

RGGS Comparative Genomics 2 – Computational Methods (Session 1)

Jose Barba

Gerstner Scholar in Bioinformatics & Computational Biology

Session 1 outline

- **Personal backgrounds and research interests**
- **Course overview**
- **Reproducibility and organization in computational biology**
- **Introduction to Unix and Unix-like operating systems**
- **Basic navigation in the terminal**

Course overview

- This project-based course introduces computational tools for interpreting molecular sequencing data, focusing on genome and transcriptome assembly, comparative analyses (both de novo and reference-based), and phylogenomic inference.
- The course emphasizes documentation, reproducibility, and computational proficiency.
- Students will address experimental questions starting from raw data, relevant to their research, with the goal of laying a foundation for a publishable final product.
- https://github.com/josebarbamontoya/rggs_comparative_genomics_2

Reproducibility and organization in computational biology

- **Reproducibility**

- Documentation: Maintain comprehensive documentation for every step of the analysis, including code, parameters, and software versions. Tools like **GitHub**, **Jupyter Notebooks**, or **RMarkdown** can integrate code with narrative text, making it easier to follow the analysis.
- Version control: Use version control systems like **Git** to track changes in scripts and collaborate with others. This ensures that any modification can be traced back and the exact state of the code at any point in time can be recovered.

Reproducibility and organization in computational biology

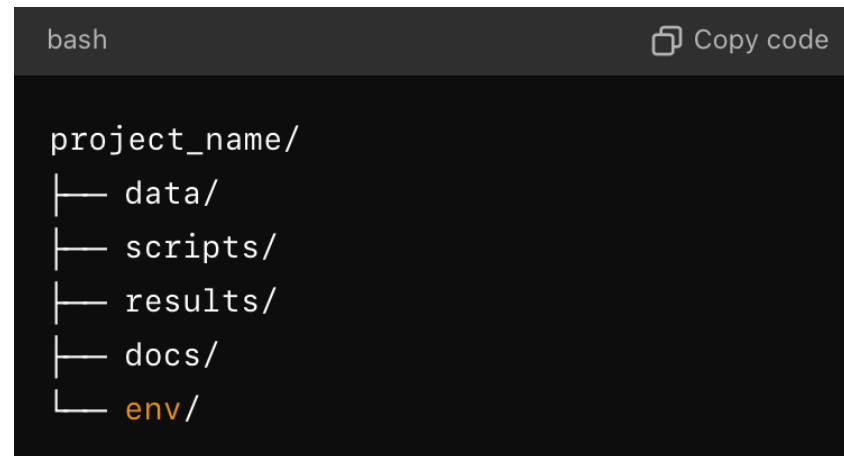
- **Reproducibility**

- Containerization: Employ containerization tools such as **Docker** or **Singularity** to encapsulate the computational environment. This includes software, libraries, and dependencies, ensuring that the analysis can be replicated on different systems without compatibility issues.
- Automated workflows: Tools like **Snakemake** or **Nextflow** help automate and standardize workflows. They ensure that the entire analysis pipeline can be executed in a controlled manner, reducing human error and making the process more reproducible.
- Data management: Organize raw and processed data systematically, using consistent naming conventions and directory structures (e.g., **Globus**). Data should be stored with appropriate metadata, making it easier to understand and reuse.

Reproducibility and organization in computational biology

- **Organization**

- Project Structure: Adopt a standardized directory structure for projects. A common approach is to separate raw data, processed data, scripts, and results into distinct directories. For example:

A terminal window with a dark background. The title bar shows 'bash' on the left and a 'Copy code' button on the right. The terminal content shows a directory tree for 'project_name/'. The tree has a root 'project_name/' with five subdirectories: 'data/', 'scripts/', 'results/', 'docs/', and 'env/'. The 'env/' directory is highlighted in orange.

```
bash                                                                    Copy code

project_name/
├─ data/
├─ scripts/
├─ results/
├─ docs/
└─ env/
```

- Naming Conventions: Use clear and consistent naming conventions for files, variables, and functions. This makes it easier to understand the purpose of each component in your project.

Reproducibility and organization in computational biology

- **Organization**

- Modular Code: Write modular code that separates different parts of the analysis into functions or classes. This not only makes the code more readable but also reusable for other projects.
- Backups and Data Integrity: Regularly back up your data and use versioned storage to ensure data integrity. This is especially important for long-term projects where data may be accessed or reused after extended periods.
- Collaboration: Foster a collaborative environment by sharing code, data, and documentation with colleagues. Use platforms like **GitHub** or **GitLab** to collaborate efficiently on code and data.

