\usepackage{booktabs} \usepackage{longtable} \usepackage{array} \usepackage{multirow} \usepackage{wrapfig} \usepackage{float} \usepackage{colortbl} \usepackage{pdflscape} \usepackage{tabu} \usepackage{threeparttable} \usepackage{threeparttablex} \usepackage[normalem]{ulem} \usepackage{makecell} \usepackage{xcolor}

Transcriptome Data Analysis in Non-model Organisms

Jiratchaya Nuanpirom

Ponsit Sathapondecha

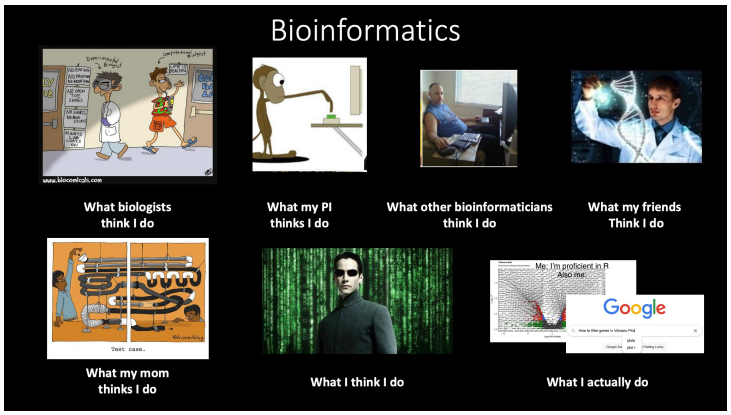
Prasert Yodsawat

3/10/23

Table of contents

Preface

Welcome to the book of Transcriptome Data Analysis in Non-Model Organisms.

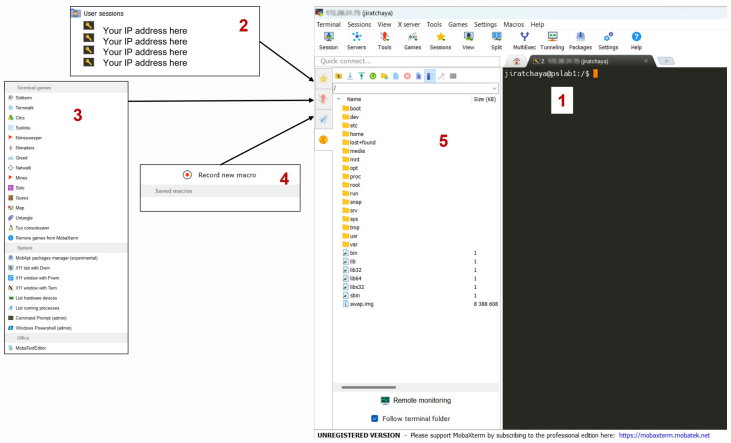


A confusion matrix. From twitter @ninadoak.

Introduction to MobaXterm, Terminal, and SSH

MobaXterm (for Windows)

MobaXterm is a toolbox for remote computing. In a single Windows application, it provides loads of functions that are tailored for programmers, webmasters, IT administrators and pretty much all users who need to handle their remote jobs in a more simple fashion. MobaXterm provides all the important remote network tools, such as SSH, X11, RDP, VNC, FTP, MOSH, and of course, Unix commands, and many more!



MobaXterm user interface. In the context of remote access through SSH and FTP, mobaXterm provides easy-to-access route as (1) a secure shell (SSH) terminal of the remote server, (2) a list of remote server you've accessed, (3) Utilities facilitating remote server access including entertainment, like Swiss army knife!, (4) If you want to reduce redundant typing, just set macro to it, and (5) a files available in the current working directory in the remote server, you can also transfer files from remote server to your local computer using this route!

There are many advantages of having an All-In-One network application for your remote tasks, e.g. when you use SSH to connect to a remote server, a graphical SFTP browser will automatically pop up in order to directly edit your remote files.

Visit MobaXterm official website to see a demo: https://mobaxterm.mobatek.net/demo.html

Terminal (for macOS)

Terminal provides a command-line interface to macOS. Each window in Terminal represents an instance of a shell process. The window contains a prompt that indicates you can enter a command. The prompt you see depends on your Terminal and shell settings, but it often includes the name of the host you're logged in to, your current working folder, your user name, and a prompt symbol. For example, if a user named michael is using the default zsh shell, the prompt appears as:

```
michael@MacBook-Pro ~ %
```

This indicates that the user named michael is logged in to a computer named MacBook-Pro, and the current folder is his home folder, indicated by the tilde (~).

MacOS features a built-in SSH client called Terminal which allows you to quickly and easily connect to a server. Starting from open the "terminal" app, and enter the standard SSH command:

```
ssh user@IPAddress
```

This will connect to the server via SSH with the username `user` and the default SSH port 22. The connection will look similar to the following:

Connecting to Remote Server

Bioinformatics data processing tasks require more computing power than our laptops, so we need large servers or clusters. It's likely you'll work mostly over a network connection with remote machines on some projects. It can be frustrating for beginners to work with a remote machine. So, This part will introduce you to some commonly used bash commands. To make it easier for beginners to manage their remote machines, there are a range of different tools and technologies available, such as SSH, FTP, and terminal commands, which can be used to access and manage the environment of the machine. Additionally, there are a variety of bash commands which can be used to streamline the process of managing the machine Buffalo (2015).

What you need to know for connecting to a remove server:

1. Your username and password in the remote server
2. IP address of the remote server, and which port to connect to server
3. You should know whether the remote server accessible via local network or a public IP address

By default, SSH uses port 22 but it can be changed to a non-standard port. To securely connect the client to the remote server, SSH uses symmetric encryption, asymmetric encryption, and hashing. If you're connecting for the first time, you'll be asked to verify the server's public key. Whenever you connect to the same server in the future, the client will reference this verified public key. During an SSH connection, the client and server negotiate a session key used to encrypt and decrypt data.

```
PS C:\Users\[redacted]> ssh jiratchaya@[redacted]
The authenticity of host '[redacted] ([redacted])' can't be established.
ED25519 key fingerprint is SHA256:h+AtPyZBUpNh3uWKSTb6/K/Mr5b6WjBXywpfCJpeuEc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

In order to establish a connection, SSH needs to verify SHA keys once connected for the first time. Once authentication is complete, the SSH connection is secure and can be trusted for future access.

Upon connecting to the remote server, you'll see a welcome message like this

```
PS C:\Users\[redacted]> ssh jiratchaya@[redacted]
jiratchaya@[redacted]'s password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-60-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Feb 27 12:30:54 PM +07 2023

  System load:  0.00048828125     Processes:                688
  Usage of /:   36.5% of 2.58TB   Users logged in:           1
  Memory usage: 0%                IPv4 address for docker0:  [redacted]
  Swap usage:   0%                IPv4 address for eno12399: [redacted]
  Temperature:  39.0 C            IPv4 address for eno8303:  [redacted]

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

7 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Mon Feb 27 11:02:32 2023 from [redacted]
jiratchaya@pslab1:~$ |
```

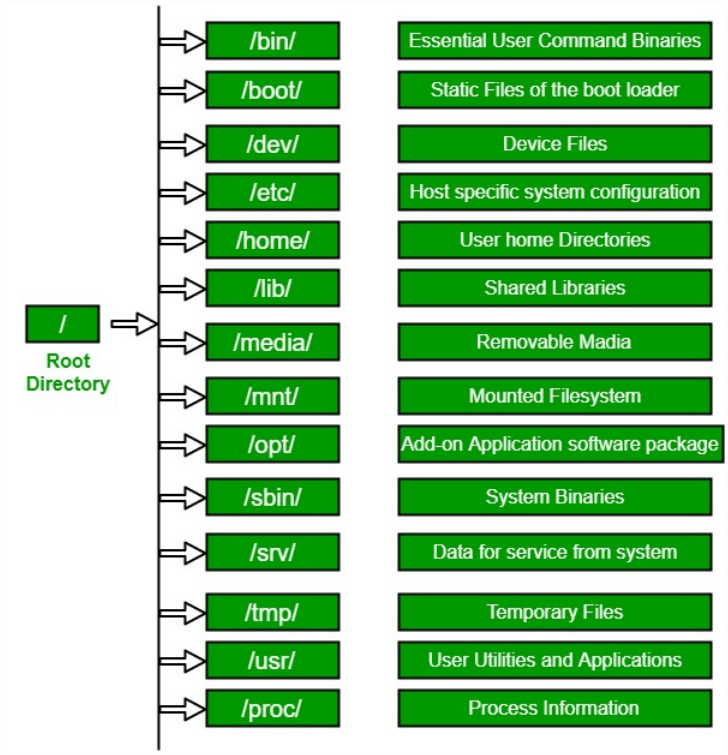
An example welcome message of server using Ubuntu, including general software and hardware status, information of the latest connection, as well as a prompt for user command.

Bash Command Language for Biologists

Shell scripts (or shell or UNIX) are widely used in bioinformatics because they're the interface for large bioinformatics programs. In this workshop, you'll learn how to use the necessary Bash command concepts. This will allow you to focus on the content of the commands in the following chapters rather than on understanding shell syntax. However, before we start learning bash, it's good to understand Linux file systems a little bit.

Linux File Systems

In Unix-like operating systems, the Linux file system defines the directory structure and contents. Even if they're located on different physical or virtual hard disks, all files and directories are located under the root directory.



Schematic hierarchy of Linux file systems. The figure is adopted from https://www.geeksforgeeks.org/linux-file-hierarchy-structure.

Root (/)

It is the root directory of the entire file system hierarchy and the primary hierarchy root. The root directory is where everything begins. This directory can be written only by root.

/bin

Essential commands that must be available with all users, for example, `cat`, `ls`, `cp`, `cd`, `top`, `mkdir` and many more.

/dev

Essential device files such as `/dev/null`, `/dev/shm`. This includes terminal devices, USB or other devices connected to the system.

/etc

System-wide configuration files for the host, contain files that all programs need. Also included are startup and shutdown shell scripts for starting and stopping individual programs, such as `/etc/fstab` for permanently mounting external disks, `/etc/netplan` for configuring the network and IP address, and more.

/home

Users' home directories, where they keep their saved files and settings. These directories are used to store all of a user's files and settings in one place so that they can easily access their data and keep it organized. For example `/home/ponsit`, `/home/jiratchaya`, `/home/prasert`.

/lib

Contain essential libraries for the binaries in `/bin/` and `/sbin/`.

/media

Mount points for removable media such as CD-ROMs (deprecated).

/mnt

Temporary mount directory where sysadmins can mount file systems, such as `/mnt/external_disk_1`, `/mnt/removable_drive_1`, etc.

/opt

Optional application software packages, including add-on applications from individual vendors.

/sbin

Essential system binaries, e.g., `fsck`, `init`, route.

/tmp

Temporary files that aren't preserved between reboots and may be severely restricted.

/usr

A secondary hierarchy for read-only user data. Most utilities and applications are located here.

Basic Bash Commands

Bash is a Unix shell that allows you to enter commands that are then interpreted and executed by the computer. Commands can be used to perform tasks such as creating a directory, running a program, or deleting a file. Bash is a type of interpreter that takes user input and converts it into a language that the computer can understand and execute. Commands usually consist of keywords, arguments, and flags that allow the user to control how the command is interpreted and executed by the computer.

Creating directories

Keeping all your files in a single directory makes things much easier for you and your collaborators, and makes it easier to reproduce. Suppose you're working on a transcriptome analysis of *Cyanophora paradoxa*. Your first step would be to choose a short, appropriate project name and create some basic directories.

> ☐**Note:** In Linux file system, **directory** is exactly the same with **folder**.

To keep it short and clear, 'Cpa' is used as an alias article name for *C. paradoxa*, and as the name of the directory, followed by words describing your work, for example.

> ☐ **warning:** Avoid using spaces ( ) or special characters such as slashes (/), backslashes (\ ), accented characters ('), tilde (~), and many others. It is **recommended to use underscore (_)** or hyphen (-) instead of these special characters.

- Create a directory name 'Cpa_RNASeq' from current working directory

```
mkdir Cpa_RNASeq
```

This will create a directory named 'Cpa_RNASeq' in your current working directory. Let us create some subdirectories!

- Create subdirectory '01_Rawdata' under the 'Cpa_RNASeq' directory

```
mkdir Cpa_RNASeq/01_Rawdata
```

This will create a subdirectory name '01_Rawdata' in the directory 'Cpa_RNASeq'

- Create multiple directory at once

For example, if you want to create 2 directories named '02_QC' and '03_assembly' under the 'Cpa_RNASeq' directory, then simply type

```
mkdir Cpa_RNASeq/{02_QC,03_assembly}
```

> **Activity**
>
> A well-organized project directory can make life easier. Your project directory should be organized in a consistent and understandable way. A clear project organization makes it easier for both you and your collaborators to find out exactly where and what everything is located, which improves the reproducibility of research. It's also much easier to automate tasks when files are organized and clearly named.
>
> In this workshop, we'll learn transcriptome data analysis in many steps from downloading reads to transcriptome annotation. Therefore, we'll divide each analysis step into subdirectories as follows. Let's assume that we have already created the directory `Cpa_RNASeq`.
>
> ```
> .
> └── Cpa_RNASeq
>  ├── 01_Rawdata
>  ├── 02_QC
>  ├── 03_assembly
>  ├── 04_DE_analysis
>  └── 05_annotation
> ```
>
> Could you generate bash command(s) to create these directories ?

> **Answer**
>
> ```
> mkdir Cpa_RNASeq/{01_Rawdata,02_QC,03_assembly,04_DE_analysis,05_ar
> ```

or

```
mkdir Cpa_RNASeq/01_Rawdata Cpa_RNASeq/02_QC Cpa_RNASeq/03_assembly
```

or

```
cd Cpa_RNASeq
mkdir 01_Rawdata 02_QC 03_assembly 04_DE_analysis 05_annotation
```

Navigating your file system

The file system manages the files and directories of the operating system. It organizes our data into files, which store information, and directories. When you see the prompt below on your terminal's screen, it means that your terminal has processed the command you entered and is ready for the next command.

```
jiratchaya@DESKTOP-P2DD13C:~$
```

`jiratchaya` is username using this terminal. The address @ symbol followed by `DESKTOP-P2DD13C` in a computer or server name. And, the dollar sign `$` is a prompt, which shows us that the shell is waiting for input. Your shell may use a different character as a prompt and may add information before the prompt.

If you want to find out where we are now, type

```
pwd
```

pwd stands for print working directory. Without explicit specification, the computer assumes that we want to execute commands in our current working directory. This can be a user's home directory (~).

If you want to change the directory, e.g. to the 'Cpa_RNASeq' directory we just created, just type the following

```
cd Cpa_RNASeq
```

`cd` stands for "change directory". You can change our working directory by typing `cd` followed by a directory name. In this case you change from the current directory to the directory named 'Cpa_RNASeq'.

Listing directories

We can see what files and subdirectories are in this directory by running ls, which stands for "listing":

```
ls
```

Expected result:

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ ls
01_Rawdata  02_QC  03_assembly
```

Let us look at the other way. This way is to list all the files and directories, including the users who own them, the permissions, and the file size in bytes.

```
ls -l
```

Expected result:

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ ls -l
total 12
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 01_Rawdata
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 02_QC
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 03_assembly
```

List files and folders, permissions and size in a human readable format.

```
ls -lh
```

Expected result:

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ ls -l
total 12
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 01_Rawdata
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 02_QC
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 03_assembly
```

See all hidden files and directories

```
ls -la
```

Expected result:

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ ls -la
total 20
drwxr-xr-x 5 jiratchaya jiratchaya 4096 Mar  1 21:02 .
drwxr-x--- 3 jiratchaya jiratchaya 4096 Mar  1 21:02 ..
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 01_Rawdata
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 02_QC
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:02 03_assembly
```

Files and directories handling

Creating and editing files

When you work on the command line, you often need to create or edit text files. In this workshop, we recommend using `nano` as a text editor. Other Unix text editors you may have heard of include vi, vim, emacs, vscode, and many more.

We'll create the file `test.fasta`. To open an existing file or create a new file, type nano followed by the filename:

```
nano test.fasta
```

This will bring up the text editing screen on your terminal. Here you can type anything you want, but in this case we'll create a sequence like this.

