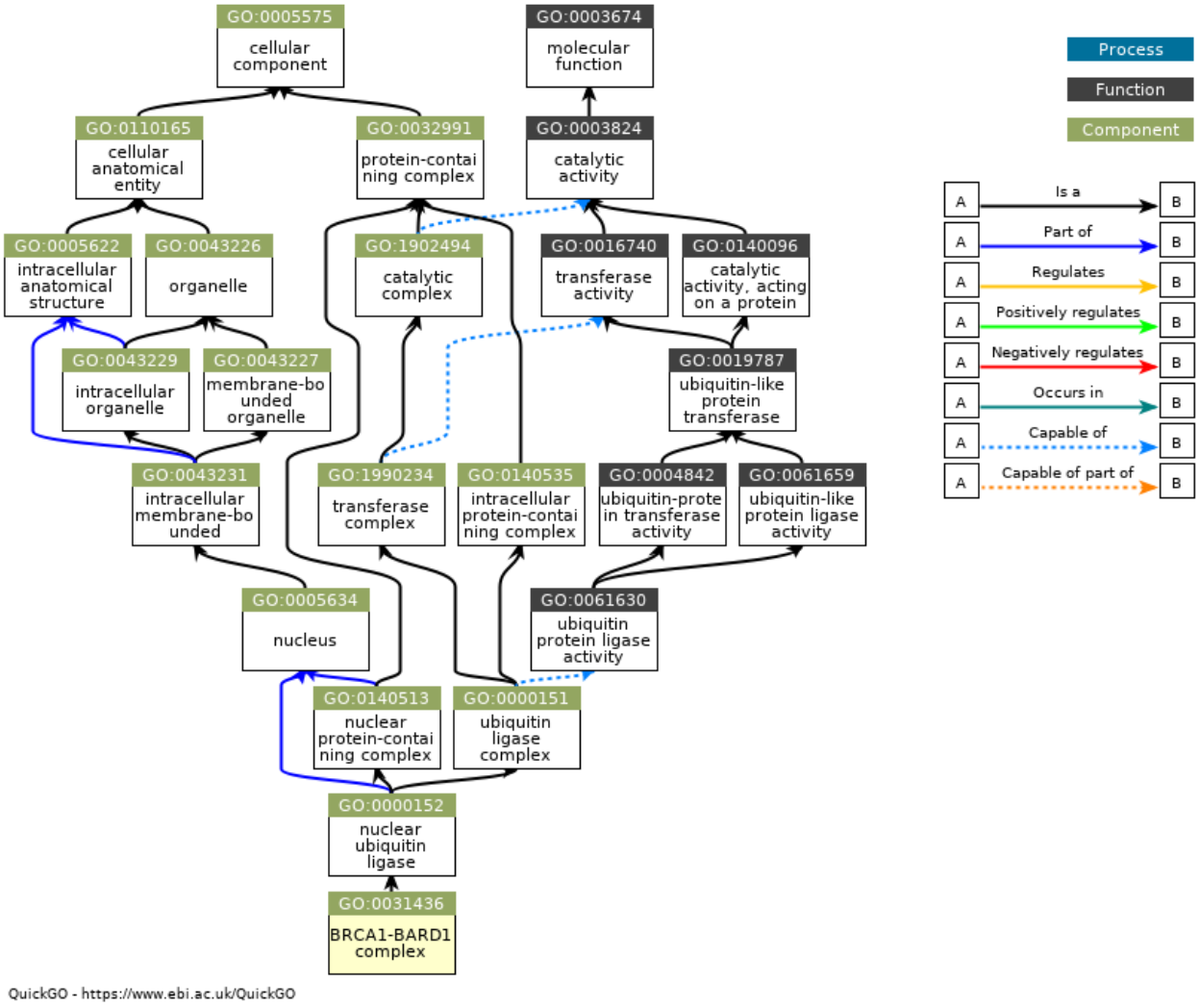


Image from https://www.ebi.ac.uk/QuickGO/GTerm?id=GO:0031436 *(https://www.ebi.ac.uk/QuickGO/GTerm?id=GO:0031436)* .

What is a GO term?

- GO terms provide information about a gene product
- GO terms as a vocabulary are species agnostic, but there are species constraints
- ontology and annotations are updated regularly
- computer readable - suitable for bioinformatics

GO integrates information about gene product function in the context of three domains:

1. molecular function (F) - "the molecular activities of individual gene products" (e.g., kinase)
2. cellular component (C) - "where the gene products are active" (e.g., mitochondria)
3. *biological process (P) - "the pathways and larger processes to which that gene product's activity contributes " (e.g., transport)

*Commonly used for pathway enrichment analysis

What is GO Slim? Reducing Semantic Similarity

There is a lot of redundancy in pathway analysis results, and there are different tools that can be used for reducing this redundancy (e.g., DAVID's Functional Annotation Clustering, REVIGO, GOSemSim). Alternatively, users can focus on sets of terms at different levels in the GO hierarchy, which range from broad to more specific. There are also GO Slim terms *(https://www.ebi.ac.uk/training/online/courses/goa-and-quickgo-quick-tour/what-is-quickgo/what-can-i-do-with-a-go-slim/)*, which are simplified subsets from GO. GO slim terms are useful for providing a high level summary of functions.

Checkout geneontology.org *(https://geneontology.org/)* to learn more about GO.

Approaches to gene set analysis / pathway analysis?

Functional enrichment and pathway analysis have broad and varying definitions. For our purposes, there are three general approaches: 1. Over-Representation Analysis (ORA), 2. Functional Class Scoring (FCS), and 3. Pathway Topology (PT) (Khatri et al. 2012 *(https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375#pcbi-1002375-t001)*).

Examining genes in a set allows us to:

1. increase the statistical power in our analysis
2. ease interpretation
3. predict new roles for genes
4. better integrate data from different methods

Over-representation analysis (ORA)

> statistically evaluates the fraction of genes in a particular pathway found among the set of genes showing changes in expression --- Khatri et al. 2012 *(https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375#pcbi-1002375-t001)*

From this, ORA determines which pathways are over or under represented by asking "are there more annotations in the gene list than expected?"

Things to know:

- Tests based on hypergeometric, fisher's exact, chi-square, or binomial distribution; these determine the probability that the number of genes in our gene list and found in a given gene set are observed by chance.
- Assumes independence between genes.
- Requires an appropriate background gene set for comparison.
    - This could be:
    - all genes in the organism of interest

    - all protein coding genes
    - only the genes that are measured (microarray)
    - **only the genes that are expressed (RNA-Seq) *(https://genomebiology.biomedcentral.com/articles/10.1186/s13059-015-0761-7)***
    - genes in a gene set collection
- Prioritizes a subset of genes using an arbitrary, user determined threshold.
- Doesn't require the data, just the gene identifiers.
- Example tools include DAVID and Qiagen IPA.

To understand the statistics behind ORA, see this guide *(https://www.pathwaycommons.org/guide/primers/statistics/fishers_exact_test/)* from Pathway Commons and this video *(https://www.youtube.com/watch?v=H1cUs6pql9s&list=PLRGSLMDpRxCrTybN8g8GjNk39lmzJ2UTF&index=2)* from Biostatsquid.

Functional Class Scoring (FCS)

> Includes 'gene set scoring' methods such as GSEA, which first compute DE scores for all genes measured, and subsequently compute gene set scores by aggregating the scores of contained genes. --- Geistlinger et al. 2021 *(https://academic.oup.com/bib/article/22/1/545/5722384?login=true)*

These methods are more sensitive than ORA methods, but are more challenging to implement.

GSEA

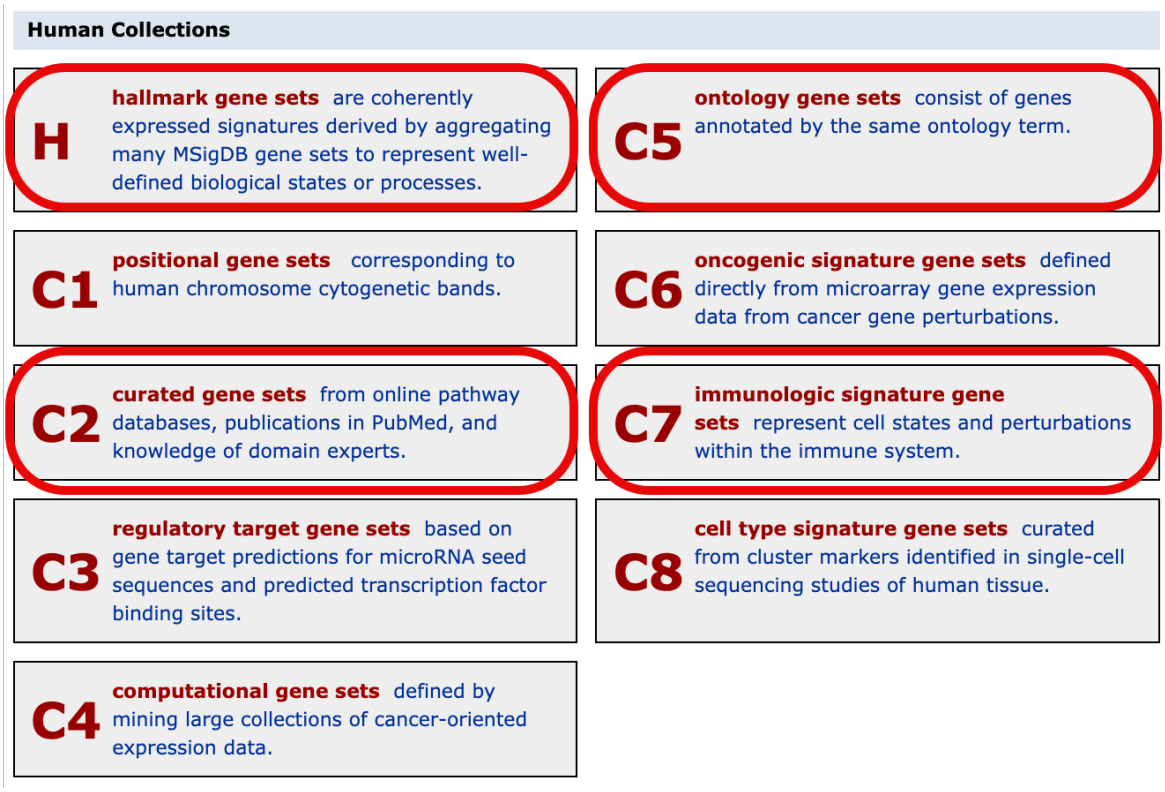
- Ignores gene position and role
- Does not pre-select genes; considers all gene expression in the form of a ranked list. You must include data with gene identifiers for ranking.
    - Ranking by magnitude of change in gene expression between conditions and p-value.
    - Top = upregulated + significant
    - Bottom = down-regulated + significant
    - Middle = non-significant
    - Determines where genes from a gene set fall in the ranking (i.e., at the top of the list or the bottom of the list).
    - Creates a running sum statistic
    - uses a permutational approach to determine significance (e.g., Kolmogorov-Smirnov)
- Broad Institute software but also available using web-based tools, R (See fgsea for a modified approach), and Qlucore (proprietary).
- also considered a strategy encompassing a range of methods
  - self-contained methods vs competitive methods

Check out this video for a nice overview of GSEA *(https://www.youtube.com/watch?v=egO7Lt92gDY)*.

What is MSigDB *(https://www.gsea-msigdb.org/gsea/msigdb/index.jsp)* and how does it relate to GSEA?

The Molecular Signatures Database (MSigDB) is a curated resource of thousands of gene sets by the Broad Institute. These sets were curated for use with GSEA software but are used with other tools as well.

- Includes both human and mouse collections.
- There are 34,837 gene sets in the Human Molecular Signatures Database (MSigDB) (not all gene sets are related to pathways).
    - Includes 9 larger, themed collections



    - Highlighted examples:
        - C5 - the gene ontology (GO) collection
        - C2 - curated gene sets from publications and pathway databases (e.g., KEGG and REACTOME).
        - Hallmark collection - a summary list of well-defined gene sets with decreased redundancy. A good choice for many studies.
        - C7 - great for immunological research.

Pathway Topology

ORA and FCS discard a large amount of information. These methods use gene sets, and even if the gene sets represent specific pathways, structural information such as gene product

interactions, positions of genes, and types of genes is completely ignored. Pathway topology methods seek to rectify this problem.

PT methods are mostly considered network based.

Some examples:

Impact analysis (iPathwayGuide)

> constructs a mathematical model that captures the entire topology of the pathway and uses it to calculate a perturbation for each gene. Then, these gene perturbations are combined into a total perturbation for the entire pathway and a p-value is calculated by comparing the observed value with what is expected by chance. (https://advaitabio.com/ipathwayguide/more-accurate-pathway-rankings-using-impact-analysis-instead-of-enrichment/)

Other tools include Pathway-Express, SPIA, NetGSA, TPEA, etc. (See Nguyen et al. 2019 *(https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1790-4)* for a review of PT methods.)

These methods have shown to produce more accurate results when the user wishes to understand the types and directions of gene interactions and underlying mechanisms. However, such methods often "require experimental evidence for pathway structures and gene–gene interactions, which is largely unavailable for many organisms." *(https://www.sciencedirect.com/science/article/pii/S0168952523000185?via%3Dihub#bb0055)*

What tools are available?

This is neither a comprehensive list of tools nor an endorsement of certain tools, but rather a list of semi-popular tools with different approaches.

Note

There are a ton of tools out there. Be aware of the background methods used and the quality of results returned. Also, check to see when the tool was last updated.

Gene set analysis tools

Tool	Author	Year	Citations[1]	Availability	Gene sets	Methods[2]
WEBGESTALT	Zhang *et al.* [73]	2005	1423	Web server	GO, KEGG, +20 more	ORA, GSEA
GOSTATS	Falcon and Gentleman [74]	2007	1437	R package	GO	ORA
G:PROFILER	Reimand *et al.* [75]	2007	534	Web server	GO, KEGG, +7 more	ORA
GENETRAIL	Backes *et al.* [76]	2007	360	Web server	GO, KEGG, +28 more	ORA, GSEA
DAVID	Huang *et al.* [8]	2009	19 569	Web server	GO, KEGG, +38 more	ORA
GORILLA	Eden *et al.* [77]	2009	1881	Web server	GO	ORA
TOPPGENE	Chen *et al.* [78]	2009	1200	Web server	GO, KEGG, +45 more	ORA
CLUSTER-PROFILER	Yu *et al.* [10]	2012	1305	R package	GO, KEGG, +8 more	ORA, GSEA
PANTHER	Mi *et al.* [79]	2013	1405	Web server	GO, +2 more	ORA, GSEA
ENRICHR	Chen *et al.* [9]	2013	1246	Web server	GO, KEGG, +33 more	ORA

[1]Google Scholar, July 2019.
[2]Detailed summary of implemented methods in Supplementary Methods S1.2.

Table from Geistlinger et al. 2020 *(https://academic.oup.com/bib/article/22/1/545/5722384)*

Other and related tools

- Gene Set Enrichment Analysis (GSEA) *(http://software.broadinstitute.org/gsea)*
- EnrichmentMap *(https://apps.cytoscape.org/apps/enrichmentmap)*
- REVIGO *(http://revigo.irb.hr/)* (reducing and visualizing gene ontology)
- Pathview *(https://pathview.uncc.edu/)*
- iPathwayGuide *(https://advaitabio.com/ipathwayguide/)* (proprietary)
- Qiagen IPA *(https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/software/ipa/)* (proprietary, CCR license)
- Qlucore *(https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/software/qlucore/)* (proprietary, CCR license)
- GeneMANIA *(https://apps.cytoscape.org/apps/genemania)*
- CellNetAnalyzer *(http://www2.mpi-magdeburg.mpg.de/projects/cna/cna.html)*
- PARADIGM *(http://paradigm.five3genomics.com)*

Other databases

There are many databases devoted to relating genes and gene products to pathways, processes, and other phenomenon. Again, the following is not meant to be a comprehensive list.

Kyoto Encyclopedia of Genes and Genomes (KEGG) *(https://www.genome.jp/kegg/kegg1a.html)*

- Curated database
- Biological pathways
- Molecular interaction networks
- Includes very nice pathway maps - metabolic pathways, disease pathways, drug-target interactions, etc.
- System-level integration
- Restricted licenses

Reactome *(https://reactome.org/)*

> The Reactome Knowledgebase systematically links human proteins to their molecular functions, providing a resource that functions both as an archive of biological processes and as a tool for discovering novel functional relationships in data such as gene expression studies or catalogs of somatic mutations in tumor cells. --- Jassal et al. 2019 *(https://academic.oup.com/nar/article/48/D1/D498/5613674)*

- Curated database including metabolism, signaling, and other biological processes
- Human specific
- Also includes disease super-pathways
- Several built-in pathway analysis tools

Pathway Commons *(https://www.pathwaycommons.org/)*

- A meta-database of pathways from other pathway databases
- Standardized format

PANTHER *(http://www.pantherdb.org/pathway/)*

> The PANTHER (Protein ANalysis THrough Evolutionary Relationships) Classification System was designed to classify proteins (and their genes) in order to facilitate high-throughput analysis. The core of PANTHER is a comprehensive, annotated "library" of gene family phylogenetic trees. --- pantherdb.org/about.jsp *(http://pantherdb.org/about.jsp)*

- Especially useful when considering evolutionary relationships

WikiPathways *(https://www.wikipathways.org/index.php/WikiPathways)*

- Community driven meta-database of pathways
- A great source of less well-established pathways

NDEx *(https://home.ndexbio.org/index/)*

> The NDEx Project provides an open-source framework where scientists and organizations can store, share, manipulate, and publish biological network knowledge. - ndexbio.org *(https://home.ndexbio.org/about-ndex/)*

HumanCyc *(https://humancyc.org/)* (See BioCyc)

> HumanCyc provides an encyclopedic reference on human metabolic pathways, the human genome, and human metabolites. --- humancyc.org *(https://humancyc.org/)*

DisGeNET *(https://disgenet.com/)*

- A knowledgebase of genes and associated diseases.

Pathguide *(http://pathguide.org/?organisms=all&availability=all&standards=all&order=alphabetic&DBID=none)*

- Looking for a specific database? Pathguide contains a resource list of pathways searchable by organism and resource type.
- Most recent update was 2017

Importance of Gene IDs

To use various tools for functional analysis, you will need a list of annotated genes. Gene annotations come in a variety of flavors and not all flavors are compatible with every tool. For example, Gene Ontology (GO) is associated with Entrez, Ensemble, and offical gene symbols (assigned by the HUGO Gene Nomenclature Committee (HGNC)).

Note: Genome builds will have differences in the names and coordinates of genomic features, which will impact gene ID conversions. See this tutorial *(https://github.com/hbctraining/Training-modules/blob/master/DGE-functional-analysis/lessons/Genomic_annotations.md)* from the Harvard Chan Bioinformatics Core.

Some tools to help with annotation / conversion:
- g:Convert *(https://biit.cs.ut.ee/gprofiler/convert)*
- Ensembl Biomart
- AnnotationHub *(https://bioconductor.org/packages/release/bioc/html/AnnotationHub.html)*
- DAVID Gene ID Conversion *(https://davidbioinformatics.nih.gov/conversion.jsp)*

Other Considerations

- Describe what method(s) you use for functional enrichment or pathway analysis clearly in any resulting publications. This should include parameters used and software versions.
- For ORA, select appropriate background genes.
- Use p-values corrected for multiple comparisons. If more than one gene set is tested, the p-values should be corrected to minimize false positive.

- Many of the classic methods developed within the above approaches were designed specifically for transcriptomic data. Other types of -omic data (e.g., proteomics, metabolomics, scRNA-seq, GWAS) differ intrinsically from transcriptomic data, and these differences should be considered when selecting a method. For methods/tools specific to proteomics, GWAS, epigenomics, and multi-omics, see Zhao and Rhee 2023 *(https://www.sciencedirect.com/science/article/pii/S0168952523000185?via%3Dihub#bb0055)*.

