Week 1: Computing Infrastructure Basics

Duration: 2 hours

Focus: Understanding the computational backbone of bioinformatics

Slide 1: Welcome & Course Overview

Computing Infrastructure Basics in Bioinformatics

Week 1 - Research-Oriented Framework

Learning Objectives:

- Understand computational requirements in bioinformatics
- Master fundamental Linux operations
- Set up development environments
- Introduction to version control with GitHub

Research Context:

- Modern bioinformatics relies heavily on computational infrastructure
- Skills learned today are essential for genomics research
- Foundation for advanced bioinformatics workflows

Slide 2: What is Bioinformatics?

The Intersection of Biology, Computer Science, and Statistics

Definition:
Bioinformatics is the application of computational tools and techniques to capture, store, analyze, and interpret biological data.

Key Research Areas:

- **Genomics**: DNA/RNA sequence analysis
- **Proteomics**: Protein structure and function prediction
- **Systems Biology**: Network analysis and pathway modeling
- **Phylogenetics**: Evolutionary relationships
- **Metagenomics**: Microbial community analysis

Research Impact:

- Drug discovery and development
- Personalized medicine
- Agricultural genomics
- Infectious disease surveillance

Slide 3: Role in Microbiology Research

Computational Microbiology Applications

Pathogen Genomics:

- Whole genome sequencing (WGS) analysis
- Antimicrobial resistance (AMR) detection
- Virulence factor identification
- Outbreak investigation and source tracking

Microbiome Studies:

- 16S rRNA gene sequencing
- Metagenomic shotgun sequencing
- Functional annotation and pathway analysis
- Host-microbe interaction studies

Clinical Applications:

- Diagnostic tool development
- Treatment selection guidance
- Epidemiological surveillance
- Public health decision support

Slide 4: Computing Requirements

Why Computational Infrastructure Matters

Data Volumes:

- Single bacterial genome: ~5 MB
- Human genome: ~3 GB
- Metagenomic dataset: 1-100+ GB
- Multi-sample studies: TB to PB scale

Computational Challenges:

- **Memory**: Large reference databases (BLAST, Kraken2)
- **Processing**: CPU-intensive algorithms (assembly, alignment)
- **Storage**: Long-term data archiving and backup
- **Network**: Data transfer and collaborative sharing

Research Implications:

- Infrastructure directly impacts research scope
- Scalability enables larger studies
- Reproducibility requires standardized environments

Slide 5: Operating Systems Comparison

Choosing the Right Platform for Research

Feature	Linux/Unix	Windows	macOS
Bioinformatics Software	Extensive native support	Limited, WSL required	Good support
Command Line	Native, powerful	PowerShell/CMD, WSL	Native Unix terminal
Package Management	apt, yum, conda	Package managers limited	Homebrew, conda
Server Deployment	Industry standard	Less common	Development focused
Cost	Free (most distributions)	Licensed	Hardware dependent
Research Adoption	~80% of bioinformatics	~15% (mainly Windows-specific tools)	~5% (mainly development)

Recommendation: Linux for production research environments

Slide 6: Linux in Research Computing

Why Linux Dominates Bioinformatics

Advantages:

- **Open Source**: Free, customizable, transparent
- **Stability**: Designed for long-running processes
- **Performance**: Efficient resource utilization
- **Security**: Robust permission system
- **Scalability**: From desktop to supercomputer
- **Package Ecosystem**: Extensive bioinformatics software

Popular Distributions for Research:

- **Ubuntu**: User-friendly, extensive documentation
- **CentOS/RHEL**: Enterprise stability, HPC clusters
- **Debian**: Stable, minimal overhead
- **Arch**: Cutting-edge packages, advanced users

Research Infrastructure:

- Most HPC clusters run Linux
- Cloud platforms default to Linux
- Bioinformatics pipelines designed for Linux

Slide 7: Computing Environments

From Personal Workstations to Cloud Computing

Personal Workstations

- Development and small-scale analysis
- 16-64 GB RAM, multi-core CPUs
- Local storage for immediate access
- Cost-effective for routine tasks

High-Performance Computing (HPC) Clusters

- Large-scale computational analysis
- Shared resources: 100s-1000s of cores
- Job scheduling systems (SLURM, PBS)
- Parallel processing capabilities

Cloud Computing

- Scalable, on-demand resources
- AWS, Google Cloud, Microsoft Azure
- Pay-per-use model
- Global accessibility and collaboration

Hybrid Approaches

- Local development + cloud execution
- Data preprocessing locally
- Intensive analysis in cloud/HPC

Slide 8: Understanding Servers and Clusters

Distributed Computing for Genomics

Server Architecture:

- **Compute Nodes**: CPU/GPU processing units
- **Storage Nodes**: High-capacity data storage
- **Head Nodes**: Job submission and management
- **Network Infrastructure**: High-speed interconnects

Cluster Management:

- **Job Schedulers**: SLURM, Torque/PBS, SGE
- **Resource Allocation**: CPU, memory, time limits
- **Queue Systems**: Priority-based job execution
- **Monitoring**: Resource utilization tracking

Research Applications:

- Genome assembly projects
- Phylogenetic reconstructions
- Machine learning model training
- Large-scale comparative genomics

Slide 9: Cloud Computing Platforms

Leveraging Cloud Resources for Research

Amazon Web Services (AWS)

- **EC2**: Scalable compute instances
- **S3**: Object storage for datasets
- **Batch**: Large-scale job processing
- **Genomics-specific**: HealthOmics, GATK workflows

Google Cloud Platform (GCP)

- **Compute Engine**: Virtual machines
- **Cloud Storage**: Distributed file storage
- **Cloud Life Sciences**: Genomics pipelines
- **BigQuery**: Large-scale data analytics

Specialized Platforms

- **Galaxy Project**: Web-based analysis platform
- **Seven Bridges**: Bioinformatics workflow platform
- **DNAnexus**: Secure genomics cloud platform

Slide 10: Introduction to Version Control

Why Every Researcher Needs Git

Research Challenges Without Version Control:

- Multiple script versions: `analysis_v1.py`, `analysis_final_FINAL.py`
- Collaboration difficulties
- Lost work due to overwrites
- Unable to reproduce previous results
- No change tracking or blame assignment

Version Control Benefits:

- **History Tracking**: Complete change history
- **Collaboration**: Multiple contributors, conflict resolution
- **Branching**: Parallel development tracks
- **Backup**: Distributed repository copies
- **Reproducibility**: Tagged versions, rollback capability

Research Applications:

- Analysis script development
- Pipeline version management
- Collaborative manuscript writing
- Data processing workflow tracking

Slide 11: Git vs GitHub

Understanding the Ecosystem

Git (Local Version Control)

- Command-line tool for version control
- Tracks changes in local repositories
- Works offline
- Created by Linus Torvalds (2005)

GitHub (Remote Platform)

- Web-based hosting service for Git repositories
- Social coding platform
- Issue tracking and project management
- Continuous integration/deployment (CI/CD)
- Free public repositories, paid private options

Research Workflow:

```
Local Development (Git) ←→ Remote Collaboration (GitHub)
     ↓                           ↓
- Track changes               - Share code
- Commit history              - Collaborate
- Branch/merge                - Release software
- Offline work                - Community engagement
```

Slide 12: File Systems and Directory Structure

Organizing Research Data Effectively

Linux File System Hierarchy

```
/ (root)
├── home/          # User home directories
├── usr/           # User programs and libraries
├── etc/           # System configuration
├── var/           # Variable data (logs, databases)
├── tmp/           # Temporary files
├── opt/           # Optional software packages
└── mnt/           # Mount points for external drives
```

Research Directory Structure Best Practices

```
project_name/
├── data/
│   ├── raw/        # Original, unmodified data
│   ├── processed/  # Cleaned, filtered data
│   └── results/    # Analysis outputs
├── scripts/        # Analysis code
├── docs/           # Documentation, protocols
├── figures/        # Plots, visualizations
└── README.md       # Project description
```

Slide 13: Hands-On Setup Overview

Practical Session Structure (90 minutes)

Phase 1: Environment Setup (30 minutes)

- Install VirtualBox or enable WSL2
- Download Ubuntu 20.04 LTS
- Configure virtual machine settings
- First boot and initial setup

Phase 2: Command Line Basics (30 minutes)

- Terminal navigation: `pwd`, `ls`, `cd`
- Directory operations: `mkdir`, `rmdir`
- File operations: `touch`, `cp`, `mv`, `rm`
- Text viewing: `cat`, `less`, `head`, `tail`

Phase 3: GitHub Integration (30 minutes)

- Create GitHub account
- Install git and configure credentials
- Initialize repository and first commit
- Push project structure to GitHub

Slide 14: Linux Command Essentials

Navigation and File Management

Basic Navigation

```
pwd                 # Print working directory
ls                  # List directory contents
ls -la             # Detailed listing with hidden files
cd /path/to/dir    # Change directory
cd ~               # Go to home directory
cd ..              # Go to parent directory
```

Directory Operations

```
mkdir project_dir          # Create directory
mkdir -p data/raw/fastq   # Create nested directories
rmdir empty_dir           # Remove empty directory
rm -rf directory_name     # Remove directory and contents
```