```
>seq_01
TCGCTAGTC

>seq_02
TAGCGAGTT
```

> ☐**Note:** Always leave an enter in the last line. This is advantageous if this file is further used by many programmes.

> The text editing screen is displayed once you have typed nano into some files. At the bottom of the window is a list of the most important keyboard shortcuts for the nano editor. All commands are preceded by either a `^` or an `M` character. The caret symbol (^) stands for the `Ctrl` key. For example, the commands `^J` mean that you press the `Ctrl` and `J` keys simultaneously. The letter `M` stands for the `Alt` key.

To edit a file, you can use the navigation keys such as arrow keys, End, Home, PgUp or PgDn to control the cursor.

To save the changes you made to the file, press `Ctrl+o`. If the file doesn't exist yet, it'll be created after saving.

To exit nano, press `Ctrl+x`. If there are unsaved changes, you'll be asked if you want to save the changes. Nano will ask you 'Save modified buffer?', then type `y` to confirm the edit.

Copying files and directories

To copy files and directories the command `cp` can be used. `cp` stands for copy and is used to copy files and directories in Linux. An example: You copy the file `test.fasta` to `01_Rawdata` with the following syntax

```
cp [source file] [target_directory]/
```

For example

```
cp test.fasta 01_Rawdata/
```

Copy file to another file, using the syntax

```
cp [source_file] [new_file_name]
```

For example

```
cp test.fasta test_2.fasta
```

You can copy a file to a new file in the directory by using the following syntax

```
cp [source_file] [target_directory]/[new_file_name]
```

For example

```
cp test.fasta 01_Rawdata/another_test.fasta
```

To copying directory, use additional flag as follow

```
cp -r [source_directory] [new_directory_name]
```

The flag `-r` stands for recursive, i.e. all files and subdirectories in this directory are copied repeatedly. For example, `01_Rawdata` already contains `test.fasta`, which we copied before, and we want to duplicate this directory.

```
cp -r 01_Rawdata/ 01_Rawdata_new
```

Moving files and directories

To copy files and directories, the command `mv` can be used. `mv` stands for move and is used to move files and directories in Linux. For example, move the file `test_2.fasta` to the directory `01_Rawdata_new` with the following syntax

```
mv [file_to_move] [target_directory]
```

```
mv test_2.fasta 01_Rawdata_new/
```

Specifically, to move files and directories, no flags are required as with `cp`. So if we want to move `01_Rawdata_new` to a subdirectory of `01_Rawdata`, this can be done as follows

```
mv [source_file_or_dir] [target_file_or_dir]
```

```
mv 01_Rawdata_new/ 01_Rawdata
```

Moving file within the directory up to the current directory

```
mv [source_dir]/[source_file] .
```

The dot (.) stands for the current directory, which means you want to move something to the current directory. For example, we want to move the file `another_test.fasta`, which is in the directory `01_Rawdata`, to the current directory by typing

```
mv 01_Rawdata/another_test.fasta .
```

Deleting files and directories

Removing files and directories can be done with the command `rm`. `rm` stands for remove and is used to delete files and directories in Linux. It's simple and straightforward with the following syntax.

```
rm [file_to_delete]
```

For example, you are deleting file `another_test.fasta`

```
rm another_test.fasta
```

To delete directories, use additional flags

```
rm -rf [directory_to_delete]
```

The flag `-r` means that it does something recursive, which means that it deletes all files and subdirectories of the directory you want to delete. The flag `f` can help us delete some protected files and directories that you should think twice before deleting.

For example you want to delete `03_adapter_trimming` directory

```
rm -rf 03_adapter_trimming
```

Or delete subdirectory `01_Rawdata_new` by

```
rm -rf 01_Rawdata/01_Rawdata_new
```

Don't worry~ the `01_Rawdata` is still with us

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ ls -l
total 20
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 22:33 01_Rawdata
drwxr-xr-x 2 jiratchaya jiratchaya 4096 Mar  1 21:59 02_QC
-rw-r--r-- 1 jiratchaya jiratchaya   38 Mar  1 22:00 another_test.fasta
-rw-r--r-- 1 jiratchaya jiratchaya   38 Mar  1 21:59 test.fasta
-rw-r--r-- 1 jiratchaya jiratchaya   38 Mar  1 22:00 test_2.fasta
```

> ☐ **Danger zone:** Be sure to check the path of the location where you want to delete something with the command `rm -rf`, otherwise you'll unintentionally delete necessary files or directories.
>
> 
>
>
> Remove everything with sudo privilege. From meme-arsenal.

Downloading file from URL

There are numerous ways to download a file from a URL via the command line on Linux, and two of the best tools for this task are `wget` and `curl`. Both tools have their advantages and disadvantages, depending on the download task at hand. However, in this workshop we'll mainly focus on downloading with `curl`.

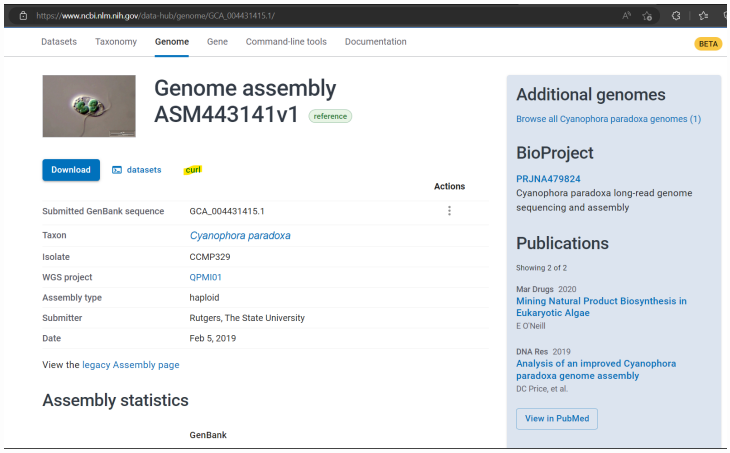
For example, we want to download the latest (draft) genome assembly report of *Cyanophora paradoxa* from the NCBI genome database via `curl` as follows.

```
curl -O https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/004/431/415/GCA_004
```

Expected output

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ curl -O
https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/004/431/415/GCA_004431415.1
t
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 61357  100 61357    0     0  21604      0  0:00:02  0:00:02 --:--:--
21604
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ ls -l
total 80
drwxr-xr-x 2 jiratchaya jiratchaya  4096 Mar  1 22:33 01_Rawdata
drwxr-xr-x 2 jiratchaya jiratchaya  4096 Mar  1 21:59 02_QC
-rw-r--r-- 1 jiratchaya jiratchaya 61357 Mar  1 22:56
GCA_004431415.1_ASM443141v1_assembly_report.txt
-rw-r--r-- 1 jiratchaya jiratchaya    38 Mar  1 22:00 another_test.fasta
-rw-r--r-- 1 jiratchaya jiratchaya    38 Mar  1 21:59 test.fasta
-rw-r--r-- 1 jiratchaya jiratchaya    38 Mar  1 22:00 test_2.fasta
```

Tips: The alternative way to retrieve genome information from NCBI, you can just go to [NCBI Genome Data Hub]() and specify species name to get information. NCBI provides several routes to download files including `curl`!



A genome assembly of *C. paradoxa* in NCBI genome data hub (Accessed: 1 March 2023)

Inspecting file

We'll inspect the assembly report file `GCA_004431415.1_ASM443141v1_assembly_report.txt` that we just downloaded from NCBI

- Count how many lines in that file

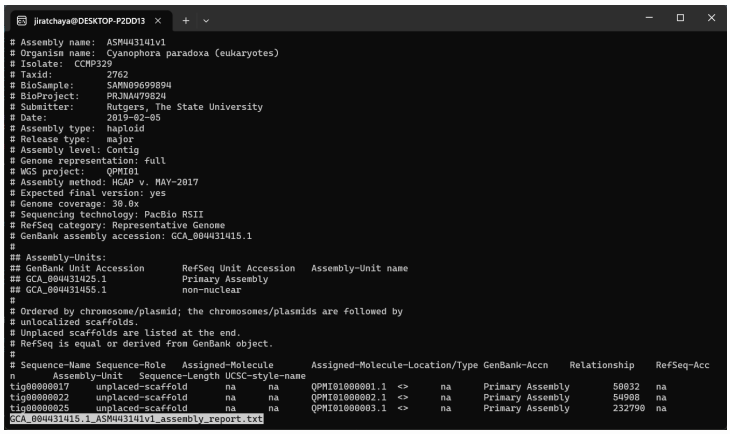
```
wc -l GCA_004431415.1_ASM443141v1_assembly_report.txt
```

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ wc -l
GCA_004431415.1_ASM443141v1_assembly_report.txt
743 GCA_004431415.1_ASM443141v1_assembly_report.txt
```

- Print some contents at a time.

Now you will see the number of lines that fit on your screen, and you can scroll `up` and `down` with the arrow keys. Then press `q` when you have checked your file.

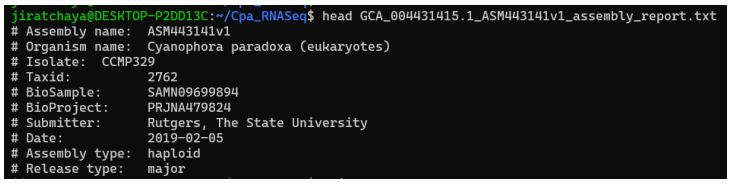
```
less GCA_004431415.1_ASM443141v1_assembly_report.txt
```



Example of inspecting a file with the `less` command. Users can scroll up and down with the arrow keys and exit by pressing `q`.

- Print top 10 lines of file

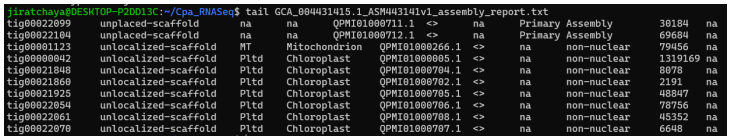
```
head GCA_004431415.1_ASM443141v1_assembly_report.txt
```



The first 10 lines of *C. paradoxa* assembly report file

- Print bottom 10 lines of file

```
tail GCA_004431415.1_ASM443141v1_assembly_report.txt
```

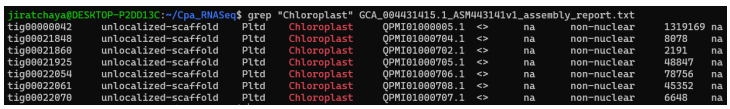


The last 10 lines of *C. paradoxa* assembly report file.

- Print only lines with a specific pattern of word.

For example, we'll print only the lines contain the word "Chloroplast"

```
grep "Chloroplast" GCA_004431415.1_ASM443141v1_assembly_report.txt
```



```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ grep "Chloroplast" GCA_004431415.1_ASM443141v1_assembly_report.txt
tig00000042     unlocalized-scaffold    Pltd    Chloroplast     QPMI01000005.1  <>      na      non-nuclear     1319169 na
tig00021848     unlocalized-scaffold    Pltd    Chloroplast     QPMI01000704.1  <>      na      non-nuclear     8078    na
tig00021860     unlocalized-scaffold    Pltd    Chloroplast     QPMI01000702.1  <>      na      non-nuclear     2191    na
tig00021925     unlocalized-scaffold    Pltd    Chloroplast     QPMI01000705.1  <>      na      non-nuclear     48847   na
tig00022054     unlocalized-scaffold    Pltd    Chloroplast     QPMI01000706.1  <>      na      non-nuclear     78756   na
tig00022061     unlocalized-scaffold    Pltd    Chloroplast     QPMI01000708.1  <>      na      non-nuclear     45352   na
tig00022070     unlocalized-scaffold    Pltd    Chloroplast     QPMI01000707.1  <>      na      non-nuclear     6648    na
```

Extracted lines with a specific word “Chloroplast” in assembly report file.

Show latest commands we used

You can simply press arrow keys `up` or `down` to see your latest commands that you typed in the terminal.

Another way to see the latest command by typing below in the terminal

```
history
```

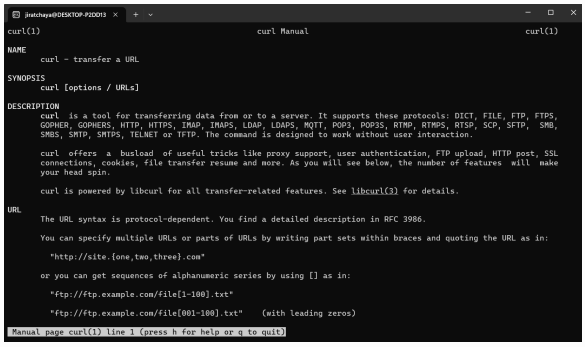
History is able to keep track of the command lines you use, associate any data with each line, and use information from previous lines when writing new lines.

Shortcut: Tab Completion

When typing file or directory names, it’s easy to mistype. Instead, we can use ‘tab’ to complete what we want to type. The shell will try to fill in the rest of a directory or file name if you press tab after typing.

Have no idea what this command can do

Basically, the built-in Linux system commands store usage in their command manual. You can call `man` followed by a command you want to learn more about. for example `man curl`.



```
curl(1)                                curl Manual                                curl(1)

NAME
       curl - transfer a URL

SYNOPSIS
       curl [options / URLs]

DESCRIPTION
       curl  is a tool for transferring data from or to a server. It supports these protocols: DICT, FILE, FTP, FTPS,
       GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP,  SMB,
       SMBS, SMTP, SMTPS, TELNET or TFTP. The command is designed to work without user interaction.

       curl  offers  a  busload  of useful tricks like proxy support, user authentication, FTP upload, HTTP post, SSL
       connections, cookies, file transfer resume and more. As you will see below, the number of features  will  make
       your head spin.

       curl is powered by libcurl for all transfer-related features. See libcurl(3) for details.

URL
       The URL syntax is protocol-dependent. You find a detailed description in RFC 3986.

       You can specify multiple URLs or parts of URLs by writing part sets within braces and quoting the URL as in:

         "http://site.{one,two,three}.com"

       or you can get sequences of alphanumeric series by using [] as in:

         "ftp://ftp.example.com/file[1-100].txt"

         "ftp://ftp.example.com/file[001-100].txt"    (with leading zeros)

 Manual page curl(1) line 1 (press h for help or q to quit)
```

A manual page of `curl`. Users can scroll using arrow keys up and down, and quit reading by press `q`.

Maintaining Long-Running Jobs with tmux

When we run programs through the Unix shell, they run until they terminate successfully or are terminated with an error. Multiple processes running simultaneously on your computer, such as system files, web browser, email application, bioinformatics programs, and so on. In bioinformatics, we often work

with processes that run for a long period of time. Therefore, it's important that we know how to work with processes and manage them using the Unix shell. In this section, we'll learn the basics of dealing with processes.

In addition, processes are also terminated if the connection to the servers is interrupted, the network connection drops immediately, or the power fails. Since we're constantly working with remote computers in our daily work in bioinformatics, we need a way to prevent the accidental termination of long-running applications. Leaving the local terminal's connection to a remote computer open while a program is running is an unsafe solution, even the most reliable networks can experience short outages.

> How tmux increase you pruductivity :/ (From Billy uses tmux in Reddit)

Some software offers the user the possibility to run their work as a background process, e.g. Nohup, Screen and Tmux. In this workshop, we propose **Terminal Multiplexer (Tmux)**, which allows you to create a session with multiple windows, each of which can run its own processes. The Tmux sessions are persistent, which means that all the windows and their processes can be easily restored by reattaching the session.

