Day 01 Workshop Instruction Manual

Disclaimer: The primary goal of this workshop is to explore how to use a high-performance computing (HPC) environment. We will use various bioinformatics tools as examples to understand the cluster environment. Please be sure to consult the application manual for each tool, and choose the options and flags that are most appropriate for your own research question.

Table of Contents

I. Connect to the cluster via SSH

Learn how to securely log in to a remote Linux cluster using **SSH (Secure Shell)** from **Windows** or **macOS/Linux**.

a. Open a Terminal

Use a terminal or command-line window to connect to the server.

- *Windows:*
  Open Command Prompt, PowerShell, or Windows Terminal.
  (Click Start → type "cmd" or "PowerShell" → press Enter.)
- *macOS:*
  Press Command + Space, type Terminal, and press Enter.

b. Connect to the Remote Server

In the terminal, type:

```
ssh user01@omics.c3.ca
```

Replace **`user01`** with your assigned username.

c. Verify the Server's Authenticity (First-Time Connection)

When we connect for the first time, we will see a message like this:

```
The authenticity of host 'omics.c3.ca
(206.12.93.206)' can't be established.
ED25519 key fingerprint is
SHA256:ZtOn1jn7PJRwxue5mt339GlCFzAVgWpCSaz984s6gpY.
Are you sure you want to continue connecting
(yes/no/[fingerprint])?
```

Type:

`yes`

This tells SSH to trust the server and saves its identity in your known hosts list for future logins.

d. Enter Your Password

We will be prompted:

```
user01@omics.c3.ca's password:
```

Type your password and press Enter.

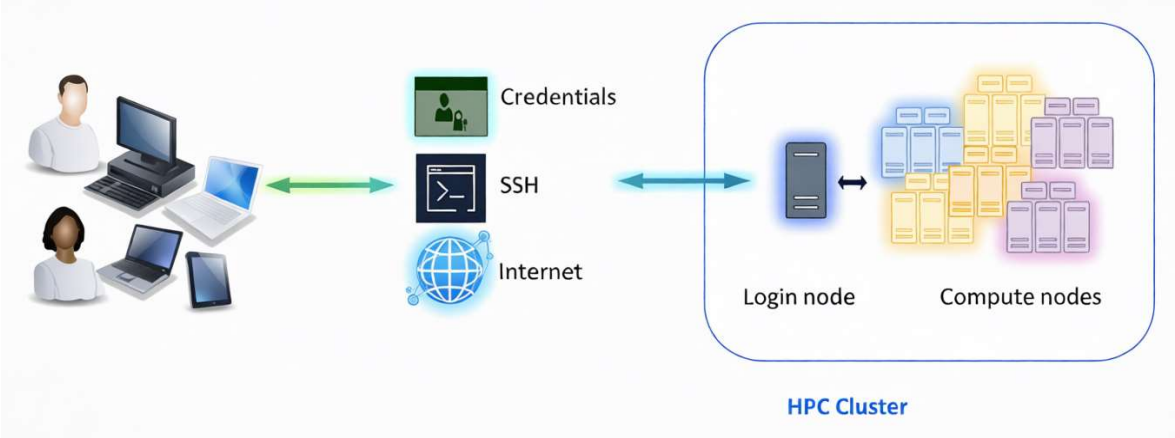
Note: Nothing will appear as we type — this is normal for password security.

e. Successful Login

Once authenticated, we will see something like this:

```
##################################################
##
Welcome to the 'OMICS' Cluster!
##################################################
##

We are glad to have you here!

This cluster has been set up specifically for the
OMICS Workshop (November 18-20, 2025).

1. All data generated during the workshop will
remain available **until Sunday, November 23,
2025**. Please remember to **download any files or
scripts** you would like to keep before that date.

2. This workshop is designed to help you learn
**how to efficiently use an HPC cluster** by
exploring some bioinformatics application along the
way. Please be sure to consult the application
manuals for each tool and choose the options and
flags that best suit your research questions.

3. You will have access to the workshop handouts
each day before the session begins.

If you have any questions, feel free to reach out
to us.

Happy learning and computing!

- The OMICS Workshop Team
Last login: Tue Oct 7 16:16:05 2025 from
137.82.20.161
[user01@login1 ~]$
```

We are now connected to the remote Linux cluster.


Credentials
SSH
Internet
Login node
Compute nodes
HPC Cluster

II. Understand the Command Prompt:

Once logged in, this is our shell prompt.

```
[user01@login1 ~]$
```

It provides useful context about our session:

Part	Meaning	Description
user01	Username	The account we are logged in with.
login1	Hostname	The name of the cluster node we are connected to. (login1 is one of the nodes for omics.c3.ca.)
~	Home Directory	The tilde (~) indicates our home directory on the cluster.
$	Dollar Symbol	Shows us that the shell is waiting for input

Example:
The prompt changes as we move through directories. For example:

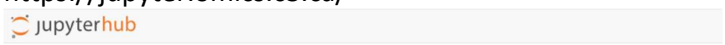
```
[user01@login1 scratch]$
```

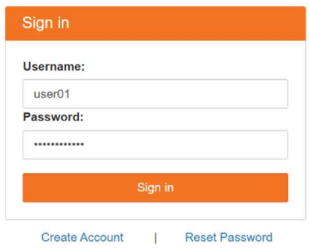
means we are now inside a directory named `scratch`.

III. Understand Graphical User Interface

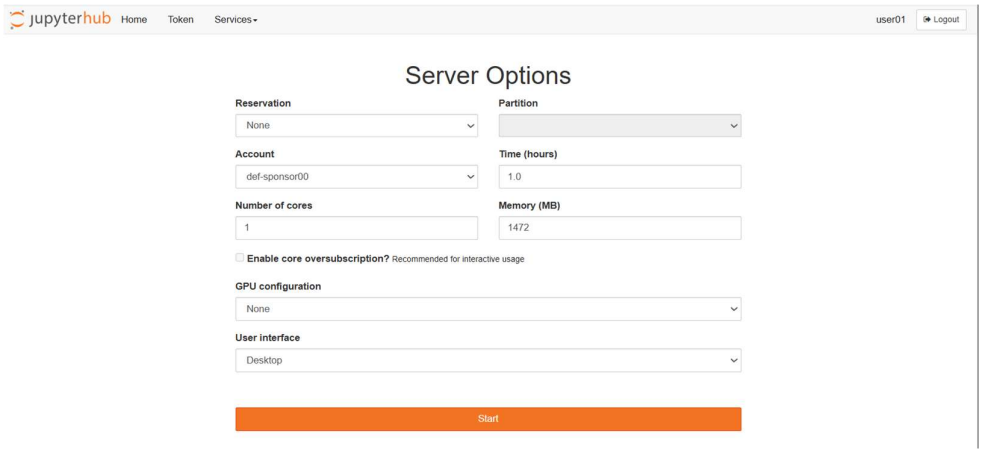
Let us open an internet browser – Google Chrome or Safari and point it to the link below:

