Chapter 2: Unix Shell Operating System

Introduces the Unix shell as a foundational tool for interacting with a computer system through text-based commands. It highlights how the command line enables efficient file management, automation, and reproducible workflows.

<u>Key Points</u>

- Core commands include:
    - ls (list files)
    - cd (change directory)
    - pwd (show current directory)
    - mkdir (create directory)
    - touch (create file)
    - rm (remove files)
- Redirection and pipes (>, >>, |) enable chaining commands and writing output to files.
- Using the shell supports reproducibility, especially when running data processing workflows across different systems.

Chapter 3: Version Control

Explains the importance of version control for managing changes in software and data analysis projects. It focuses on Git, a widely-used tool that tracks modifications, supports collaboration, and preserves project history.

<u>Key Points</u>

- Version control helps track who made changes, when, and why.
- Git workflow:
    1. **git init** to start a repository
    2. **git add** to stage changes
    3. **git commit** to save changes with a message
    4. **git log** to review project history
- Branches allow development without affecting the stable main version.
- Collaboration with GitHub:
    - Remote repositories allow multiple contributors.
    - **git push** uploads changes, **git pull** downloads changes.
    - Pull requests enable review and discussion before merging.
- Merge conflicts can occur when multiple users edit the same lines; these must be resolved manually.
- Git is not suitable for large datasets; Datalad is recommended for data versioning.

Chapter 4: Computational Environments & Containers

Focuses on managing computational environments to ensure reproducibility and portability. It introduces containers as a modern solution for packaging software, dependencies, and configurations together.

<u>Key Points</u>

- A computational environment includes:
    - Operating system
    - Software packages and dependencies
    - Configuration settings
- Containers package code, libraries, and runtime environments into a self-contained unit.
- Docker is the most widely-used container platform:
    - Dockerfile defines the environment setup.
    - Image is the packaged environment.
    - Container is a running instance of an image.
- Containers ensure code runs consistently across different machines, improving reproducibility.
- They are essential for sharing research workflows and running complex analysis pipelines.