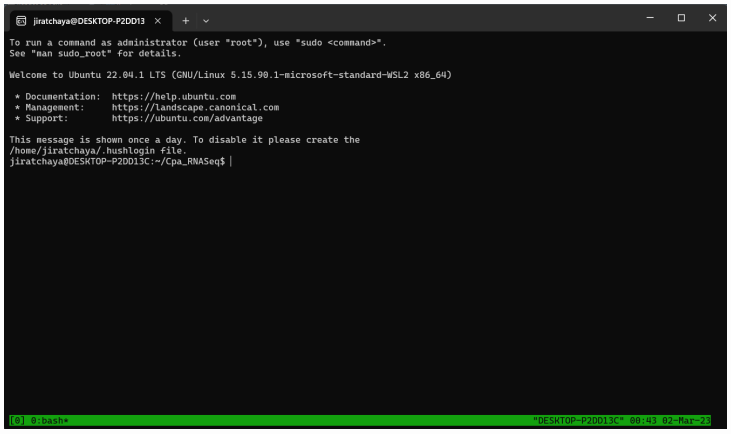
When Tmux is running on a remote machine, you can maintain a persistent session that isn't lost when the connection drops or you close your terminal window to go home (or even exit your terminal programme). Rather, all Tmux sessions can be reattached to the terminal you're currently at - simply log back into the remote machine via SSH and reattach the Tmux session. All windows remain undisturbed and all processes continue to run.

A simple usage of Tmux

Open a terminal and use the following command

```
tmux
```

You see a command prompt as usual, but you now see a taskbar-style menu at the bottom of the terminal that contains something like bash 0 *. The asterisk indicates that this is your active window.



Tmux windows

Detach a session

This allows you to leave the tmux session, but it continues to run in the background. Just press the key

```
[ctrl + b] + d
```

Your terminal will print

```
jiratchaya@DESKTOP-P2DD13C:~/Cpa_RNASeq$ tmux
[detached (from session 0)]
```

This should take you back to a standard prompt. Remember that the Tmux session continues in the background, and you can recall it at any time.

Name the Tmux session

You may find it helpful to name your sessions with meaningful titles to keep things organized. Let's try naming your first session with Tmux.

You can name it anything that we want, but in this case I will name it 'process2'. Enter the following command:

```
tmux new -s process2
```

You should now have a new Tmux session running. If you look in the lower left area of the window, you will see the name of your session rather than the generic 'bash'.

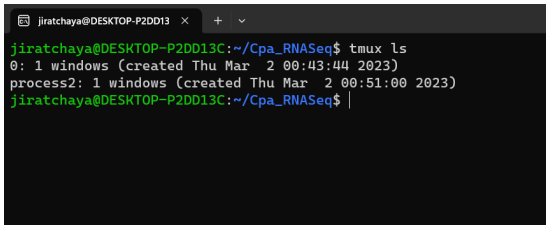
List tmux sessions

What happened to your session? It is still running in the background. You can reopen the session by name or number ID, but what if you forgot the session name?

There is a list function built into tmux:

```
tmux ls
```

This lists all your current tmux sessions. When you run it, you get output like this:



List of running tmux sessions.

Reenter (aka reattach) a session in Tmux

To reopen your tmux session, you can use the tmux command with the attach or attach-session option as follows:

```
tmux a -t [session_name]
```

For example, we'll reenter to the process2 session.

```
tmux a -t process2
```

Exit tmux when finish running

Quitting tmux is exactly the same as quitting the standard terminal by pressing the keys `Ctrl+d` or by entering

```
exit
```

Resources

- Buffalo, V. (2015). *Bioinformatics data skills: Reproducible and robust research with open source tools*. " O'Reilly Media, Inc.".
- Introduction to the command line interface by Harvard Chan Bioinformatics Core (Accessed on 27 Feb 2023).
- Introducing the Shell, from the course Introduction to the Command Line for Genomics in bioinformatics-core-shared-training (Accessed on 28 Feb 2023)
- Bash cheat sheet from RehanSaeed GitHub repository (Accessed on 1 March 2023).
- Getting Started with Tmux [Beginner's Guide]. By linuxhandbook.com (Accessed on 2 March 2023)

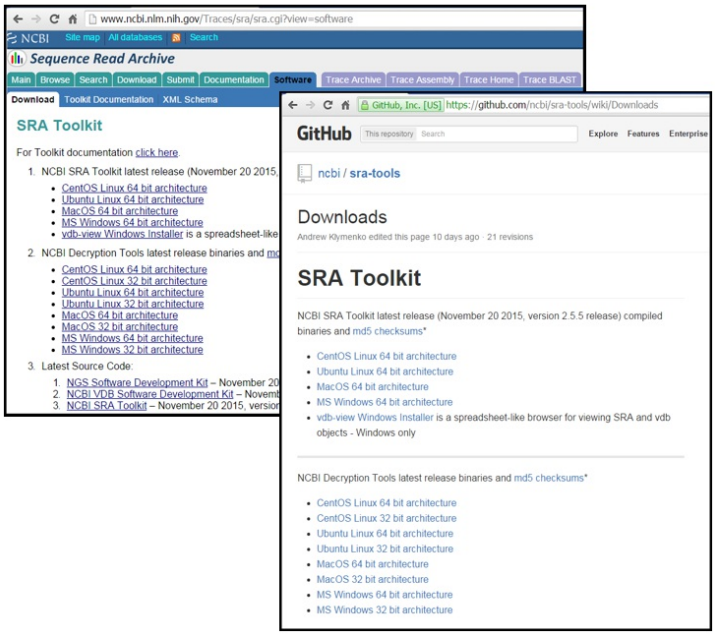
Data Retrieval with NCBI SRA Toolkit

NCBI (Natianal Center for Biotechnology Information) is a major source of biological databases related to life and health sciences research, as well as a major source of bioinformatics tools and services. NCBI hosts various types of biological data submitted by researchers from around the world, such as GenBank for nucleotide sequence submissions, Sequence Read Archive (SRA) for raw sequence data, Genome for submitting full or draft genomes, Gene Expression Omnibus (GEO) for quantitative gene expression data sets, and many more.

NCBI SRA toolkit is a set of utilities for downloading, viewing, and searching large amounts of high-throughput sequencing data from the NCBI SRA database.

SRA toolkit can

- Effectively download the large volume of high-throughput sequencing data
- Convert SRA file into other biological file format
- Retrieve a small subset of large files
- Search within SRA files and fetch specific sequences



Screenshots of NCBI Sequence Read Archives

What is Sequence Read Archives (SRA)

The Sequence Read Archive (SRA) is the largest publicly accessible repository for high-throughput sequencing data. SRA accepts data from all areas of sequencing projects as well as metagenomics and environmental studies. Sequencing data may be isolated from a single species or from multiple species as in metagnomics studies.

SRA also refers in the file description to the format defined by NCBI for NGS data in the SRA database. All data submitted to NCBI must be stored in SRA format and can be converted back to a FASTQ, FASTA, or BAM file depending on the original submission by the researchers. Here, the SRA Toolkit provides tools for downloading data, converting various data formats to SRA format and vice versa,

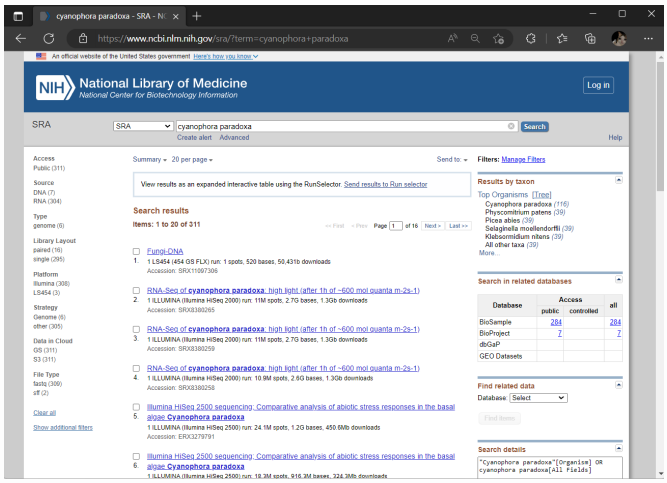
and extracting SRA data to other formats.

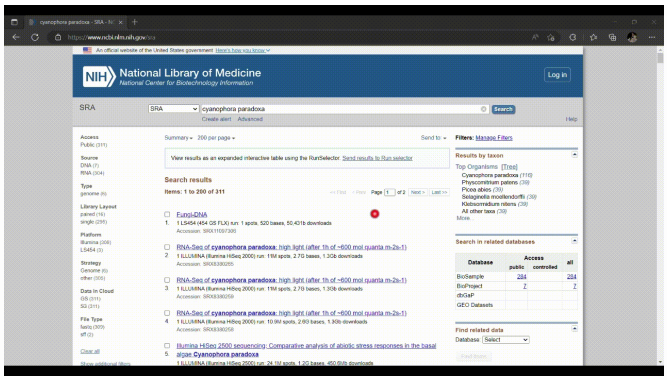
Researchers often use SRA data to make discoveries and conduct reproducible research. Data sets can be compared using the SRA web interface. However, if you want to download files for local use on your computer, you should use a command line interface, and the SRA Toolkit is highly recommended by NCBI.

Searching RNA-Sequencing datasets in NCBI

The databases in NCBI are linked by some common features. This means that you can start wherever you have your research problems in NCBI. In this workshop, we will investigate transcriptional changes during light exposure of the alga *Cyanophora paradoxa*, a representative species of Glaucophytes. For more information about this alga, please see [this article in Science](#).

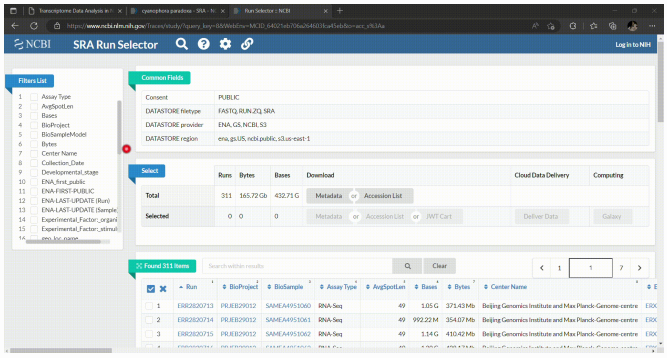
In this workshop we'll retrieve transcriptome sequencing libraries of C. paradoxa under the normal light and dark conditions. This dataset is generated by (Knopp et al. 2020).

> **Activity**
>
> You can easily search the SRA database for any keywords of interest related to your research. In this context, we search for all SRA studies related to *C. paradoxa* and see what SRA provides us. Note that the SRA database contains not only transcriptome studies, but also genomes and metagenomes.
>
> 1. Go to SRA database: https://www.ncbi.nlm.nih.gov/sra, and search for 'cyanophora paradoxa'.
>
> 
>
>
> Screenshot of the search result of *C. paradoxa* in the database NCBI SRA. Here you can see all sequencing libraries of this species that have been submitted to NCBI. You can specify which items are of interest or customize the search using the filter box on the left and right sides of the screen.
>
> 2. We'll adjust our selection using the tool in the SRA database, the ***SRA Run Selector***, as follows.



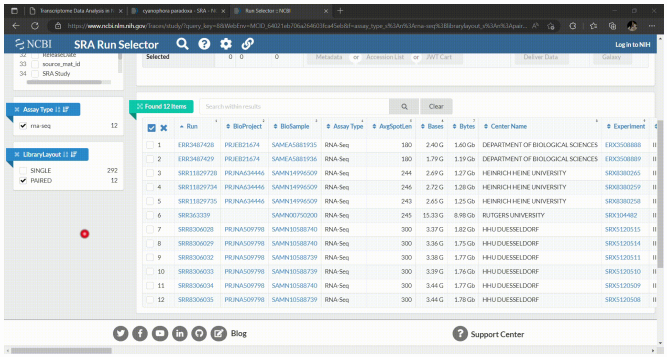
Browsing sequencing data in NCBI SRA database

3. In the SRA Run Selector, you can customize the filters based on the metadata columns of all runs. In this case, we filter the SRA runs based on the assay type as RNA-Seq and select only paired-end sequencing data as follows.



Customizing filters in SRA Run Selector

4. Then you can select which runs you want to download and perform analysis. In this workshop we'll select *C. paradoxa* RNA-Seq reads from **SRR8306028**, **SRR8306029**, **SRR8306032**, **SRR8306033**, **SRR8306034** and **SRR8306035**.



Exporting the selected metadata in SRA Run Selector.

5. The downloaded metadata is in comma-separated file format. So you can open them with spreadsheet programs like Microsoft Excel on your local laptop. The metadata looks like this.

Downloading SRA runs using *fasterq-dump*

`fasterq-dump` are tools in the SRA toolkit used to connect from our remote server to the NCBI server and download sequencing data from SRA.

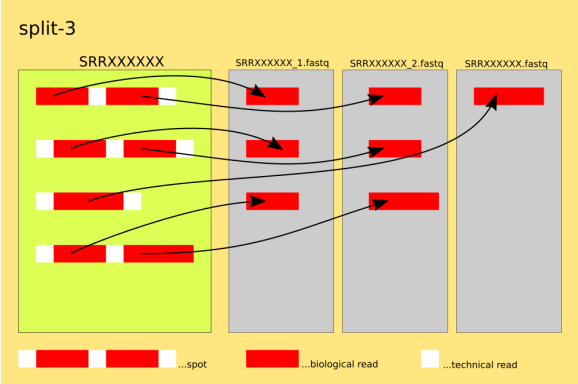
According to the NCBI sra-tools' guideline, using `fasterq-dump` in combination with another tool, `prefetch`, is the better way to download data because `prefetch` can be invoked at any time if the previous download accidentally failed. So it is not necessary to start the download from the beginning.

However, `prefetch` can sometimes be skipped if you want to download a small amount of data. In this workshop we'll use only `fasterq-dump` to download and process SRA file format to FASTQ file for further analysis.

> **Activity**
>
> We'll do the following command in Terminal or MobaXterm, by access to the username and password that we've provided.
>
> - For mobaXterm, enter to your session.
> - For terminal, type
>
> ```
> ssh <username>@<server IP address>
> ```
>
> Now we download RNA-Seq libraries from *C. paradoxa* using the SRA accessions listed in the first column of the metadata above using the following command.
>
> Activate analysis environment
>
> ```
> conda activate ncbi
> ```
>
> Go to working directory
>
> ```
> cd ~/Cpa_RNASeq/01_Rawdata
> ```
>
> Run fasterq-dump
>
> ```
> fasterq-dump --threads 2 --progress \
> SRR8306028 SRR8306029 SRR8306032 \
> SRR8306033 SRR8306034 SRR8306035
> ```
>
> From the `fasterq-dump` command,
>
> `--threads` refer to how many threads to use (default = 6).
>
> `--progress` force the terminal to print the progress of downloading and processing file to the screen.
>
> Expected output files.

```
01_Rawdata
├── SRR8306028_1.fastq
├── SRR8306028_2.fastq
├── SRR8306029_1.fastq
├── SRR8306029_2.fastq
├── SRR8306032_1.fastq
├── SRR8306032_2.fastq
├── SRR8306033_1.fastq
├── SRR8306033_2.fastq
├── SRR8306034_1.fastq
├── SRR8306034_2.fastq
├── SRR8306035_1.fastq
└── SRR8306035_2.fastq
```

By default, `fasterq-dump` processes a single SRA file format of paired-end reads by splitting reads into forward (`*_1.fastq`) and reverse (`*_2.fastq`), if singletons (unpaired between forward and reverse reads) present, it will be written to another fastq file as described in this figure.



Sequence read processing by fasterq-dump using default parameters. Figure adopted from https://github.com/ncbi/sra-tools/wiki/HowTo:-fasterq-dump.

Reference Sources

- Price, Dana C., et al. "*Cyanophora paradoxa* genome elucidates origin of photosynthesis in algae and plants." *Science* 335.6070 (2012): 843-847. https://doi.org/10.1126/science.1213561.
- SRA Toolkit: the SRA database at your fingertips from NCBI Insights. Accessed 4-Mar-2023.
- How to use NCBI SRA Toolkit effectively by Renesh Bedre, Data science blog. Accessed 4-Mar-2023.
- HowTo: fasterq dump by NCBI sra-tools GitHub Wiki. Accessed 4-Mar-2023.