https://jupyter.omics.c3.ca/

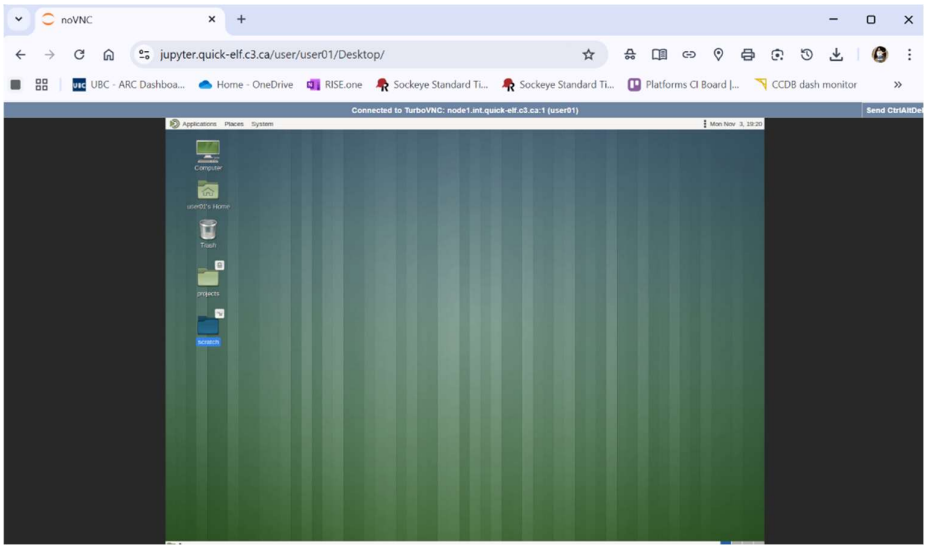




Then, click on Sign-in, to land on the following page to request the resources:



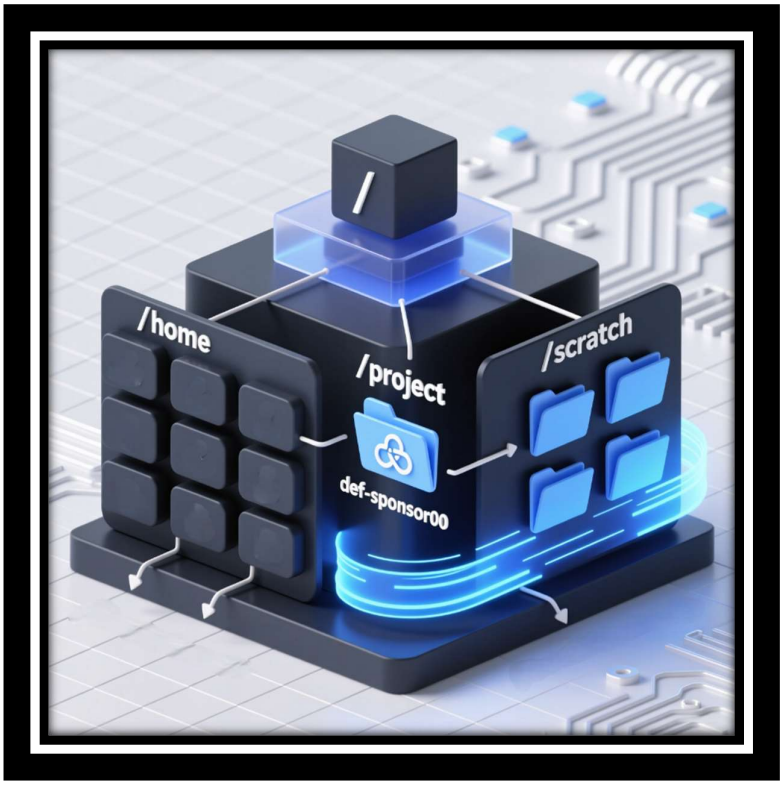
Click on ‘start’ to launch the Desktop



IV. Navigating Files and Directories in Linux

Once logged in via SSH, we will need to move around the filesystem to access our files, shared datasets, and computation space.

Linux organizes all files in a hierarchical tree structure, starting from the root directory /



Let's try it out on the cluster….

```
/
├── home
│   └── user01          ← Personal home directory
├── project
│   └── def-sponsor00   ← Shared space for datasets & results
└── scratch
    └── user01          ← Temporary read-write space for computations
```

V. Setting Up the Project Directory Structure

a. Navigate to the Working Directory

On the terminal, run the following command:

```
$ cd /home/user01/scratch
```

Note: We will replace user01 with our own user name.

---

b. Create the Workshop Directory

```
$ mkdir omics_workshop
$ cd omics_workshop
```

This ensures everyone is starting in a clean, directory named **omics_workshop**.

c. Create the Subdirectory Structure (Step-by-Step)

Run the following commands in sequence. We may choose to type out (to reinforce understanding of each part) or copy-paste one line at a time.

```
# Create input_data directory and its subdirectories
$ mkdir -p input_data/raw_data
$ mkdir -p input_data/metadata


# Create all other directories
$ mkdir preprocessed_data
$ mkdir scripts
$ mkdir output_data
$ mkdir references
$ mkdir figures
```

After running these, the directory tree should look like:

```
.
└── omics_workshop
    ├── figures
```

```
├── input_data
│   ├── metadata
│   └── raw_data
├── output_data
├── preprocessed_data
├── references
└── scripts
```

(Optional) Combine into a Single Command

```
$ mkdir -p input_data/raw_data input_data/metadata \
preprocessed_data scripts output_data references \
figures
```

d. Verify the Directory Structure

To verify everything was created correctly:

```
$ tree .
```

We should see something like:

```
.
├── figures
├── input_data
│   ├── metadata
│   └── raw_data
├── output_data
├── preprocessed_data
├── references
└── scripts
```

If `tree` is not installed, we can use:

```
$ find . -type d | sort
```

We should see something like:

```
.
./figures
./input_data
./input_data/metadata
./input_data/raw_data
./output_data
./preprocessed_data
./references
./scripts
```

