SIRE511 : LINUX AND BIOINFORMATICS DATA SKILLS

Bioinformatics Data Skill II

8th Week, 22/10/2024

Kwanrutai Mairiang, Ph.D

Bioinformatics Data

- Data is required in all bioinformatics projects. Most genomics data are large and complex, which can be challenging to handle, including:
  - Retrieving data
    - wget, curl, Rsync, and scp
  - Ensuring data integrity
    - SHA, MD5 checksums
  - Compression
    - gzip, gunzip
    - zcat, zgrep, zdiff, zless

Retrieving Bioinformatics Data

Download data from web using **`wget`** and **`curl`**

Download data from web using wget and curl

Downloading Data with wget

Wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies. (Manual page wget)

- Install wget

```
sudo apt install wget
```

- Syntax:

```
wget [option] URL
```

Wget command examples

1. Using wget to download single files
   - This is the most basic wget command example. A single file will be downloaded and stored in current working directory.
     - Ex. Download file “chr22.fa.gz” from genome database:

```
$ wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz
```

Wget command examples

2. Using Wget Command to Download Multiple Files

- To download multiple files using wget in a single command, you need to create a text file containing a list of download URLs.

```
$ nano download_url.txt
```

```
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr21.fa.gz
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chrY.fa.gz
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chrM.fa.gz
```

Use -i option to download files from list URL in “download_url.txt”

```
$ wget -i download_url.txt
```

Wget command examples

3. Using Wget Command to Get Files Under Different Names

- The downloaded file will be saved with a specified name using the –O (uppercase o) option.
  - Ex. Download the file "chr22.fa.gz" from the genome database and save it as "hg38_chr22.fa.gz."

```
$ wget -O hg38_chr22.fa.gz
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz
```

Wget command examples

4. Using Wget Command to Save Files in Specified Directory
   - The downloaded file will be saved to the location specified by the -P option.
     - Ex. Download the file "chr22.fa.gz" from the genome database and store it in the /data/download/ folder.

```
$ wget -P /data/download/
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chrom
osomes/chr22.fa.gz
```

Wget command examples

5. Using Wget Command to Limit Download Speed
   - Download the file while limiting the download speed. This is useful when fetching large files, as it prevents them from using all of your available bandwidth.
     - Ex. Download file “chr22.fa.gz” from genome database by limiting download speed to 500k :

```
$ wget --limit-rate=500k
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz
```

Wget command examples

6. Using Wget Command to Download in Background
   - To handle large files, you can make use of the -b feature, which allows your content to be downloaded in the background.

```
$ wget -b
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chr
omosomes/chrX.fa.gz
```

A "wget-log" file will be generated in the working directory to record the progress status. The '**`tail -f`**' command can be used to monitor the progress status.

Wget command examples

7. Using Wget Command to Continue Interrupted Downloads

- Download interruptions due to internet or power issues are common with large files. To avoid restarting, utilize the -c function for resuming downloads.

```
$ wget -c
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chr
omosomes/chrX.fa.gz
```

Wget command examples

8. Using Wget Command to Download Numbered Files

- Brace expressions can be used to automate the downloading of lists of files.

Download file chromosome 20, 21, 22

```
$ wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr{20..22}.fa.gz
```

Download file chromosome 1, 12, X, Y

```
$ wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr{1,12,X,Y}.fa.gz
```

Wget command examples

9. Downloading recursively with wget
   - To download all linked files on a website with wget, you can use -r option.

```
$ wget -r https://website.to.download.com/
```

Downloading Data with curl

Curl behaves similarly to wget, although it typically writes the file to standard output by default.

1. Download file by curl.
   - Save the downloaded file in the current directory with the same filename using the -O (uppercase o) option.