RNA-Seq Data Quality Control

What is FASTQ file format

Next-generation sequencing and data analysis projects typically begin with the processing of sequence read data and their quality tags from the sequencer in FASTQ format. The FASTQ format is the most commonly used format in sequence analysis and is generated by a sequencer. The FASTQ file contains the sequence data from the clusters that pass the filter of a flow cell. Many analysis tools require this format because it contains much more information than FASTA. In this workshop, we will mainly explain the FASTQ file format, which comes from the Illumina sequencer.

The FASTQ format is similar to the fasta format, but differs in syntax and in the integration of quality values. Each sequence requires at least 4 lines:

```
@SRR8306028.1 K00137:70:H3LGMBBXX:2:1101:3340:2195 length=150
GCCCTGCTTGCTGATGAGGGACTTGTGGATGTGCGGAATGACACCACCACGAGCATGGTNGCCTNGANNAGGGNNTCNAGCTCNNCATCACCACNAATNGCGAGCTGAAGATGCCGGGGAGTGATANGCTTCACCTTCAGATCTTTCGG
+SRR8306028.1 K00137:70:H3LGMBBXX:2:1101:3340:2195 length=150
AAFFFFAKKKKKKKKKFKAKKKFFFKKK<FKFAKFFKKKFKKKKFAKKFKKFKKK<AAFKAFKBFA,F#<F#BK,,A##7A8AAA,<B#<FKKKKFF,#AA78#<FKKK7<<<<AFAKA,<A<A,7A,AAK<B<<,,AA7A,,,,,A,,,,,,<,
```

Example of FASTQ file format

1. A sequence identifier with information about the sequencing run and the cluster. The exact contents of this line vary by based on the BCL to FASTQ conversion software used.
2. The sequence (the base calls; A, C, T, G and N).
3. A separator, which is simply a plus (+) sign.
4. The base call quality scores. These are Phred +33 encoded, using ASCII characters to represent the numerical quality scores.

The FastQ sequence descriptor generally follows a specific format that includes all information about the sequencer and its position on the flow cell. The sequence descriptor also follows a specific format and contains information about the sample information.

FASTQ sequence descriptor, particulary in Illumina sequence reads look like:

```
@HWUSI-EAS100R:6:73:941:1973#0/1
```

where

HWUSI-EAS100R	The unique instrument name
6	Flowcell lane
73	Tile number within the flowcell lane
941	‘x’-coordinate of the cluster within the tile
1973	‘y’-coordinate of the cluster within the tile
#0	Index number for a multiplexed sample (0 for no indexing)
/1	The member of a pair, /1 or /2 (paired-end or mate-pair reads only)

As mentioned earlier, line 4 contains the quality score of the nucleotide at the same position. The quality scores are represented by the code ASCII, which indicates how confident of the correctly called base is.

We can calculate the quality score of a base,if P is the error probability, then:

$$Q = -10log_{10}(P)$$

The following figure shows the representative ASCII code for the score value. Base quality scoring for analysis is important when identifying types of genomic variation such as SNPs, but it is also an indicator of the overall quality of the sequencing as well.

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			

ASCII_BASE=64 Old Illumina

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	64 @	11	0.07943	75 K	22	0.00631	86 V	33	0.00050	97 a
1	0.79433	65 A	12	0.06310	76 L	23	0.00501	87 W	34	0.00040	98 b
2	0.63096	66 B	13	0.05012	77 M	24	0.00398	88 X	35	0.00032	99 c
3	0.50119	67 C	14	0.03981	78 N	25	0.00316	89 Y	36	0.00025	100 d
4	0.39811	68 D	15	0.03162	79 O	26	0.00251	90 Z	37	0.00020	101 e
5	0.31623	69 E	16	0.02512	80 P	27	0.00200	91 [	38	0.00016	102 f
6	0.25119	70 F	17	0.01995	81 Q	28	0.00158	92 \	39	0.00013	103 g
7	0.19953	71 G	18	0.01585	82 R	29	0.00126	93]	40	0.00010	104 h
8	0.15849	72 H	19	0.01259	83 S	30	0.00100	94 ^	41	0.00008	105 i
9	0.12589	73 I	20	0.01000	84 T	31	0.00079	95 _	42	0.00006	106 j
10	0.10000	74 J	21	0.00794	85 U	32	0.00063	96 `			

Tables converting between integer Q scores, ASCII characters and error probabilities. Figure adopted from https://www.drive5.com/usearch/manual/quality_score.html

What software use FASTQ

To date, Almost NGS analysis software requires FASTQ format. For example:

- QC such as fastQC used FASTQ to determine how good of the sequence read library, generate an informative report, and also determining the presence of adapter sequences which can also be trimmed by some integrated QC tools such as FASTP.
- Aligners such as bowtie2, BWA, STAR, and so on, use reads, and quality sometimes, to align to the reference sequence. The mapping information can be further used for quantifying expression, constructing sequence assembly, and variant calling.
- De novo assembly tools, for example Trinity, Spades, Velvet, etc., also use FASTQ to construct contig library and scaffolding. Some de novo assembler tools not only use FASTQ to contruct draft assembly but also used in the polishing process to refine assembly, such as Flye, Unicycler, Canu, etc.

Quality assessment using FastQC

FastQC is designed for quality control of raw sequence data from high-throughput sequencing technology. It provides a modular set of analyses that you can use to get a quick overview of the quantity and quality of your data, and to help you decide on the raw data whether you should perform adapter or low-quality read trimming or whether you can perform further analyses. For sequence reads that require adapter trimming before further analysis, we recommended to

assessing the quality both before and after trimming.

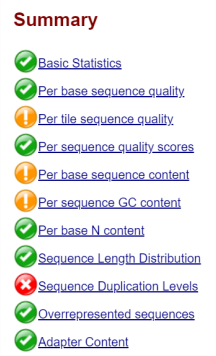
Most sequencers will generate a QC report as part of their analysis pipeline, but this is usually only focused on identifying problems which were generated by the sequencer itself. FastQC aims to provide a QC report which can spot problems which originate either in the sequencer or in the starting library material.

> **Activity**
>
> The following will perform on you user account by activating your working environment at first.
>
> ```
> conda activate qc
> ```
>
> Then, create a directory for QC result before adapter trimming
>
> ```
> mkdir 02_QC/fastQC_before_trim
> ```
>
> Run FastQC all file at once. Here, we'll use a wildcard `*.fastq` to select all FASTQ files in 01_Rawdata directory. We also specify number of CPU threads in `--threads` and QC output files in 02_QC/fastQC_before_trim using `--outdir` argument.
>
> ```
> fastqc --outdir 02_QC/fastQC_before_trim \
> --threads 2 \
> /opt/Cpa_RNASeq/01_Rawdata/*.fastq
> ```
>
> Estimated time: ~10min

Interpreting FastQC results

FastQC also provided excellent explanation of each analysis step in their documentation. So we encouraged you to learn more at their web page along with the documentation.

The analysis in FastQC is performed by a series of analysis modules. The left hand side of the main interactive display or the top of the HTML report show a summary of the modules which were run, and a quick evaluation of whether the results of the module seem entirely normal (green tick), slightly abnormal (orange triangle) or very unusual (red cross).



FastQC sidebar

Basic statistics

The Basic Statistics module generates some simple composition statistics for the file analysed.

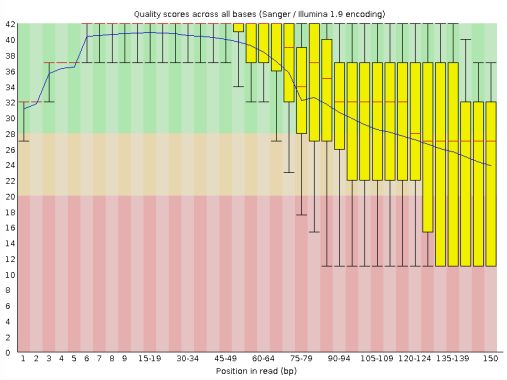
Measure	Value
Filename	SRR8306028_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	11244423
Total Bases	1.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	62

Basic statistics of SRR8306028.

- **Filename:** The original filename of the file which was analysed
- **File type:** Says whether the file appeared to contain actual base calls or colorspace data which had to be converted to base calls
- **Encoding:** Says which ASCII encoding of quality values was found in this file.
- **Total Sequences:** A count of the total number of sequences processed. There are two values reported, actual and estimated. At the moment these will always be the same. In the future it may be possible to analyse just a subset of sequences and estimate the total number, to speed up the analysis, but since we have found that problematic sequences are not evenly distributed through a file we have disabled this for now.
- **Sequence Length:** Provides the length of the shortest and longest sequence in the set. If all sequences are the same length only one value is reported. %GC: The overall %GC of all bases in all sequences

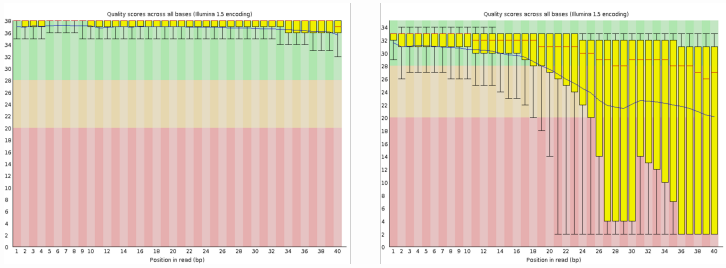
Per Base Sequence Quality

The Per base sequence quality plot shows an overview of the range of quality values across all bases at each position in the FastQ file.



Per Base Sequence Quality plot of SRR8306028. In which the central red line is the median value. The yellow box represents the inter-quartile range (25-75%). The upper and lower whiskers represent the 10% and 90% points. The blue line represents the mean quality.

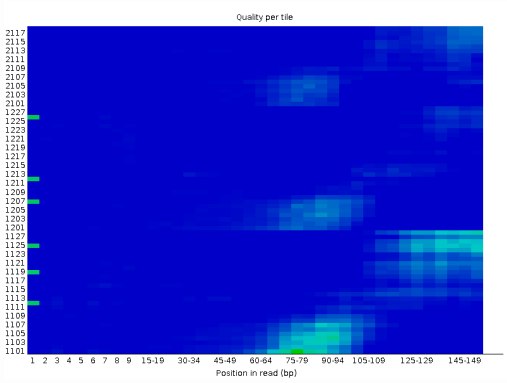
The higher the score, the better the base call, i.e., the box plots fall into the very good quality area (green background), the mediocre quality area (orange background), and the poor quality area (red background). The following figures show a comparison of the good and poor quality results of Illumina sequencing technology.



A comparison of good (left) and bad (right) per base sequence quality plots. Figures adopted from example reports in https://www.bioinformatics.babraham.ac.uk/projects/fastqc.

Per tile sequence quality

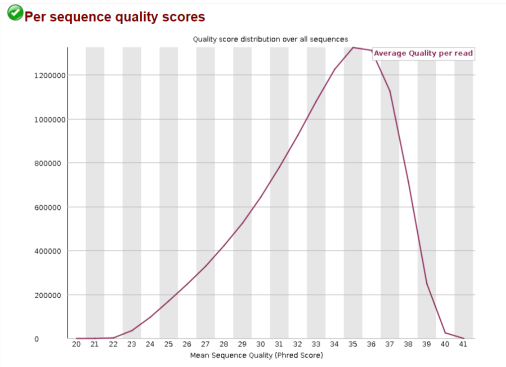
This plot is specific to Illumina sequencing libraries and shows colour shading of quality score by position on the flow cell. The colours are on a scale from cold to hot, with cold colours representing positions where the quality was at or above average for that base in the run, and hotter colours indicating that a tile had worse qualities than other tiles for that base. In the example below, you can see that certain tiles have consistently poor quality. A good chart should be blue throughout.



Per tile sequence quality plot of SRR8306028.

Per Sequence Quality Scores

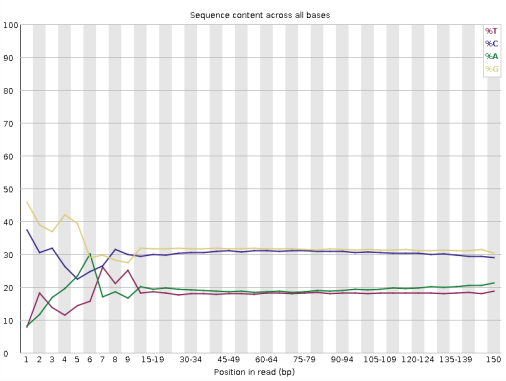
The per sequence quality score report allows you to see if a subset of your sequences have universally low quality values. It is often the case that a subset of sequences will have universally poor quality, often because they are poorly imaged (on the edge of the field of view etc), however these should represent only a small percentage of the total sequences.



Per Sequence Quality Scores plot of SRR8306028.

Per Base Sequence Content

Per Base Sequence Content plots out the proportion of each base position in a file for which each of the four normal DNA bases has been called. The plot shows the quality of nucleotide A T C and G separately into 4 lines.



Per Base Sequence Content plot of SRR8306028.

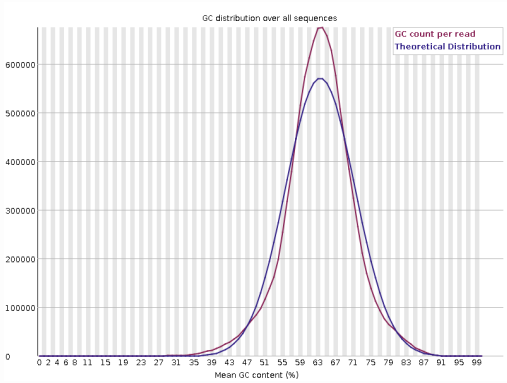
Usually ambiguous base values are found at the beginning of the read. Libraries made with random hexamer primers, with almost all RNA-Seq libraries using them, and those that were fragmented libraries. This bias does not affect an absolute sequence, but provides enrichment of a number of different K-mers at the 5' end of the reads. While this is a true technical bias, it cannot be corrected by trimming and does not appear to affect downstream analysis in most cases. However, a warning or error is generated in this module. This module issues a warning if the difference between A and T, or G and C is greater than 10% in any position.

Per Sequence GC Content

This module measures the GC content over the entire length of each sequence in a file and compares it to a normal distribution of GC content. Normally, one would expect an approximately normal distribution of GC content, where the central peak corresponds to the total GC content of the underlying genome of interest.

The skewness of the distribution may indicate some unusual events such as

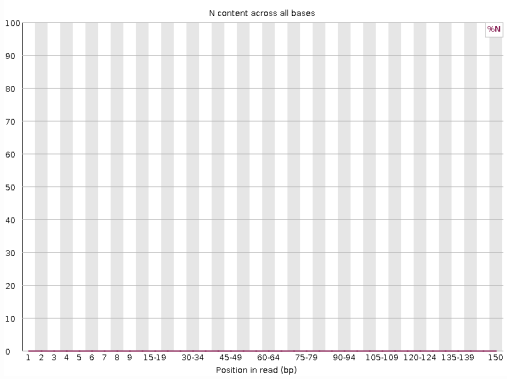
contamination or systematic bias in your sequencing library. However, the GC content signature of different organisms may depend on their nature.



Per Sequence GC Content plot of SRR8306028.

Per base N content

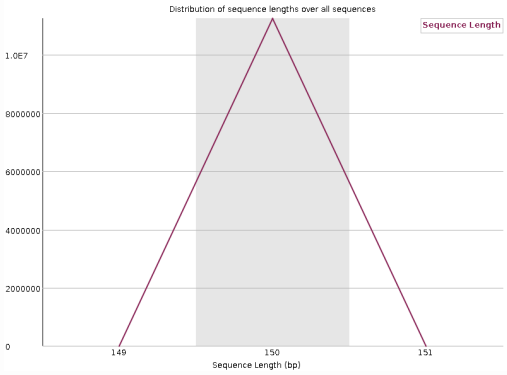
This module represents the percentage of base calls at each position for which an N was called. The 'N' base is found when the sequencer is not able to make a confident base call, then it will normally substitute an N.