Resources:

- ClusterProfiler tutorial *(https://bioinformatics.ccr.cancer.gov/docs/btep-coding-club/CC2023/FunctionalEnrich_clusterProfiler/)*
- Functional enrichment and comparison with R *(https://alexslemonade.github.io/refinebio-examples/03-rnaseq/pathway-analysis_rnaseq_01_ora.html)* .
- ClusterProfiler, pathview, and good introductory information *(https://github.com/hbctraining/Training-modules/blob/master/DGE-functional-analysis/lessons/functional_analysis_2019.md)*
- Article on the impact of the evolving GO *(https://www.nature.com/articles/s41598-018-23395-2)*
- Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges, PLOS Computation Biology, 2012 *(https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375)*
- Pathway enrichment analysis and visualization of omics data using g:Profiler, GSEA, Cytoscape and EnrichmentMap *(https://www.nature.com/articles/s41596-018-0103-9)*
- Toward a gold standard for benchmarking gene set enrichment analysis, Briefings in Bioinformatics, 2021 *(https://academic.oup.com/bib/article/22/1/545/5722384)*
- Introductory lectures on functional enrichment and R from DIY Transcriptomics *(https://diytranscriptomics.com/project/lecture-10)*

Database for Annotation, Visualization and Integrated Discovery (DAVID) - An Overview

Lesson 15 review

In Lesson 15, we learned about functional enrichment, pathway analysis, and related concepts. These analyses help us put RNA sequencing results into biological context by informing us of the biomolecular pathways, biological functions, cellular localities, etc. of genes in our study.

Learning objectives

This lesson will provide an overview of the Database for Annotation, Visualization and Integrated Discovery (DAVID) *(https://david.ncifcrf.gov/home.jsp)*. DAVID is one of many tools that can be used to perform functional enrichment analysis.

In this lesson, we will

- Learn more about DAVID
    - Available tools
    - Underlying statistical methods
    - Some expected outputs
    - Data size limits
    - How to get help
- Run an example and interpret results
    - Starting An Analysis in DAVID
    - DAVID Inputs - Gene lists and background
    - Results
        - Annotation Results Summary
        - Functional Annotation Chart
        - Functional Annotation Cluster
        - Functional Annotation Table
        - Gene Functional Classification

Potential Files of Interest:

Files used in this tutorial or related to files used in this tutorial include:

1. Up-regulated gene list - up_logfold3.txt
2. Background gene list - background_expressed.txt
3. Differential Expression Results - hcc1395_deg.csv

Background on DAVID

What does DAVID do?

> DAVID is a popular bioinformatics resource system including a web server and web service for functional annotation and enrichment analyses of gene lists. It consists of a comprehensive knowledgebase *(https://davidbioinformatics.nih.gov/helps/update.html)* and a set of functional analysis tools. - Sherman et al. 2022 *(https://pubmed.ncbi.nlm.nih.gov/35325185/)*
>
> Over 40 functional categories from dozens of independent public sources (databases) are collected and integrated into the DAVID Knowledgebase. - https://davidbioinformatics.nih.gov/helps/knowledgebase/DAVID_gene.html#coverage *(https://davidbioinformatics.nih.gov/helps/knowledgebase/DAVID_gene.html#coverage)*

DAVID was created and is maintained by the Laboratory of Human Retrovirology and Immunoinformatics *(https://frederick.cancer.gov/research/laboratory-human-retrovirology-and-immunoinformatics)* at Frederick National Lab. It has been cited in 72,287 papers since its debut in 2003 as of 23 July 2024.

Tools available in DAVID include:

1. Functional Annotation Clustering
2. Functional Annotation Chart
3. Functional Annotation Table
4. Gene Functional Classification
5. Gene ID Conversion
6. Gene Name Batch Viewer
7. Ortholog Tool

With DAVID, we can:

- Identify enriched biological themes, particularly GO terms.
- Discover enriched functional-related gene groups.
- Cluster redundant annotation terms.
- Visualize genes on BioCarta & KEGG pathway maps.
- Display many-genes-to-many-terms relationships in 2D.
- Search for functionally related genes not in the list.
- List interacting proteins.
- Explore gene names in batch mode.

Note

DAVID is not just for gene lists associated with humans; it includes annotations relevant to thousands of species.

Basic statistics behind DAVID

Fisher Exact Test

DAVID performs over representation analysis (ORA) at its core, which aims to find enriched molecular functions, pathways, or other annotations represented by the input gene list.

With DAVID, we are essentially looking at contingency tables (Figure 1). The example in Figure 1 shows the number of user input genes and background genes (selected from the whole genome) that fall onto a particular pathway (ie. p53 signaling). We want to know the probability that the number of user input genes that map to a given pathway is by random chance. In other words, do user input genes fall onto a pathway more often as compared to the background than expected? DAVID uses a modified Fisher exact test to determine whether a pathway is over-represented in our gene list.

DAVID uses an EASE score (default = 1) *(https://davidbioinformatics.nih.gov/helps/functional_annotation.html#fisher)*, which puts a more conservative spin on the Fisher's exact test by subtracting from the left hand side of our contingency table.

	User Genes	Genome	
In Pathway	3	37	40
Not in Pathway	297	29663	29960
	300	29700	30000

Exact p-value = 0.007. Since p-value < 0.05, this user's gene list is specifically associated (enriched) in the p53 signaling pathway by more than random chance.

Figure 1: Contingency table showing the number of user input genes and background genes from the genome that fall onto a certain pathway. DAVID help documentations *(https://david.ncifcrf.gov/helps/functional_annotation.html#bonfer)*

Background gene list

DAVID compares the overlap of a user provided gene list to an annotation to the overlap of a background gene list to the same annotation. So what is an appropriate background gene list?

Ideally, a background gene list should represent "the 'universe' of possible genes that could be called as significantly regulated in the experiment" (Timmons et al. 2015) *(https://genomebiology.biomedcentral.com/articles/10.1186/s13059-015-0761-7)*. In RNA-Seq this would not be the whole genome, but rather the genes that were expressed. A background gene list representing the whole genome would be more appropriate for experiments surveying the entire genome (e.g., genetic variation experiments). See Wijesooriya et al. (2022) *(https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1009935#pcbi.1009935.ref017)* for a more detailed discussion.

The default background gene list in DAVID is the whole genome; however, you also have the option to upload your own background gene list.

A word of caution.

> If you do use a custom background set, "make sure that the genes in your list are found in the background set that you have selected in DAVID; otherwise, DAVID will ignore them." -- DAVID FAQ *(https://david.ncifcrf.gov/content.jsp?file=FAQs.html#14)*

Below are some resources for you to learn about or review this statistical procedure.

- Fisher exact test from Wikipedia *(https://en.wikipedia.org/wiki/Fisher%27s_exact_test)*
- Hypergeometric distribution from Wikipedia *(https://en.wikipedia.org/wiki/Hypergeometric_distribution)*
- Fisher exact test from Pathway Commons *(https://www.pathwaycommons.org/guide/primers/statistics/fishers_exact_test/)*
- Hypergeometric distribution from Pathway Commons *(https://www.pathwaycommons.org/guide/primers/statistics/distributions/#hypergeometric)*
- EASE score *(https://www.ncbi.nlm.nih.gov/pmc/articles/PMC328459/)*

Multiple Testing Correction

A problem that arises with enrichment analysis is the need to perform multiple statistical tests across many gene sets. In short, the number of type I errors or false positives increases as the number of tests performed increases -- Pathway Commons multiple testing *(https://www.pathwaycommons.org/guide/primers/statistics/multiple_testing/)*. With DAVID, users can choose to use either Bonferroni, Benjamini-Hochberg, or False Discovery Rate (FDR) *(https://davidbioinformatics.nih.gov/helps/functional_annotation.html#bonfer)* to correct for multiple testing.

Reducing Redundancy

The Functional Annotation Clustering tool can be used to reduce the redundancy evident in the Functional Annotation Chart results. See more below.

Briefly, DAVID uses the Kappa statistic *(https://www.sciencedirect.com/topics/medicine-and-dentistry/kappa-statistics)* to measure the level of similarities in genes between annotations and then applies fuzzy heuristic clustering *(https://david.ncifcrf.gov/helps/functional_classification.html#heuristic)* to cluster groups of similar annotations.

Data Size Limits

DAVID works best with gene lists comprised of <= 3000 genes, and the Functional Annotation Clustering and Gene Functional Classification tools have a 3000 gene limit.

Getting help

DAVID question and forum *(https://david-bioinformatics.freeforums.net/)*

Contact the DAVID Bioinformatics Team via email *(mailto:weizhong.chang@nih.gov)*

DAVID FAQ *(https://david.ncifcrf.gov/content.jsp?file=FAQs.html)*

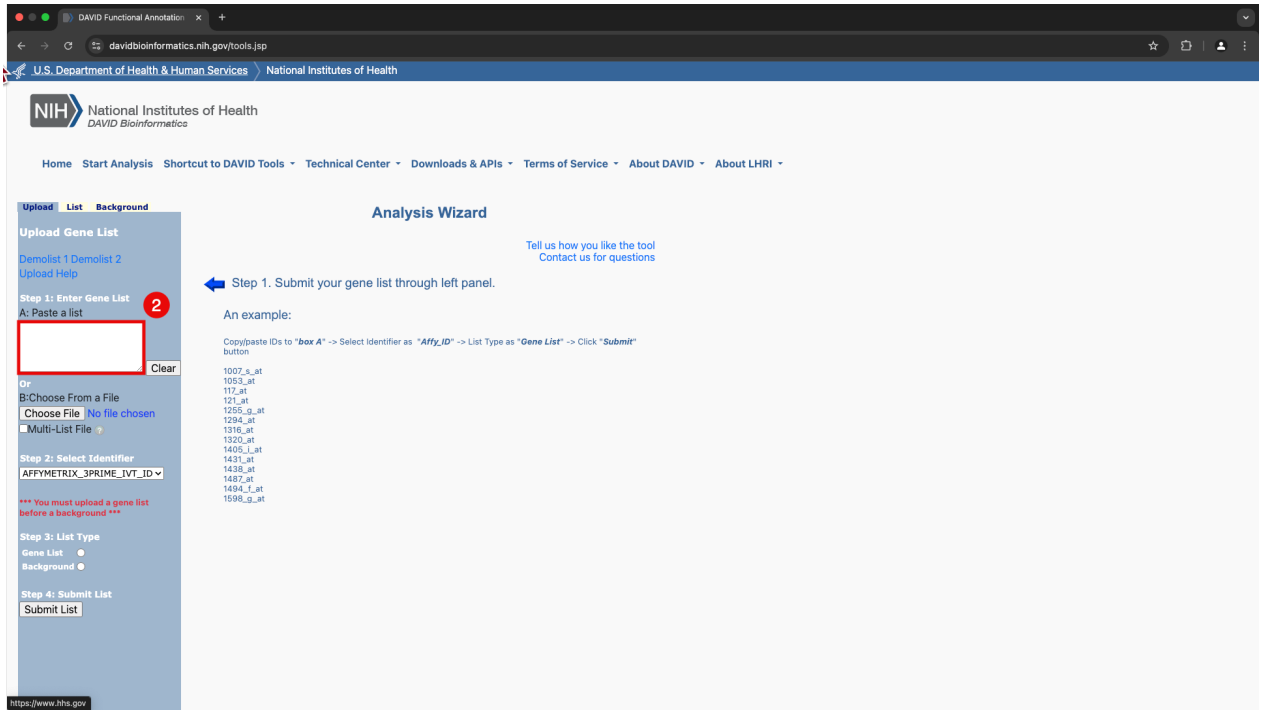
DAVID help documentations *(https://david.ncifcrf.gov/helps/functional_annotation.html#bonfer)*

DAVID quick start tutorial *(https://david.ncifcrf.gov/helps/tutorial.pdf)*

Starting an analysis in DAVID

USE GOOGLE CHROME TO INTERACT WITH DAVID

Click on the Start Analysis button to initiate an analysis, this will take us to the Analysis Wizard.



Supplying input

Tasks to do at the Analysis Wizard:

1. Provide an input gene list (either copy paste or upload as a text file)
2. Specify the gene identifier type. Gene identifiers can be gene symbol, Ensembl IDs, Entrez IDs, Genbank IDs, Refseq IDs, etc.)

> **Note**
>
> DAVID only recognizes “Official Gene Symbol” based on the latest update. Therefore, names changes for "Official Gene Symbols" (e.g., SEPT5 -> SEPTIN5) can impact results.

1. Specify whether the input gene list in the "gene list" or "background gene list".
2. Submit the list for analysis

Here, we will provide the genes that are upregulated in the Hcc1395 Tumor samples. However, we will not be using results obtained in Module 2, as these were derived from filtered fastqs, which only included reads from Chromosome 22. Instead, we will use up-regulated genes from non-filtered fastqs described here *(https://pmbio.org/module-02-inputs/0002/05/01/Data/)*.

Note

These samples were quickly processed using the CCBR pipeliner workflow Renee. The counts file was obtained without checking any of the QC reports to save time.

Up-regulated genes (3,008 total) were obtained by filtering differential expression results obtained from DESeq2 for log2 fold change values greater than or equal to 3 with a false discovery rate of less than or equal to 0.01. This choice was arbitrary, and was primarily driven by DAVID input thresholds. You can download the gene list here.

A background gene list including genes with greater than 0 expression can be found here.

background gene list

For simplicity, I am using the default background gene list in DAVID. However, see the above note regarding appropriate gene lists. You can use the file above (background_expressed.txt) to get an idea concerning how your selection of a background can affect the results.

More on the HCC1395 cell line

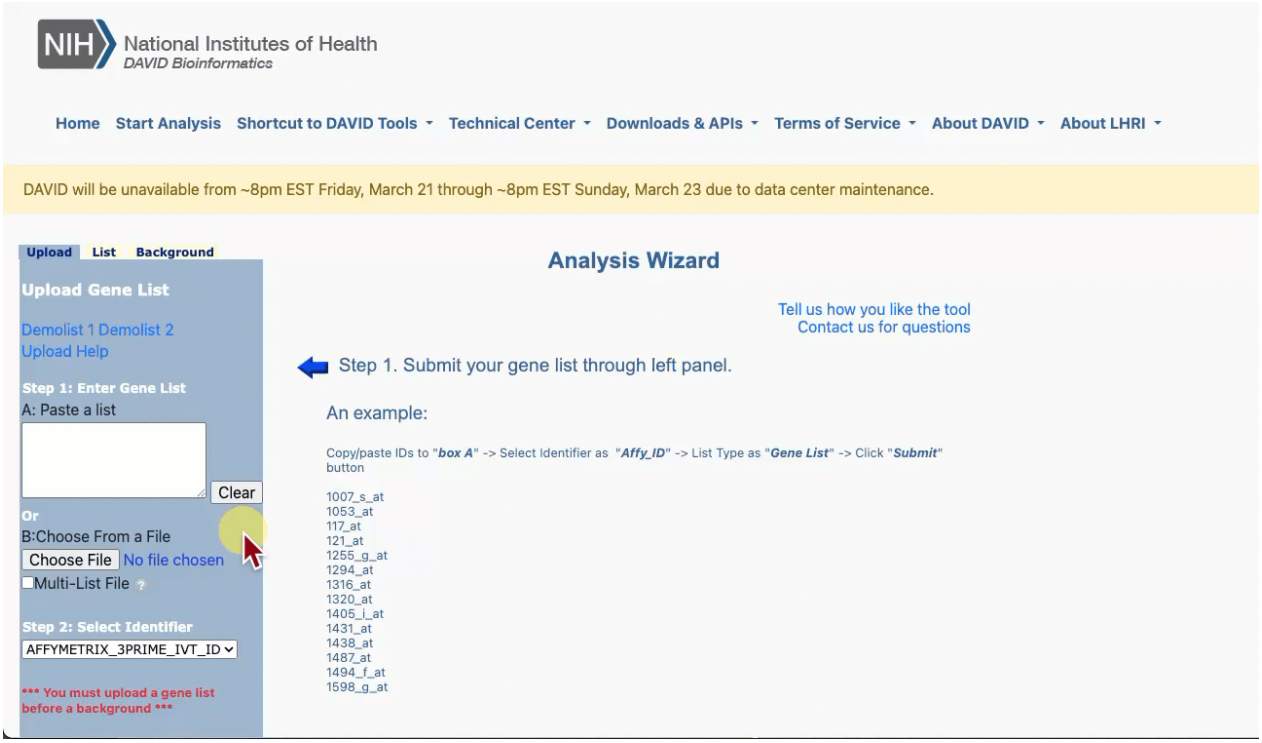
The HCC1395 cell line was obtained from a 43 year old Caucasian female patient. The HCC1395 cell line is described as being of tissue orgin: mammary gland; breast/duct. The HCC1395BL cell line was created from a B lymphoblast that was tranformed by the EBV virus. The patient's cancer was described as: TNM stage I, grade 3, primary ductal carcinoma. This cell line was initiated in the 1990s from a patient with a family history of cancer (patient's mother had breast cancer). The cell line took 14 months to establish. The patient received chemotherapy prior to isolation of the tumor (PMID: 9833771). This tumor is concidered "Triple-Negative" by classic typing: ERBB2 -ve (aka HER2/neu), PR -ve, and ER -ve). Otherwise it is one of those difficult to classify by expression-based molecular typing but is likely of the "Basal" sub-type (PMID: 22003129). The tumor cell line is known to be polyploid. The tumor is also described as TP53 mutation positive. - Griffith Lab *(https://pmbio.org/module-02-inputs/0002/05/01/Data/)*

Step 1: Open the up-regulated genes (up_logfold3.txt) locally; copy the identifiers, and paste the gene list as input in the DAVID Analysis Wizard. Alternatively, attach the file (up_logfold3.txt) directly. Uploaded files must be tab-delimited with one gene id / protein id per row. *(https://davidbioinformatics.nih.gov/content.jsp?file=list_manager.html)* Then select the appropriate identifier (e.g., ENSEMBL_GENE_ID) and specify the list type (Gene List). As indicated by DAVID, a gene list must be submitted prior to any background.

Note

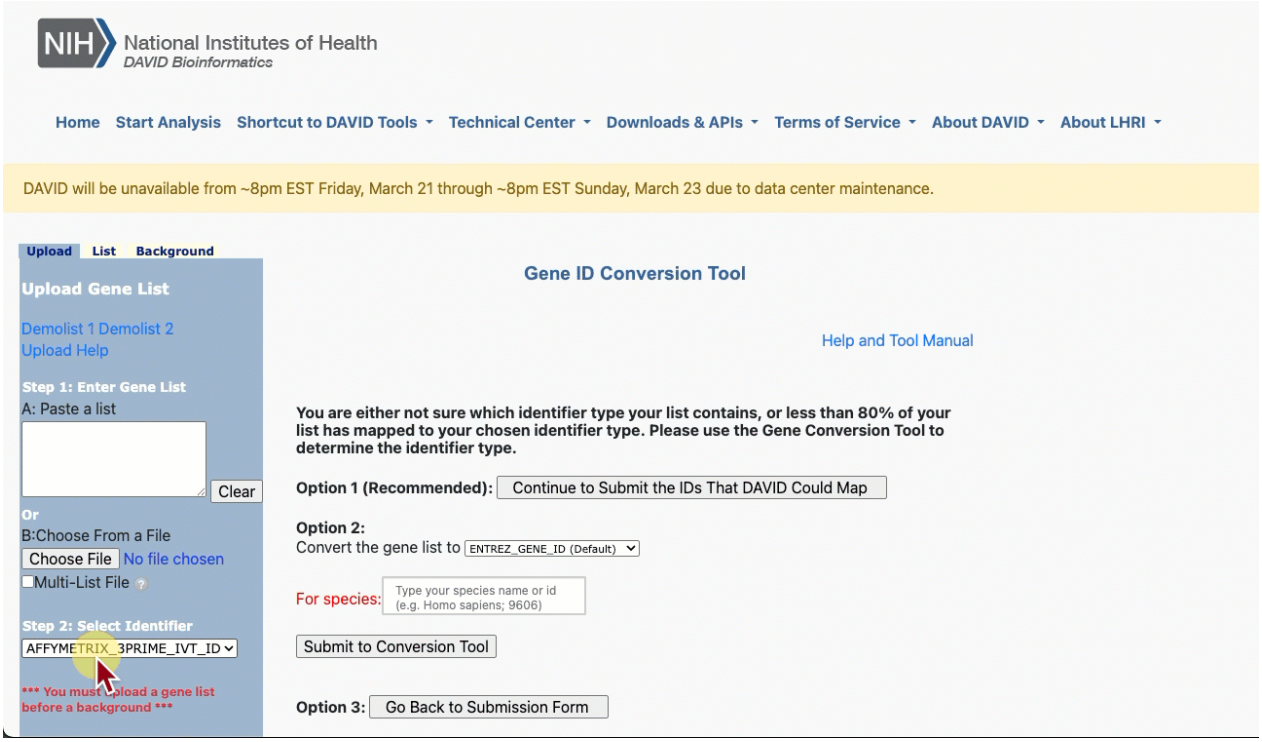
If you are not sure what type of identifier you are working with, you can select "Not Sure" from the list. This will open the Gene ID Conversion Tool. The GENE ID Conversion Tool is used to convert the input gene list to a set of IDs that

are recognized by DAVID in the event that one does not know or DAVID does not recognize the identifier type in the input.