     ```
     $ curl –O https://url.to.download/chrY.fa.gz
     ```
   - Save the downloaded file with a specified name and location using the -o (lowercase o) option

     ```
     $ curl -o /data/download/hg38_chrY.fa.gz
     https://url.to.download/chrY.fa.gz
     ```

Example:

```
$ curl -O
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chrY.fa.gz
```

Downloading Data with curl

2. Using curl Command to Continue Interrupted Downloads

- Use '-C -' option (uppercase c) to tell curl to automatically find out where/how to resume the transfer.

```
$ curl -C - -O https://url.to.download/chrY.fa.gz
$ curl -C - -o hg38_chrY.fa.gz
https://url.to.download/chrY.fa.gz
```

-`C` = continue or resume a previous transfer

- (hyphen) = curl will figure out where to resume the download by checking the size of local file.

Downloading Data with curl

3. Using curl Command to download multiple files

- Download multiple files from multiple URLs.

```
$ curl -O https://url/chrX.fa.gz -O
https://url/chrY.fa.gz
```

- Download files from list of URLs in a file. (Ex. download_url.txt)

```
$ xargs -n 1 curl -O < download_url.txt
```

`xargs -n 1` command will get items from “download_url.txt” and use one argument (one URL) at a time to run command “**`curl -O`**”

Downloading Data with curl

4. Use curl Command for HTTP
   - Use curl command when there is a proxy server.

```
$ curl -x
http://proxy_user:proxy_password@proxy_server:pr
oxy_port -O https://url/file.gz
```

Example:

```
$ curl -x http://kwanrutai.chi:123pass@proxy-
si.mahidol.ac.th:8080 -O https://url/chr21.fa.gz
```

Downloading Data with curl

5. Use curl for FTP authentication
   - Download file from FTP.

```
curl -u username:password -O
ftp://ftp.server/chrX.fa.gz
```

   - Upload file to FTP

```
curl -u username:password -T
Yourfile.tar.gz ftp://ftp.server.com
```

-T = transfer

Downloading Data with curl

6. Limit bandwidth
   - Example: restrict the bandwidth to 500K

```
curl --limit-rate 100K -O https://
url.to.download/chrY.fa.gz
```

Retrieving Bioinformatics Data

Transfer file over SSH (typically between local and server) using **`rsync`** and **`scp`**

Rsync and Secure Copy (scp)

- What are the differences between rsync and scp?
  - **Rsync:** It synchronizes files and directories by only transferring the differences between the source and destination. This makes it efficient for incremental backups and updates.
  - **SCP:** It simply copies files from the source to the destination without any synchronization or differential transfer.

How to use rsync?

- **Install rsync**

```
sudo apt-get install rsync
```

- **Basic Syntax**

```
rsync [OPTIONS] SOURCE DESTINATION
```

  - OPTION: add more rsync option
  - SOURCE: the source directory or file you want to copy or synchronize.
  - DESTINATION: The destination directory where the source data will be copied or synchronized.

Use rsync command to copy files and and directory

- Copy all files in source directory to destination

```
rsync original/* destination/
```

- Copy all file and subdirectories in source directory to destination

```
rsync -r original/* destination/
```

  - Exclude a particular subdirectory:

```
rsync -r --exclude=subdirectory_name
original/ destination/
```

How to synchronize files using rsync?

- To sync or update files between two folders

```
rsync -avz original/ duplicate/
```

- Pull data from a remote system to local machine:

```
rsync -avz -e ssh user@remote_host:/path/to/source/
/path/to/local/destination/
```

- Push data from local file to a remote directory

```
rsync -avz /path/to/local/source/
user@remote_host:/path/to/remote/destination/
```

User scp to copy file between server

- Copy file from local machine to remote server

```
scp file_name
remoteuser@remotehost:/remote/directory
```

- Copy file from remote server to local machine

```
scp user@remotehost:/home/user/file_name
```

- Copy file from local machine to remote server by specify port

```
scp -P port file_name
remoteuser@remotehost:/remote/directory
```