Per base N content plot of SRR8306028.

Sequence Length Distribution

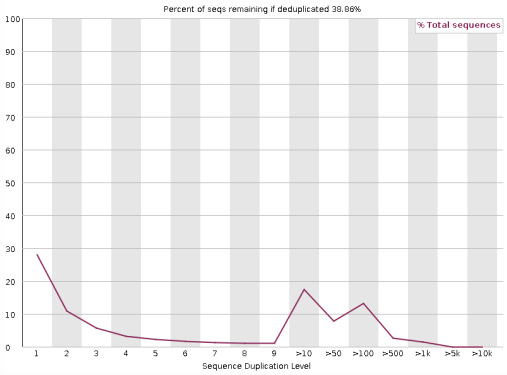
This module generates a histogram of distribution of sequence reads in the file which was analyzed.



Sequence Length Distribution plot of SRR8306028.

Sequence Duplication Levels

This module counts the degree of duplication for every sequence in a library and creates a plot showing the relative number of sequences with different degrees of duplication. A low level of duplication may indicate a very high level of coverage of the target sequence, but a high level of duplication is more likely to indicate some kind of enrichment bias (eg PCR over amplification).



Sequence Duplication Levels plot of SRR8306028.

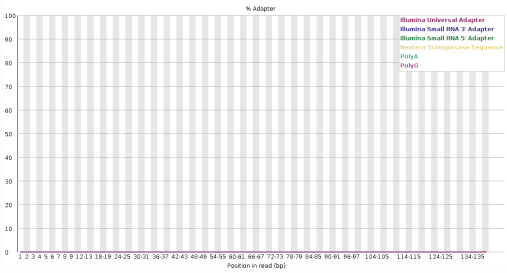
Overrepresented sequences

This module lists all sequences that make up more than 0.1% of the first 100,000 sequences examined. For each overrepresented sequence, the program searches for matches in a database of common impurities and reports the best match found. However, finding a hit doesn't mean that this is the source of the contamination, but may point you in the right direction.

Adapter Content

This plot shows the cumulative percentage of adapter sequences used for sequencing this library at each position. Most adapter sequences found in Illumina RNA-Seq libraries are Illumina Universal Adapters. This module issues a warning if a sequence is present in more than 5% of all reads. This module issues a warning if a sequence is present in more than 10% of all reads. If the adapter sequence is present in more than 1% of the sequence library, adapter

trimming is considered.



Adapter Content of SRR8306028.

> **Activity**
>
> To further combine all the QC results into a single interactive HTML file, we'd suggested to use `multiqc` software to combine it.
>
> ```
> conda activate qc
> ```
>
> Then, run multiqc
>
> ```
> multiqc --filename QCreport_before_trim \
> --outdir 02_QC/ \
> --dirs 02_QC/fastQC_before_trim/
> ```
>
> Estimated time: < 1 min
>
> Output files:
>
> ```
> - QCreport_before_trim_data
>  ├── multiqc_citations.txt
>  ├── multiqc_data.json
>  ├── multiqc_fastqc.txt
>  ├── multiqc_general_stats.txt
>  ├── multiqc.log
>  └── multiqc_sources.txt
> - QCreport_before_trim.html
> ```

Adapter Trimming with Cutadapt

Cutadapt is a tool to remove sequencing adapters, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads. Cutadapt supports both FASTQ and FASTA file format for trimming.

Several types of sequencing adapters have been used nowaday. We have to know which adapter found in our sequencing library. Fortunately, Illumina provide a manual of [Illumina Adapter Sequences]() that used in different types of sequencing. As mentioned, most of RNA-Seq library sequenced by Illumina used Illumina TruSeq Single Indexes, which is AGATCGGAAGAGCACACGTCTGAACTCCAGTCA and AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT flanked at the 5' end of forward reads and 3' end of reverse reads, respectively.

Fortunately, the dataset that will be used is free of apadter sequences examined from the [adapter content]() from FastQC result. ***So this command will just***

show as a demo for your future project.

An example command of Cutadapt as follow.

```
cutadapt --cores 2 \
-u 10 -U 10 \
-a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA \
-A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
-o <output_forward.fastq> \
-p <output_reverse.fastq> \
<input_forward.fastq> <input_reverse.fastq>
```

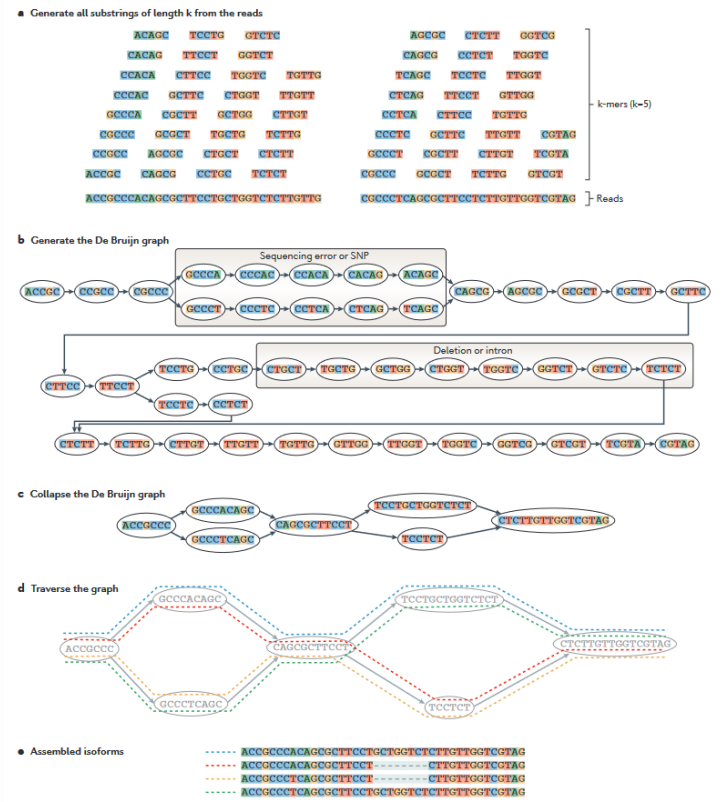
According to the command, we specify number of CPU threads in `--cores`. We remove 10 bases directly from each end of read, `-u` for forward and `-U` for reverse reads. Then we specify the adapter sequences as mentioned above in `a` and `A`. And the paths for forward and reverse reads output files in `-o` and `-p`, respectively.

Reference Sources

- Quality Control of FASTQ files from Harvard Chan Bioinformatics Core (HBC) training (Accessed on 1 Mar 2023).
- FastQC official website from Babraham Bioinformatics (Accessed on 1 Mar 2023).
- FastQC Documentation from Babraham Bioinformatics (Accessed on 1 Mar 2023).
- Cutadapt 4.2 Documentation (Accessed on 1 Mar 2023).
- Illumina Adapter Sequences (Accessed on 1 Mar 2023).

De novo Assembly with Trinity

Trinity is a promising tool for de novo full-length transcriptome assembly that continually developed since 2011. Trinity assembles reads by constructs many individual de Bruijn graphs, each representing the transcriptional complexity at a given gene or locus, that originated from the different nucleotide in the same position, and then processes each graph independently to extract full-length splicing isoforms and to tease apart transcripts derived from paralogous genes (Grabherr et al. 2011; Haas et al. 2013). Each assembled contig is will refer to a transcript.



Overview of the concept of de novo transcriptome assembly. Clean reads are divided into k-mers, i.e., in this figure k = 5, which means that a read is divided into many fragments, each fragment containing 5 bases. Then, de Bruijn graphs are stitched from a pool of billions of k-mers (b). During sequencing, read fragments originating from the same spot or derived from the same gene may even have a nucleotide change at the same position, which may be either true polymorphisms or sequencing errors, so that similar k-mer sequences are joined together and routing to adjacent k-mers (c). The bulges in the graphs represent variations within the graph complex. Each graph complex represents a gene that can split into many transcript isoforms during traversion (d) to eventually obtain the assembled transcript library (e). For more details, please see Martin and Wang (2011).

Trinity can construct genomes without genome information and enables transcript construction in non-model organisms where genome assembly is not yet available, or that do not achieve successful chromosome-level or full assembly. Downstream processes, such as transcript assembly completeness analysis, transcript abundance estimation, and identification of differentially expressed genes, can also be performed with Trinity and its built-in utilities commands.

For de novo assembly in fast and efficient way for limited computational resources available, we prepared the downsized reads that derived from the SRA accessions that we already retrieved from NCBI SRA database from the previous chapter.

During de novo assembly, the longest and the heaviest computation resource

required for constructing and stitching billions of de Bruijn graphs. Very deep sequencing libraries may failed of these processes. Therefore, normalizing or downsizing sequence reads before de novo assembly is efficient way to proceed it. In brief, we downsized sequence reads using built-in Trinity command `insilico_read_normalization.pl` as follow. *This just inform all participant to the source of raw data that they will perform assembly.*

```
insilico_read_normalization.pl \
--seqType fq \
--JM 100G \
--max_cov 10 \
--left 01_Rawdata/*_1.fastq \
--right 01_Rawdata/*_2.fastq \
--pairs_together \
--CPU 50 \
--output 03_assembly
```

Trinity `insilico_read_normalization.pl` uses forward and reverse reads input from `--left` and `--right` parameters by reduce the maximum coverage depth (`--max_cov`) observed to 10x, and retain only paired reads in `--pairs_together`.

The output file given as follow:

```
-rw-r--r--  1 jiratchaya jiratchaya 1005390788 Mar  8 10:34 left.fq
-rw-r--r--  1 jiratchaya jiratchaya 1005390788 Mar  8 10:34 right.fq
```

Running Trinity

Trinity is run via the script `Trinity`:

```
Trinity --seqType fq \
--max_memory 6G \
--CPU 2 \
--left /opt/Cpa_RNASeq/insilico_norm_reads/left.fq \
--right /opt/Cpa_RNASeq/insilico_norm_reads/right.fq \
--output Trinity_2023-03-08
```

Estimated time usage: ~24 hr

By this command, Trinity take the input forward and reverse reads from `--left` and `--right`, respectively. By default, de novo assembly with Trinity will perform in silico read normalization by itself. Since we have already normalized the sequence reads file prior to assembly, this step will skip the normalization step by add `--no_normalize_reads` to the command. All other arguments will use with the default parameters.

The given results in `~/Cpa_RNASeq/03_assembly` directory are:

```
-rw-r--r--  1 jiratchaya jiratchaya  77M Mar  8 11:34 Trinity_2023-03-08.Trinity.fasta
-rw-r--r--  1 jiratchaya jiratchaya 3.0M Mar  8 11:34 Trinity_2023-03-08.Trinity.fasta.gene_trans_map
```

- `*.Trinity.fasta` is the assembled transcript files

- `*.Trinity.fasta.gene_trans_map` is tab-separated file of trinit genes (left column) and the belonging transcript (right column)

```
jiratchaya@pslab1:~/Cpa_RNASeq/03_assembly$ head Trinity_2023-03-08.Trinity.fasta.gene_trans_map
TRINITY_DN43810_c0_g1   TRINITY_DN43810_c0_g1_i1
TRINITY_DN43890_c0_g1   TRINITY_DN43890_c0_g1_i1
TRINITY_DN43840_c0_g1   TRINITY_DN43840_c0_g1_i1
TRINITY_DN43847_c0_g1   TRINITY_DN43847_c0_g1_i1
TRINITY_DN43815_c0_g1   TRINITY_DN43815_c0_g1_i1
TRINITY_DN43872_c0_g1   TRINITY_DN43872_c0_g1_i1
TRINITY_DN43845_c0_g1   TRINITY_DN43845_c0_g1_i1
TRINITY_DN43867_c0_g1   TRINITY_DN43867_c0_g1_i1
TRINITY_DN43843_c0_g1   TRINITY_DN43843_c0_g1_i1
TRINITY_DN43844_c0_g1   TRINITY_DN43844_c0_g1_i1
```

> **Remarks**
>
> The de novo assembly part may take hours to days to proceed since this process in one of the time and resource-consuming in transcriptome data analysis. So that's OK if we couldn't accomplish the assembly as of the limited time for this workshop.
>
> We have prepared the assembled transcripts for further analyses. You can use the following command to copy assembly files to your working directory.
>
> ```
> # your current working directory: ~/Cpa_RNASeq/
> cp /opt/Cpa_RNASeq/denovo_assembly/Trinity_2023-03-08.Trinity.fasta
> ```

Transcript Assembly Quality Assessment

According to suggestions from Trinity wiki of [transcriptome assembly quality assessment](), it's worth to determine how good of the quality of assembled transcript. Several approaches available for characterize the quality of your assembly. However, in this workshop we'll perform only two approaches

Examining gene and contig Nx statistics

We can compute Nx statistics from the assembled transcripts, as well as GC content, number of assembled transcripts, mean and median of contig length. from `TrinityStats.pl` command.

> **Activity**
>
> The following script in the Trinity toolkit will compute these values for you like so:

```
# Go to assembly directory
cd ~/Cpa_RNASeq/03_assembly

# Run TrinityStats
TrinityStats.pl Trinity_2023-03-08.Trinity.fasta
```

Expected time used: < 1 min

Expected output from the terminal screen:

```
################################
## Counts of transcripts, etc.
################################
Total trinity 'genes':  49389
Total trinity transcripts:      66772
Percent GC: 66.62

########################################
Stats based on ALL transcript contigs:
########################################

        Contig N10: 4225
        Contig N20: 3083
        Contig N30: 2464
        Contig N40: 2014
        Contig N50: 1678

        Median contig length: 758
        Average contig: 1106.61
        Total assembled bases: 73890513


#####################################################
## Stats based on ONLY LONGEST ISOFORM per 'GENE':
#####################################################

        Contig N10: 3987
        Contig N20: 2871
        Contig N30: 2280
        Contig N40: 1862
        Contig N50: 1531

        Median contig length: 624
        Average contig: 984.38
        Total assembled bases: 48617582
```

The N10 through N50 values show the value of at least x% of number of assembled contigs have Nx nucleotide in length. For example, in contigs (isoform) level, the N50 indicates at least half (50%) of number of the assembled transcripts are 1,678 nucleotides in length, whereas N50 of the longest isoform that represent the gene is 1,531 nucleotides in length.

Benchmarking Universal Single-Copy Orthologs (BUSCO) analysis

BUSCO reported the transcriptome assembly completeness by evaluate whether the set of assembly recovered a whole set of universal functional genes referred from orthologous sequence from neighbor species. BUSCO metric is complementary to technical metrics like N50 as we did using TrinityStats.

> **Activity**
>
> BUSCO v4 and v5 use lineage datasets information from OrthoDB v10. You can search all available lineage datasets using the following command:
>
> ```
> # Activate conda environment
> conda activate busco
> # List all lineage datasets in OrthoDB v10
> busco --list-datasets
> ```
>
> As of March 2023, more than 100 lineage datasets available in OrthoDB v10.
>
> Generally the lineage to select for your assessments should be the most specific lineage available, e.g. for assessing Cyanophora transcriptome assembly data you may choose the Viridiplantae or Chlorophyta lineages rather than the metazoa lineage. Here we'll select Viridiplantae lineage dataset to evaluate the single-copy orthologs in Cyanophora assembled transcriptome using the following command.
>
> ```
> # Go to current working directory
> cd ~/Cpa_RNASeq/03_assembly
> # Run BUSCO
> busco --mode transcriptome \
> --in Trinity_2023-03-08.Trinity.fasta \
> --lineage_dataset /opt/Cpa_RNASeq/busco_downloads/lineages/viridipl
> --out BUSCO_Viridiplantae \
> --cpu 2 \
> --offline
> ```
>
> Estimated time usage: ~30-35 min
>
> In this command, BUSCO runs in transcriptome mode by the required input file.
>
> If we didn't make it in time, the backup data in backup BUSCO results will keep in: `/opt/Cpa_RNASeq/BUSCO_Viridiplantae/`, and BUSCO Viridiplantae lineage dataset is also in `/opt/Cpa_RNASeq/busco_downloads/`.