Step 2: After submitting the gene list, DAVID will either proceed directly to the Analysis Wizard, or if more than 20% of input genes were unknown or failed to map to the chosen identifier list, DAVID will proceed to the Gene ID Conversion Tool.

Here, we were directed to the Gene ID Conversion Tool. Because these IDs are Ensembl Gene IDs, I will select "Continue to Submit the IDs That DAVID Could Map".



Ensembl Gene ID

If you are using Ensembl IDs, you will likely need to remove the version ID *(https://useast.ensembl.org/Help/Faq?id=488)* for many tools to work properly. If working with a gene list you can use the command line tool `cut` to do this:

```
cat gene_list.txt | cut -f 1 -d . > newIDs.txt
```

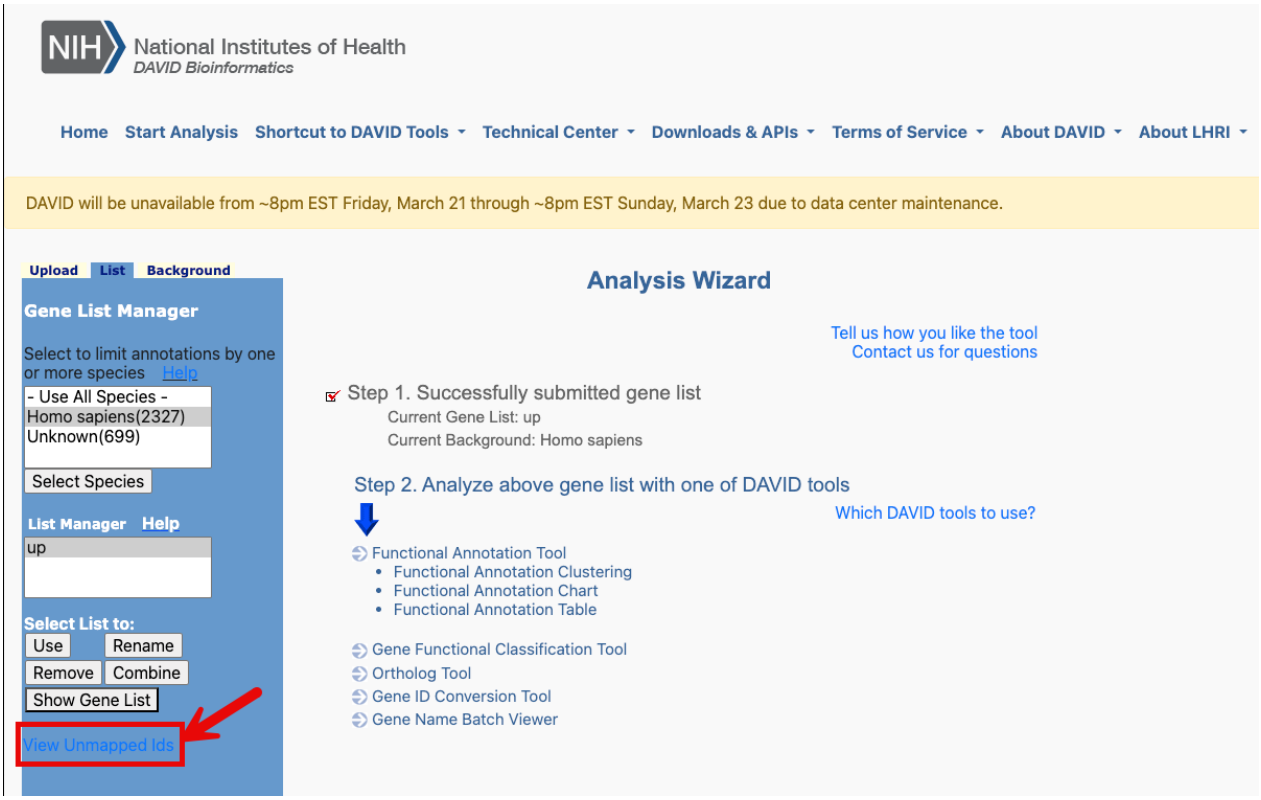
Gene ID Conversion Tool

If we instead decided to convert the identifiers. You would do that as follows:

1. Select the Identifier to convert to.
2. Designate the species associated with these IDs.
3. Select "Submit to Conversion Tool"

Once submitted to the Conversion Tool, can choose to convert all genes or convert each gene individually. Some of the genes may not be in the DAVID database, so the Gene ID conversion tool will not be able to convert those. We would need to do some data wrangling to find identifiers for those genes not in the DAVID database. After conversion, we can send the new list back to DAVID either as input or background. Here, we will submit as input.

2,327 gene IDs were able to map. If we want to check out Unmapped IDs, we can select those using "View Unmapped Ids".



This provides a list of IDs that failed to map. We could explore these IDs further outside of DAVID, for example using BioMart *(https://www.ensembl.org/info/data/biomart/index.html)* or other tool.

Results and interpretation

Annotation Results Summary

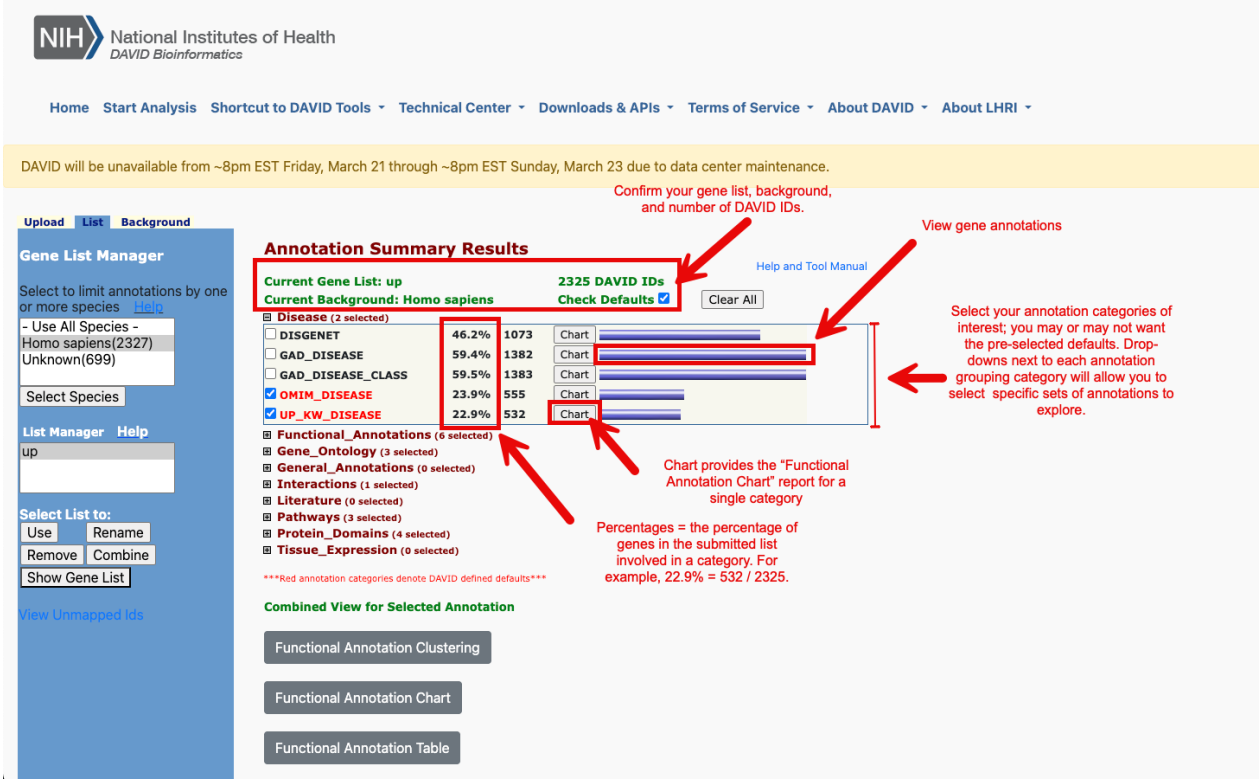
Once we have submitted our gene list for analysis, DAVID takes us to the Annotation Summary Results page. Here, we will find a summary of all of the annotations in DAVID associated with our input gene list.

This page confirms the name of our gene list (up) and, the background gene list that we are using (Homo sapiens), and the number of IDs in our list. We can navigate the Annotation Summary Results page to obtain different insights to our data.

Here we can select and explore multiple classes of annotation categories including GO terms, protein-protein interactions, protein functional domains, disease associations, bio-pathways, sequence general features, functional summaries, tissue expression, literature, etc.

From the Annotation Summary Results, we can see information about the number of genes in our list involved in a given category. The annotations for these genes can be viewed by selecting the horizontal bars.

A functional annotation chart report can be viewed for individual annotation categories by selecting "Chart", or for combined categories (See Functional Annotation Chart for more details on combined reports.)

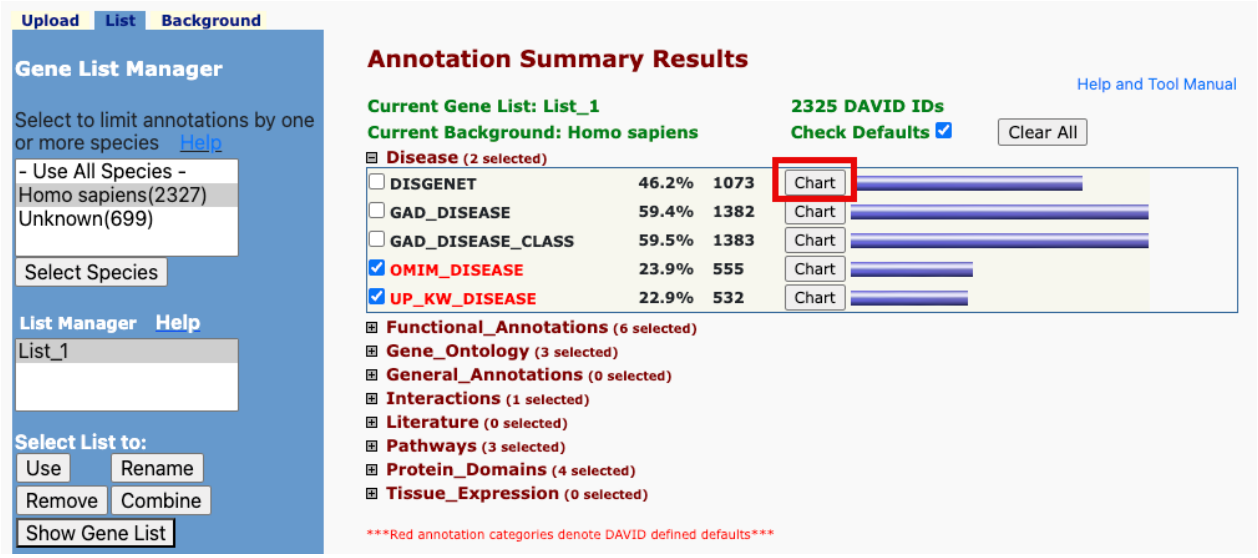


To better understand our results, let's take a closer look at the disease annotations.

Disease Annotations

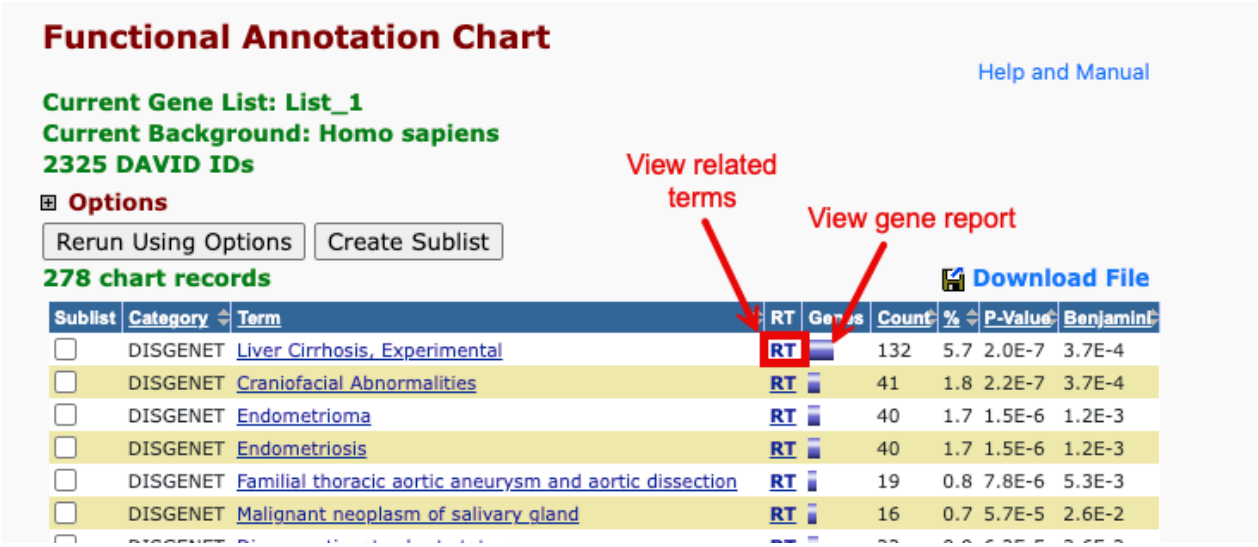
DAVID pulls disease annotations from different sources. Clicking on the Chart button will take us to a chart view showing the disease records from a given database in which our input genes map.

- DISGENET *(https://www.disgenet.org)*
- GAD_DISEASE (GAD = Genetic Association Database)
- GAD_DISEASE_CLASS
- OMIM *(https://www.omim.org)*
- UniProt - UP_KW_DISEASE where KW stands for keyword *(https://www.uniprot.org/keywords/KW-9995)*



Disease Annotation - chart view

In the chart view, we are presented with the disease(s) found in a particular database that our input genes map to. Notice that we can sort this list by gene count, percentage, p-value, or adjusted p-value. The chart view shows us the results of our modified Fisher's exact. More on this later.



Clicking on one of the disease terms in the list will provide a description of the term. The source of this description will vary depending on the category selected. For example, terms under DISGENET send us to NCBI's MedGen.

Note

The chart view for some annotation databases may not have records, or will have few records. This is because there were no annotations that met the statistical threshold (Default = 0.1).

Here, we can also look at related terms by selecting "RT", and we can access a Gene Report by selecting the blue bar.

Related Terms:

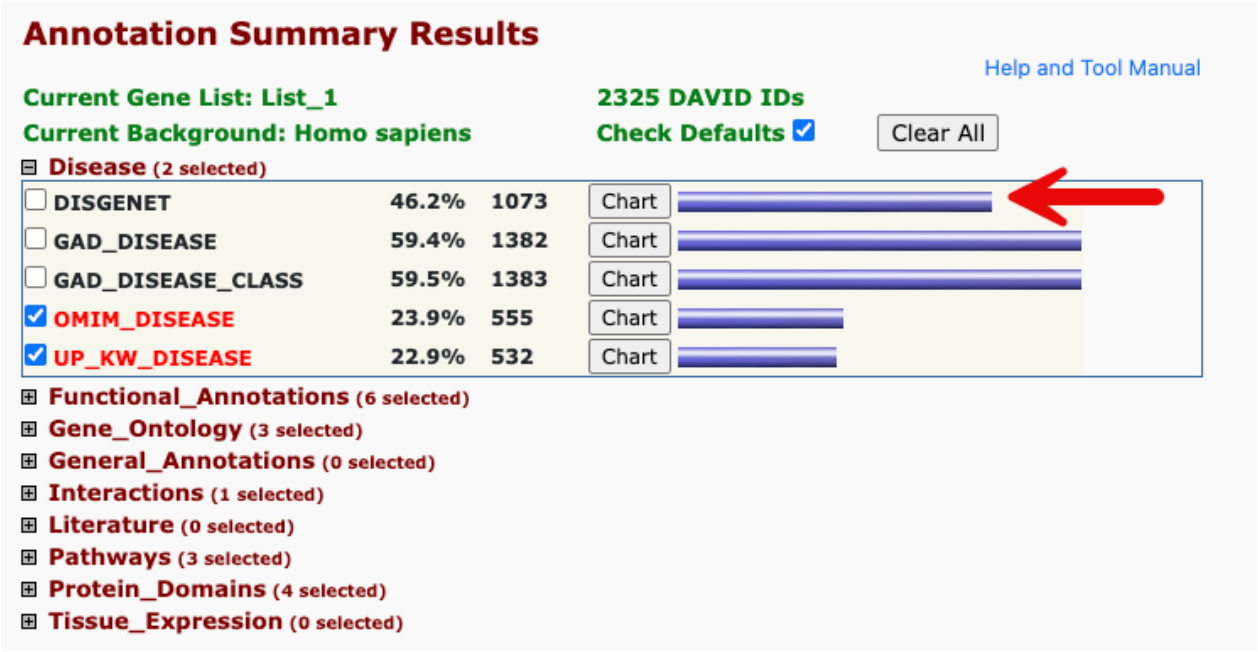
Examining the related terms can help you identify related biological processes or terms to get a better idea of the underlying biology. A kappa statistic is used to measure the degree of agreement between participating genes in terms. The closer to 1, the greater the similarity. Greater than 0.7 indicates a strong agreement. *(https://davidbioinformatics.nih.gov/helps/linear_search.html#kappa)*.

Disease Annotation - link to external resource

In the chart view shown above, we clicked on mammary neoplasm, and this took us to the corresponding record in NCBI's MedGen. MedGen is NCBI's database that contains organized information pertaining to human gene-disease relationships.