Practical: Copy file to server using rsync and scp

- Copy two files to remote server via ssh protocol with specify port 2222
  - files:
    - 18_12.dv3.consensus.fasta.gz
    - 18_13.dv3.consensus.fasta.gz

```
$ rsync -Pavz -e "ssh -p 2222" 18_* kwan@localhost:~
```

```
$ scp -P 2222 18_* kwan@localhost:~/
```

- Copy file from remote server to local machine via ssh protocol with specify port 2222

```
$ rsync -avz -e "ssh -p 2222"
kwan@localhost:~/35_1.dv4.consensus.fasta.gz ./
```

```
$ scp -P 2222 kwan@localhost:~/"18_*" ./
```

Data Integrity

SHA and MD5 Checksums

- The two common checksum algorithms for checking data integrity are MD5 and SHA-1.
- The program for MD5 is ‘md5sum’ in Linux or ‘md5’ in OS X.
- The program for SHA-1 is ‘shasum’ or ‘sha1sum’.

Let’s perform checksums using SHA-1 and MD5

- SHA-1
  - Checksums with file input

```
$ sha1sum 12_8.dv2.consensus.fasta.gz
5f1805efc82dd3dad27ed5a72cf6591db28a72a9  12_8.dv2.consensus.fasta.gz
```

  - SHA-1 checksum file for all files in the directory

```
$ sha1sum dv_consensus/*.gz > consensus_checksum.sha
$ cat consensus_checksum.sha
```

  - Use shasum’s check option (-c) to validate that these files match the original versions

```
$ sha1sum -c consensus_checksum.sha
```

- MD5
  - Use the same syntax as ‘shasum’, but replace it with 'md5sum'.

Looking at Differences Between Data

- To check the differences between two files, you can use the "diff -u" command.

```
$ diff -u file1 file2
```

- Example:

```
$ diff -u gene-1.bed gene-2.bed
```

  - gene-1.bed is original file which prefixed by ---
  - gene-2.bed is modified file which prefixed by +++
  - + indicate a line has been added to the modified file.
  - - indicates lines removed in the modified file.

Compressing Data and Working with Compressed Data

gzip and gunzip

1. Compress result from standard input
   ```
   $ trimmer in.fastq.gz | gzip > out.fastq.gz
   ```
2. qzip and gunzip can output their results to standard out using option -c
   ```
   $ gzip -c in.fastq > in.fastq.gz
   $ gunzip -c in.fastq.gz > duplicate_in.fastq
   ```
3. Concatenate gzip compressed output directly to an existing gzip file.
   in.fastq.gz in2.fastq
   ```
   $ gzip -c in2.fastq >> in.fastq.gz
   ```
4. Compress multiple files piped from ‘cat’ command.
   ```
   $ cat in.fastq in2.fastq | gzip > in.fastq.gz
   ```

Working with Gzipped Compressed Files

- zgrep
- zdiff
- zcat
- zless
- zmore

Case study:
Reproducibly Downloading Data

Download genomic and sequence resources for mouse (Mus musculus)

- Here is an example of downloading the GRCm38 mouse reference genome and annotation.
  - Document the name and version of file
  - Document the command that used for downloading the file
  - Download checksum file and check with the downloaded data
  - Record the checksum
- Ex. README.txt

References

- wget
  - https://www.hostinger.com/tutorials/wget-command-examples/#:~:text=Downloading%20files%20with%20wget%20is,begin%20after%20you%20press%20enter.
- curl
  - https://www.hostinger.com/tutorials/curl-command-with-examples-linux/#What_Is_cURL_Command
  - https://www.tutorialspoint.com/linux-commands-comparison-curl-vs-wget#:~:text=Overview%20of%20curl%20and%20wget,supports%20HTTP%20and%20FTP%20protocols.
- rsync
  - https://www.hostinger.com/tutorials/how-to-use-rsync#:~:text=rsync%20is%20a%20powerful%20and,the%20source%20and%20destination%20files.
- scp
  - https://www.geeksforgeeks.org/scp-command-in-linux-with-examples/