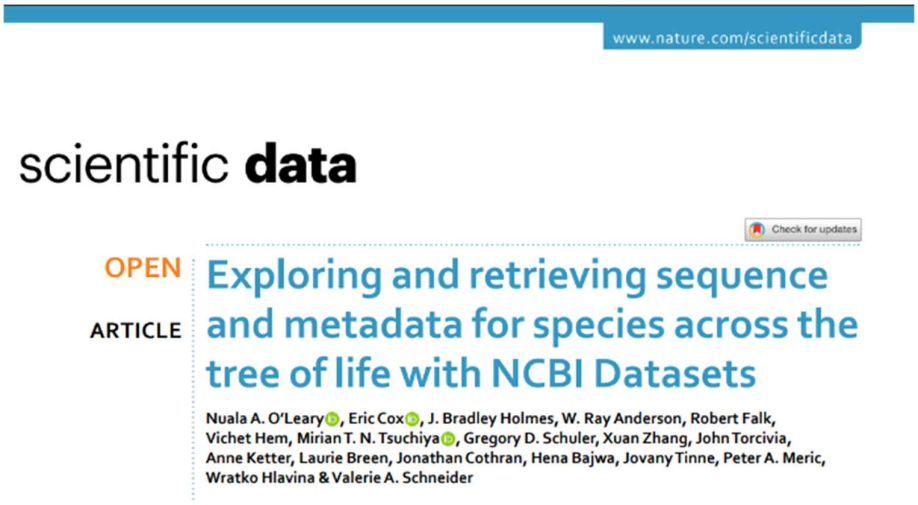
Exercise 1: Access Genomic Data from public repositories

In this exercise, we will use the `datasets` and `dataformat` command-line tools to retrieve and process data from NCBI. We will extract one representative (the longest) protein sequence per gene from a set of orthologs

There are many repositories for public data. Some model organisms or fields have specific databases. Two of the most comprehensive public repositories are:

National Center for Biotechnology Information (NCBI)
European Bioinformatics Institute (EMBL-EBI)

We will be using both NCBI's Sequence Read Archive (SRA) database and the EMBL-EBI's Nucleotide Archive (ENA) for this workshop.



a. Download and Install the tool in a cluster

```
$ cd /home/user01/scratch/omics_workshop/scripts

$ curl -o datasets \
'https://ftp.ncbi.nlm.nih.gov/pub/datasets/command-line/v2/linux-amd64/datasets'

$ curl -o dataformat \
'https://ftp.ncbi.nlm.nih.gov/pub/datasets/command-line/v2/linux-amd64/dataformat'
```

```
$ ls -l
total 38052
-rw-r-----. 1 user01 user01 19578600 Oct 20 19:57 dataformat
-rw-r-----. 1 user01 user01 19382506 Oct 20 19:56 datasets

$ chmod +x datasets dataformat

$ ls -l
total 38052
-rwxr-x---. 1 user01 user01 19578600 Oct 20 19:57 dataformat
-rwxr-x---. 1 user01 user01 19382506 Oct 20 19:56 datasets

$ export \
PATH=/home/user01/scratch/omics_workshop/scripts:$PATH
```

- Did both files download successfully?
- Are they executable? (Use ls -l datasets dataformat to check)

b. Download Metadata for Orthologs

Navigate to the metadata directory where we want to download the metadata:

```
$ cd /home/user01/scratch/omics_workshop/input_data
$ ls -l
total 0
drwxr-x---. 2 user01 user01 6 Oct 14 17:49 metadata
drwxr-x---. 2 user01 user01 6 Oct 14 17:49 raw_data

$ cd metadata
$ mkdir metadata_01
$ mkdir metadata_02 # We will use this tomorrow
$ cd metadata_01
```

We will focus on the gene **BRCA1** and its orthologs in human, mouse, and ferret.

```
$ datasets summary gene symbol brca1 \
  --ortholog 'homo sapiens,mus musculus,mustela putorius furo' \
--report product --as-json-lines > brca1_orthologs.jsonl
```

- Does the file `brca1_orthologs.jsonl` exist?
- How many lines does it have? (`wc -l brca1_orthologs.jsonl`)

c. Convert Metadata into Tabular Format

```
$ dataformat tsv gene-product --inputfile brca1_orthologs.jsonl \
 --fields gene-id,tax-name,symbol,transcript-accession,\
transcript-length,transcript-protein-accession,transcript-protein-length \
> transcript_protein.tsv
```

- View the first few rows: head -5 transcript_protein.tsv

e. Identify the Longest Protein per Gene

In this step, we will begin by organizing our workspace and preparing to extract information about the longest protein for each gene.

We will first create a directory where our input data will be downloaded and stored.

```
$ mkdir \
/home/user01/scratch/omics_workshop/input_data/raw_data/raw_data_01


$ ls /home/user01/scratch/omics_workshop/input_data/raw_data/
```

Now, let us use a few Linux commands to start exploring the input file (`transcript_protein.tsv`)

```
$ tail -n +2 transcript_protein.tsv | \
cut -f1 | \
sort -u | \
while read GENE_ID;
do
  echo $GENE_ID;
done
```

Let us take a closer look at what this code does

1. `tail -n +2 transcript_protein.tsv`
   - The `tail` command is used here with `-n +2` to *start reading the file from line 2 onward*, i.e., skipping the first (header) line.
   - So, this takes the file `transcript_protein.tsv` and emits all lines except the header.

2. `| cut -f1`
   - The pipe | takes the output of the previous command and uses it as input for cut.
   - cut -f1 extracts field 1 (i.e., the first column) from each line (columns are tab-delimited by default).
   - So now we have a list of all the “GENE_ID” (first column) from the data, one per line (still may have duplicates).

3. `| sort -u`
   - `sort` arranges lines in sorted order. The `-u` option means "unique" — i.e., remove duplicates.
   - As a result, we now have a **unique list of GENE_IDs**, one line each.

4. `| while read GENE_ID; do … done`
   - Now for each `GENE_ID` (that is, for each unique gene), the script enters the loop body.
   - Everything inside the `do … done` will be performed **once per GENE_ID**.

Next, we will add the commands that should be executed for each **GENE_ID**

```
$ tail -n +2 transcript_protein.tsv | \
cut -f1 | \
sort -u | \
while read GENE_ID;
do
  grep -w "^$GENE_ID" transcript_protein.tsv | \
    sort -t$'\t' -nr -k7 | \
    head -n1 | \
                cut -f4,6 | \
               tr '\t' '\n' >>
/home/user01/scratch/omics_workshop/input_data/raw_data/raw_
data_01/longest.list
done
```

Now, let us explore what is happening in this section of the code.

5. Inside the loop:
   - `grep -w "^$GENE_ID" transcript_protein.tsv`
     - `grep -w` means "match the whole word" exactly; `^$GENE_ID` means "match lines starting with this gene ID".
     - This finds **all lines in the file** that belong to that particular gene.

   - `| sort -t$'\t' -nr -k7`
     - This sorts those lines **numerically** (`-n`), in **reverse order** (`-r`), based on **field 7** (`-k7`).
     - The `-t$'\t'` tells `sort` that the delimiter is a tab character.
     - In the metadata, field 7 is the "protein length" (the length of the protein sequence) — so sorting highest to lowest means the first line after this sort will represent the **longest protein isoform** for that gene.

