# **Class 3: Linux Commands and Bioinformatics Analysis Using Bash**

## **1. Introduction**

In this class, we’ll build on our foundational skills with Linux and delve into using Bash scripting for bioinformatics tasks. Bash scripting enables automation, making repetitive tasks more efficient—especially important in bioinformatics where tasks like RNA-seq analysis involve multiple steps that can be cumbersome to run manually.

## **2. Review: Key Linux Commands for Bioinformatics**

Before jumping into scripting, let's review essential Linux commands that serve as building blocks for bioinformatics:

* **File Operations**:
  + cp, mv, rm: Copy, move, and remove files.
  + cat, less: View file contents.
* **Searching**:
  + grep, find: Search within files or directories.
* **Process Handling**:
  + ps, top, kill: Monitor and manage processes.
* **Permissions**:
  + chmod, chown: Manage file and directory permissions.

These commands, combined into scripts, form the foundation for automating bioinformatics data analysis.

## **3. Bash Scripting Fundamentals**

A Bash script is a text file containing a sequence of commands that can automate complex workflows. Here's a simple script structure:

**Creating a Script**:  
Every Bash script begins with a *shebang* (#!/bin/bash) to indicate the script is to be executed in the Bash shell.  
bash  
Copy code  
#!/bin/bash

echo "Hello, Bioinformatics World!"

1. Save the script as myscript.sh, make it executable (chmod +x myscript.sh), and run it with ./myscript.sh.

**Variables**:  
Variables allow flexibility by storing values like file paths or parameters. Define variables with a straightforward syntax:  
bash  
Copy code  
input="data.fasta"

echo "Processing file: $input"

## **4. Automating Bioinformatics Workflows with Bash**

Bioinformatics workflows often involve multiple steps, such as data processing, alignment, and quantification. For example, an RNA-seq analysis may involve downloading data, unzipping files, and aligning sequences to a reference genome. Here’s how we could automate these tasks:

bash

Copy code

#!/bin/bash

# Define variables

input\_dir="RNA\_seq\_data"

output\_dir="aligned\_data"

index="genome\_index"

# Step 1: Download data

wget https://example.com/sample\_1.fastq.gz -P $input\_dir

# Step 2: Unzip data

gunzip $input\_dir/sample\_1.fastq.gz

# Step 3: Sequence alignment

hisat2 -x $index -U $input\_dir/sample\_1.fastq -S $output\_dir/aligned.sam

# Step 4: Convert SAM to BAM

samtools view -bS $output\_dir/aligned.sam > $output\_dir/aligned.bam

This script automates multiple stages of RNA-seq data processing, enhancing efficiency when handling large datasets.

## **5. Control Structures in Bash**

### **Conditionals**

Conditionals allow scripts to make decisions based on file existence or variable values. Here’s an example checking if a file exists:

bash

Copy code

#!/bin/bash

if [ -f "data.fasta" ]; then

echo "File exists."

else

echo "File not found!"

fi

### **Loops**

Loops automate operations over multiple files or directories, streamlining tasks like quality control or sequence alignment:

bash

Copy code

#!/bin/bash

for file in RNA\_seq\_data/\*.fastq

do

echo "Processing $file"

fastqc $file

done

Using loops, you can easily apply a bioinformatics tool across all relevant files in a directory.

## **6. Practical Bioinformatics Task: RNA-seq Workflow Pipeline**

A complete RNA-seq analysis pipeline might include:

* **Quality Control**: FastQC
* **Trimming**: Trimmomatic (optional)
* **Alignment**: Hisat2
* **Quantification**: featureCounts

Here’s a simplified pipeline script to automate these tasks:

bash

Copy code

#!/bin/bash

input\_dir="RNA\_seq\_data"

output\_dir="results"

qc\_dir="qc\_reports"

# Quality control

fastqc $input\_dir/\*.fastq -o $qc\_dir

# Alignment

for file in $input\_dir/\*.fastq

do

hisat2 -x genome\_index -U $file -S $output\_dir/$(basename $file .fastq).sam

done

# Convert SAM to BAM

for file in $output\_dir/\*.sam

do

samtools view -bS $file > ${file%.sam}.bam

done

## **7. Fetching Data from NCBI Using Bash**

NCBI’s datasets can be accessed programmatically, which is helpful for large-scale analyses. Using wget or curl, data can be fetched directly within a Bash script.

Example:

bash

Copy code

#!/bin/bash

# Download data from NCBI

wget ftp://ftp.ncbi.nlm.nih.gov/genomes/ASSEMBLY\_ACCESSION.fa.gz

gunzip ASSEMBLY\_ACCESSION.fa.gz

Fetching data directly from the source saves time, ensuring you always have up-to-date sequences.

## **8. Version Control with Git and GitHub in Bash**

Using Git within Bash is essential for managing script versions, tracking changes, and collaborating. To begin, navigate to your project directory and initialize a Git repository:

bash

Copy code

git init

git add .

git commit -m "Initial commit for RNA-seq pipeline"

Push changes to GitHub:

bash

Copy code

git remote add origin [GitHub repository URL]

git push -u origin main

## 