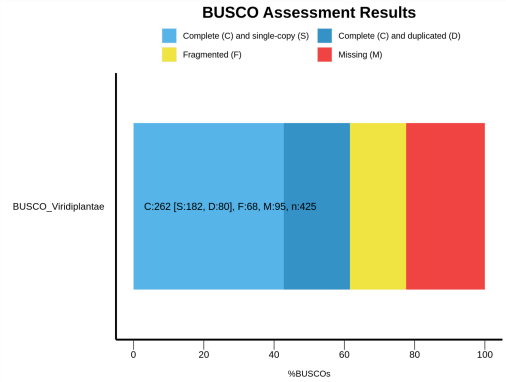
The classification results from BUSCO will save to `BUSCO_Viridiplantae` in your working directory, as well as print out to the terminal as follow:

```
2023-03-08 00:05:05 INFO:       Results:        C:61.6%
[S:42.8%,D:18.8%],F:16.0%,M:22.4%,n:425

2023-03-08 00:05:06 INFO:

        ---------------------------------------------------
        |Results from dataset viridiplantae_odb10         |
        ---------------------------------------------------
        |C:61.6%[S:42.8%,D:18.8%],F:16.0%,M:22.4%,n:425   |
        |262    Complete BUSCOs (C)                       |
        |182    Complete and single-copy BUSCOs (S)       |
        |80     Complete and duplicated BUSCOs (D)        |
        |68     Fragmented BUSCOs (F)                     |
        |95     Missing BUSCOs (M)                        |
        |425    Total BUSCO groups searched               |
        ---------------------------------------------------
```

The BUSCO result shows the composition of the expected gene content within the assembled transcriptome. The BUSCO result can be divided into Complete and Single-Copy, Complete and Duplicated, Fragmented or Missing BUSCOs.

> **Activity**
>
> Now we'll generate a BUSCO plot.
>
> ```
> # Working directory: cd ~/Cpa_RNASeq/03_assembly/assessment
> generate_plot.py --working_directory BUSCO_Viridiplantae
> ```
>
> Estimated time usage: < 1 min

Expected graphical should be as follows. And explanation on each type of BUSCO results can be found at the BUSCO's Documentation.



Estimating Abundance and Differential Expression Analysis of Genes

Differential gene expression analysis is a statistical method that uses count data to determine significant changes between experimental groups. For example, transcriptional changes of stress-induced genes in plant leaves under water deficit

compared to normal conditions are examined. Count data can be transcript, gene, exon, and noncoding characteristics.

To perform differential gene expression analysis with the Trinity integrated extensions, you will need the assembled transcripts/genes from the previous step and clean reads (and their replicates) from your experiment to assign to the assembled transcripts/genes and count the number of reads assigned to those transcripts/genes.

Estimating Transcript Abundance

This part will adopted from Trinity's Wiki [Trinity Transcript Quantification](#).

There are two different methods for quantifying reads mapped to the reference, by using the alignment-based (RSEM) and alignment-free (salmon, kallisto) qualtifiers. In this workshop, we'll use the salmon, an ultra-fast alignment and quantification tool, to count number of reads mapped to the assembled gene.

> **Tip**
>
> You can see the usage of the command you wsh to perform by type the command followed by `--help`, or `-h`. For example, to see the usage of
>
> `align_and_estimate_abundance.pl -h`

> **Activity**
>
> Now we will align and count reads mapped to the reference assembled transcripts using buit-in utility `align_and_estimate_abundance.pl` using the following commands.
>
> 1. Go to working directory and activate conda environment
>
> Go to working directory
>
> ```
> cd ~/Cpa_RNASeq
> ```
>
> The working directory should contain the following subdirectories.
>
> ```
> Cpa_RNASeq
>  ├── 01_Rawdata
>  ├── 02_QC
>  ├── 03_assembly
>  ├── 04_DE_analysis
>  └── 05_annotation
> ```
>
> Then, activate conda environment
>
> ```
> conda activate trinity
> ```
>
> 2. Estimating Transcript Abundance
>
> your current working directory is: `~/Cpa_RNASeq`

```
align_and_estimate_abundance.pl \
--transcripts 03_assembly/Trinity_2023-03-08.Trinity.fasta \
--seqType fq \
--samples_file /opt/Cpa_RNASeq/sample_list.tsv \
--est_method salmon \
--gene_trans_map 03_assembly/Trinity_2023-03-08.Trinity.fasta.gene_
--thread_count 2 \
--prep_reference
```

Then, type `ls` to see the results in your working directory. These directories are the results from the above command. Each folder represents each biological replicate in your experiment.

```
drwxrwxr-x  5 jiratchaya jiratchaya 4096 Mar  9 20:37 dark_1/
drwxrwxr-x  5 jiratchaya jiratchaya 4096 Mar  9 20:38 dark_2/
drwxrwxr-x  5 jiratchaya jiratchaya 4096 Mar  9 20:38 dark_3/
drwxrwxr-x  5 jiratchaya jiratchaya 4096 Mar  9 20:38
normal_light_1/
drwxrwxr-x  5 jiratchaya jiratchaya 4096 Mar  9 20:39
normal_light_2/
drwxrwxr-x  5 jiratchaya jiratchaya 4096 Mar  9 20:39
normal_light_3
```

Estimated time usage: ~ 20 min per user

Parameter descriptions of `align_and_estimate_abundance.pl`: the assembled transcripts flagged in `--transcripts`, `--seqType` indicates file format of reads that will be mappped to the reference transcripts. We define the read count estimation tool in `--est_method`, in this workshop we use salmon, and also estimate read counts using information of gene-transcript relationships from the `trinity_out_dir.Trinity.fasta.gene_trans_map` file that we specified in the `--gene_trans_map` parameter.

A list of read files will be contained in the metadata file `sample_list.tsv` in the parameter `--samples_file`, which we have prepared for you. In short, the sample list will be prepared in a tab-delimited text file indicating the relationships between biological replicates. For example,

```
jiratchaya@pslab1:~$ cat /opt/Cpa_RNASeq/sample_list.tsv
dark    dark_1  /opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306034_1.fastq
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306034_2.fastq
dark    dark_2  /opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306029_1.fastq
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306029_2.fastq
dark    dark_3  /opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306028_1.fastq
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306028_2.fastq
normal_light    normal_light_1
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306033_1.fastq
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306033_2.fastq
normal_light    normal_light_2
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306032_1.fastq
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306032_2.fastq
normal_light    normal_light_3
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306035_1.fastq
/opt/Cpa_RNASeq/Cyanophora_rawdata/SRR8306035_2.fastq
```

The first column indicates the study experimental groups, followed by their biological replicates in the second column, and the forward and reverse sequence read files belong to their biological replicate. It's important that the file path begins with the directory in which you'll be working so that the programs can correctly route to the files.

Output results are created in the current working directory separately for experimental groups and biological replicates as follow.

```
dark_1
    ├── aux_info/
    ├── cmd_info.json
    ├── lib_format_counts.json
    ├── libParams/
    ├── logs/
    ├── quant.sf
    └── quant.sf.genes
.
.
.
normal_light_3
    ├── aux_info/
    ├── cmd_info.json
    ├── lib_format_counts.json
    ├── libParams/
    ├── logs/
    ├── quant.sf
    └── quant.sf.genes
```

> **Activity**
>
> According to the previous part, now we'll organize the directory to make it tidy by moving all the results to the directory `04_DE_analysis`.

```
# make sure you're in ~/Cpa_RNASeq directory so that you can move t
mv dark* normal* 04_DE_analysis/

# Then enter to the directory `04_DE_analysis`
cd 04_DE_analysis
ls ./*
```

Expected result:

```
(trinity) jiratchaya@pslab1:~/Cpa_RNASeq/04_DE_analysis$ ls ./*
./dark_1:
aux_info  cmd_info.json  lib_format_counts.json  libParams  logs
quant.sf  quant.sf.genes

./dark_2:
aux_info  cmd_info.json  lib_format_counts.json  libParams  logs
quant.sf  quant.sf.genes

./dark_3:
aux_info  cmd_info.json  lib_format_counts.json  libParams  logs
quant.sf  quant.sf.genes

./normal_light_1:
aux_info  cmd_info.json  lib_format_counts.json  libParams  logs
quant.sf  quant.sf.genes

./normal_light_2:
aux_info  cmd_info.json  lib_format_counts.json  libParams  logs
quant.sf  quant.sf.genes

./normal_light_3:
aux_info  cmd_info.json  lib_format_counts.json  libParams  logs
quant.sf  quant.sf.genes
```

after running salmon you'll find output files:

- `quant.sf` : transcript abundance estimates (generated by salmon)
- `quant.sf.genes` : gene-level abundance estimates (generated here by summing transcript values)

Here's an example of `quant.sf.genes` file:

Building Transcript and Gene Expression Matrices

We'll estimates abundance matrices with the filename `quant.sf`, which are available in all results directories. In this step, the utility `abundance_estimates_to_matrix.pl` is used to combine all separate count matrices from the file `quant.sf` in all result directories into a single matrix file. By using salmon as `--est_method` and specifying the parameter `--gene_trans_map`, a

gene abundance matrix is created.

> **Activity**
>
> 1. Create abundance matrix
>
> Your current working directory: `~/Cpa_RNASeq/04_DE_analysis`
>
> ```
> abundance_estimates_to_matrix.pl --est_method salmon \
> --gene_trans_map ../03_assembly/Trinity_2023-03-08.Trinity.fasta.ge
> --name_sample_by_basedir \
> */quant.sf
> ```
>
> Expected result:
>
> ```
> (trinity) jiratchaya@pslab1:~/Cpa_RNASeq/04_DE_analysis$ ls -l
> total 4945
> drwxrwx--- 1 root PSLab    4096 Mar  5 16:16 dark_1
> drwxrwx--- 1 root PSLab    4096 Mar  5 16:16 dark_2
> drwxrwx--- 1 root PSLab    4096 Mar  5 16:16 dark_3
> drwxrwx--- 1 root PSLab    4096 Mar  5 16:16 normal_light_1
> drwxrwx--- 1 root PSLab    4096 Mar  5 16:16 normal_light_2
> drwxrwx--- 1 root PSLab    4096 Mar  5 16:16 normal_light_3
> -rw-rw---- 1 root PSLab      67 Mar  6 13:29
> salmon.gene.counts.matrix
> -rw-rw---- 1 root PSLab      67 Mar  6 13:29
> salmon.gene.TPM.not_cross_norm
> -rw-rw---- 1 root PSLab 2736773 Mar  6 13:29
> salmon.isoform.counts.matrix
> -rw-rw---- 1 root PSLab 2297101 Mar  6 13:29
> salmon.isoform.TPM.not_cross_norm
> ```
>
> This command will generate 4 result files:
>
> - `salmon.gene.counts.matrix` is the estimated raw RNA-Seq counts in GENE level in all experimental groups.
> - `salmon.gene.TPM.not_cross_norm` is the Transcript per Million (TPM) of RNA-Seq counts in GENE level in all experimental groups.
> - `salmon.isoform.counts.matrix` is the estimated raw RNA-Seq counts in TRANSCRIPTS level in all experimental groups.
> - `salmon.isoform.TPM.not_cross_norm` is the Transcript per Million (TPM) of RNA-Seq counts in TRANSCRIPTS level in all experimental groups.

Quality Control of Sample Read Counts and Biological Replicates

Once you've performed quantification for each experimental group, it's good to examine the data to ensure that your biological replicates are well correlated, and also to investigate relationships among your samples. It is critical that you identify any obvious differences between the relationships between your sample and replicates, such as those resulting from accidental mislabeling of sample

replicates, strong outliers, or batch effects, prior to further data analysis. The Trinity's utility called `PtR` (pronounced as 'Peter', stands for Perl to R) can generate some exploratory data analysis rely on count matrix, such as compare difference between replicate, compare difference between experimental groups, principal component analysis, and so on.

> **Activity**
>
> Recheck the current working directory
>
> ```
> pwd
> ```
>
> You must be in `~/Cpa_RNASeq/04_DE_analysis`
>
> TheThen prepare the sample metadata from differential expression analysis (DE). The sample metadata table for the DE analysis is different from the table used for abundance estimation. We only need the first two columns from this file to create a metadata table for the analysis of DE. Therefore, we can use the following Bash command to create and edit a new file.
>
> EExtract the first 2 columns of metadata to estimate the read count into a new file in `04_DE_analysis`
>
> ```
> cut -f 1-2 /opt/Cpa_RNASeq/sample_list.tsv > samples.txt
> ```
>
> See expected result file
>
> ```
> (trinity) jiratchaya@pslab1:~/Cpa_RNASeq/04_DE_analysis$ cat
> samples.txt
> dark    dark_1
> dark    dark_2
> dark    dark_3
> normal_light    normal_light_1
> normal_light    normal_light_2
> normal_light    normal_light_3
> ```

Compare replicates for each of your samples

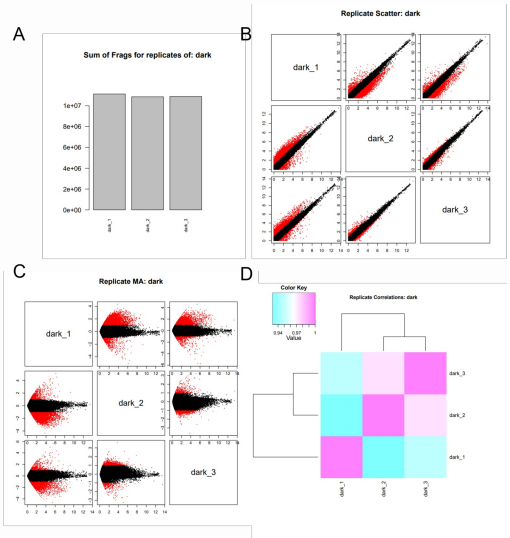
This step will use PtR to reads the matrix of counts, performs a counts-per-million (CPM) data transformation followed by a log2 transform, and then generates a multi-page pdf file `named ${sample}.rep_compare.pdf` for each of your samples, including several useful plots

> **Activity**
>
> Compare replicates for each of your samples
>
> ```
> # Current workdir: ~/Cpa_RNASeq/04_DE_analysis
> PtR --matrix salmon.isoform.counts.matrix \
> --samples samples.txt --log2 --CPM \
> --min_rowSums 10 \
> --compare_replicates
> ```

These files will append more to your current working directories:

```
-rw-rw---- 1 root PSLab    4695 Mar  6 14:37
salmon.isoform.counts.matrix.R
-rw-rw---- 1 root PSLab 1990182 Mar  6 14:37 dark.rep_compare.pdf
-rw-rw---- 1 root PSLab 1828692 Mar  6 14:37
normal_light.rep_compare.pdf
-rw-rw---- 1 root PSLab 3558147 Mar  6 14:37
salmon.isoform.counts.matrix.minRow10.CPM.log2.dat
```

The last 3 files are newly generated by this step. There's two PDF files separated by experimental groups, `dark.rep_compare.pdf` and `normal_light.rep_compare.pdf`, and raw data for plots in `.dat` file.



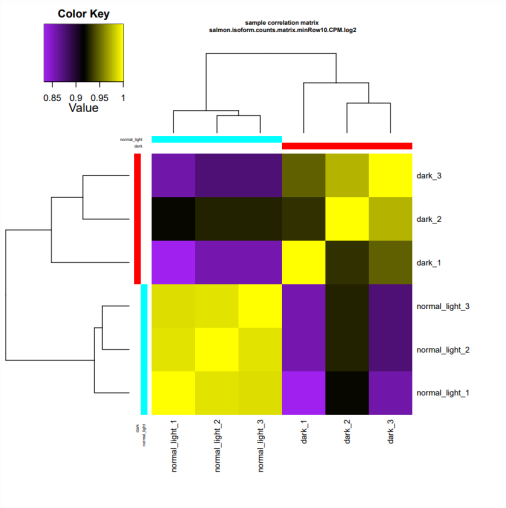
Example result of comparing biological replicates in Dark samples. The figures were captured from `dark.rep_compare.pdf` file. (A) The sum of mapped fragments. (B) Pairwise comparisons of replicate log(CPM) values, in which the data points more than 2-fold different are highlighted in red. (C) The pairwise MA plots (x-axis: mean log(CPM), y-axis log(fold_change)). And, (D) A Replicate Pearson correlation heatmap.