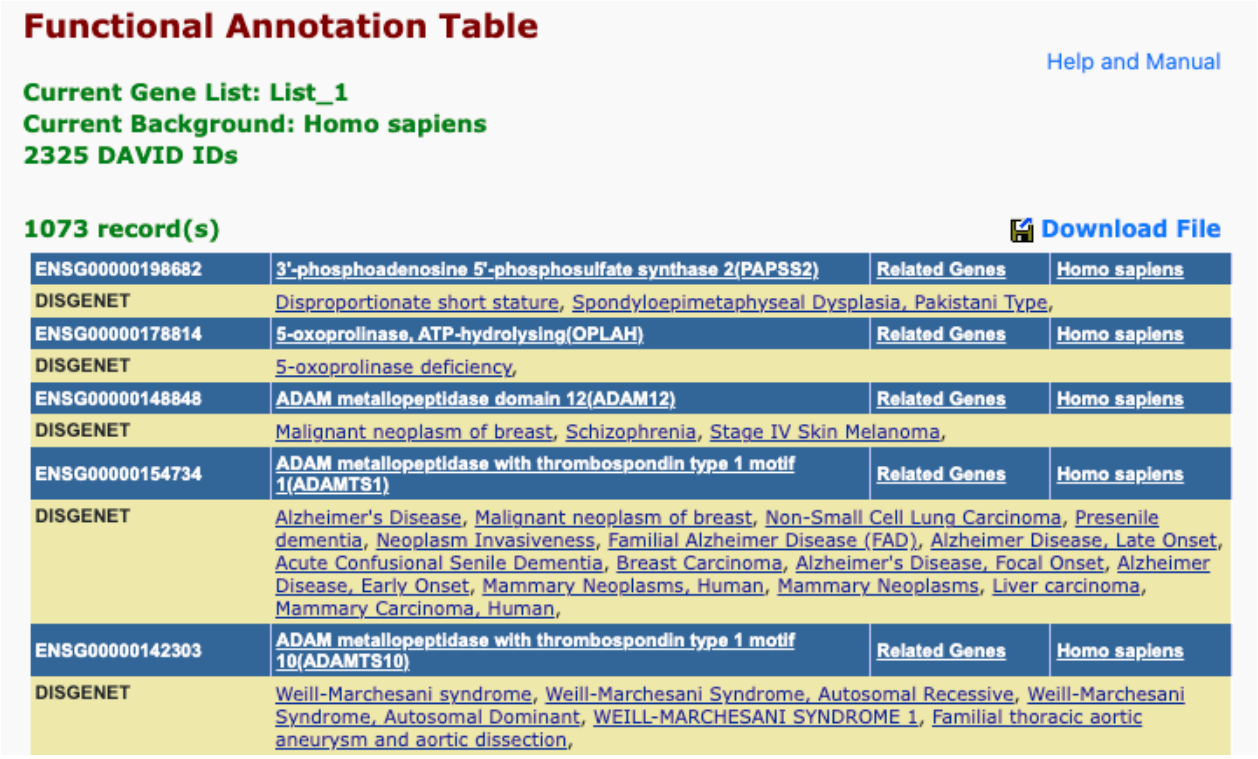
Disease Annotation - gene view

If we click on the blue bar next to the chart button, we will be taken to a gene view of the disease terms.



Disease Annotation - gene view results

The gene view lists the disease terms for each gene in the annotation category of interest. This is the Functional Annotation Table.



Other annotations

We see a similar organization of information for other annotation categories. For instance, DAVID pulls information on biomolecular processes, functions, and pathways from several sources such as UniProt, GO, KEGG, Reactome, and Wikipathways.

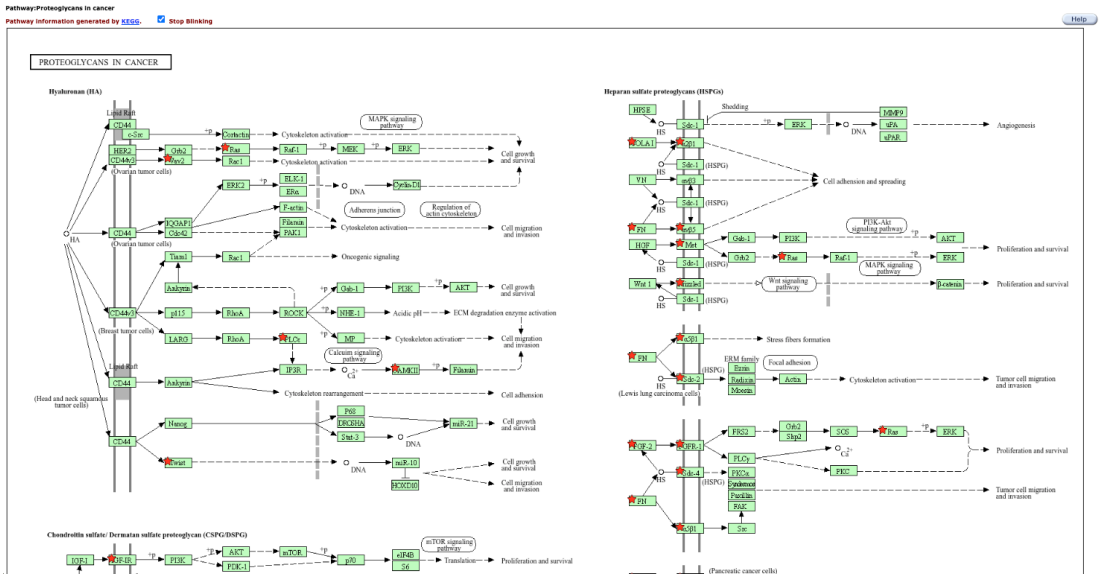
For Gene Ontology (GO), the GO Direct categories are selected by default. These provide GO mappings directly annotated by the source database (no parent terms included). The user can also opt for all levels from GO or specific levels, with level one including broader terms and level 5 including more specific terms. The GO FAT category filters out very broad GO terms based on a measured specificity of each term (not level-specificity).

We can look at results across multiple categories by selecting annotation sources (databases) of interest. By Default, DAVID will select annotation sources in red, but these can also be deselected.

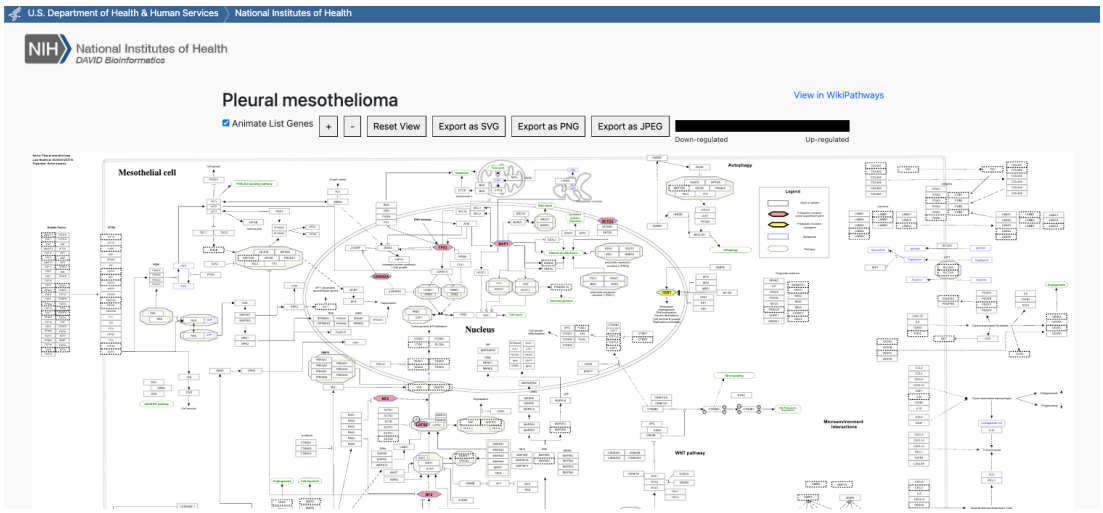
Pathway Maps

DAVID includes several pathway databases, and also includes a Pathway Viewer for annotations from KEGG, BioCarta, and WikiPathways. This viewer displays the user's genes on pathway maps.

> **BioCarta**
>
> Pathway information generated by BioCarta is no longer maintained by BioCarta or CGAP but is retained in DAVID.



DAVID Pathway Viewer: Proteoglygans in Cancer from KEGG

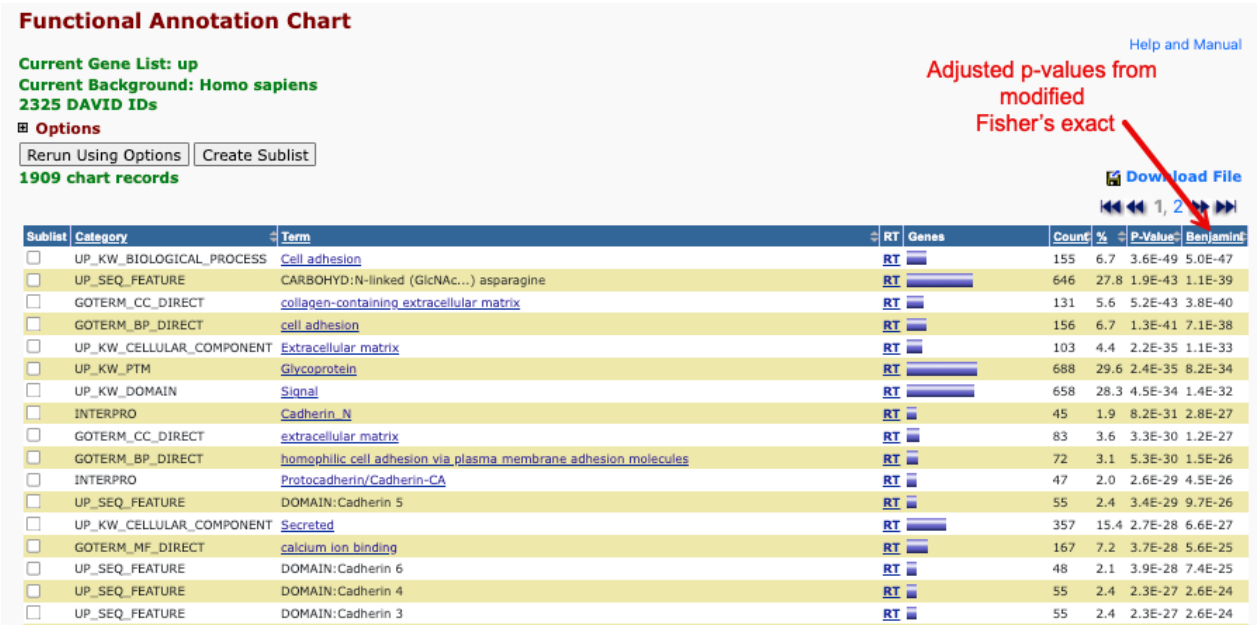


DAVID Pathway Viewer: Pleural mesothelioma from WikiPathways

Functional Annotation Chart

> Chart Report is an annotation term focused view which lists annotation terms and their associated genes under study. -- DAVID help documents *(https://david.ncifcrf.gov/helps/functional_annotation.html#summary)*

As we have seen, you can view the Functional Annotation Chart for specific annotation categories. However, the "Functional Annotation Chart button" provides our enrichment results across multiple categories. Without customizing which categories (or databases) to include, DAVID will automatically use pre-selected defaults. This view does not eliminate redundancies across annotation categories and databases.



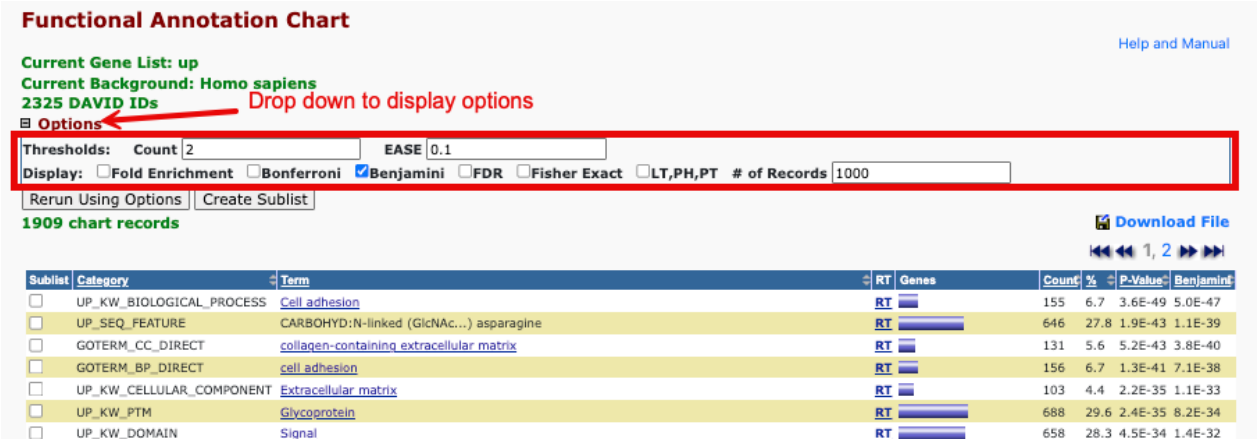
Sublist	Category	Term	RT	Genes	Count	%	P-Value	Benjamini
	UP_KW_BIOLOGICAL_PROCESS	Cell adhesion	RT		155	6.7	3.6E-49	5.0E-47
	UP_SEQ_FEATURE	CARBOHYD:N-linked (GlcNAc...) asparagine	RT		646	27.8	1.9E-43	1.1E-39
	GOTERM_CC_DIRECT	collagen-containing extracellular matrix	RT		131	5.6	5.2E-43	3.8E-40
	GOTERM_BP_DIRECT	cell adhesion	RT		156	6.7	1.3E-41	7.1E-38
	UP_KW_CELLULAR_COMPONENT	Extracellular matrix	RT		103	4.4	2.2E-35	1.1E-33
	UP_KW_PTM	Glycoprotein	RT		688	29.6	2.4E-35	8.2E-34
	UP_KW_DOMAIN	Signal	RT		658	28.3	4.5E-34	1.4E-32
	INTERPRO	Cadherin_N	RT		45	1.9	8.2E-31	2.8E-27
	GOTERM_CC_DIRECT	extracellular matrix	RT		83	3.6	3.3E-30	1.2E-27
	GOTERM_BP_DIRECT	homophilic cell adhesion via plasma membrane adhesion molecules	RT		72	3.1	5.3E-30	1.5E-26
	INTERPRO	Protocadherin/Cadherin-CA	RT		47	2.0	2.6E-29	4.5E-26
	UP_SEQ_FEATURE	DOMAIN:Cadherin 5	RT		55	2.4	3.4E-29	9.7E-26
	UP_KW_CELLULAR_COMPONENT	Secreted	RT		357	15.4	2.7E-28	6.6E-27
	GOTERM_MF_DIRECT	calcium ion binding	RT		167	7.2	3.7E-28	5.6E-25
	UP_SEQ_FEATURE	DOMAIN:Cadherin 6	RT		48	2.1	3.9E-28	7.4E-25
	UP_SEQ_FEATURE	DOMAIN:Cadherin 4	RT		55	2.4	2.3E-27	2.6E-24
	UP_SEQ_FEATURE	DOMAIN:Cadherin 3	RT		55	2.4	2.3E-27	2.6E-24

Chart results have to meet certain criteria to be included:

- EASE Score Threshold (Maximum Probability, p-value) <= 0.1

- Count Threshold (Minimum gene count for an annotation term) >= 2

These thresholds are customizable using "Options".



You can also add additional columns to the output under "Display". For example, you can look at other methods for adjusting p-values, fold enrichment, Fisher's Exact p-value, etc.

What is "fold enrichment"?

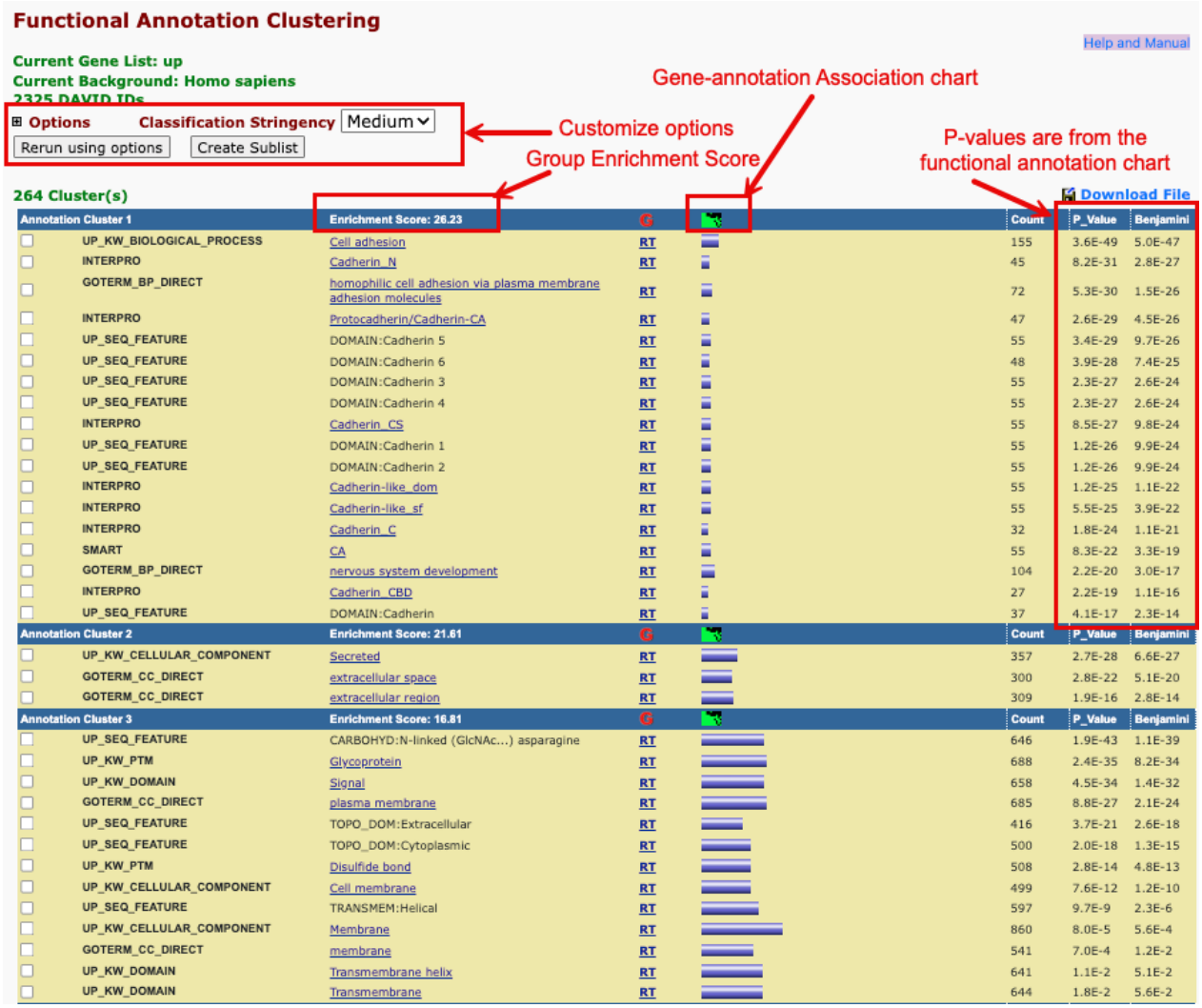
In this context, "fold enrichment" refers to the enrichment of the term in your gene list as compared to the background population of genes. For example, if 40/400 (i.e. 10%) of the genes from your list are involved in "kinase activity" and the background population ratio of genes involved in "kinase activity" is 300/30000 genes (i.e. 1%). There is a 10-fold(10%/1%) enrichment of genes from your list involved in "kinase activity" compared to the population. *(https://david-bioinformatics.freeforums.net/thread/64/fold-change-result-window)*

Functional Annotation Clustering

The Functional Annotation Clustering tool groups similar annotations together to reduce the redundancy seen in the Functional Annotation Chart results. This eases the interpretation of the findings.