- `| head -n1`

  Take only the first line of that sorted list. That line corresponds to the longest protein/transcript pair for that gene.

- `| cut -f4,6`
  - From that line, `cut` fields 4 and 6. Field 4 = "transcript accession", field 6 = "transcript-protein accession" in our earlier table.
  - So, we extract those two accessions.

- `| tr '\t' '\n'`

  `tr` (translate) replaces the tab (`\t`) between the two accessions with a newline (`\n`), so we output one per line.

- `>> /home/user01/scratch/omics_workshop/input_data/raw_data/raw_data_01/longest.list`

  Append (>>) these two lines (transcript accession + protein accession) to the file `longest.list` in the specified directory.

- What's in longest.list? (cat longest.list)
- How many accessions are listed?
- Do they match the genes we targeted?

```
$ cat longest.list
      XM_004772608.3
      XP_004772665.1
      XM_030245495.2
      XP_030101355.1
      NM_001407582.1
      NP_001394511.1
```

What is an accession number?
An accession number is a unique identifier assigned by the National Center for Biotechnology Information (NCBI) RefSeq project to each curated or predicted sequence record (mRNA, protein, transcript, genome). For example: `NM_001407582.1`.

Every accession ends with a "dot + version number" (here: `.1`), which increases when the sequence record is updated.

Why prefixes like "NM_", "XM_", "XP_" matter
These prefixes tell us both the molecule type (mRNA transcript, protein sequence etc.) and the curation status (known vs. predicted/model) of the sequence. According to NCBI's documentation:

"Model RefSeq records … have accession prefixes XM_, XR_, and XP_ … Known RefSeq records … use prefixes NM_, NR_, and NP_."

Here is a quick reference table:

Prefix	Molecule Type	Meaning
NM_	mRNA transcript (protein-coding)	Known RefSeq
NP_	protein sequence	Known RefSeq
XM_	mRNA transcript (predicted/model)	Model RefSeq
XP_	protein sequence (predicted/model)	Model RefSeq

Examples (from our list) & what they tell us

- XM_004772608.3 → a **predicted mRNA transcript** (model RefSeq, version .3)

- XP_004772665.1 → a **predicted protein sequence** (model RefSeq, version .1)
- XM_030245495.2 → predicted mRNA transcript (version .2)
- XP_030101355.1 → predicted protein sequence (version .1)
- NM_001407582.1 → a **curated (known) mRNA transcript** (version .1)
- NP_001394511.1 → a **curated (known) protein sequence** (version .1)

f. Download the Selected Sequences

```
$ datasets download gene accession \
  --inputfile longest.list \
  --fasta-filter-file longest.list \
  --filename longest.zip
```

Unzip and inspect:

```
$ unzip longest.zip
$ ls ncbi_dataset/data/
```

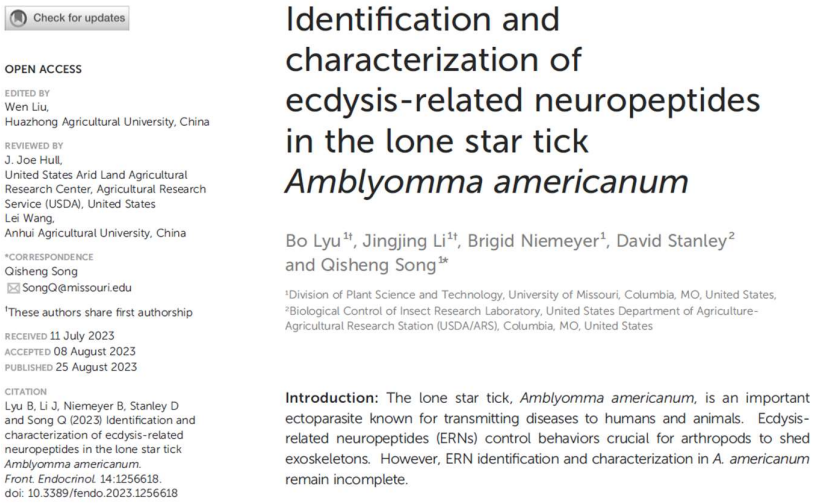
We will see:

`protein.faa` → one protein sequence per gene
`rna.fna` → corresponding transcript sequences

- Are the files present and check the file size?
- View the first few lines of `protein.faa` to inspect the FASTA headers.

Exercise 2: Examining Data on the NCBI SRA and ENA Database

We will use the project under accession PRJNA982785 that corresponds to the study described in Lyu et al. (2023) (Identification and characterization of ecdysis-related neuropeptides in the lone star tick *Amblyomma americanum* — PMCID: PMC10490126).



Check for updates

OPEN ACCESS

EDITED BY
Wen Liu,
Huazhong Agricultural University, China

REVIEWED BY
J. Joe Hull,
United States Arid Land Agricultural Research Center, Agricultural Research Service (USDA), United States
Lei Wang,
Anhui Agricultural University, China

*CORRESPONDENCE
Qisheng Song
SongQ@missouri.edu

†These authors share first authorship

RECEIVED 11 July 2023
ACCEPTED 08 August 2023
PUBLISHED 25 August 2023

CITATION
Lyu B, Li J, Niemeyer B, Stanley D and Song Q (2023) Identification and characterization of ecdysis-related neuropeptides in the lone star tick *Amblyomma americanum*. *Front. Endocrinol.* 14:1256618. doi: 10.3389/fendo.2023.1256618

Identification and characterization of ecdysis-related neuropeptides in the lone star tick *Amblyomma americanum*

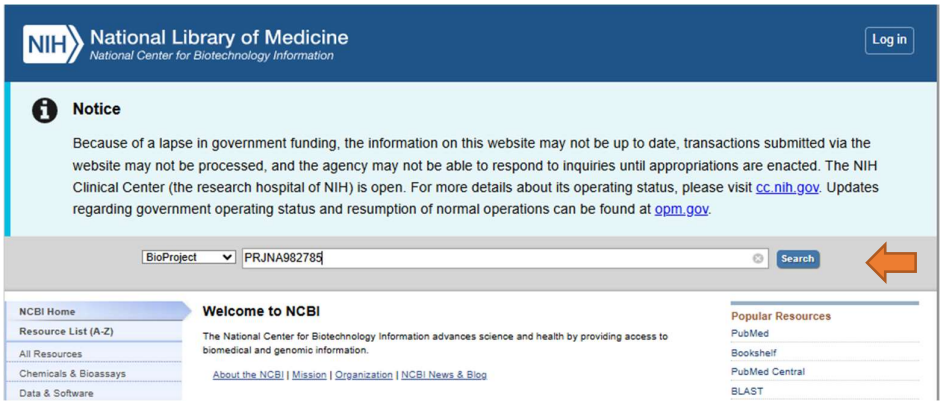
Bo Lyu[1†], Jingjing Li[1†], Brigid Niemeyer[1], David Stanley[2] and Qisheng Song[1*]

