Lecture 9: Bioinformatics Tools & Databases

Student Handout & In-Class Exercises

Course: BINF301 — Computational Biology
Instructor: Tom Michoel
Date: 11/2/2026
Created with Copilot

1 ELIXIR & Core Data Resources (Slides 3–7)

ELIXIR is a Europe-wide infrastructure that coordinates and develops:

- Life-science data resources,
- computing platforms,
- standardized metadata and interoperability tools,
- training and community networks.

ELIXIR promotes the FAIR principles:

- **Findable:** indexed, searchable, identifiable;
- **Accessible:** open protocols and metadata access;
- **Interoperable:** shared vocabularies, structured data;
- **Reusable:** rich metadata, clear licensing, provenance.

Solution. Make clear to students that ELIXIR is about *both* data and tools, and that FAIR principles guide modern research data management.

2 What is a Bioinformatic Tool? (Slides 9–12)

A bioinformatic tool is software that:

- implements one or more computational algorithms,
- accepts biological data as input,
- produces processed/analyzed/annotated biological output,
- may have various interfaces (CLI, GUI, web, API).

Metadata describing tools (format expectations, algorithms, dependencies) is essential for reproducibility.

Solution. Students often think "tools = big software packages." Instead, stress that even a small script can be a tool if it implements a meaningful computational function.

3 Finding & Using Tools (Slides 13–17)

Researchers typically discover tools through:

- publications,
- community recommendations,
- conferences,
- tool registries such as **bio.tools**,

- the Galaxy tools ecosystem.

3.1 bio.tools Registry (Slides 14–18)

- ELIXIR-funded tool registry,
- 29,227 entries (Feb. 2024),
- uses unique identifiers and ontology-based tags.

Solution. Highlight key teaching point: metadata-rich registries like bio.tools enable better discovery, evaluation, and reuse of computational tools.

4 Installing & Running Tools (Slides 19–22)

Bioinformatics tools are typically installed via:

- Linux/Unix native binaries,
- macOS (sometimes compatible),
- Windows via WSL,
- Conda/Bioconda/Mamba for dependency management,
- Docker or Singularity containers for full reproducibility,
- GitHub repositories and manual compilation.

Solution. Students should understand the tradeoffs: Conda = easy dependency resolution; Containers = fully reproducible environment; Manual builds = flexible but error-prone.

5 Bioinformatic Workflows (Slides 24–28)

Large analyses combine multiple steps $\rightarrow$ workflows.

5.1 Workflow Languages (Slide 25)

- **CWL**: command-line focused.
- **Snakemake**: Python-based; hybrid scripting + rules.
- **Nextflow**: scalable, portable, used by NF-core.

5.2 Workflow Databases (Slide 28)

- **WorkflowHub**: FAIR-compliant, domain-agnostic repository.
- **NF-core**: curated Nextflow pipelines.

Solution. Explain reproducibility: workflows encode both tool calls and metadata, enabling robust, shareable analyses.

6 Databases (Slides 31–45)

A true database is:

- structured, machine-readable,
- built on a database management system (e.g. SQL),
- consistent and queryable.

6.1 Data Deposition (Slide 33)

Many journals require deposition in:

- **SRA** (NCBI),
- **ENA** (Europe),
- **DDBJ** (Japan),

which synchronize globally.

6.2 Widely Used Databases (Slides 38–45)

- **ENA, GenBank, SRA** — sequence archives
- **UniProt** — protein knowledgebase
- **Ensembl** — annotated genomes
- **OrthoDB** — ortholog groups
- **BRENDA** — enzyme function
- **KEGG** — pathways
- **PDB** — protein structures
- **InterPro/PFAM** — protein domains

Solution. Point out that each database solves different biological questions; knowing which one to query is a key skill in bioinformatics.

7 Programmatic Access (Slides 47–55)

Modern datasets support machine-readable interfaces:

- **REST APIs** for structured HTTP output (JSON, XML),
- **BioMart** for data mining,
- **NCBI E-utils** for scripted access (ESearch, ESummary, EFetch),
- **NCBI Datasets** for bulk genome/gene downloads.

Solution. Instructor should clarify the role of REST: all parameters in URL, standardized output, ideal for scripting and automation.

8 Exercises (with Solutions)

8.1 Exercise 1 — FAIR Principles

Match:

a. Dataset includes rich metadata + license.
b. Database provides JSON API.
c. Sequence has stable accession number.
d. Workflow uses controlled vocabularies.

with: Findable, Accessible, Interoperable, Reusable.

Solution. a→Reusable; b→Accessible; c→Findable; d→Interoperable.

8.2 Exercise 2 — bio.tools Exploration

Pick a known tool and identify:

- tool name,
- bio.tools identifier,
- input type,
- output type.

Solution. Example: **FastQC**. Inputs: FASTQ files. Outputs: HTML reports + per-base quality statistics. bio.tools identifier varies (conceptual example).

8.3 Exercise 3 — Tool Installation Methods

Choose best installation method:

1. Tool with many dependencies all available in Conda.
2. Tool requiring an old Linux library.
3. Complex workflow with many collaborators.
4. Simple script with no dependencies.

Options: Conda, container, workflow language, manual install.

Solution. 1→Conda; 2→Container (ensures environment consistency); 3→Nextflow/Snakemake workflow; 4→Manual install (lightweight).

8.4 Exercise 4 — Database Choice

Match task → database:

- Retrieve all human protein sequences.
- Find 3D kinase structure.
- Download metagenome reads.
- Retrieve orthologs across insects.

Solution. Human proteins → UniProt; Kinase structure → PDB; Metagenome reads → SRA/ENA; Orthologs → OrthoDB.

8.5 Exercise 5 — REST vs E-utils

Explain the difference.

Solution. REST: generic web API using structured URLs, returns JSON/XML, widely used across databases. E-utils: NCBI-specific REST-like API with ESearch/EFetch operations for querying and retrieving NCBI data.

8.6 Exercise 6 — Workflow Languages True/False

1. CWL focuses on command-line tools.
2. Snakemake is Bash-based.
3. Nextflow emphasizes scalability.
4. NF-core workflows use Galaxy.

Solution. 1: True. 2: False (Python-based). 3: True. 4: False (Nextflow-only).