File Operations

```
touch filename.txt        # Create empty file
cp source.txt dest.txt    # Copy file
cp -r dir1/ dir2/         # Copy directory recursively
mv oldname.txt newname.txt # Move/rename file
rm filename.txt           # Delete file
```

Slide 15: File Viewing and Text Processing

Essential Commands for Data Inspection

Text File Viewing

```
cat filename.txt          # Display entire file content
less filename.txt         # Page through file (q to quit)
head -n 10 data.txt       # Show first 10 lines
tail -n 20 log.txt        # Show last 20 lines
tail -f logfile.txt       # Follow file changes in real-time
```

File Information

```
wc -l sequences.fasta     # Count lines in file
wc -w document.txt        # Count words
file mysterious_data      # Determine file type
```

```
du -sh directory/         # Directory size
find . -name "*.fastq"    # Find files by pattern
```

Research Applications

- Quick quality assessment of sequence files
- Log file monitoring during long-running analysis
- Dataset size estimation
- File format verification

Slide 16: GitHub Workflow for Research

Version Control Best Practices

Initial Setup

```
# Configure git (one-time setup)
git config --global user.name "Your Name"
git config --global user.email "email@example.com"

# Initialize repository
git init
git remote add origin https://github.com/username/repository.git
```

Daily Workflow

```
# Check status
git status

# Stage changes
git add filename.txt        # Stage specific file
git add .                   # Stage all changes

# Commit changes
git commit -m "Descriptive commit message"

# Push to GitHub
git push origin main
```

Research Commit Messages

- "Add initial genome assembly script"
- "Fix bug in quality filtering parameters"
- "Update analysis pipeline for new data format"
- "Add statistical testing to differential expression"

Slide 17: Assignment Overview

Practical Project Structure Creation

Objective

Create a well-organized directory structure for a mock bioinformatics project and practice version control workflows.

Project Scenario

"Comparative genomics analysis of antibiotic-resistant *E. coli* isolates from hospital samples"

Required Directory Structure

```
ecoli_resistance_study/
├── data/
│   ├── raw/
│   │   ├── genomes/
│   │   └── metadata/
│   ├── processed/
│   └── results/
├── scripts/
│   ├── preprocessing/
│   ├── analysis/
│   └── visualization/
├── docs/
│   ├── protocols/
│   └── notes/
├── figures/
└── README.md
```

Slide 18: Assignment Tasks

Step-by-Step Implementation

Task 1: Directory Creation (20 minutes)

1. Create the complete directory structure using `mkdir -p`
2. Add placeholder files using `touch` in each subdirectory
3. Create a comprehensive README.md with project description
4. Document directory purposes and expected file types

Task 2: File Operations Practice (15 minutes)

1. Create sample files representing different data types
2. Practice copying files between directories
3. Use text viewing commands to inspect file contents
4. Organize files into logical subdirectories

Task 3: GitHub Integration (25 minutes)

1. Create GitHub account and new repository
2. Initialize git in project directory
3. Add all files and commit with meaningful messages
4. Push complete project structure to GitHub
5. Verify repository accessibility online

Slide 19: Assessment Criteria

Evaluation Standards for Assignment

Technical Competency (60%)

- Correct directory structure implementation
- Proper use of Linux commands
- Successful GitHub repository creation
- File organization follows best practices

Documentation Quality (25%)

- Clear, comprehensive README.md
- Appropriate file and directory naming
- Logical project organization
- Professional presentation

Research Relevance (15%)

- Understanding of bioinformatics workflows
- Appropriate choice of directory names
- Realistic file type expectations
- Scalable project structure design

Submission Requirements

- GitHub repository URL
- Screenshot of directory structure (`tree` command output)
- Brief reflection on challenges encountered

Slide 20: Next Steps & Resources

Continuing Your Bioinformatics Journey

Week 2 Preview: Data Management

- Large file handling strategies
- Database systems for biological data
- Data backup and archiving
- File format standards in bioinformatics

Additional Learning Resources

Linux Mastery:

- Linux Command Line Bootcamp (Udemy)
- "The Linux Command Line" by William Shotts
- Ubuntu official documentation

Git/GitHub:

- GitHub Learning Lab
- "Pro Git" book (free online)
- Git branching interactive tutorial

Bioinformatics Platforms:

- Galaxy Project tutorials
- Bioconda package manager
- QIIME2 documentation

Office Hours: Available for individual consultation on setup challenges and advanced topics

Contact Information

Instructor: [Your Name]
Email: [your.email@institution.edu]
Office: [Location]
GitHub: [github.com/yourusername]

Course Repository: [github.com/course/bioinformatics-infrastructure]
Discussion Forum: [Link to course forum/Slack]