AWS/UNIX 2025

Material by Jose Hector Galvez, Zhibin Lu & Rob Syme

Contents

Chapter 1

AWS/UNIX primer info

1.0.1 Data download

We will be using Amazon Web Services for the primer session. If you are not able to attend this session live, you can download the data we used here:

- data.zip
- gff3

You will need a UNIX environment (like a Linux/Mac computer or Windows Subsystem for Linux) to work through the material.

1.0.2 Schedule

RNA-seq and scRNA-seq Analysis 2025: Friday, October 31, 2025

(ADT)	Module
10:00	Welcome Introduction
10:10	Module 1: Logging into AWS
11:10	Short Break
11:15	Module 2: Introduction to the UNIX Command Line
11:35	Module 3: File Manipulation
12:05	Long Break
13:05	Module 4: Searching and Sorting Through Files

(ADT)	Module
14:00	Short Break
14:05	Module 5: Shell scripts
15:00	Finished

Chapter 2

Meet Your Faculty

2.0.1 Instructors

2.0.1.1 Dr. Gustavo Sganzerla Martinez (he/him)

Postdoctoral Fellow Dalhousie University Halifax, NS, Canada

— gustavo.sganzerla@dal.ca

Dr. Gustavo Sganzerla Martinez is a postdoctoral fellow at Dalhousie University. Gustavo's work sits at the very intersection between computer science and biology. By being fascinated by coding and how computers work, Gustavo develops computational solutions for biological problems.

2.0.2 Teaching Assistants

2.0.2.1 Sarah Russell (she/her)

Genetic Analysis Specialist IWK Health Halifax, Nova Scotia, Canada

— sarah.russell@iwk.nshealth.ca

Sarah Russell is a Genetic Analysis Specialist at IWK Health, where she applies bioinformatics tools to support the clinical interpretation of genomic data. Her work focuses on developing and maintaining NGS workflows that ensure the accuracy and quality of genetic test results and clinical reporting. Her previous work involved the development and validation of bioinformatic solutions for somatic profiling of solid tumor panels used in clinical diagnostics.

2.0.3 Facilitator

2.0.3.1 Ben Fisher (he/him)

Regional Coordinator, CBH Atlantic Canadian Bioinformatics Hub, Dalhousie University Halifax, NS, Canada

— atlantic@bioinformatics.ca

Ben has a Master of Science degree in Microbiology and Immunology, completing his bioinformatics training under Dr. Morgan Langille at Dalhousie University. Throughout his training, he has instructed others in genetics, molecular biology, and microbiome data science. Ben is passionate about continued education of trainees and professionals, and firmly believes that enhancing bioinformatics and computational biology competencies will support the success of Canada's current and future scientists.

Chapter 3

1. Logging into AWS

3.0.1 Logging into the Amazon Cloud During the Workshop

- These instructions will **ONLY** be relevant in class, as the Cloud will not be accessible from home in advance of the class.
- On the cloud, we're going to use the default username: **ubuntu**

3.0.2 Logging in with ssh (Mac/Linux)

- Make sure the permissions on your private key are secure. Use chmod on your downloaded pem file:

```
chmod 600 CBW.pem
```

- To log in to the instance, use the -i command line argument to specify your private key:

```
ssh -i CBW.pem ubuntu@##.uhn-hpc.ca
```

(where ## is your assigned student number.)

3.0.3 Logging in with OpenSSH client (Windows)

- Make sure OpenSSH client is installed on your machine. Settings > System > Optional features, then search for "OpenSSH" in your added features.

- open the "Command Prompt" or "Windows PowerShell" and run

```
C:\Windows\System32\OpenSSH\ssh.exe -i CBW.pem ubuntu@##.uhn-hpc.ca
```

3.1 File System Layout

When you log in, you'll notice that you have two directories: **CourseData** and **workspace**.

- The **CourseData** directory will contain the files that you'll need to complete your lab assignments.
- The **workspace** directory is where you will work on your labs. Files here will be accessible through your browser.

3.2 Workspace

- Everything created in your workspace on the cloud is also available by a web server on your cloud instance. Simply go to the following in your browser:

```
http://##.uhn-hpc.ca/
```

(where ## is your assigned student number.)

3.3 RStudio

- RStudio server is installed on your instance. It is accessible in your browser

```
http://##.uhn-hpc.ca:8080
```

(where ## is your assigned student number.) The username is ***ubuntu***. We will give you the password in class.

3.4 Jupyter Notebook

- Jupyter Notebook is installed on your instance. It is accessible in your browser

```
http://##.uhn-hpc.ca:8000
```

(where ## is your assigned student number.) We will give you the password in class.