> The Functional Annotation Clustering integrates the same techniques of Kappa statistics to measure the degree of the common genes between two annotations, and fuzzy heuristic clustering (used in Gene Functional Classification Tool *(https://davidbioinformatics.nih.gov/helps/functional_classification.html)*) to classify the groups of similar annotations according to Kappa values. In this sense, the more common genes annotations share, the higher chance they will be grouped together.
> - DAVID Documentation - Functional Annotation Clustering *(https://davidbioinformatics.nih.gov/helps/functional_annotation.html#funcannoclus)*

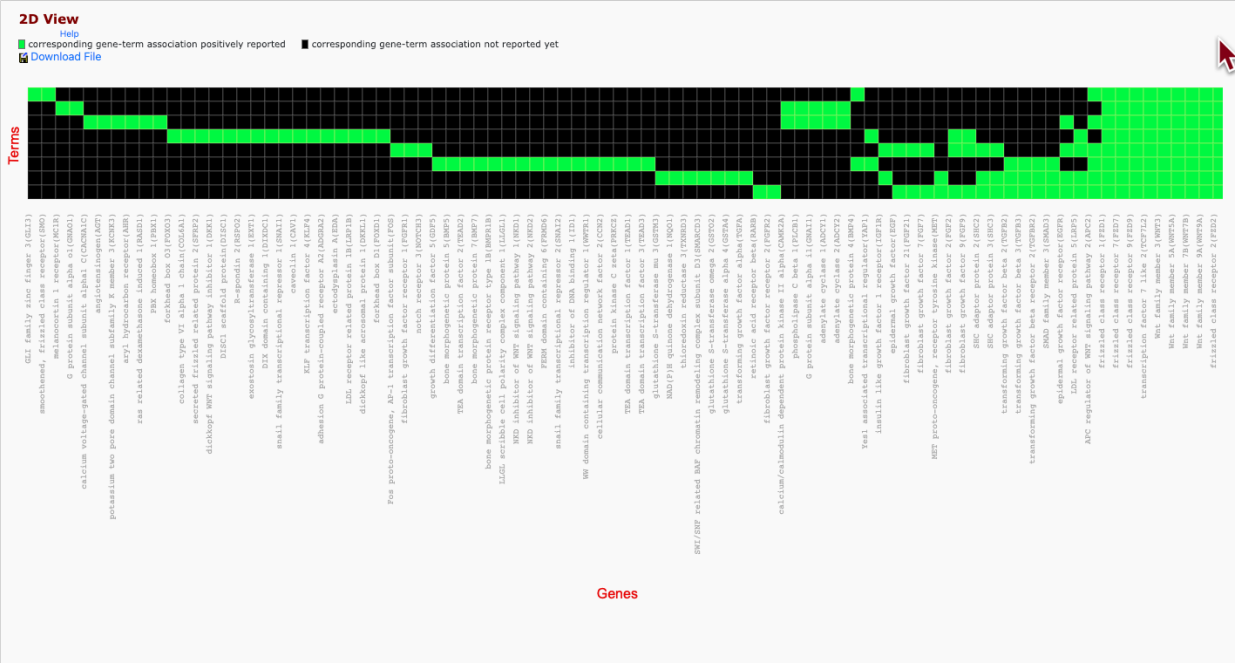
For a more in-depth understanding of the methods used, see Huang et al. 2007 *(https://genomebiology.biomedcentral.com/articles/10.1186/gb-2007-8-9-r183)*.



What is the Group Enrichment Score?

This is the geometric mean (in -log scale) of member's p-values in a corresponding annotation cluster. More enriched terms will be toward the top with higher group enrichment scores. This is used to rank biological significance.

Gene-Annotation Association View:
The Gene-Annotation Association Viewer can be used to examine the relationships between genes and annotation terms. Green indicates a term is associated with a given gene. We can use this to look at genes shared across terms.



Modifying Parameters:

We can fine tune how DAVID clusters the annotations using the parameters below.

Adjusting parameters for clustering

- "Clustering Stringency (lowest → highest): A high-level single control to establish a set of detailed parameters involved in functional classification algorithms. In general, the higher stringency setting generates less functional groups with more tightly associated genes in each group, so that more genes will be unclustered. The default setting is Medium, which gives balanced results for most cases based on our studies. Customization allows you to control Advanced options." -- DAVID functional classification documentations *(https://david.ncifcrf.gov/helps/functional_classification.html)*
- "Similarity Term Overlap (any value ≥ 0; default = 4): The minimum number of annotation terms overlapped between two genes in order to be qualified for kappa calculation. This parameter is to maintain necessary statistical power to make the kappa value more meaningful. The higher the value, the more meaningful the result is." -- DAVID functional classification documentations *(https://david.ncifcrf.gov/helps/functional_classification.html)*
- "Similarity Threshold (any value between 0 to 1; default = 0.35): The minimum kappa value to be considered significant. A higher setting will lead to more genes going unclustered, which leads to a higher quality functional classification result with fewer groups and fewer gene members. Kappa value of 0.3 starts giving meaningful biology based on our genome-wide distribution study. Anything below 0.3 has a good chance to be noise." -- DAVID functional classification documentations *(https://david.ncifcrf.gov/helps/functional_classification.html)*
- "Initial Group Members (any value ≥ 2; default = 4): The minimum gene number in a seeding group, which affects the minimum size of each functional group in the final cluster. In general, the lower value attempts to include more genes in functional groups, and may generate a lot of small size groups." -- DAVID functional classification documentations *(https://david.ncifcrf.gov/helps/functional_classification.html)*
- "Final Group Members (any value ≥ 2; default = 4): The minimum gene number in one final group after a 'cleanup' procedure. In general, the lower value attempts to include more genes in functional groups and may generate a lot of small size groups. It cofunctions with previous parameters to control the minimum size of functional groups. If you are interested in functional groups containing only 2 or 3 genes, you need to set

it to a very low value. Otherwise, the small group will not be displayed and the genes will go unclustered." -- DAVID functional classification documentations *(https://david.ncifcrf.gov/helps/functional_classification.html)*

- "Multi-linkage Threshold (any value between 0% to 100%; default = 50%): This parameter controls how seeding groups merge with each other, i.e. two groups sharing the same gene members over the percentage will become one group. A higher percentage, in general, gives sharper separation (i.e. it generates more final functional groups with more tightly associated genes in each group). In addition, changing the parameter does not cause additional genes to go unclustered." -- DAVID functional classification documentations *(https://david.ncifcrf.gov/helps/functional_classification.html)*

Tip

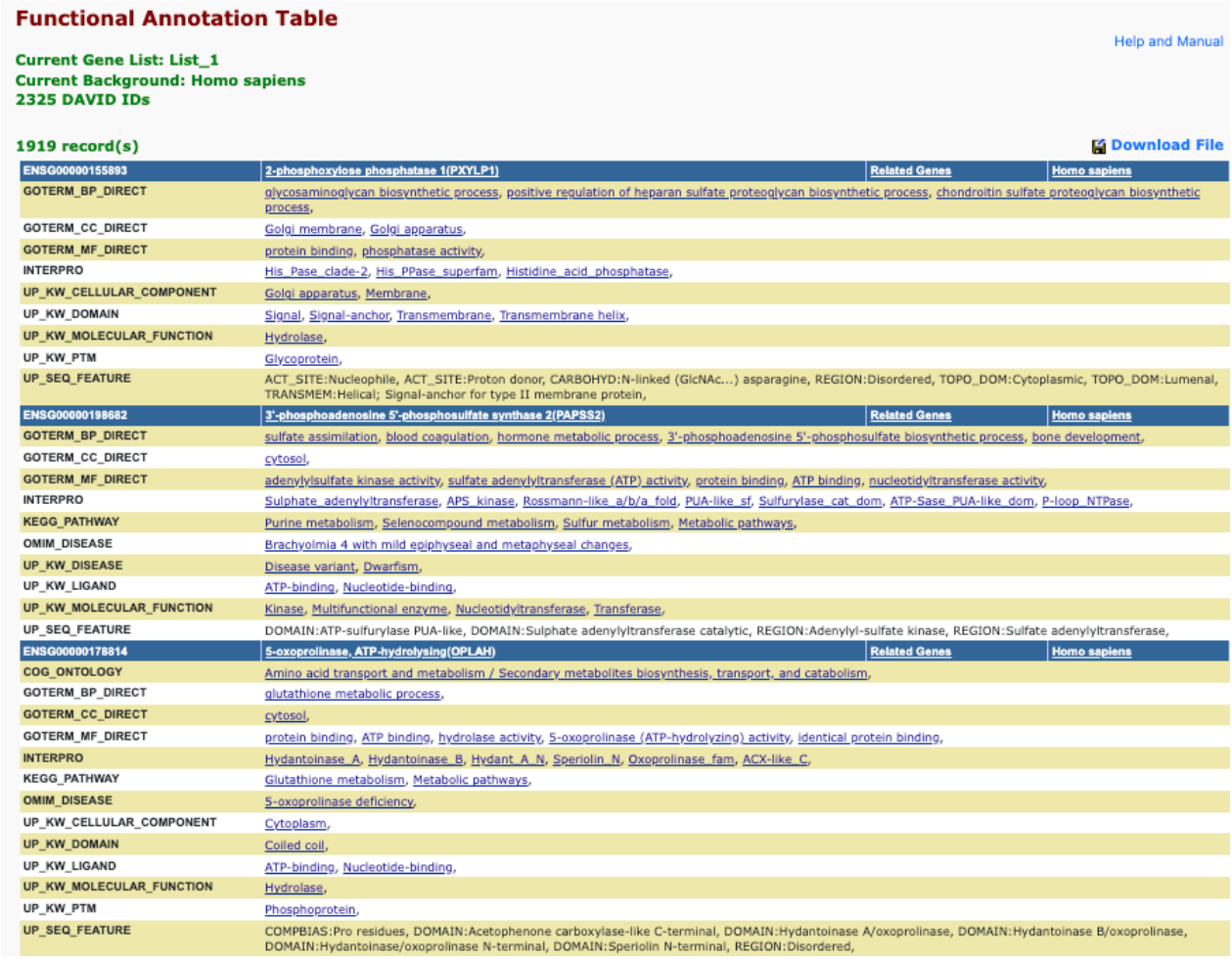
If you do not understand the parameters, stick to the defaults.

Note

The parameters for the Functional Annotation Clustering tool and Functional Classification Tool are the same. These tools apply the same methods and concepts for clustering.

Functional Annotation Table

> Provides a gene-centric view which lists the genes and their associated annotation terms... -- DAVID help documents *(https://david.ncifcrf.gov/helps/functional_annotation.html#summary)*



Functional Annotation Table

Help and Manual

Current Gene List: List_1
Current Background: Homo sapiens
2325 DAVID IDs

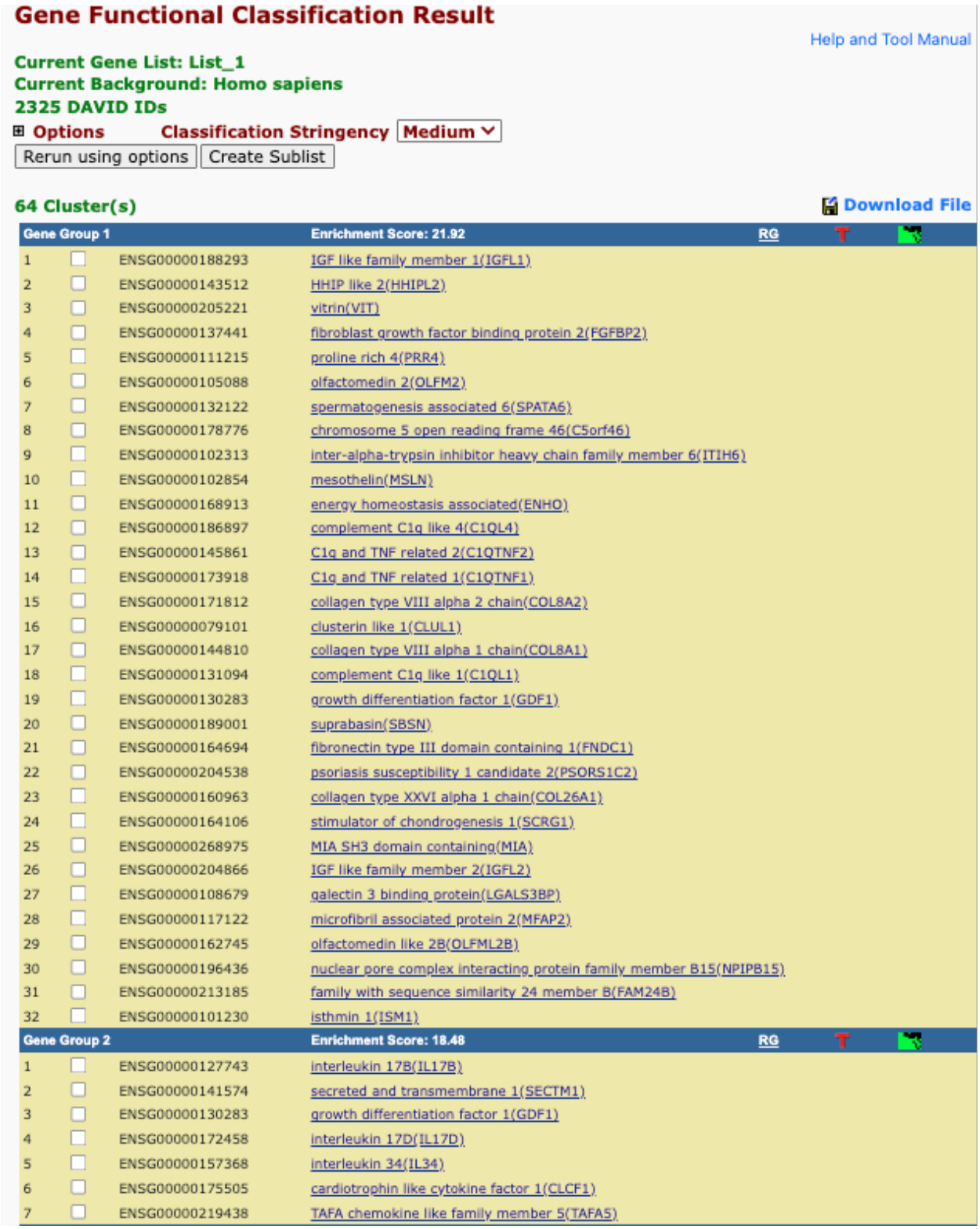
1919 record(s) Download File

ENSG00000155893	2-phosphoxylose phosphatase 1(PXYLP1)	Related Genes	Homo sapiens
GOTERM_BP_DIRECT	glycosaminoglycan biosynthetic process, positive regulation of heparan sulfate proteoglycan biosynthetic process, chondroitin sulfate proteoglycan biosynthetic process,		
GOTERM_CC_DIRECT	Golgi membrane, Golgi apparatus,		
GOTERM_MF_DIRECT	protein binding, phosphatase activity,		
INTERPRO	His_Pase_clade-2, His_PPase_superfam, Histidine_acid_phosphatase,		
UP_KW_CELLULAR_COMPONENT	Golgi apparatus, Membrane,		
UP_KW_DOMAIN	Signal, Signal-anchor, Transmembrane, Transmembrane helix,		
UP_KW_MOLECULAR_FUNCTION	Hydrolase,		
UP_KW_PTM	Glycoprotein,		
UP_SEQ_FEATURE	ACT_SITE:Nucleophile, ACT_SITE:Proton donor, CARBOHYD:N-linked (GlcNAc...) asparagine, REGION:Disordered, TOPO_DOM:Cytoplasmic, TOPO_DOM:Lumenal, TRANSMEM:Helical; Signal-anchor for type II membrane protein,		
ENSG00000198682	3'-phosphoadenosine 5'-phosphosulfate synthase 2(PAPSS2)	Related Genes	Homo sapiens
GOTERM_BP_DIRECT	sulfate assimilation, blood coagulation, hormone metabolic process, 3'-phosphoadenosine 5'-phosphosulfate biosynthetic process, bone development,		
GOTERM_CC_DIRECT	cytosol,		
GOTERM_MF_DIRECT	adenylylsulfate kinase activity, sulfate adenylyltransferase (ATP) activity, protein binding, ATP binding, nucleotidyltransferase activity,		
INTERPRO	Sulphate_adenylyltransferase, APS_kinase, Rossmann-like_a/b/a_fold, PUA-like_sf, Sulfurylase_cat_dom, ATP-Sase_PUA-like_dom, P-loop_NTPase,		
KEGG_PATHWAY	Purine metabolism, Selenocompound metabolism, Sulfur metabolism, Metabolic pathways,		
OMIM_DISEASE	Brachyolmia 4 with mild epiphyseal and metaphyseal changes,		
UP_KW_DISEASE	Disease variant, Dwarfism,		
UP_KW_LIGAND	ATP-binding, Nucleotide-binding,		
UP_KW_MOLECULAR_FUNCTION	Kinase, Multifunctional enzyme, Nucleotidyltransferase, Transferase,		
UP_SEQ_FEATURE	DOMAIN:ATP-sulfurylase PUA-like, DOMAIN:Sulphate adenylyltransferase catalytic, REGION:Adenylyl-sulfate kinase, REGION:Sulfate adenylyltransferase,		
ENSG00000178814	5-oxoprolinase, ATP-hydrolysing(OPLAH)	Related Genes	Homo sapiens
COG_ONTOLOGY	Amino acid transport and metabolism / Secondary metabolites biosynthesis, transport, and catabolism,		
GOTERM_BP_DIRECT	glutathione metabolic process,		
GOTERM_CC_DIRECT	cytosol,		
GOTERM_MF_DIRECT	protein binding, ATP binding, hydrolase activity, 5-oxoprolinase (ATP-hydrolyzing) activity, identical protein binding,		
INTERPRO	Hydantoinase_A, Hydantoinase_B, Hydant_A_N, Speriolin_N, Oxoprolinase_fam, ACX-like_C,		
KEGG_PATHWAY	Glutathione metabolism, Metabolic pathways,		
OMIM_DISEASE	5-oxoprolinase deficiency,		
UP_KW_CELLULAR_COMPONENT	Cytoplasm,		
UP_KW_DOMAIN	Coiled coil,		
UP_KW_LIGAND	ATP-binding, Nucleotide-binding,		
UP_KW_MOLECULAR_FUNCTION	Hydrolase,		
UP_KW_PTM	Phosphoprotein,		
UP_SEQ_FEATURE	COMPBIAS:Pro residues, DOMAIN:Acetophenone carboxylase-like C-terminal, DOMAIN:Hydantoinase A/oxoprolinase, DOMAIN:Hydantoinase B/oxoprolinase, DOMAIN:Hydantoinase/oxoprolinase N-terminal, DOMAIN:Speriolin N-terminal, REGION:Disordered,		

Gene Functional Classification Result

> The Functional Classification Tool provides a rapid means to organize large lists of genes into functionally related groups to help unravel the biological content captured by high throughput technologies. - DAVID documentation - Functional Classification *(https://davidbioinformatics.nih.gov/helps/functional_classification.html)*

Genes are grouped on the basis of shared annotation terms. Similar to the Functional Annotation Clustering tool, this tool uses the kappa statistics and fuzzy heuristic clustering algorithm. "The fuzziness feature of the agglomeration method allows a gene or term to participate in more than one functional group, better reflecting the true 'multiple-roles' nature of genes" (Huang et al. 2007) *(https://genomebiology.biomedcentral.com/articles/10.1186/gb-2007-8-9-r183)*



Gene Functional Classification Result

Help and Tool Manual

Current Gene List: List_1
Current Background: Homo sapiens
2325 DAVID IDs

⊞ Options Classification Stringency: Medium

Rerun using options | Create Sublist

64 Cluster(s) Download File

Gene Group 1			Enrichment Score: 21.92	RG	T	
1	☐	ENSG00000188293	IGF like family member 1(IGFL1)			
2	☐	ENSG00000143512	HHIP like 2(HHIPL2)			
3	☐	ENSG00000205221	vitrin(VIT)			
4	☐	ENSG00000137441	fibroblast growth factor binding protein 2(FGFBP2)			
5	☐	ENSG00000111215	proline rich 4(PRR4)			
6	☐	ENSG00000105088	olfactomedin 2(OLFM2)			
7	☐	ENSG00000132122	spermatogenesis associated 6(SPATA6)			
8	☐	ENSG00000178776	chromosome 5 open reading frame 46(C5orf46)			
9	☐	ENSG00000102313	inter-alpha-trypsin inhibitor heavy chain family member 6(ITIH6)			
10	☐	ENSG00000102854	mesothelin(MSLN)			
11	☐	ENSG00000168913	energy homeostasis associated(ENHO)			
12	☐	ENSG00000186897	complement C1q like 4(C1QL4)			
13	☐	ENSG00000145861	C1q and TNF related 2(C1QTNF2)			
14	☐	ENSG00000173918	C1q and TNF related 1(C1QTNF1)			
15	☐	ENSG00000171812	collagen type VIII alpha 2 chain(COL8A2)			
16	☐	ENSG00000079101	clusterin like 1(CLUL1)			
17	☐	ENSG00000144810	collagen type VIII alpha 1 chain(COL8A1)			
18	☐	ENSG00000131094	complement C1q like 1(C1QL1)			
19	☐	ENSG00000130283	growth differentiation factor 1(GDF1)			
20	☐	ENSG00000189001	suprabasin(SBSN)			
21	☐	ENSG00000164694	fibronectin type III domain containing 1(FNDC1)			
22	☐	ENSG00000204538	psoriasis susceptibility 1 candidate 2(PSORS1C2)			
23	☐	ENSG00000160963	collagen type XXVI alpha 1 chain(COL26A1)			
24	☐	ENSG00000164106	stimulator of chondrogenesis 1(SCRG1)			
25	☐	ENSG00000268975	MIA SH3 domain containing(MIA)			
26	☐	ENSG00000204866	IGF like family member 2(IGFL2)			
27	☐	ENSG00000108679	galectin 3 binding protein(LGALS3BP)			
28	☐	ENSG00000117122	microfibril associated protein 2(MFAP2)			
29	☐	ENSG00000162745	olfactomedin like 2B(OLFML2B)			
30	☐	ENSG00000196436	nuclear pore complex interacting protein family member B15(NPIPB15)			
31	☐	ENSG00000213185	family with sequence similarity 24 member B(FAM24B)			
32	☐	ENSG00000101230	isthmin 1(ISM1)			
Gene Group 2			**Enrichment Score: 18.48**	RG	T	
1	☐	ENSG00000127743	interleukin 17B(IL17B)			
2	☐	ENSG00000141574	secreted and transmembrane 1(SECTM1)			
3	☐	ENSG00000130283	growth differentiation factor 1(GDF1)			
4	☐	ENSG00000172458	interleukin 17D(IL17D)			
5	☐	ENSG00000157368	interleukin 34(IL34)			
6	☐	ENSG00000175505	cardiotrophin like cytokine factor 1(CLCF1)			
7	☐	ENSG00000219438	TAFA chemokine like family member 5(TAFA5)			

Tools are available to investigate consensus terms, enriched terms, and gene-to-term relationships.