[1]Division of Plant Science and Technology, University of Missouri, Columbia, MO, United States, [2]Biological Control of Insect Research Laboratory, United States Department of Agriculture-Agricultural Research Station (USDA/ARS), Columbia, MO, United States

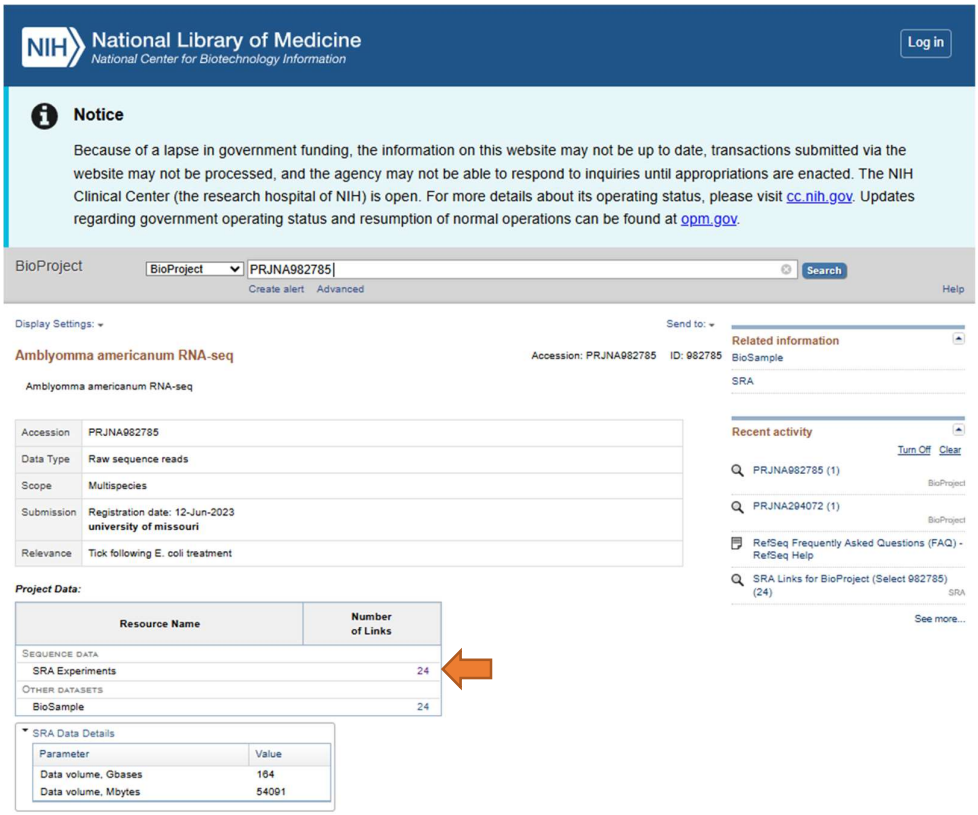
Introduction: The lone star tick, *Amblyomma americanum*, is an important ectoparasite known for transmitting diseases to humans and animals. Ecdysis-related neuropeptides (ERNs) control behaviors crucial for arthropods to shed exoskeletons. However, ERN identification and characterization in *A. americanum* remain incomplete.

The research article references **BioProject PRJNA982785**.

1. Let us launch the National Center for Biotechnology Information (NCBI) website and search for *"PRJNA982785"*. This will take us to the BioProject page for that accession.

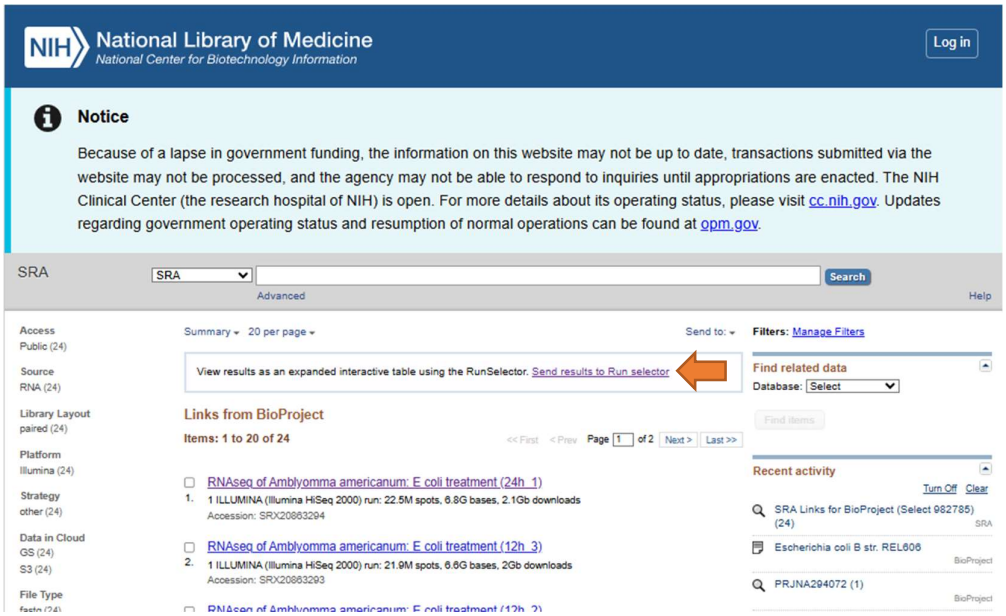


2. Once on the BioProject page, look for the section or table labelled something like "Project Data". This will show a description, and a table **"Project Data"** that has a link to the 24 SRA files.



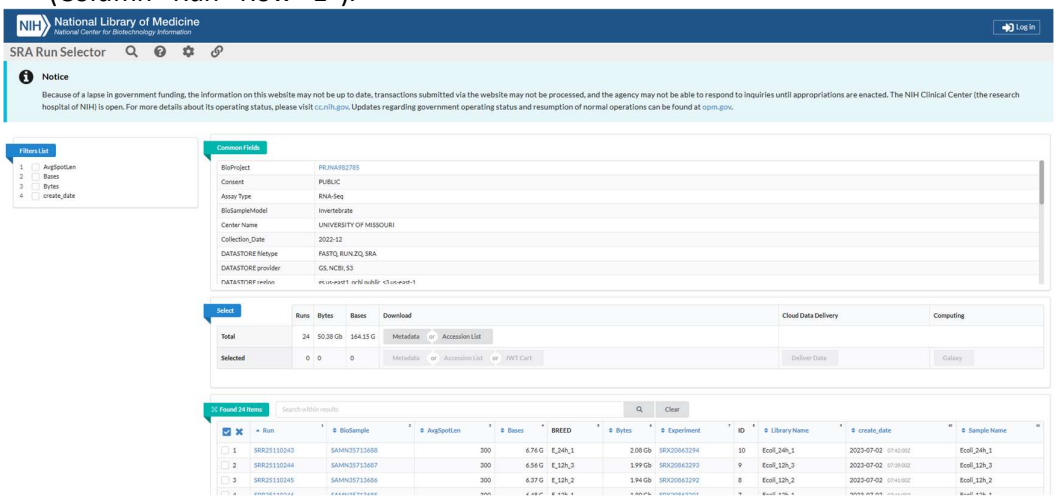
In that table we will find links associated with PRJNA982785 (for example, individual sample series or SRA experiments).