Chapter 4

2. Introduction to the command line

4.0.1 Exercise: Exploring the filesystem

1. Connect to your AWS instance
2. Type the `ls` command - what is the output?

Solution

```
$ ls
CourseData  R  workspace
```

The `ls` command lists the contents of a working directory.

3. Type the `pwd` command - what is the output?

Solution

```
$ pwd
/home/ubuntu
```

The `pwd` command shows the absolute *path to the working directory*.

Chapter 5

3. File manipulation

5.0.1 Exercise: Reading text files

Using the commands you just learned, explore the .gff3 file in the Home directory.

1. Is the file a text file?

Solution

Yes. You can use `less`, `cat`, `head`, or `tail` and get human-readable info. Note that this doesn't have anything to do with its file extension.

2. How many lines does the file have?

Solution

```
$ wc -l GCF_009858895.2_ASM985889v3_genomic.gff
67 GCF_009858895.2_ASM985889v3_genomic.gff
```

There are 67 lines in this file.

3. Can you read the content of the file using less?

Solution

```
$ less GCF_009858895.2_ASM985889v3_genomic.gff
```

5.0.2 Exercise: Editing text files

Using the commands you just learned, create a file called helloworld.txt and edit it using nano.

1. Write "Hello world" into the file. Save the file and exit nano.
2. Create a subdirectory called "test"; move the helloworld.txt file into test.
3. Create a copy of the helloworld.txt file called helloworld2.txt

Solution

1. First, use the `nano` command to open a file called `helloworld.txt`

```
$ nano helloworld.txt
```

Inside the nano editor, write "Hello world", then use the `^O` option to write the changes and `^X` to exit.

2. Create a subdirectory called "test"; move the helloworld.txt file into test.

Use the command `mkdir` to create this new directory. Then, use `mv` to move `helloworld.txt` into this directory.

```
$ mkdir test
$ mv helloworld.txt test/
```

3. Create a copy of the `helloworld.txt` file called `helloworld2.txt`.

Change the working directory using `cd`, then use the `cp` command to create the copy.

```
$ cd test
$ cp helloworld.txt helloworld2.txt
```

5.0.3 Reference material

Here are two cheat-sheets that can be useful to have as a reference for common UNIX/Linux commands:

- FOSSwire.com Unix/Linux Command Reference
- SUSO.org Unix/Linux Command Syntax and Reference

Chapter 6

4. Searching and sorting through files

6.0.1 Setup

We want to explore some data on our instances, but it is a compressed ZIP file, so we'll need to unzip it first. First move into your home directory, and then unzip the file named data.zip:

```
cd ~
unzip data.zip
cd data
```

Take a look at what's in your home directory after unzipping using ls:

```
ls
```

```
GCF_009858895.2_ASM985889v3_genomic.gff  __MACOSX  data  data.zip
```

Note that the original zipped file (data.zip) is still present, but there is now a folder (data) that you can enter. There is also a file called ___MACOSX. This is because the folder was zipped using a Mac; you can safely ignore it here.

6.1 Data Exploration

We'll begin by looking at files are in Genbank gbff. This is a text file that describes the nucleotide sequence and annotation features on those nucleotide

sequences. First of all, we run the `ls` command to view the names of the files in the `ecoli_genomes` directory:

```
$ ls genomes
```

```
atlanta.gbff      braunschweig.gbff lab-strain.gbff   london.gbff       muenster.gbff
```

Let's go into that directory with `cd` and run an example command `wc london.gbff`:

```
$ cd genomes
$ wc london.gbff
```

```
212462  899913 13993868 london.gbff
```

`wc` is the 'word count' command: it counts the number of lines, words, and characters in files (from left to right, in that order).

If we run the command `wc *.gbff`, the `*` wildcard matches zero or more occurences of any character, so the shell turns `*.gbff` into a list of all gbff files in the current directory:

```
$ wc *.gbff
```

```
  217040  915629 14251456 atlanta.gbff
  204528  837084 13332241 braunschweig.gbff
  191859  685480 11851718 lab-strain.gbff
  212462  899913 13993868 london.gbff
  202036  823021 13167730 muenster.gbff
  216788  883045 14082598 nevada.gbff
  208514  849571 13555200 texas.gbff
 1453227 5893743 94234811 total
```

Note that `wc *.gbff` also shows the total number of all lines in the last line of the output.