Gene Functional Classification vs. Functional Annotation Clustering:

These are related methods. Gene Functional Classification identifies groups of genes sharing similar biological terms, while Functional Annoation Clustering identifies groups of terms sharing similar genes.

DAVID Orthology

The DAVID Orthology tool is a new tool that allows a user to convert a gene list between species.

> DAVID Ortholog can convert a gene list from the species under study to a list of orthologs in a targeted species using OMA and Ensembl ortholog pair information. The ortholog lists converted from lesser to better studied species will provide more annotation information thereby helping the user to further understand the gene list under study. - Sherman et al. 2024 *(https://pmc.ncbi.nlm.nih.gov/articles/PMC11520416/)*

The converted gene list can then be used directly with DAVID functional enrichment tools.

Pathway Analysis with Reactome

Presenter - Nancy Li, Ph.D. (Outreach Coordinator, Reactome)

Reactome is a free, open-source, curated and peer-reviewed pathway database that includes bioinformatics tools for the visualization, interpretation and analysis of pathway knowledge. This lesson showcased the capabilities of Reactome in garnering biological meaning from an example gene list derived from differential expression results.

Help Sessions

Lesson 2 Practice

The instructions that follow were designed to test the skills you learned in Lesson 2. Thus, the primary focus will be navigating directories and manipulating files.

1. Let's navigate our files using the command line. Begin in your data directory.

2. Copy the Practice_L2 directory (`/data/classes/BTEP/B4B_2025/Module_1/Practice_Sessions/Practice_L2`) and all of its contents to your class working directory.

   Solution

   ```
   cd /data/$USER/Module_1
   cp -R /data/classes/BTEP/B4B_2025/Module_1/Practice_Sessions/Practice_L2 .
   ```

3. Change directories to `Practice_L2/`.

   Solution

   ```
   cd Practice_L2
   ```

4. List the contents of `Practice_L2`. If there are files, when were they last modified and what is the file size?

   Solution

   ```
   ls -lh
   ```

   The `-lh` flag will allow you to obtain a list of directory content in long format, which provides information including the date the file was last modified and the file size.

5. Rename `sample_names.txt` to `treatment_groups.txt`.

   Solution

   ```
   mv sample_names.txt treatment_groups.txt
   ```

6. Within `Practice_L2`, create a new directory called `Analysis`.

Solution

```
mkdir Analysis
```

7. Copy `treatment_groups.txt` to the newly created `Analysis` directory.

Solution

```
cp treatment_groups.txt ./Analysis
```

8. List the contents of the `Analysis` directory.

Solution

```
ls -l ./Analysis
```

9. Change directories to `Analysis`. What is the path to `Analysis`?

Solution

```
cd Analysis
pwd
```

10. Change to the `data` directory within the lesson practice directory (`Practice_L2`).

Solution

```
cd ../data
```

11. List the contents of `data`.

Solution

```
ls
```

12. View `A_R1.fastq`.

Solution

```
less A_R1.fastq
q
```

13. Copy and paste the first line of `A_R1.fastq` into a new file called `Line1.txt` using keyboard shortcuts and `nano`.

14. Move `Line1.txt` to `Practice_L2` using a relative file path.

Solution

```
mv Line1.txt ..
```

15. Change to the `Practice_L2` directory and list the content of the directory.

Solution

```
cd ..
ls
```

16. Remove the `Analysis` directory.

Solution

```
rm -ir Analysis
```

Lesson 3 Practice

For today's practice, we are going to embark on a Unix treasure hunt created by the **Sanders Lab *(https://sanderslab.github.io/code/)*** at the University of California San Francisco. Note: the treasure hunt materials can be obtained directly from the Sanders lab code repository linked above.

UNIX treasure hunt tutorial

This perl script will install a series of directories and clues that teaches basic UNIX command line skills including `cd`, `ls`, `grep`, `less`, `head`, `tail`, and `nano`. Run the perl script from the command line on a UNIX based machine (e.g. Mac or Linux) using the command: `perl treasureHunt_v2.pl`. Then use `ls` to find the first clue. A PDF of command line commands is also available to download.

- Source
- Manual

To begin create a directory called `treasure_hunt` in your class working directory and run the perl script from `/data/classes/BTEP/B4B_2025/Module_1/Practice_Sessions/Practice_L4` in the `treasure_hunt` directory.

Solution

```
mkdir treasure_hunt
cd treasure_hunt
perl /data/classes/BTEP/B4B_2025/Module_1/Practice_Sessions/Practice_L4/treasureHur
ls -l
```

Read the first clue and begin.

Recommendation: Create an environment variable to store the path to the treasure hunt directory to facilitate movement through the directory.

Solution

```
THUNT=`pwd`
echo $THUNT
```

When you have found the treasure, answer or do the following:

1. How many words are in the last line of the file containing the teasure?

Solution

```
tail -n 1 openTheBox.txt | wc -w
```

2. Save the last line of `openTheBox.txt` to a new file called `finallyfinished.txt` without copying and pasting.

Solution

```
tail -n 1 openTheBox.txt > finallyfinished.txt
```

3. Now append the first line of `openTheBox.txt` to the same file that you just saved the last line.

Solution

```
head -n 1 openTheBox.txt >> finallyfinished.txt
```

Congratulations! You have found the treasure and have gained some useful unix practice throughout your hunt.

Lesson 4 Practice

The following can be used to practice skills learned in Lesson 4.

Login to Biowulf

If you are already logged in, exit the remote connection and reconnect. Remember, you must be on the NIH network.

Solution

```
ssh username@biowulf.nih.gov
```

Navigate to your working directory.

```
cd /data/$USER/Module_1/testscript
```

Let's run `fastqc` *(https://hpc.nih.gov/apps/fastqc.html)*, a quality control program, on the files we downloaded from the SRA.

Using sbatch

Start a new script, named `fastqc.sh` in the same directory in which you downloaded data from Lesson (`/data/$USER/testscript`).

The command you will include in the script is as follows:

```
mkdir fastqc
fastqc -o ./fastqc/ -t 4 *.fastq
```

This command will output the `fastqc` results to a directory named `fastqc` inside the current working directory. It will also run using four threads and will run on all fastq files in your working directory.

You need edit this script in order to submit as a job on Biowulf. What is missing?

Solution

```
#!/bin/bash
#SBATCH --cpus-per-task=4

module load fastqc
mkdir fastqc
fastqc -o ./fastqc/ -t 4 *.fastq
```

Submit the job.

Solution

```
sbatch fastqc.sh
```

How can we check on our job? What is the job's status? How much memory is it using?

Solution

```
squeue -u $USER
sjobs -u $USER
```

How can we cancel this job?

Solution

```
scancel job-id
```

where `job-id` is the id of the job. Check the output of `squeue -u $USER` if you are unsure what the job id is.

Using sinteractive

What if we want to run this same task interactively? How can we access an interactive session to run the same code above?

Solution

```
sinteractive --cpus-per-task=4
```

Additional practice materials from hpc.nih.gov

- Submit a job using `sbatch`. *(https://hpc.nih.gov/training/intro_biowulf/hands-on-sbatch.html)*
- Submit a multi-threaded job. *(https://hpc.nih.gov/training/intro_biowulf/hands-on-multi.html)*
- Sumbit a `swarm` job. *(https://hpc.nih.gov/training/intro_biowulf/hands-on-swarm.html)*

Lesson 5 Practice

The following was designed to practice skills learned in lesson 5.

Find the data

Here *(https://journals.asm.org/doi/full/10.1128/mSystems.00065-20?rfr_dat=cr_pub+0pubmed&url_ver=Z39.88-2003&rfr_id=ori%3Arid%3Acrossref.org)* is a paper examining the relationship between the oral microbiome and nasopharyngeal carcinoma. Where can you find the associated data?

Notice that the data was originally submitted to the ENA.

Download the data

Make a directory called `Lesson5_practice` and change directories.

Solution

```
mkdir Lesson5_practice
cd Lesson5_practice
```

Navigate to the NCBI website *(https://www.ncbi.nlm.nih.gov/)* and grab the accession information for the associated BioProject.

1. Download the first 10 samples using `fastq-dump` with `parallel`.

Solution

You can download the accession list directly from the SRA Run Selector.

From the command line:

```
esearch -db sra -query PRJEB37445 | efetch -format runinfo | cut -f 1 -d ',' |sort

cat accessions.txt | parallel -j 1 fastq-dump --split-3 {}
```

Or the same command with prefetch...

```
cat accessions.txt | parallel -j 1 'prefetch {} | fastq-dump --split-3' {}
```

1. Download five samples with the aid of the sra-explorer.

2. Navigate to the ENA. How might you go about downloading the data?

Unix Review Practice Session

Author: Stephan Sanders, PhD (UCSF)

In this exercise, we are going to embark on a Unix treasure hunt created by the **Sanders Lab *(https://sanderslab.github.io/code/)*** at the University of California San Francisco. Note: the treasure hunt materials can be obtained directly from the Sanders lab code repository linked above.

UNIX treasure hunt tutorial

This perl script will install a series of directories and clues that teaches basic UNIX command line skills including `cd`, `ls`, `grep`, `less`, `head`, `tail`, and `nano`. Run the perl script from the command line on a UNIX based machine (e.g. Mac or Linux) using the command: `perl treasureHunt_v2.pl`. Then use `ls` to find the first clue. A PDF of command line commands is also available to download.

- Source
- Manual

Note to start at the `/data/username` folder for this exercise (replace username with the student account ID). To begin create a directory called `treasure_hunt` in your data directory (ie. `/data/username`) and change into it. Next, run the `treasureHunt_v2.pl` perl script in `/data/classes/BTEP`.

Solution

```
mkdir treasure_hunt
cd treasure_hunt
perl /data/classes/BTEP/treasureHunt_v2.pl
ls -l
```

Read the first clue and begin.

Recommendation: Create an environment variable to store the path to the treasure hunt directory to facilitate movement through the directory.

Solution

```
THUNT=`pwd`
echo $THUNT
```

When you have found the treasure, answer or do the following:

1. How many words are in the last line of the file containing the teasure?

Solution

```
tail -n 1 openTheBox.txt | wc -w
```

1. Save the last line to a new file called `finallyfinished.txt` without copying and pasting.

Solution

```
tail -n 1 openTheBox.txt > finallyfinished.txt
```

1. Now append the first line to the same file that you just saved the last line.

Solution

```
head -n 1 openTheBox.txt >> finallyfinished.txt
```

Congratulations! You have found the treasure and have gained some useful unix practice throughout your hunt.

Lesson 7 Practice

This session will allow users to practice assessing FASTQ quality using data from the HBR-UHR study (see https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/ *(https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/)*).

First step is to make sure that the participant is signed onto Biowulf. If not then do the following to sign on. Remember to replace user with the participant's own Biowulf sign on ID.

Solution

```
ssh user@biowulf.nih.gov
```

Then change into the `data` directory.

Solution

```
cd /data/user
```

The practice data is stored in the folder `hbr_uhr_b4b` in the folder `/data/classes/BTEP`. Copy it to the data directory.

Solution

```
cp -r /data/classes/BTEP/hbr_uhr_b4b .
```

Change into the `hbr_uhr_b4b` folder in the participant's `data` directory.

```
cd hbr_uhr_b4b
```

Request an interactive session with 12 gb of RAM or memory and 10 gb of local temporary storage space.

Solution

```
sinteractive --mem=12gb --gres=lscratch:10
```

What are the contents in `hbr_uhr_b4b`?

Solution

```
ls -l
```

```
drwxr-x---. 2 wuz8 wuz8 4096 Dec 20 15:41 reads
drwxr-x---. 2 wuz8 wuz8 4096 Dec 20 14:37 references
```

There are two folders, `reads` and `references`.

```
ls -1 reads
```

The `reads` folder contains the FASTQ files. There are 12 of these, two for each sample.

```
HBR_Rep1_R1.fq
HBR_Rep1_R2.fq
HBR_Rep2_R1.fq
HBR_Rep2_R2.fq
HBR_Rep3_R1.fq
HBR_Rep3_R2.fq
UHR_Rep1_R1.fq
UHR_Rep1_R2.fq
UHR_Rep2_R1.fq
UHR_Rep2_R2.fq
UHR_Rep3_R1.fq
UHR_Rep3_R2.fq
```

```
ls references
```

The references folder contains the chromosome 22 reference (fa or FASTA) and annotation (gtf) files.

```
22.fa  22.gtf
```

There is also a file, `hbr_uhr_samples.txt` that contains the HBR-UHR sample IDs.

```
cat hbr_uhr_samples.txt
```

```
HBR_Rep1
HBR_Rep2
HBR_Rep3
UHR_Rep1
UHR_Rep2
UHR_Rep3
```

Make a directory called `hbr_uhr_b4b_raw_qc` inside `/data/user/hbr_uhr_b4b`.

Solution

```
mkdir hbr_uhr_b4b_raw_qc
```

Load FASTQC.

Solution

```
module load fastqc
```

Run FASTQC for raw FASTQ files and save the results in `hbr_uhr_b4b_raw_qc`. Stay in `/data/user/hbr_uhr_b4b` for this.

Solution

```
fastqc reads/*.fq -o hbr_uhr_b4b_raw_qc
```

Change into `hbr_uhr_b4b_raw_qc`.

Solution

```
cd hbr_uhr_b4b_raw_qc
```

Load and run MultiQC to combine all of the FASTQC reports in `hbr_uhr_b4b_raw_qc`. Name the MultiQC results with prefix `hbr_uhr_b4b_raw_qc`.

Solution

```
module load multiqc
```

```
multiqc --filename hbr_uhr_b4b_raw_qc .
```

Copy the `hbr_uhr_b4b_raw_qc.html` MultiQC report to local `Downloads` to view the report.

Solution

Open a new Terminal (Mac) or Command Prompt (Windows). Then change into the local `Downloads` folder.

```
cd Downloads
```

```
scp user@helix.nih.gov:/data/user/hbr_uhr_b4b/hbr_uhr_b4b_raw_qc/hbr_uhr_b4b_raw_qc
```

Based on the MultiQC report for the raw FASTQ files, what is the next step.

Solution

Align the data to reference genome since the quality of the data is good and there does not appear to be adapter contamination.

Lesson 8 Practice

Participants will practice trimming using `trimmomatic` in this help session but with data downloaded from the SRA instead as the HBR-UHR FASTQC report indicated that there are no adapter contamination.

Before getting started, be sure to sign onto Biowulf.

Solution

```
ssh user@biowulf.nih.gov
```

Then change into the participant's `data` directory.

Solution

```
cd /data/user
```

Create a new folder called `practice_trim` and change into it.

Solution

```
mkdir practice_trim
```

```
cd practice_trim
```

Request an interactive session with 12 gb of RAM (or memory) and 10 gb of local temporary storage.

Solution

```
sinteractive --mem=12gb --gres=lscratch:10
```

Use the SRA Tool kit to obtain the first 10,000 sequences for SRR1553606 *(https://www.ncbi.nlm.nih.gov/sra/?term=SRR1553606)*. These are paired end sequences.

Solution

```
module load sratoolkit
```

```
fastq-dump --split-files -X 10000 SRR1553606
```

```
Read 10000 spots for SRR1553606
Written 10000 spots for SRR1553606
```

Stay in `practice_trim` and list its contents.

```
ls
```

```
SRR1553606_1.fastq  SRR1553606_2.fastq
```

QC the two FASTQ files for SRR1553606 (ie. `SRR1553606_1.fastq` and `SRR1553606_2.fastq`).

Solution

```
module load fastqc
```

```
fastqc *.fastq
```

Copy the HTML FASTQC report to local computer `Downloads` folder to view the results. Is there adapter contamination?

Solution

Change into local `Downloads` and obtain the HTML FASTQC reports using `scp`.

```
scp user@helix.nih.gov:/data/user/practice_trim/SRR1553606_1_fastqc.html  .
```

```
scp user@helix.nih.gov:/data/user/practice_trim/SRR1553606_2_fastqc.html  .
```

There appears to be Nextera Transposase contamination in this data. Would this data require quality trimming?

Try trimming away the adapter sequence below from the FASTQ files and rerun FASTQC.

```

```

```
>nextera
CTGTCTCTTATACACATCTCCGAGCCCACGAGAC
```

Hint: copy the above sequence and save it to the file `nextera.fa`.

Solution

```
nano nextera.fa
```

Copy and paste the Nextera sequence above, hit control-x, save and exit the editor.

Trim away the sequence in `nextera.fa` using Trimmomatic.

- For quality trimming, use a SLIDINGWINDOW of size 4 bases and average score in the window of 30.
- For adapter trimming, set the adapter strigencies
- seed mismatches: 2
- palindrome clip threshold: 30
- simple clip threshold: 5
- Retain on those trimmed sequences whose length is greater than 50 bases

Solution

```
module load trimmomatic
```

```
java -jar $TRIMMOJAR PE SRR1553606_1.fastq SRR1553606_2.fastq SRR1553606_trimmed_1.
```

QC the two paired and trimmed FASTQ files.