3. Click on the number “24” next to “SRA Experiments” and it will take us to the SRA page. From there we can access all of the raw sequence run entries associated with this BioProject.



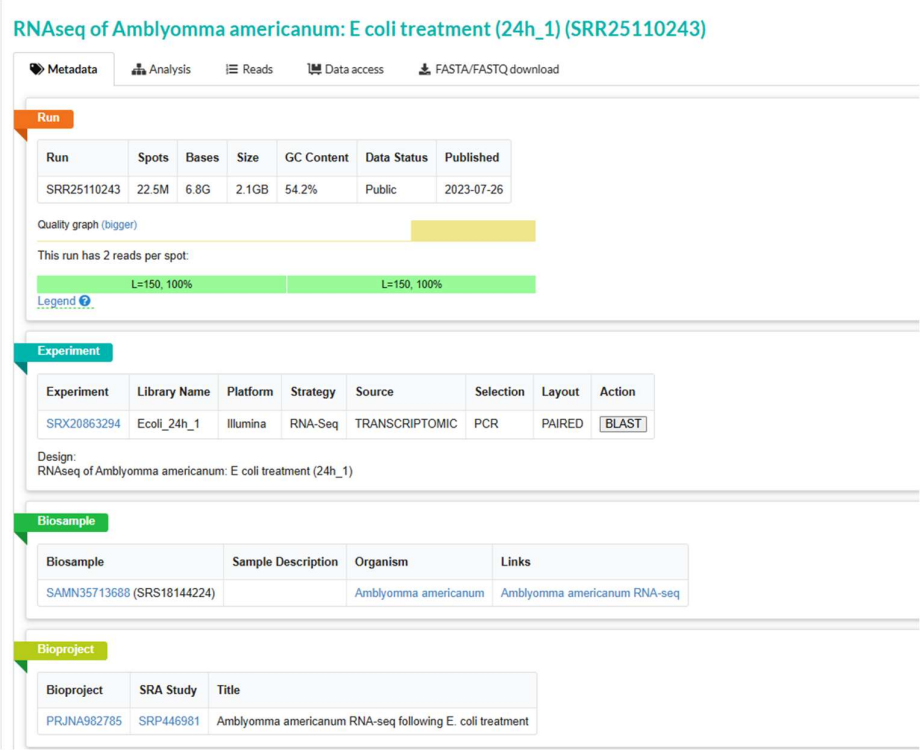
4. For a more organized table, select "Send results to Run selector". This takes us to the Run Selector page for BioProject PRJNA982785.

   Notice on this page there are three sections. "Common Fields" "Select", and "Found 24 Items". Within "Found 24 Items", click on the first Run Number (Column "Run" Row "1").

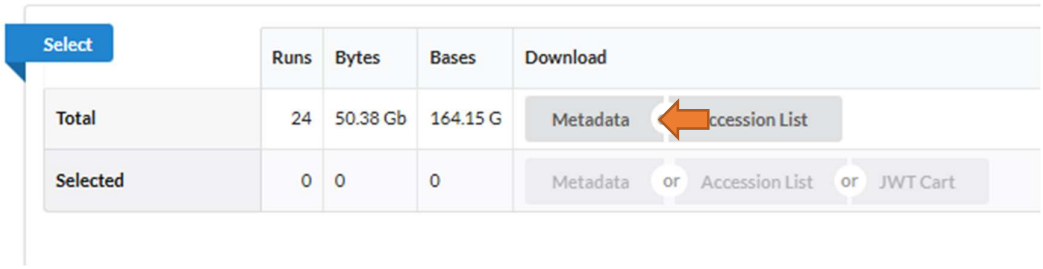


5. Download the SRA data from the SRA Run Selector Table.

This will take us to a page that is a run browser. We will examine some of the descriptions on the page.



6. Use the browser's back button to go back to the 'previous page'. As shown in the figure below, the second section of the page ("Select") has the **Total** row showing us the current number of "Runs", "Bytes", and "Bases" in the dataset to date. On 2025-10-21 there were 24 runs, 50.38 Gb data, and 164.15 G bases of data.



7. Click on the "Metadata" button to download the data for this lesson. The filename is "SraRunTable.csv" and save it on our computer Desktop.

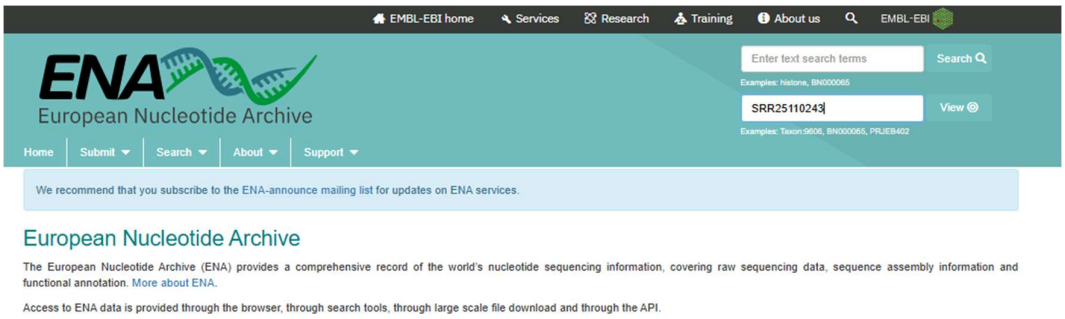
We should now have a file called `SraRunTable.csv` on our desktop.