If we run `wc -l` instead of just `wc`, the output shows only the number of lines per file:

```
$ wc -l *.gbff
```

```
  217040 atlanta.gbff
  204528 braunschweig.gbff
  191859 lab-strain.gbff
  212462 london.gbff
  202036 muenster.gbff
  216788 nevada.gbff
  208514 texas.gbff
 1453227 total
```

Which of these files contains the fewest lines? It's an easy question to answer when there are only six files, but what if there were 6000? Our first step toward a solution is to run the command:

```
$ wc -l *.gbff > lengths.txt
```

The greater than symbol, `>`, tells the shell to redirect the command's output to a file instead of printing it to the screen. (This is why there is no screen output: everything that wc would have printed has gone into the file `lengths.txt` instead.) The shell will create the file if it doesn't exist. If the file exists, it will be silently overwritten, which may lead to data loss and thus requires some caution. `ls lengths.txt` confirms that the file exists:

```
$ ls lengths.txt
lengths.txt
```

We can now send the content of `lengths.txt` to the screen using `cat lengths.txt`. The cat command gets its name from 'concatenate' i.e. join together, and it prints the contents of files one after another. There's only one file in this case, so cat just shows us what it contains:

```
$ cat lengths.txt
```

```
  217040 atlanta.gbff
  204528 braunschweig.gbff
  191859 lab-strain.gbff
  212462 london.gbff
  202036 muenster.gbff
  216788 nevada.gbff
  208514 texas.gbff
 1453227 total
```

6.2 Sorting

The `sort` command rearranges the lines in a file in order. There are different methods of sorting - lexigraphically (a-z1-9) or numerically. The default sort type is lexigraphically, where numbers are treated one character at a time. Given a the file "numbers.txt" that looks like:

```
cd ../sorting
cat numbers.txt
```

```
10
2
19
22
6
```

If we run `sort` on this file:

```
sort numbers.txt
```

```
10
19
2
22
6
```

If we run `sort -n` on the same input - specifying that we want to sort numerically, we get this instead:

```
sort -n numbers.txt
```

```
2
6
10
19
22
```

Explain why `-n` has this effect.

Solution

The -n option specifies a numerical rather than an alphanumerical sort.

We will sort our lengths.txt file using the `-n` option to specify that the sort is numerical instead of alphanumerical. Note that running sort does not modify the file; instead, it sends the sorted result to the screen:

```
cd ../genomes
$ sort -n lengths.txt
```

```
  191859 lab-strain.gbff
  202036 muenster.gbff
  204528 braunschweig.gbff
  208514 texas.gbff
  212462 london.gbff
  216788 nevada.gbff
  217040 atlanta.gbff
 1453227 total
```

We can put the sorted list of lines in another temporary file called `sorted-lengths.txt` by putting `> sorted-lengths.txt` after the command, just as we used `> lengths.txt` to put the output of `wc` into `lengths.txt`. Once we've done that, we can run another command called `head` to get the first few lines in sorted-lengths.txt:

```
$ sort -n lengths.txt > sorted-lengths.txt
$ head -n 1 sorted-lengths.txt
```

```
  191859 lab-strain.gbff
```

Using `-n 1` with head tells it that we only want the first line of the file; `-n 20` would get the first 20, and so on. Since `sorted-lengths.txt` contains the lengths of our files ordered from least to greatest, the output of head must be the file with the fewest lines.

6.2.1 What Does » Mean?

We have seen the use of `>`, but there is a similar operator `>>` which works slightly differently. We'll learn about the differences between these two operators by printing some strings. We can use the echo command to print strings e.g.

```
$ echo The echo command prints text
```

```
The echo command prints text
```

Now test the commands below to reveal the difference between the two operators:

```
$ echo hello > testfile01.txt
```

and:

```
$ echo hello >> testfile01.txt
```

6.2.2 Running Commands Together

If you think this is confusing, you're in good company: even once you understand what `wc`, `sort`, and `head` do, all those intermediate files make it hard to follow what's going on. We can make it easier to understand by running sort and head together:

```
$ sort -n lengths.txt | head -n 1
```

```
  191859 lab-strain.gbff
```

The vertical bar, `|`, between the two commands is called a pipe. It tells the shell that we want to use the output of the command on the left as the input to the command on the right.

Nothing prevents us from chaining pipes consecutively. That is, we can for example send the output of `wc` directly to sort, and then the resulting output to head. Thus we first use a pipe to send the output of `wc` to `sort`:

```
$ wc -l *.gbff | sort -n
```

```
  191859 lab-strain.gbff
  202036 muenster.gbff
  204528 braunschweig.gbff
  208514 texas.gbff
  212462 london.gbff
  216788 nevada.gbff
  217040 atlanta.gbff
 1453227 total
```