Solution

```
fastqc *trimmed_*.fastq
```

Copy the two HTML reports for the trimmed FASTQ files to local `Downloads` folder to review the results.

Solution

```
scp user@helix.nih.gov:/data/user/practice_trim/SRR1553606_trimmed_1_fastqc.html .
```

```
scp user@helix.nih.gov:/data/user/practice_trim/SRR1553606_trimmed_2_fastqc.html .
```

Did trimming remove the adapters and improve the data quality?

Solution

Yes

Lesson 9 Practice

Since the HBR-UHR data did not need trimming, this practice session will have participants align this data to the chromosome 22 human reference.

Sign onto Biowulf and change into the `/data/user/hbr_uhr_b4b` folder.

Solution

```
ssh user@biowulf.nih.gov
```

```
cd /data/user/hbr_uhr_b4b
```

Request an interactive session with 12 gb RAM (or memory) and 10 gb of local temporary storage.

Solution

```
sinteractive --mem=12gb --gres=lscratch:10
```

Create a new folder in `/data/user/hbr_uhr_b4b` called `hbr_uhr_hisat2`.

Solution

```
mkdir hbr_uhr_hisat2
```

Load HISAT2.

Solution

```
module load hisat2
```

Stay in the folder `hbr_uhr_b4b` for this exercise.

Build HISAT2 indices for the chromosome 22 reference genome. The file for the chromosome 22 genome is located in the folder `references` and the file name is `22.fa`. Give the indices a

base name (ie. file name without the extension) of `22`. List the contents of the `references` folder to make sure that the build succeeded (ie. the `.ht2` files are present).

Solution

```
hisat2-build references/22.fa references/22
```

After building the indices, align the HBR-UHR FASTQ files to genome. Use the `parallel` command for this and store the alignment output in the folder `hbr_uhr_hisat2`.

Solution

```
cat hbr_uhr_samples.txt | parallel "hisat2 -x references/22 -1 reads/{}_R1.fq -2 re
```

Look at the overall alignment rates in the HISAT2 alignment summary files. Why are they low?

Solution

```
cat hbr_uhr_hisat2/*summary.txt | grep "overall"
```

```
52.75% overall alignment rate
52.51% overall alignment rate
52.68% overall alignment rate
47.91% overall alignment rate
51.26% overall alignment rate
46.94% overall alignment rate
```

"In addition, a spike-in control was used. Specifically we added an aliquot of the ERCC ExFold RNA Spike-In Control Mixes to each sample." -- (Griffith lab RNA Bio, https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/) This dataset has ERCC spike-ins as a quality control measure so not all sequences will map to the human chromosome 22 genome.

Create sorted BAM files and BAM indices for the alignment results.

Solution

Create BAM from SAM.

```
cat hbr_uhr_samples.txt | parallel "samtools sort -o hbr_uhr_hisat2/{}.bam hbr_uhr_
```

Index BAM.

```
cat hbr_uhr_samples.txt | parallel "samtools index -b hbr_uhr_hisat2/{}.bam hbr_uhr
```

Lesson 10 Practice

In this session, participants will practice generating a gene expression matrix for the HBR-UHR data.

Sign onto Biowulf and change into the `/data/user/hbr_uhr_b4b` folder.

Solution

```
ssh user@biowulf.nih.gov
```

```
cd /data/user/hbr_uhr_b4b
```

Create a new folder called `hbr_uhr_expression`.

Solution

```
mkdir hbr_uhr_expression
```

Request an interactive session with 12 gb of RAM and 10 gb of local temporary storage space.

Solution

```
sinteractive --mem=12gb --gres=lscratch:10
```

Generate a gene expression table using `featureCounts` for all of the samples in one go. Change into `/data/user/hbr_uhr_b4b/hbr_uhr_hisat2` for this.

Solution

```
module load subread
```

```
featureCounts -p --countReadPairs -a ../references/22.gtf -g gene_name -o ../hbr_uh
```

Change back to the `hbr_uhr_expression` folder after the gene express matrix has been generated.

Solution

```
cd ../hbr_uhr_expression
```

Convert the gene expression matrix `hbr_uhr_gene_expression.txt` to a CSV file without a header line containing `featureCounts` information as well as with only the following columns:

- Gene name
- Columns for the expression of each sample

Do this using | (or pipe) to avoid writing intermediate files.

Solution

```
sed '1d' hbr_uhr_gene_expression.txt | cut -f1,7-12 | tr '\t' ',' > hbr_uhr_gene_e›
```

Make sure the result is correct.

```
column -t -s ',' hbr_uhr_gene_expression.csv | less -S
```

Lesson 11 Practice

In this session, participants will practice making bigWig files from the HBR-UHR alignment results.

As always, be sure to be signed on to Biowulf for this set of exercises.

Solution

```
ssh user@biowulf.nih.gov
```

Change into the `/data/user/hbr_uhr_b4b` folder.

Solution

```
cd `/data/user/hbr_uhr_b4b`
```

If not done already, request an interactive session with 12gb of RAM and 10gb of local temporary storage space.

Solution

```
sinteractive --mem=12gb --gres=lscratch:10
```

Load `bedtools`.

Solution

```
module load bedtools
```

Create `bedGraph` files from the HBR-UHR alignment results. Stay in `/data/user/hbr_uhr_b4b` and recall that these are RNA sequencing data. Write the `bedGraph` files to the folder `hbr_uhr_hisat2`.

Solution

```
cat hbr_uhr_samples.txt | parallel "bedtools genomecov -ibam hbr_uhr_hisat2/{}.bam
```

Next, index the chromosome 22 reference genome stored in the folder `references` with file name `22.fa`. Which module is needed?

Solution

```
module load samtools
```

```
samtools faidx references/22.fa
```

Create `bigWig` files from the HBR-UHR `bedGraph` files. Which module needs to be loaded to do this? Stay in `/data/user/hbr_uhr_b4b` for this exercise.

Solution

```
module load ucsc
```

```
cat hbr_uhr_samples.txt | parallel "bedGraphToBigWig hbr_uhr_hisat2/{}_hisat2.bg re
```

Lesson 12 Practice

Participants can work on these practice questions on their own time as a Biowulf account or local installation of IGV is needed.

Open the `bigWig` files for HBR_Rep1 (HBR_Rep1_hisat2.bw) and UHR_Rep1 (UHR_Rep1_hisat2.bw). Search of the gene TEF. Does it appear to be more highly expressed in the HBR_Rep1 sample? Use "Human (hg38)" as reference.

Solution

Open the HBR_Rep1_hisat2.bw and UHR_Rep1_hisat2.bw `bigWig` files. Zoom to chromosome 22. Select both `bigWig` tracks, right click and choose "Group Autoscale". Search for TEF. From the IGV view, it does appear that TEF is more highly expressed in HBR_Rep1.

How many transcript isoforms are there for TEF.

Solution

Right click on the Gene track and select expand. There are 5 transcript isoforms associated with TEF, although 3 of these are predicted as the accession starts with "XM" and "XR" (see https://www.ncbi.nlm.nih.gov/books/NBK50679/).

Which of the TEF isoforms is likely to be expressed?

Solution

Load the BAM files for these two samples. It appears that from the splice junctions that NM_003216.4 is expressed. Although in the HBR_Rep1 sample, reads are mapping to the first exon of NM_001145398.2.

Zoom to chromosome 22, position 41,387,655. Notice anything in the HBR_Rep1 sample?

Solution

It looks like there is a heterozygous SNP, where the reference is C but some reads are showing T.

Lesson 13 Practice

In this practice session, participants will filter and perform some quality checks on the HBR-UHR gene expression data.

Before getting started, make sure to be connected to Biowulf and an interactive session with 12 gb of memory and 10 gb of local temporary storage is created.

Change into the `/data/user/hbr_uhr_b4b` folder.

Solution

```
cd /data/user/hbr_uhr_b4b
```

Create directory called `hbr_uhr_deg` to store the differential expression analysis outputs.

Solution

```
mkdir hbr_uhr_deg
```

What is folder is the HBR-UHR gene expression data table stored and what is the file name?

Solution

The gene expression table is stored in the folder `hbr_uhr_expression/`. The name of the gene expression table is `hbr_uhr_gene_expression.csv`. To reference this from `hbr_uhr_b4b` use `hbr_uhr_expression/hbr_uhr_gene_expression.csv`.

Load R.

Solution

```
module load R
```

The scripts are the in the folder `b4b_script`

Filter low expressing genes out of `hbr_uhr_gene_expression.csv`. Set the minimum number of samples per group that have greater than 0 expression to be 2. Assign hbr_uhr to the study name and write the output to hbr_uhr_deg.

Solution

```
Rscript b4b_scripts/filter_expression.R hbr_uhr_expression/hbr_uhr_gene_expression.
```

How many genes are in the filtered expression table?

Solution

```
wc -l hbr_uhr_deg/hbr_uhr_gene_expression_filtered.csv
```

Since the filter gene expression CSV file has 562 lines then this means 561 genes are left after filtering.

Run QC on the filtered expression data. Assign hbr_uhr as the study name and write the output to the folder `hbr_uhr_deg`.

Solution

```
Rscript b4b_scripts/quality_check.R hbr_uhr_deg/hbr_uhr_gene_expression_filtered.cs
```

After running QC, download the images to the `Downloads` folder of personal computer to view the results. To do this, open and a new terminal (mac) or command prompt window (Windows 10 or above) and change into the local `Downloads` folder.

```
cd Downloads
```

Then use the `scp` command construct below to download. Remember to replace `user` with the participants assigned Biowulf student ID.

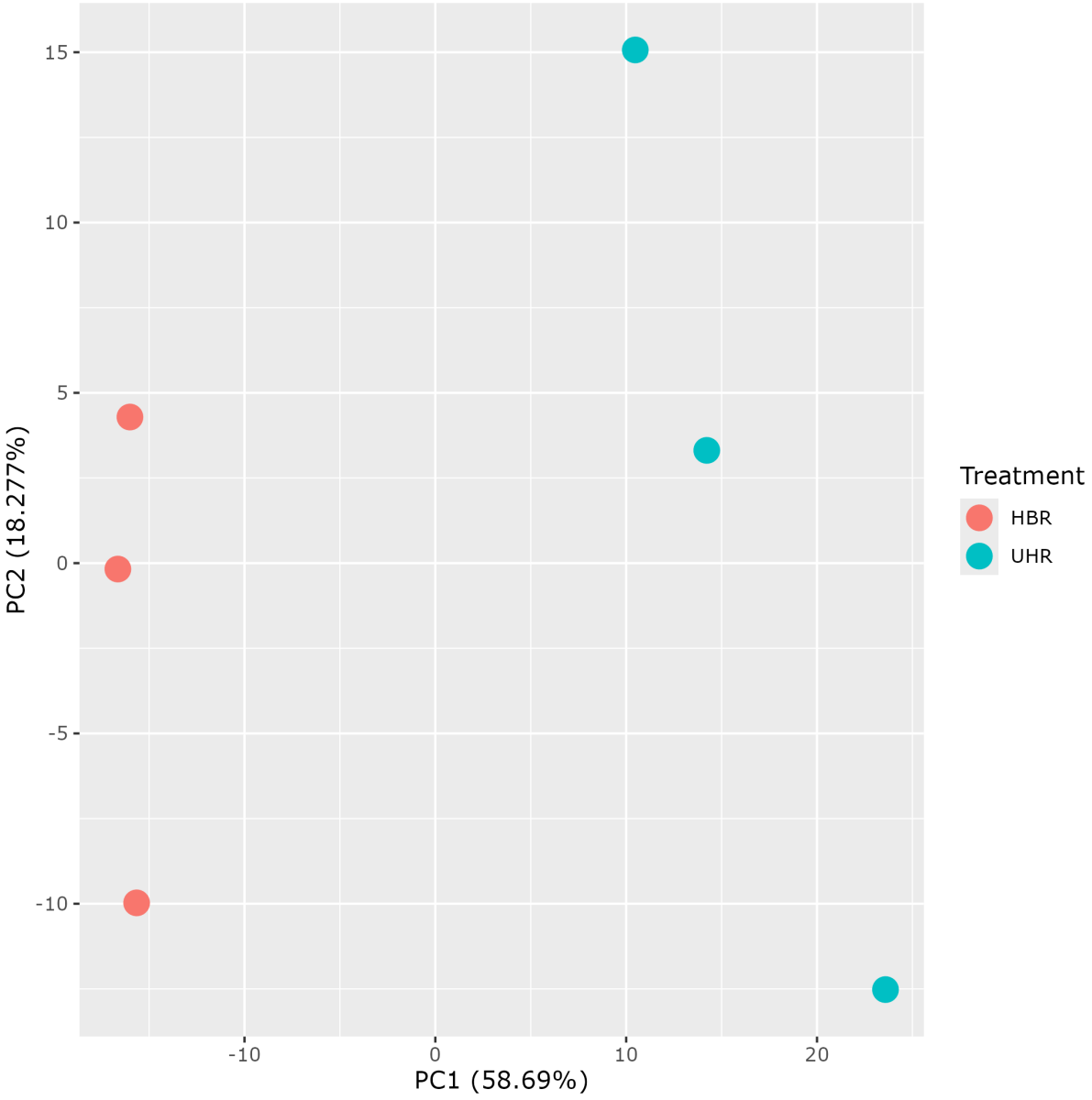
```
scp -r user@helix.nih.gov:/data/user/hbr_uhr_b4b/hbr_uhr_deg .
```

Use the Mac Finder or Windows Explorer to navigate to `hbr_uhr_deg` in the local `Downloads` directory to begin exploring the results.

Does it look like the samples are separated by biology?

Solution

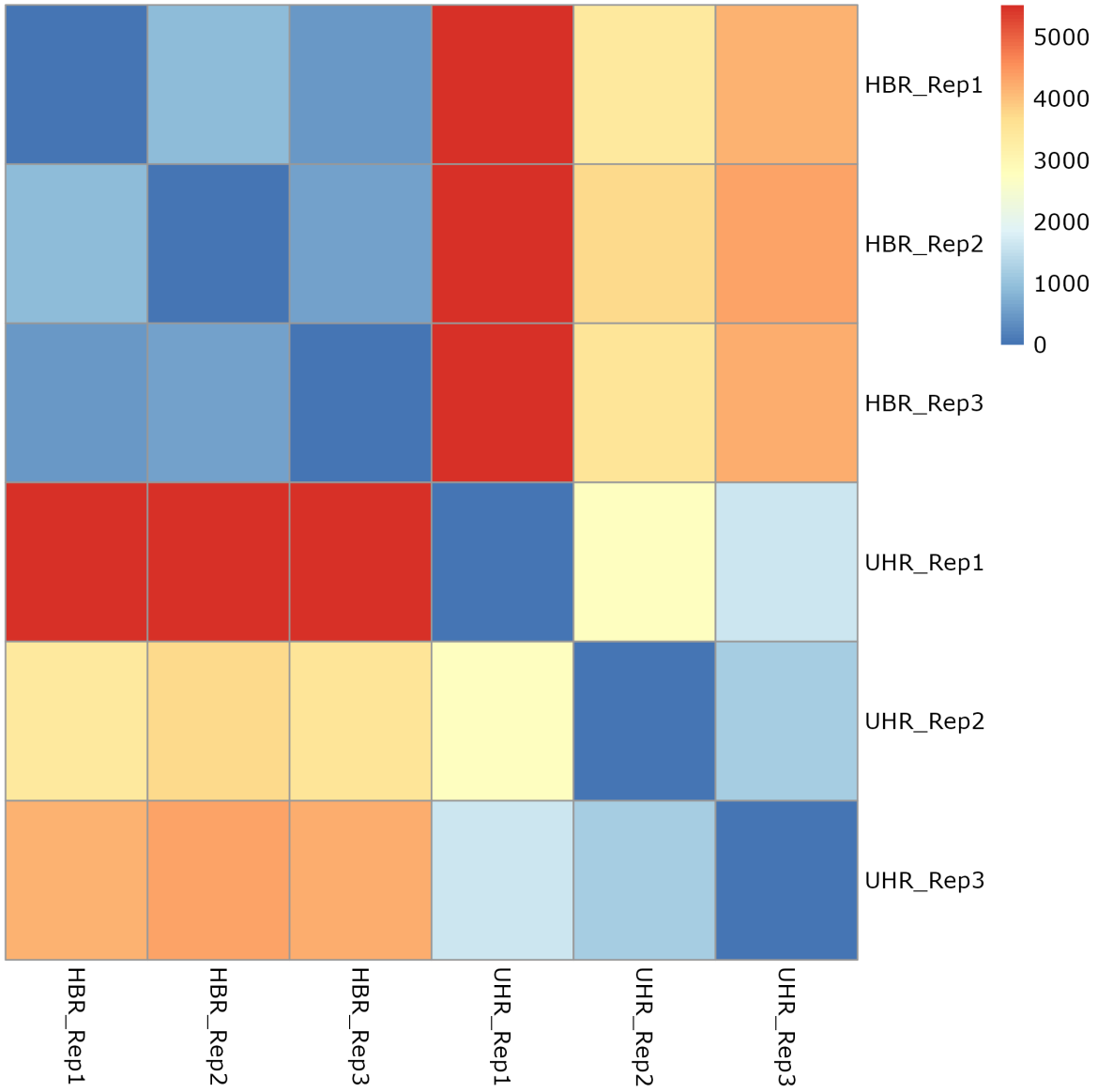
Yes, it appears that biology is driving the difference between the HBR and UHR samples, which are separated along the first principal component axis.



What is the distance plot informing of?

Solution

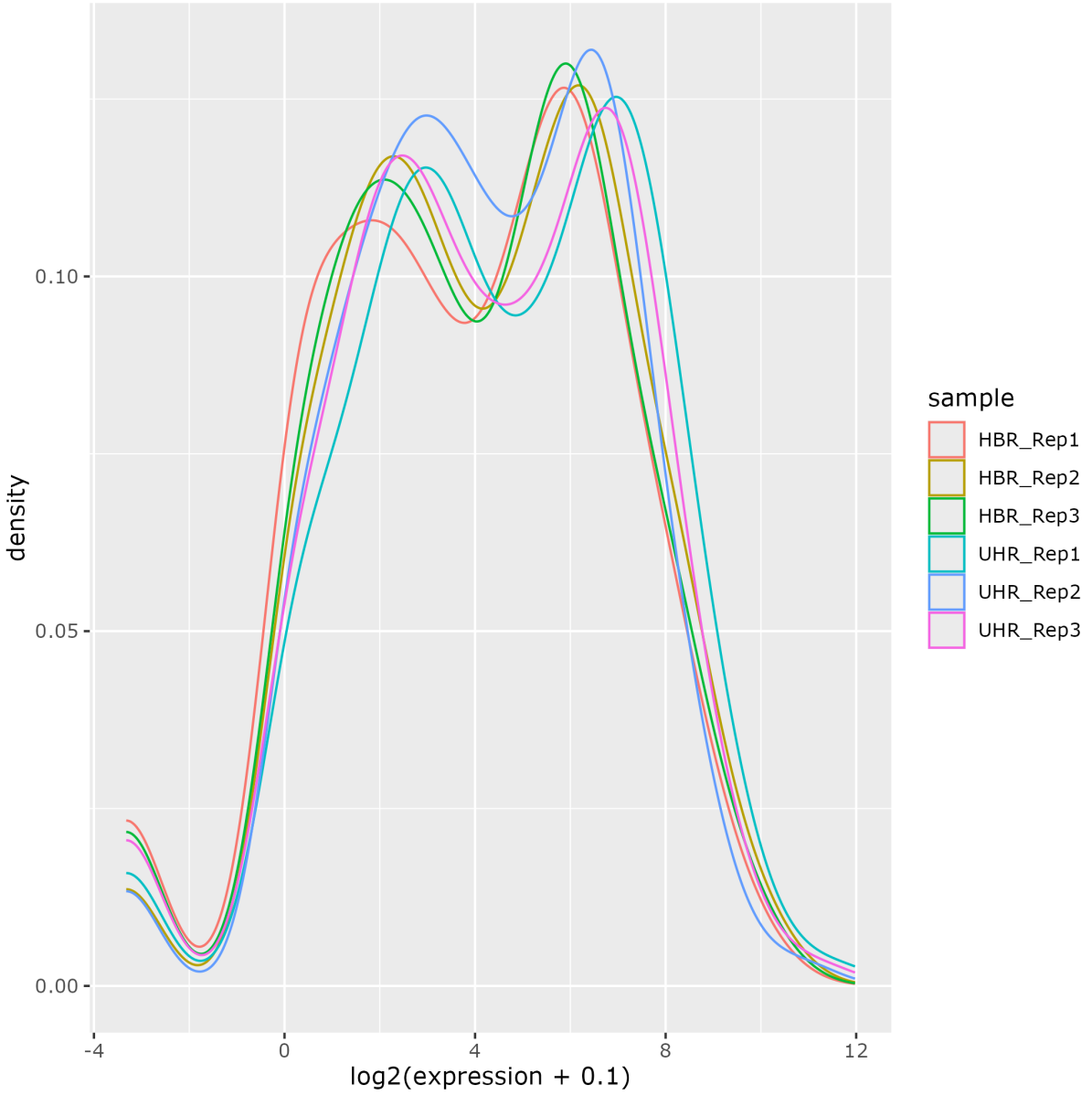
The distance plot is not really optimal as samples within group are not as close together.