Review the SraRunTable metadata in a spreadsheet program

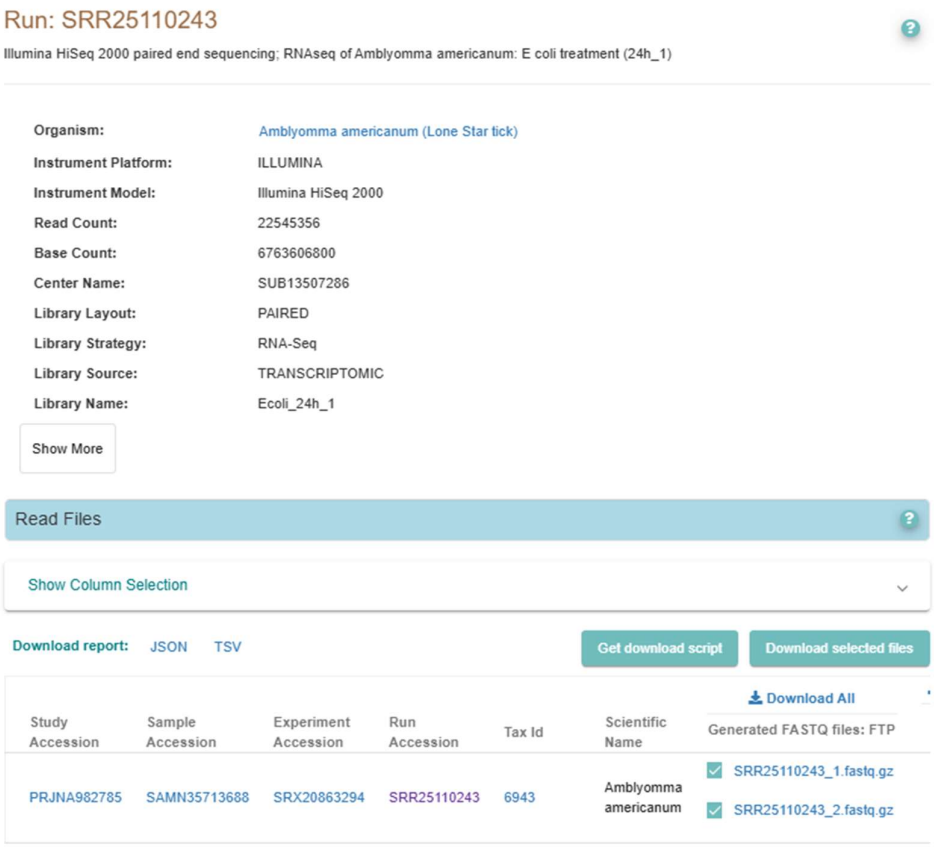
Using any spreadsheet program, open the SraRunTable.csv file

8. Download sequencing files
    a. From SRA: The SRA does not support direct download of fastq files from its webpage.

    b. From ENA: Let us navigate to the European Nucleotide Archive (ENA). Near the top right, in the box next to “View”, type in SRR25110243 and click the “View” button.



This will take us to a page with information about the data. Near the bottom we will have the option to download the data by FTP. We could download the. fastq read files here, but we do not need to download these files right now and they are large. Alternatively, right click and copy the URL to save it for later.



It is not recommended to download large numbers of sequencing files this way. For that, the NCBI has made a software package called the sra-toolkit.

9. Using the SRA Toolkit on the cluster
   In this section, we will learn how to download raw sequencing data from the NCBI Sequence Read Archive (SRA) using the **SRA Toolkit** on a cluster.

   The dataset we will use is from the BioProject PRJNA982785 (see the reference publication: *PMID 10490126*). This project includes high-throughput sequencing runs that can be retrieved directly via their SRA accessions.

1. Set Up our Working Directory

Navigate to our workshop input data directory and create a new directory for this exercise.

```
$ cd \
/home/user01/scratch/omics_workshop/input_data/raw_data/

$ mkdir raw_data_02
$ cd raw_data_02
```

This keeps our data organized and separate from other exercises.

2. *Check and Load the SRA Toolkit Module*

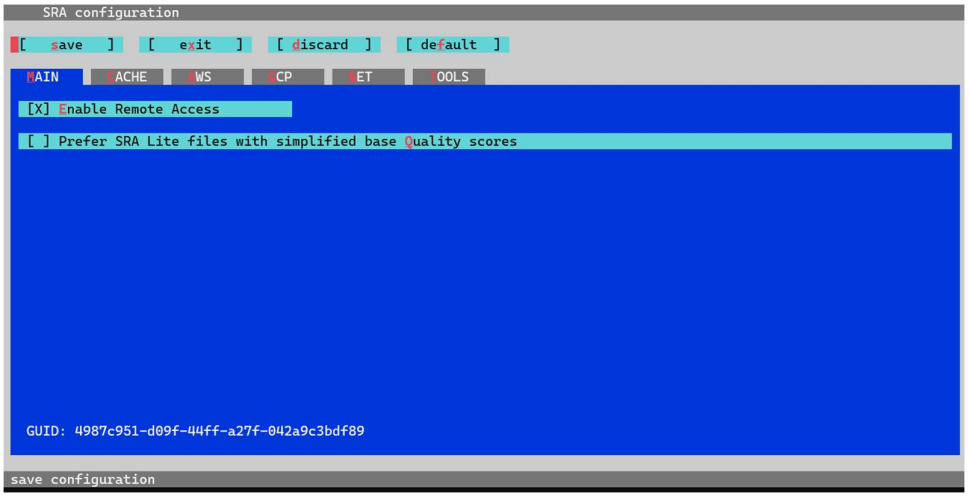
Check if the **SRA Toolkit** module is available on our system and then load it.

```
$ module spider sra-toolkit
$ module spider sra-toolkit/3.0.0
$ module load StdEnv/2020 gcc/9.3.0
$ module load sra-toolkit/3.0.0
```

This prepares our environment to run SRA Toolkit commands such as `prefetch` and `fasterq-dump`.

This version of sra-toolkit prompts to run

$ vdb-config --interactive



3. *Download the SRA Run*

Use the `prefetch` command to download the raw SRA data file.
Here we will use **SRR25110243**, one of the runs from BioProject PRJNA982785.

```
$ prefetch SRR25110243 &
```

#During download, we will see progress messages similar to:

```
2025-10-22T17:59:38 prefetch.3.0.9: Current preference is set to retrieve SRA
Normalized Format files with full base quality scores.
2025-10-22T17:59:38 prefetch.3.0.9: 1) Downloading 'SRR25110243'...
2025-10-22T17:59:38 prefetch.3.0.9: SRA Normalized Format file is being retrieved,
if this is different from your preference, it may be due to current file
availability.
2025-10-22T17:59:38 prefetch.3.0.9:  Downloading via HTTPS...
2025-10-22T18:46:57 prefetch.3.0.9:  HTTPS download succeed
2025-10-22T18:47:10 prefetch.3.0.9:  'SRR25110243' is valid
2025-10-22T18:47:10 prefetch.3.0.9: 1) 'SRR25110243' was downloaded successfully
2025-10-22T18:47:10 prefetch.3.0.9: 'SRR25110243' has 0 unresolved dependencies
```

We can monitor the download progress as:

```
$ ls -l SRR25110243/
```

```
total 2031236
-rw-r-----. 1 user01 user01          0 Oct 22 17:59 SRR25110243.sra.lock
-rw-r-----. 1 user01 user01         48 Oct 22 18:25 SRR25110243.sra.prf
-rw-r-----. 1 user01 user01 1204742645 Oct 22 18:25 SRR25110243.sra.tmp
```