Compare Replicates Across Samples

> **Activity**
>
> This command will generate a useful heatmap of pearson correlation matrix of samples from two different experimental groups.
>
> ```
> PtR --matrix salmon.isoform.counts.matrix \
> --min_rowSums 10 \
> -s samples.txt \
> --log2 --CPM \
> --sample_cor_matrix
> ```

These files will append more to your current working directories:

```
-rw-rw---- 1 root PSLab    4012 Mar  6 15:02
salmon.isoform.counts.matrix.R
-rw-rw---- 1 root PSLab 3558147 Mar  6 15:02
salmon.isoform.counts.matrix.minRow10.CPM.log2.dat
-rw-rw---- 1 root PSLab     678 Mar  6 15:02
salmon.isoform.counts.matrix.minRow10.CPM.log2.sample_cor.dat
-rw-rw---- 1 root PSLab    6429 Mar  6 15:02
salmon.isoform.counts.matrix.minRow10.CPM.log2.sample_cor_matrix.pdf
```



heatmap of pearson correlation coefficiant between Dark and Normal light samples.

Principal Component Analysis (PCA)

Another important analysis method to explore relationships among the sample replicates is Principal Component Analysis (PCA).

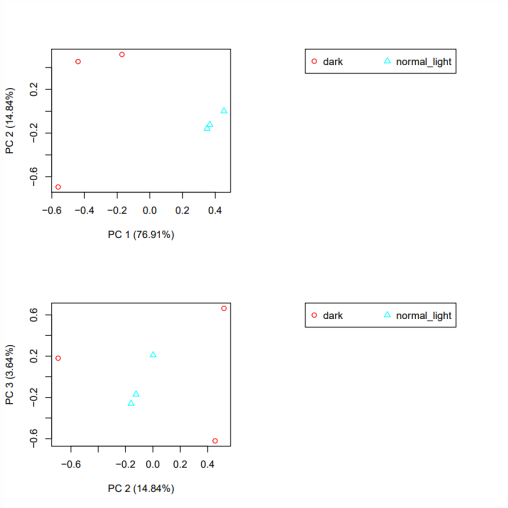
You can find more explanation about PCA here: - https://blog.bioturing.com/2018/06/14/principal-component-analysis-explained-simply/ - https://youtu.be/FgakZw6K1QQ

> **Activity**
>
> ```
> PtR --matrix salmon.isoform.counts.matrix \
> -s samples.txt \
> --min_rowSums 10 --log2 \
> --CPM --center_rows \
> --prin_comp 3
> ```

These files will append more to your current working directories:

```
-rw-rw---- 1 root PSLab    4789 Mar  6 15:18
salmon.isoform.counts.matrix.R
-rw-rw---- 1 root PSLab 4069112 Mar  6 15:18
salmon.isoform.counts.matrix.minRow10.CPM.log2.centered.dat
-rw-rw---- 1 root PSLab     756 Mar  6 15:18
salmon.isoform.counts.matrix.minRow10.CPM.log2.centered.PCA.prcomp.score
s
-rw-rw---- 1 root PSLab 4163653 Mar  6 15:18
salmon.isoform.counts.matrix.minRow10.CPM.log2.centered.PCA.prcomp.loadi
s
-rw-rw---- 1 root PSLab    5446 Mar  6 15:18
salmon.isoform.counts.matrix.minRow10.CPM.log2.centered.prcomp.principal
f
```

You can find the PCA plot in `salmon.isoform.counts.matrix.minRow10.CPM.log2.centered.prcomp.principal_components.p`



PCA plot.

We set the number of principal components (PC) to be calculated for only first 3 PCs in `--prin_comp`. Which indicates that these PCs will be plotted, as shown above, with PC1 vs. PC2 and PC2 vs. PC3. In this example, the replicates cluster tightly according to sample type, which is very reassuring.

Differential Expression Analysis

Trinity also contains a built-in utility for DE analysis called `run_DE_analysis.pl`, in which use the count matrix and sample metadata file. Trinity provides support for several differential expression analysis tools, currently including edgeR, DESeq2, limma/voom, and ROTS.

DE analysis in Trinity will perform pairwise comparison of gene/transcript expression. If the biological replicates are presented for each sample, you should indicate this as we already created in our metadata table `samples.txt`. Here we'll analyze DE genes in the 'transcript' level using the `salmon.isoform.counts.matrix` file.

> **Activity**
>
> DE analysis using DESeq2
>
> ```
> run_DE_analysis.pl \
> --matrix salmon.isoform.counts.matrix \
> --method DESeq2 \
> --samples_file samples.txt \
> --output DESeq2_result
> ```

After run the above command, the following directory will append to your current working directory:

```
drwxrwx--- 1 root PSLab     688 Mar  6 15:42 DESeq2_result
```

In this output directory, you'll find the following files for each of the pairwise comparisons performed:

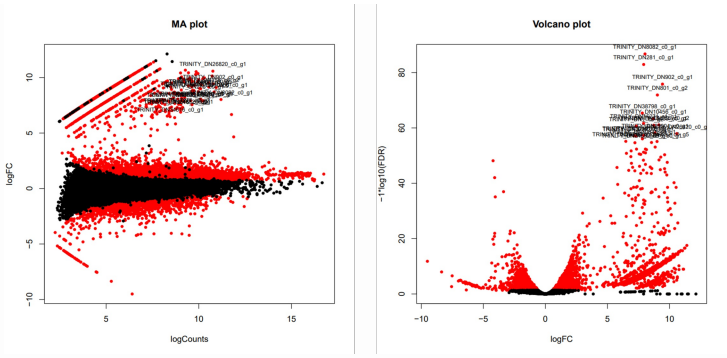
```
-rw-rw---- 1 root PSLab  972633 Mar  6 15:42
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.count_matrix
-rw-rw---- 1 root PSLab 4247784 Mar  6 15:42
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results
-rw-rw---- 1 root PSLab 2428272 Mar  6 15:42
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.MA_n
f
-rw-rw---- 1 root PSLab    1845 Mar  6 15:42
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.Rscript
```

Result explanations:

- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.Rscript` is the R-script executed to perform the DE analysis.
- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.count_matrix` is an integer matrix of read count derived from the input file `salmon.isoform.counts.matrix`.
- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results` is the DE analysis results, including log fold change and statistical significance.
- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.MA_n_Volcano.` is MA and Volcano plots features found DE at the defined FDR will be colored red.

Here's an example of DE analysis result file (`salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results`):

An example of volcano plot for transcript-level differentially expression analysis.



(left) MA plot and (right) volcano plot.

Extracting and clustering differentially expressed transcripts

An initial step in analyzing differential expression is to extract those transcripts that are most differentially expressed (most significant FDR and fold-changes) and to cluster the transcripts according to their patterns of differential expression across the samples.

> **Activity**
>
> Extracting and clustering differentially expressed transcripts can run using the following from within the DE output directory, by running the following script:
>
> ```
> cd DESeq2_result/
> analyze_diff_expr.pl \
> --matrix ../salmon.isoform.counts.matrix \
> -P 1e-3 \
> -C 2 \
> --samples ../samples.txt \
> --max_genes_clust 10000
> ```

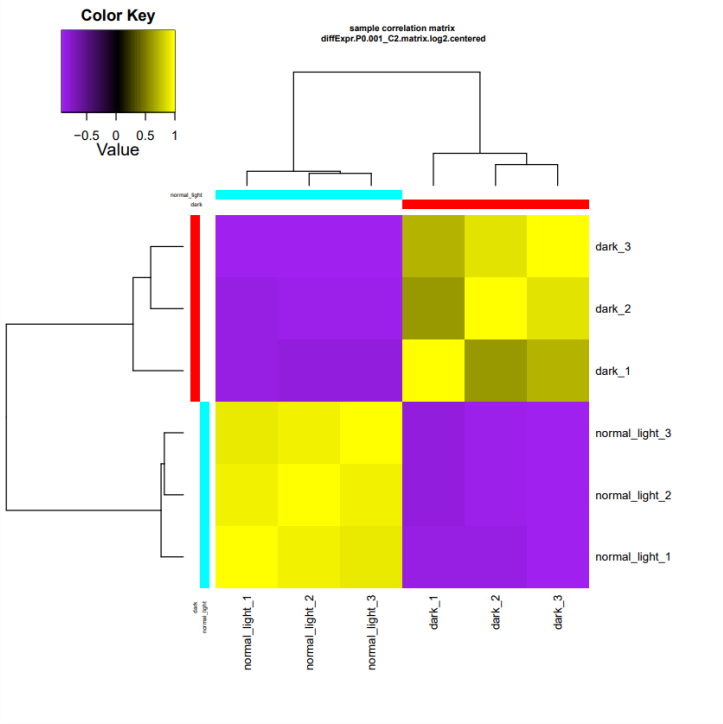
The above command use an integer count matrix from DE analysis, and define criteria for extracting differentially expressed transcripts. For example, set p-value cutoff for FDR in `-P` to 0.001, set minimum absolute log 2-fold change criteria in `-C` to 2, meaning that it will extracted only the DE transcripts that are 2^2 = 4-fold, and use only top 10,000 among all differentially transcripts in `--max_genes_clust` for hierarchical clustering analysis. However, user can customize these criteria based on their interest.

The following results will append to the current working directory `DESeq2_result`

```
-rw-rw---- 1 root PSLab      120 Mar  6 16:34
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.samp
s
-rw-rw---- 1 root PSLab   332012 Mar  6 16:34
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P0.0
-UP.subset
-rw-rw---- 1 root PSLab    42038 Mar  6 16:34
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P0.0
-UP.subset
-rw-rw---- 1 root PSLab   373901 Mar  6 16:34
salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P0.0
t
-rw-rw---- 1 root PSLab       51 Mar  6 16:34
DE_feature_counts.P0.001_C2.matrix
-rw-rw---- 1 root PSLab    73959 Mar  6 16:34 diffExpr.P0.001_C2.matrix
-rw-rw---- 1 root PSLab     4649 Mar  6 16:34
diffExpr.P0.001_C2.matrix.R
-rw-rw---- 1 root PSLab   246973 Mar  6 16:34
diffExpr.P0.001_C2.matrix.log2.centered.dat
-rw-rw---- 1 root PSLab      698 Mar  6 16:34
diffExpr.P0.001_C2.matrix.log2.centered.sample_cor.dat
-rw-rw---- 1 root PSLab     6399 Mar  6 16:34
diffExpr.P0.001_C2.matrix.log2.centered.sample_cor_matrix.pdf
-rw-rw---- 1 root PSLab   101250 Mar  6 16:34
diffExpr.P0.001_C2.matrix.log2.centered.genes_vs_samples_heatmap.pdf
-rw-rw---- 1 root PSLab 14777602 Mar  6 16:34
diffExpr.P0.001_C2.matrix.RData
```

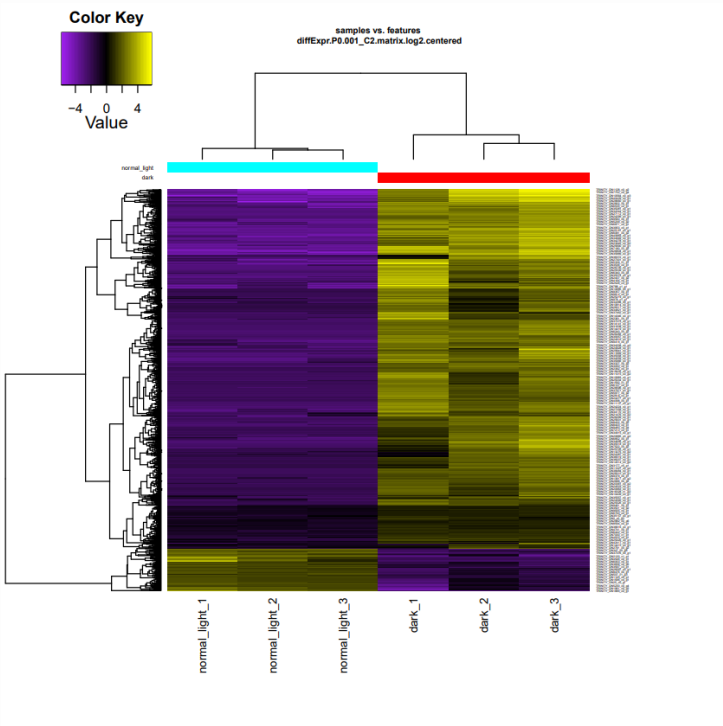
Result explanations:

- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.samples` is identical to the metadata `samples.txt` file.
- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P0.001_C2.dar` `UP.subset` is the subset of expression matrix of up-regulated transcripts in Dark group, which are down-regulated in Normal light group.
- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P0.001_C2.nor` `UP.subset` is the subset of expression matrix of up-regulated transcripts in Normal light group, which are down-regulated in Dark group.
- `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P0.001_C2.DE.` is a summary of DE transcripts results containing columns of significant values, and its normalized expression value.
- `diffExpr.P0.001_C2.matrix.log2.centered.sample_cor_matrix.pdf` is the sample correlation matrix, as follow.



Sample correlation matrix visualized only for differentially expressed transcripts.

- `diffExpr.P0.001_C2.matrix.log2.centered.genes_vs_samples_heatmap.pdf` is heatmap of differentially expressed transcripts.



Heatmap of differentially expressed transcripts.

DE gene patterning and clustering analysis

In the heat map of differentially expressed transcripts, there is a clear difference between the DE transcripts under dark and normal light conditions. Therefore, using `define_clusters_by_cutting_tree.pl`, we can divide these genes into clusters based on the same trend of expression values as follows.

> **Activity**
>
> Automatically Partitioning Genes into Expression Clusters
>
> ```
> define_clusters_by_cutting_tree.pl \
> -R diffExpr.*.matrix.RData \
> --Ptree 60
> ```

There are three different methods for dividing genes into clusters, K-Means clustering, hierarchical clustering (as used in the heatmap), and the recommended method of using criteria to truncate tree branch lengths that fall below the criteria by using '–Ptree'.

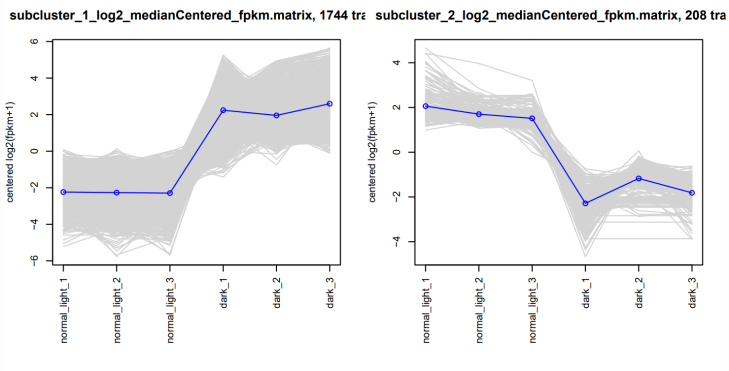
The following results will append to the current working directory `DESeq2_result` which are files and `diffExpr.P0.001_C2.matrix.RData.clusters_fixed_P_60/` directory.

```
-rw-rw---- 1 root PSLab    45837 Mar  6 16:51
clusters_fixed_P_60.heatmap.heatmap_gene_order.txt
-rw-rw---- 1 root PSLab    61467 Mar  6 16:51
clusters_fixed_P_60.heatmap.gene_cluster_colors.dat
-rw-rw---- 1 root PSLab   110890 Mar  6 16:51
clusters_fixed_P_60.heatmap.heatmap.pdf
drwxrwx--- 1 root PSLab      400 Mar  6 16:51
diffExpr.P0.001_C2.matrix.RData.clusters_fixed_P_60/
```

List of files in `diffExpr.P0.001_C2.matrix.RData.clusters_fixed_P_60/` directory are:

```
-rw-rw---- 1 root PSLab  43915 Mar  6 16:51 my_cluster_plots.pdf
-rw-rw---- 1 root PSLab 220780 Mar  6 16:51
subcluster_1_log2_medianCentered_fpkm.matrix
-rw-rw---- 1 root PSLab  26259 Mar  6 16:51
subcluster_2_log2_medianCentered_fpkm.matrix
-rw-rw---- 1 root PSLab    816 Mar  6 16:51 __tmp_plot_clusters.R
```

The DE transcript partiitoning and clustering is located in `my_cluster_plots.pdf`



DE transcript partiitoning and clustering analysis

Then, we'll subset the assembled transcriptome for only differentially expressed genes for functional annotation analysis in the next chapter.