Introduction to Unix and Unix-like operating systems

- **Unix is a multiuser, multitasking OS originally developed in the 1960s-70s at AT&T's Bell Labs**
- **Its key features include:**
 - Multitasking: Unix can execute multiple processes simultaneously.
 - Multiuser: Multiple users can access the system concurrently without interfering with each other.
 - Security and permissions: It has a strong security model with user permissions and file access controls.
 - Text-based interface: Unix uses a command-line interface (terminal) for user interactions, which is efficient for many tasks.

Introduction to Unix and Unix-like operating systems

- **Unix-like operating systems behave similarly to Unix but are not directly derived from the original Unix source code**
 - macOS: Apple's operating system for Mac computers, which is built on a Unix-based foundation.
 - Linux: A widely used Unix-like system with various distributions such as Ubuntu, Fedora, and Debian. Known for its open-source nature and flexibility.

Introduction to Unix and Unix-like operating systems

- Key Concepts

- Filesystem hierarchy: Unix-like systems organize files and directories in a hierarchical structure, starting from the root directory `/`, home directory `~`.
- Shell: The command-line interface where users can execute commands. Common shells include `bash`, `sh`, and `zsh`.
- Commands: Basic commands include `ls` (list directory contents), `cd` (change directory), `cp` (copy files), `mv` (move files), and `rm` (remove files).
- Processes: Unix-like systems handle multiple processes, and tools like `ps` and `top`, `htop` help monitor them.
- Permissions: Files and directories have permissions (read, write, execute) that control access for the owner, group, and others like `chmod`.

Basic navigation in the terminal

- Using the command line interface involves executing commands to navigate directories, view/edit/create files, among other tasks

Archiving and Compression	Networking and Remote Operations
tar - Archive files into a tarball (or extract them)	curl - Download and transfer data from remote servers
zip/unzip - Compress and extract files in zip format	exit - Close current shell, terminal prompt, interactive program
File and Directory Operations	scp - Secure copy, used for transferring files between hosts
cat - Concatenate and display file contents	sftp - Securely transfer files between computers over a network
cd - Change directory	ssh - Secure shell, used for logging into a remote machine
cp - Copy files or directories	wget - Download and transfer data from remote servers
head - Display the first few lines of a file	Operators
less - Read file one screen at a time	* - Mangle multiple files and directories
ls - List files and directories	& - Run a command in the background
mkdir - Make directory	> - Redirect the output of a command to a file
mv - Move or rename files or directories	 - Connect the output of one command to the input of another
pwd - Print working directory	Process Management
rm - Remove files or directories	kill - Terminate processes by ID
rmdir - Remove directory	ps - Display information about running processes
sort - Sort the lines of a file	top - Interactively manage processes
tail - Display the last few lines of a file	Terminal Utilities
touch - Create an empty file or update a file's timestamp	clear - Clear the terminal screen
tr - Translate or delete characters	help - Show information about built-in shell commands
File and Directory Search	man - Display manual for a command
find - Search for files and directories within a hierarchy	screen - Managing terminal sessions
grep - Search for patterns within text	Text Processing
File Permissions and Ownership	awk - Pattern scanning and processing language
chmod - Change file permissions	cut - Extract specific sections of text
chown - Change file owner and group	echo - Display a line of text or variables
File Space and Disk Usage	nano - Text editor
df - Display disk space usage	sed - Stream editor for filtering and transforming text
du - Estimate file space usage	vi - Text editor

Basic navigation in the terminal

- **Instructions to download the basic Unix navigation tutorial to the home directory:**
 1. Open the terminal
 2. Type ``cd ~``
 3. Type ``wget``
[https://raw.githubusercontent.com/josebarbamontoya/rggs_comparative_genomics_2/main/session_01/basic_unix_navigation_tutorial.sh`](https://raw.githubusercontent.com/josebarbamontoya/rggs_comparative_genomics_2/main/session_01/basic_unix_navigation_tutorial.sh)
- **Some useful links on shell and basic navigation commands**
 - Unix shell: <https://swcarpentry.github.io/shell-novice/01-intro.html>
 - Unix basic commands: <https://sandbox.bio/tutorials/terminal-basics/>
 - Data exploration with awk: <https://sandbox.bio/tutorials/awk-intro>