Once completed, we will have an `.sra` file:

```
$ ls -lh SRR25110243/
total 2.1G
-rw-r-----. 1 user01 user01 2.1G Oct 22 18:46 SRR25110243.sra
```

```
$ ls -lh
total 0
drwxr-x---. 2 user01 user01 29 Oct 22 18:47 SRR25110243
```

4. *Convert the .sra File to FASTQ Format*

Next, use fasterq-dump to convert the .sra archive into paired-end FASTQ files.

```
$ fasterq-dump SRR25110243
```

This command will generate two FASTQ files (for paired-end reads):

We can monitor the download progress as:

```
$ ls -lh
total 7.9G
drwxr-x---. 2 user01 user01   29 Oct 22 18:47 SRR25110243
-rw-r-----. 1 user01 user01 2.9G Oct 22 19:12 SRR25110243_1.fastq
-rw-r-----. 1 user01 user01 2.8G Oct 22 19:12 SRR25110243_2.fastq
drwxr-x---. 2 user01 user01 4.0K Oct 22 19:10
fasterq.tmp.login1.int.omics.c3.ca.397864
```

Once completed, we will have the `_1.fastq` and `_2.fastq` files

```
$ ls -lh
total 19G
drwxr-x---. 2 user01 user01   29 Oct 22 18:47 SRR25110243
-rw-r-----. 1 user01 user01 9.4G Oct 22 19:22 SRR25110243_1.fastq
-rw-r-----. 1 user01 user01 9.4G Oct 22 19:22 SRR25110243_2.fastq
```

The `_1.fastq` and `_2.fastq` files represent read pairs (R1 and R2), ready for downstream analysis such as quality control and alignment

5. *Verify the Output*

Confirm that both FASTQ files are created and we can also check the first few lines of each file:

```
$ head SRR25110243_1.fastq
$ head SRR25110243_2.fastq
```

Exercise 3: Plan for Omics project Plan for an OMICS Project

Working with a large sequencing dataset involves several steps and opportunities for error. This section walks us through key planning stages.

Generating Sequencing Data

1. *Submitting samples to the facility*
   Use a Sample Submission Sheet (see next page) to provide all required sample metadata.

2. *Receiving sequencing data from the facility*
   When we get data back, we will receive:
   - Raw sequence files (e.g. FASTQ)
   - Accompanying metadata (documentation about samples, sequencing runs, barcodes, etc.)

3. *Planning data storage and retention*
   The raw data is our foundation. Always keep it. We want to be able to revisit, re-run analyses, or try new approaches. Losing or corrupting raw files can compromise reproducibility

Example – Sample Submission Sheet

well_position	tube_barcode	plate_barcode	client_sample_id	replicate	Volume (µL)	concentration (ng/µL)	RIN	prep_date	ship_date
A1	151017990	LP-10624	wild type 1h1	a	64.2	211.07	8.1	Jul,07	2025-07-20
B1	151101577	LP-10624	wild type 1h-1	B	63.7	220.21	9.4	Jul,07	2025-07-20
C1	151142725	LP-10624	wildtype-1h1	c	60.2	207.57	8.9	Jul,07	2025-07-20

Discussion Questions

- What errors/ inconsistencies can we spot?
- What mistakes might be harder to detect by eye?

Example – Sequencing Metadata

sample_id	platform	sequencing_layout	barcode	#_reads	rRNA_rate (%)	filename	Size (GB)
151017990	ILLUMINA	RNA-Seq	PE	5,469,882	3.37	151017990_GTTAAG_ACA4RRCXX_R1.fastq.gz	5.77
151017990	ILLUMINA	RNA-Seq	PE	5,469,882	3.37	151017990_GTTAAG_ACA4RRCXX_R2.fastq.gz	5.77
151101577	ILLUMINA	RNA-Seq	PE	5,789,648	2.41	151101577_AAATTG_ACA4RRCXX_R1.fastq.gz	6.09
151101577	ILLUMINA	RNA-Seq	PE	5,789,648	2.41	151101577_AAATTG_ACA4RRCXX_R2.fastq.gz	6.09
151142725	ILLUMINA	RNA-Seq	PE	5,043,882	3.08	151142725_TGCTAG_ACA4RRCXX_R1.fastq.gz	5.34
151142725	ILLUMINA	RNA-Seq	PE	5,043,882	3.08	151142725_TGCTAG_ACA4RRCXX_R2.fastq.gz	5.34

Discussion Questions

- How are these samples organized in the metadata table?
- Could we link the filenames to their samples in the submission sheet?
- What do the _R1 and _R2 suffixes mean in paired-end sequencing?
- What does the .gz file extension tell us about the files?

References and additional resources

1. Data Carpentry: https://datacarpentry.org/
2. European Nucleotide Archive: https://www.ebi.ac.uk/ena/browser/home
3. National Center for Biotechnology Information: https://www.ncbi.nlm.nih.gov/
4. Digital Research Alliance of Canada: https://www.alliancecan.ca/en
5. O'Leary NA, Cox E, Holmes JB, Anderson WR, Falk R, Hem V, Tsuchiya MTN, Schuler GD, Zhang X, Torcivia J, Ketter A, Breen L, Cothran J, Bajwa H, Tinne J, Meric PA, Hlavina W, Schneider VA. Exploring and retrieving sequence and metadata for species across the tree of life with NCBI Datasets. Sci Data. 2024 Jul 5;11(1):732. doi: 10.1038/s41597-024-03571-y. PMID: 38969627; PMCID: PMC11226681.
6. Lyu B, Li J, Niemeyer B, Stanley D, Song Q. Identification and characterization of ecdysis-related neuropeptides in the lone star tick *Amblyomma americanum*. Front Endocrinol (Lausanne). 2023 Aug 25;14:1256618. doi: 10.3389/fendo.2023.1256618. PMID: 37693356; PMCID: PMC10490126.

Acknowledgments

I would like to thank the **Digital Research Alliance of Canada** for providing the infrastructure support that made this workshop possible. I am sincerely grateful to **Jayson To** for his constant support and encouragement, and for ensuring that all arrangements were in place for the smooth facilitation of the workshop. I wish to extend my deep appreciation to **Michael Tang** for his technical expertise and assistance in setting up the workshop cluster, and to the entire **Advanced Research Computing (ARC) team at UBC** for their multifaceted support throughout the process.

End of Day 1 — Take-Home Message

Thank you for your hard work and engagement today!

At this point, we have learned:

- How to connect to a HPC cluster
- How to navigate the HPC environment
- Exercises to download public datasets

What's coming tomorrow:
We will work with the downloaded data from Day 01.

Look forward to seeing you all tomorrow!