> **Activity**
>
> From the previous command, we already have a list of differentialli expressed gnees in `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P1e-3_C2.DE.subset` file. For the functional annotation analysis in the next step, we will subset only top 10 upregulated and top 10 downregulated DEGs for annotation step. Data subsetting will use the following command.
>
> 1. Extract top 10 upregulated DEGs in Normal light condition. This command will ascendingly sort the 6th column (log2foldchange) of the file `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P1e-3_C2.normal_light-UP.subset`, then selected top 10 highest log2foldchange from the last 10 lines, then select only the trinity transcript ID in the first column.
>
> ```
> sort --key=6 --numeric-sort salmon.isoform.counts.matrix.dark_vs_no
> ```
>
> 2. Do the same for dark condiiton using `salmon.isoform.counts.matrix.dark_vs_normal_light.DESeq2.DE_results.P1e-3_C2.dark-UP.subset` file.
>
> ```
> sort --key=6 --numeric-sort salmon.isoform.counts.matrix.dark_vs_no
> ```
>
> 3. Concatenate transcript ID files (`DEGtop10_NormalLight-up.txt` and `DEGtop10_dark-up.txt`) into a single file.
>
> ```
> cat DEGtop10_NormalLight-up.txt DEGtop10_dark-up.txt > DEGtop20_all
> ```
>
> 4. Retrieve FASTA sequence of top 20 DEGs using Trinity's utility.
>
> ```
> retrieve_sequences_from_fasta.pl DEGtop100_all.txt ~/Cpa_RNASeq/03_
> ```
>
> The output file locates in `~/Cpa_RNASeq/04_DE_analysis/DESeq2_result`

Transcriptome Assembly Annotation

In the absence of a well-annotated reference genome of the species of your choice, researchers will need to generate a draft genome or transcriptome through a de novo approach. Subsequent analyses after assessing the completeness of the assembly include differential gene expression analysis (quantitative approach), functional annotation (qualitative), and many others depending on your experimental design. Functional annotation involves searching for the biological meaning of the biological sequence of interest using known data sets or predicting a new one.

Functional annotation of non-model organisms is usually compared with neighboring species that have a well-annotated reference genome in terms of functions. For example, complete genome/transcriptome datasets of the black tiger shrimp Penaeus monodon or the Caridian shrimp *Marsupenaeus japonicus* can be used as reference datasets for annotating the transcriptomes of the banana shrimp *Fenneropenaeus merguiensis*. Information from the taxonomic rank, e.g., a complete genome/transcriptome of the phylum Arthropoda or subphylum Crustacea, can also be used to annotate the banana shrimp dataset. In addition, orthologous data from the associated kingdom, such as the eukaryote dataset, can be used to annotate the banana shrimp dataset.

Functional annotation is task to find the biological meaning such as biochemical and biological function of proteins or CDS. Possible analyses to annotate genes can be for example:

- **Sequence similarity searching:** such as BLAST, HMMER, Diamond, and many more. This approach uses biological sequences of interest (proteins or nucleotides) as query sequences to search a database of known sequence annotations (called subject sequences) and check how similar your query is to the subject sequence.
- **Gene Ontology (GO) annotation:** is functional classification of genes into 3 major classes; cellular component, biological process, and molecular function. More information please see Gene Ontology Overview.
- **Biological pathway and interaction network:** by searching the orthologous protein from pathway and biological interaction network such as KEGG pathway, STRING, Reactome, and many more. This approach provide information of the gene/protein of interest interacting with others within the same pathway of gene set.

Functional Annotation using EggNOG-mapper

EggNOG-mapper is a tool for functional annotation analysis in biological sequences, which can be proteomes, genomes, transcriptomes or metagenomes. EggNOG-mapper uses the EggNOG database as a reference for searching orthologous identifiers. Many search algorithms have been deployed to search against EggNOG database, such as diamond (fast and comparable), MMseqs2 fast and comparable, and HMMER3 (slowest).

Once user have already installed EggNog-mapper, you will need to download

EggNOG database to your local machine using the following command. But in this workshop, **we already prepared it for you :)**

```
download_eggnog_data.py -D -M -y -f --data_dir eggnog_db/
```

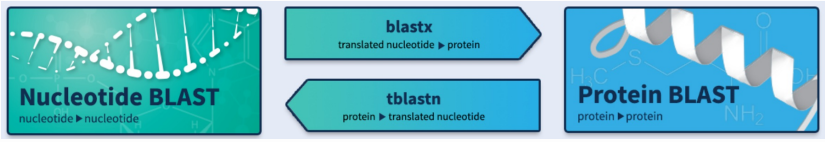
The following command `download_eggnog_data.py` tries to download the diamond search database by default, but in this workshop we will use the MMSeqs2 search tool, so we can skip downloading the diamond database with `-D` and download the MMSeqs2 database using `-M` instead.

> **Activity**
>
> After preparing the EggNOG database, we can run the EggNog mapper via the file `emapper.py` as in the following command, specifying the required arguments for the path of the downloaded EggNOG database in `--data_dir`, the input type (`--itype`) and the search algorithm MMSeqs2 (`-m`). You can see usage of emapper.py by type emapper.py –help in your terminal, or have a look at [A few recipes of using EggNOG-mapper](#).
>
> ```
> # Go to main project directory
> cd ~/Cpa_RNASeq
> # Activate conda environment
> conda activate emapper
> # Run EggNOG-mapper
> emapper.py --cpu 5 \
> --data_dir /opt/Cpa_RNASeq/eggNOG_db/ \
> --output_dir 05_annotation/ \
> -m diamond \
> --evalue 1e-5 \
> -i 04_DE_analysis/DESeq2_result/DEGtop100_all_seqs.fasta \
> --no_file_comments \
> --itype CDS \
> --excel \
> --output Cpa_emapper_2023-03-10
> ```
>
> Estimated time usage: ~ 1 hr
>
> Expected output files:
>
> ```
> -rw-r--r--  1 jiratchaya jiratchaya 20287555 Mar  8 22:54
> Cpa_emapper_2023-03-10.emapper.annotations
> -rw-r--r--  1 jiratchaya jiratchaya  5060493 Mar  8 22:54
> Cpa_emapper_2023-03-10.emapper.annotations.xlsx
> -rw-r--r--  1 jiratchaya jiratchaya  5333975 Mar  8 22:54
> Cpa_emapper_2023-03-10.emapper.hits
> -rw-r--r--  1 jiratchaya jiratchaya  2528455 Mar  8 22:54
> Cpa_emapper_2023-03-10.emapper.seed_orthologs
> ```
>
> Output explanations:
>
> - `*.emapper.annotations` is a result file from the annotation phase. Each row represents the annotation reported for a given query.
> - `*.emapper.annotations.xlsx` is as same as `*.emapper.annotations` but in Excel format.

> - `*.emapper.hits` is a result file from the MMseqs2 search phase.
> - `.emapper.seed_orthologs` is a result file from parsing the hits. Each line associates a query with a seed ortholog. This file has the same format regardless of which searcher was used, except that it can be in short format (4 fields) or full format.

Homology Searching using NCBI BLAST

BLAST (Basic Local Alignment Search Tools) is usually a first choice as sa sequence similarity search tool. BLAST search for region that similar to both of our sequence of interest, which can be nucleotides or proteins. BLAST can be used to infer functional and evolutionary relationships between sequences as well as help identify members of gene families.



.

Several types of BLAST search algorithms classified by type of query sequence (protein/nucleotide sequence you want to search for) and th expected subject sequence (protein/nucleotide in BLAST database expected to match with the query sequence) such as the following table

Search algorithms	Input type	Output type
BLASTN	Nucleotide	Nucleotide
BLASTX	CDS	Protein
TBLASTN	Protein	CDS
BLASTP	Protein	Protein

- **BLASTN (Nucleotide BLAST)** compares nucleotide sequences to nucleotide sequences in databases such as Nt, 16S rRNA, ITS, and custom nucleotide databases. A useful example of using BLASTN is to search for phylogenetic relationships between the query sequence and neighboring species and infer the evolutionary relationship between them.
- **BLASTX (Translated BLAST)** uses the nucleotide query sequence to search protein databases such as Nr, Uniprot, Protein Databank (PDB), and custom protein databases. BLASTX translates the query into six reading frames (-3, -2, -1, 1, 2, 3) before searching. Therefore, the time required for BLASTX is about 6 times slower than the straightforward BLAST. A useful example of using BLASTX is to search for possible translation frames in de novo transcriptome assembly.
- **TBLASTN (Translated BLAST)** uses protein query sequence to search nucleotide databases by translating into six reading frames. TBLASTN provides benefit in searching for coding sequence (CDS) along with its open reading frame from protein sequence.
- **BLASTP (Protein BLAST)** compares protein query sequences with protein

sequences in databases. A useful example of using BLASTP is to search for protein sequence similarities to infer their functions from conserved domains observed in the sequence.

> **Activity**
>
> In this workshop, we' will use the nucleotide database BLAST using custom nucleotide data. In short, we'll subdivide whole nucleotide sequences and create the database BLAST from the NCBI nucleotide collection database belonging to the phylum Chlorophyta, Rhodophyta and Glaucophyta. **We have already prepared these databases for you** by using the following command:
>
> `makeblastdb -in [input_seq.fasta] -dbtype [nucl|prot]`
>
> Database paths:
>
> - Chlorophyta BLASTN database: `/opt/Cpa_RNASeq/BLAST_DB/Chlorophyta.fna`
> - Rhodophyta BLASTN database: `/opt/Cpa_RNASeq/BLAST_DB/Rhodophyta.fna`
> - Glaucophyta BLASTN database: `/opt/Cpa_RNASeq/BLAST_DB/Glaucophyta.fna`
>
> Depending on your interest, you can choose which phylum you want to use as BLAST database as above. Then we' will investigate it together!
>
> ### BLASTN
>
> BLAST tools is located in the `ncbi` environment, so you will need to activate environment as follow:
>
> ```
> conda activate ncbi
> ```
>
> Then,run BLASTN:
>
> ```
> # Current working directory: ~/Cpa_RNASeq
> blastn -db /opt/Cpa_RNASeq/BLAST_DB/[database_of_interest] \
> -query 04_DE_analysis/DEG_sequence.fasta \
> -out 05_annotation/BLASTN_DEG_[phylum_of_interest].tsv \
> -evalue 1e-5 \
> -outfmt "7 std qcovhsp stitle" \
> -max_target_seqs 5 \
> -num_threads 4
> ```
>
> Estimated time: < 5 mins
>
> By this command, you'll search query sequence in your database of interest. The result will collected using BLAST if E-value <= 1e-5.

Here we' specify the output format 7, which results in the table file containing the comment lines that start with the '#' character.

Example BLASTN output format `7 std qcovhsp stitle`:

```
$ head -n 15 BLASTN_DEG_Chlorophyta.tsv
# BLASTN 2.13.0+
# Query: sampleA
# Database: /opt/Cpa_RNASeq/BLASTN/Chlorophyta.fna
# 0 hits found
# BLASTN 2.13.0+
# Query: TRINITY_DN281_c0_g1_i1
# Database: /opt/Cpa_RNASeq/BLASTN/Chlorophyta.fna
# Fields: query acc.ver, subject acc.ver, % identity, alignment length,
mismatches, gap opens, q. start, q. end, s. start, s. end, evalue, bit
score, % query coverage per hsp, subject title
# 10 hits found
TRINITY_DN281_c0_g1_i1  XM_043068042.1  99.797  1972    0       1
1       1968    2126    155     0.0     3616    100     XM_043068042.1
Chlamydomonas reinhardtii uncharacterized protein (CHLRE_12g498600v5),
mRNA
TRINITY_DN281_c0_g1_i1  DQ122889.1      94.995  1918    84      5
1       1913    1936    26      0.0     3000    97      DQ122889.1
Chlamydomonas incerta elongation factor alpha-like protein (efl) mRNA,
complete cds
TRINITY_DN281_c0_g1_i1  XM_001696516.2  98.929  1400    15      0
514     1913    1539    140     0.0     2503    71      XM_001696516.2
Chlamydomonas reinhardtii uncharacterized protein (CHLRE_06g263450v5),
mRNA
TRINITY_DN281_c0_g1_i1  XM_043062829.1  98.929  1400    15      0
514     1913    1539    140     0.0     2503    71      XM_043062829.1
Chlamydomonas reinhardtii uncharacterized protein (CHLRE_06g263450v5),
mRNA
TRINITY_DN281_c0_g1_i1  CP097822.1      100.000 1261    0       0
1       1261    3287383 3286123 0.0     2329    64      CP097822.1
Chlamydomonas reinhardtii strain CC-5816 chromosome 12
TRINITY_DN281_c0_g1_i1  CP097822.1      100.000 251     0       0
1260    1510    3285932 3285682 2.34e-128       464     13
CP097822.1 Chlamydomonas reinhardtii strain CC-5816 chromosome 12
```

The result file can be further open with spreadsheet software such as Microsoft Excel, by skipping the comment lines. The column name is described in `# Fields:` comment line, containing the following fields:

Field names	Descriptions
query acc.ver	Query sequence ID
subject acc.ver	Subject sequence ID
% identity	Percentage identity
alignment length	Alignment length
mismatches	Number of mismatches
gap opens	Number of gap openings
q. start	Query sequence alignment start position
q. end	Query sequence alignment end position
s. start	Subject sequence alignment start position
s. end	Subject sequence alignment end position
evalue	Expect value
bit score	Bit score
% query coverage per hsp	Query Coverage Per High Scoring Pairs
subject title	Subject sequence name

You can specify options to include in this tabular format by type the folowing command and take a look at the terminal.

```
blastn -help
```

Reference sources

- [De novo transcriptome assembly, annotation, and differential expression analysis] from Galaxy Training!
- NCBI Bioinformatics Resources: [An Introduction: BLAST: Compare & identify sequences]. Berkeley Library, Universiry of California.

References

Buffalo, Vince. 2015. *Bioinformatics Data Skills: Reproducible and Robust Research with Open Source Tools*. " O'Reilly Media, Inc.".

Grabherr, Manfred G, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, and Qiandong Zeng. 2011. "Trinity: Reconstructing a Full-Length Transcriptome Without a Genome from RNA-Seq Data." *Nature Biotechnology* 29 (7): 644.

Haas, Brian J, Alexie Papanicolaou, Moran Yassour, Manfred Grabherr, Philip D Blood, Joshua Bowden, Matthew Brian Couger, David Eccles, Bo Li, and Matthias Lieber. 2013. "De Novo Transcript Sequence Reconstruction from RNA-Seq Using the Trinity Platform for Reference Generation and Analysis." *Nature Protocols* 8 (8): 1494–1512.

Knopp, Michael, Sriram G. Garg, Maria Handrich, and Sven B. Gould. 2020. "Major Changes in Plastid Protein Import and the Origin of the Chloroplastida." *iScience* 23 (3): 100896. https://doi.org/10.1016/j.isci.2020.100896.

Martin, Jeffrey A., and Zhong Wang. 2011. "Next-Generation Transcriptome

Assembly." *Nature Reviews Genetics* 12 (10): 671–82. https://doi.org/10.1038/nrg3068.