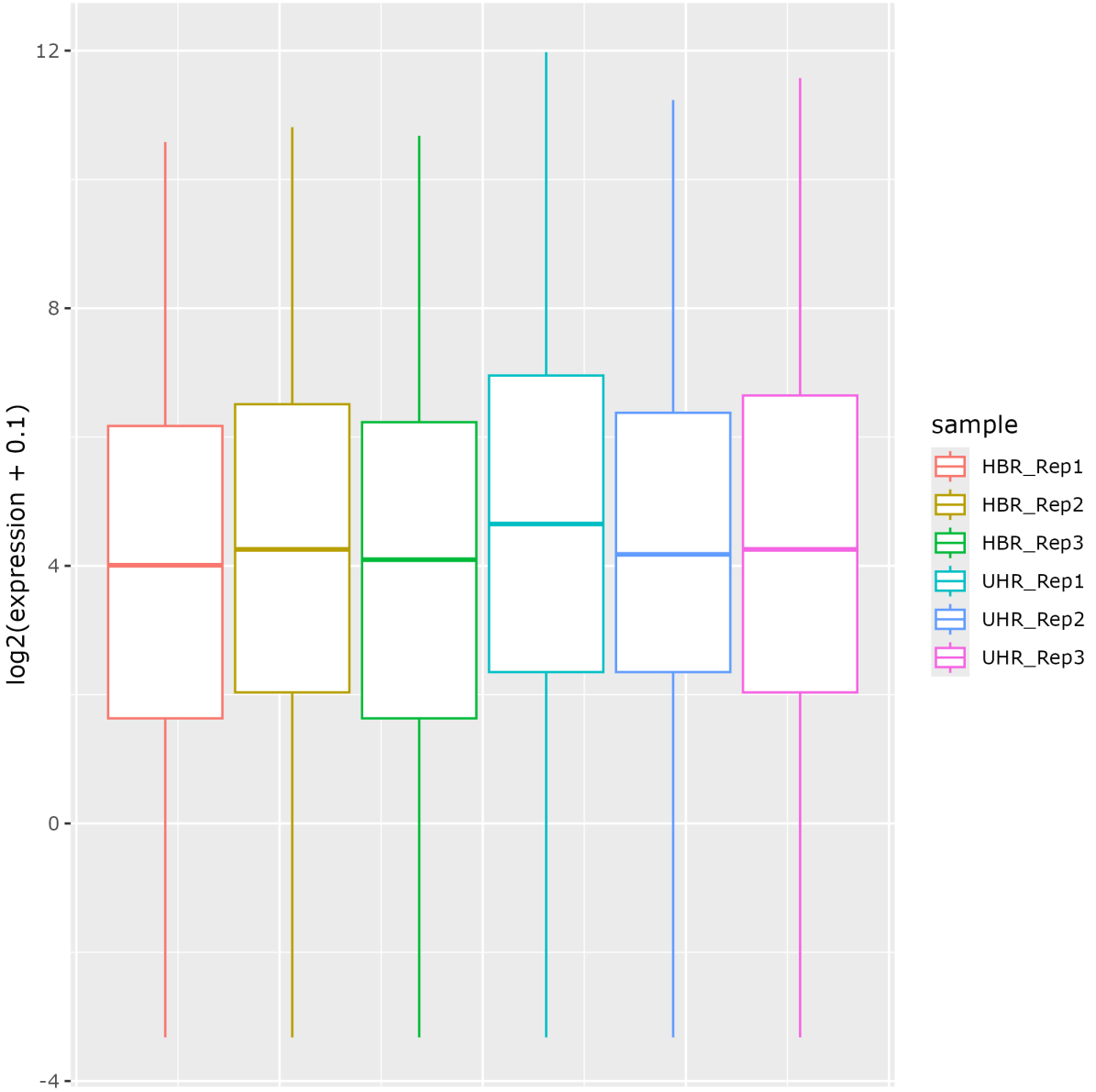


How does the expression distribution among samples look?

Solution

From the density and box plots, the expression distribution are not equal among samples. Hopefully normalization improves this.


density
0.10
0.05
0.00
-4
0
4
8
12
log2(expression + 0.1)
sample
HBR_Rep1
HBR_Rep2
HBR_Rep3
UHR_Rep1
UHR_Rep2
UHR_Rep3


12
8
4
0
-4
log2(expression + 0.1)
sample
HBR_Rep1
HBR_Rep2
HBR_Rep3
UHR_Rep1
UHR_Rep2
UHR_Rep3

Lesson 14 Practice

In this practice session, participants will run the actually differential gene expression analysis on the HBR-UHR gene expression data.

Before getting started, be sure to be connected to Biowulf and have an interactive session with 12 gb of memory and 10 gb of local temporary storage space ready. Remember to replace `user` with the participant's assigned Biowulf student account ID.

Next, change into the `/data/user/hbr_uhr_b4b` folder.

Solution

```
cd /data/user/hbr_uhr_b4b
```

Load R.

Next, change into the `/data/user/hbr_uhr_b4b` folder.

Solution

```
module load R
```

Run the `deg2.R` script in the folder `b4b_scripts` to generate differential expression analysis results. Use `hbr_uhr` for the study name and write the output to `hbr_uhr_deg`.

Solution

```
Rscript b4b_scripts/deg.R hbr_uhr_deg/hbr_uhr_gene_expression_filtered.csv hbr_uhr_
```

After running the differential expression analysis, copy the `hbr_uhr_deg` folder to local downloads folder to review the images.

To do this, open and a new terminal (mac) or command prompt window (Windows 10 or above) and change into the local `Downloads` folder.

```
cd Downloads
```

Then use the `scp` command construct below to download. Remember to replace `user` with the participants assigned Biowulf student ID.

```
scp -r user@helix.nih.gov:/data/user/hbr_uhr_b4b/hbr_uhr_deg .
```

How many genes matched the criteria of log2FoldChange>=4.5 and adjusted p-value <=0.01 or log2FoldChange<=-4.5 and adjusted p-value <=0.01?

Solution

```
wc -l hbr_uhr_deg/hbr_uhr_filter_deg.csv
```

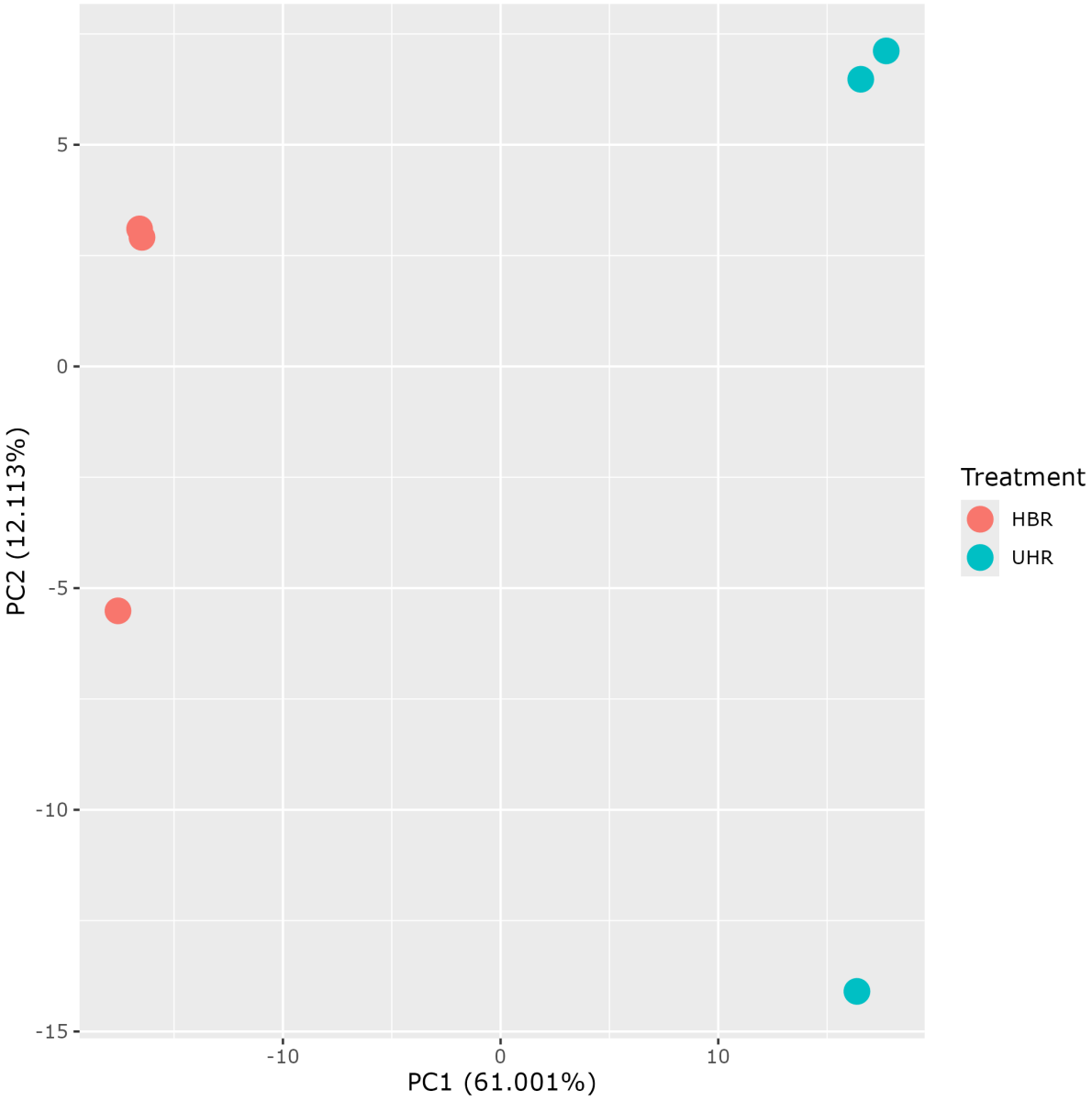
There are 23 (ie. 24 lines in hbr_uhr_filter_deg.csv, so minus 1 line for the column headers to get 23 genes) genes that matched the specified log2FoldChange and adjust p-value threshhold.

What is the PCA obtained from normalized expression table suggesting?

Solution

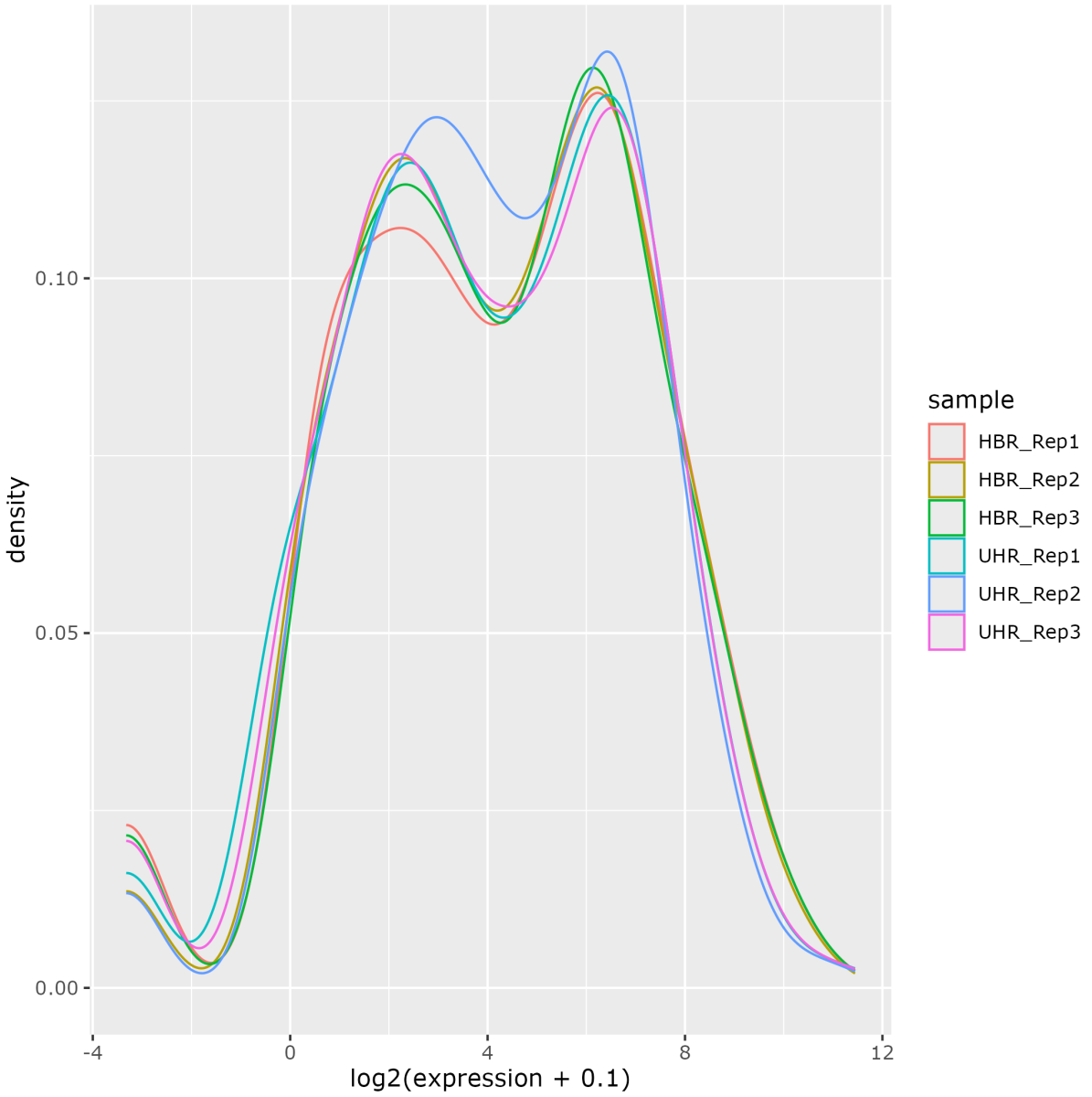
After normalization, the HBR and UHR samples still separate along the first principal components axis. But it appears that there is one sample from each group that does not cluster with the rest on the second principal

components axis. This could suggest biological differences between samples within condition or batch effect.

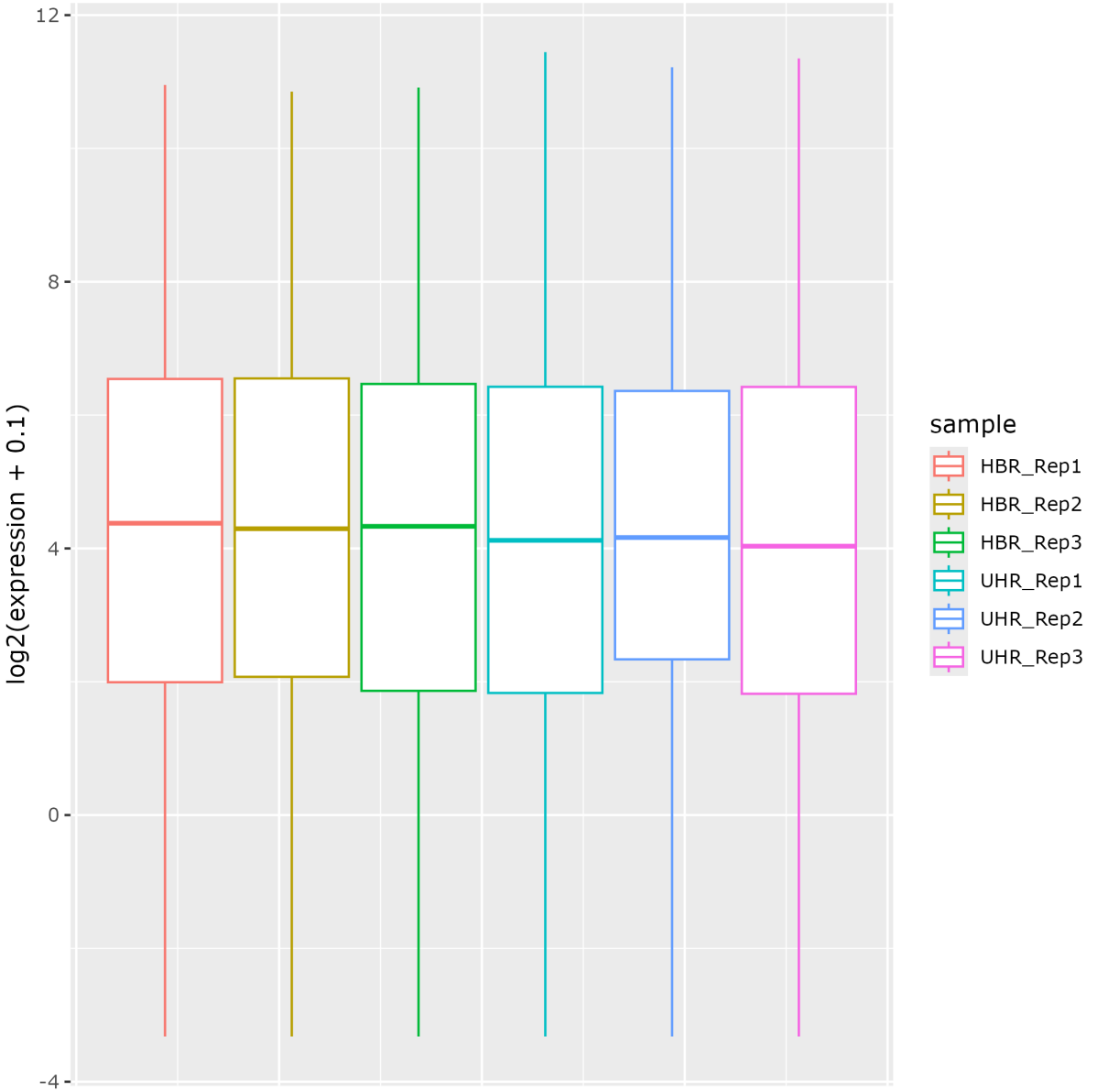


Did normalization help in terms of pushing the gene expression distribution for each sample towards the same.

Solution


0.10
0.05
0.00
density
-4
0
4
8
12
log2(expression + 0.1)
sample
HBR_Rep1
HBR_Rep2
HBR_Rep3
UHR_Rep1
UHR_Rep2
UHR_Rep3

Solution



Take a look at the rest of the visualizations (ie. the distance plot, expression heatmap, and volcano plot) to see what can be concluded.