# Fundamentals of computational biology

Lecture notes

Camilo García-Botero

2022-08-01

# Table of contents

**Preface** **1**
Learning features . . . . . 1

**Caution** **2**

**Challenges** **3**

**File format** **3**

**Introduction** **5**

**I The command line** **7**

**1 The command line** **9**
1.1 Getting started with the command line . . . . . 9
1.2 File paths . . . . . 10
1.3 Basic Unix commands . . . . . 11
1.4 Anatomy of a command . . . . . 13
1.5 Some great operators/ Metacharacteres in Bash . . . . . 14
1.6 intermediate Unix commands . . . . . 14

**2 Introduction to the control of versions** **15**
2.1 Git basics . . . . . 15
2.2 GitHub . . . . . 15

**3 Package managers** **17**
3.1 What is the importance of package managers . . . . . 17
3.2 Conda environments and other package managers . . . . . 17
3.3 Creating enviroments with Conda . . . . . 17
3.4 Package managers for OS . . . . . 17

**4 Notions of HPC** **19**
4.1 What is HPC . . . . . 19

4.2 Some important concepts of the hardware . . . . . 19
4.3 Using the Apolo cluster wirh Slurm . . . . . 19

**II Sequence analysis 21**

**5 Introduction to sequence analysis 23**
5.1 Endless debate: bioinformatics vs. computational biology . . . . 23
5.2 The duality of DNA . . . . . 23
5.3 The central ~~dogma~~ theory of molecular biology extended . . . . . 23
5.4 Sequencing strategies . . . . . 23
5.5 Sequencing over time . . . . . 23
5.6 Some insights from sequencing genomes . . . . . 23

**6 Sanger analysis 25**
6.1 Databases exploration . . . . . 25
6.2 Sanger sequencing methods . . . . . 25
6.3 Files from Sanger . . . . . 25
6.4 Sanger processing workflow . . . . . 25
6.5 The 16S rRNA and its relevance for sequencing . . . . . 25

**7 Sequence alignments 27**
7.1 Why do we align sequences? . . . . . 27
7.2 What is homology . . . . . 27
7.3 Pairwise alignments algorihtms . . . . . 27
7.4 The genetic code and Scoring matrices . . . . . 27
7.5 BLAST and its families . . . . . 27
7.6 Multiple sequence alignments . . . . . 27

**8 Phylogenetics 29**
8.1 What is a phylogenetic tree . . . . . 29
8.2 Mehtods for phylogenetic reconstruction . . . . . 29
8.3 Building a phylogenetic reconstruction . . . . . 29

**9 NGS and TGS: principles 31**
9.1 Platforms yields . . . . . 31
9.2 Reads main differences . . . . . 31
9.3 Illumina principle (sequencing by synthesis) . . . . . 31
9.4 PacBio principle (sequencing by incorporation) . . . . . 31
9.5 Oxford Nanopore Technology (ONT) principle . . . . . 31

**III Genomics 33**

**10 Genome assembly 35**
10.1 The problem of assembling genomes . . . . . 35
10.2 Main algorithms for genome asssembly . . . . . 35

10.3 Main concepts of an assembly . . . . . 35
10.4 A complete workflow for assembling genomes . . . . . 35
10.5 Assessing genomes . . . . . 35
10.6 Understanding genome difficulties . . . . . 35

**11 Genome annotation and visualizaiton** **37**
11.1 *ab initio* annotation . . . . . 37
11.2 Homolgy annotation . . . . . 37
11.3 Annotation files . . . . . 37
11.4 Visualizing genomes and annotations . . . . . 37

**12** **39**

**13 Variant calling analysis** **41**
13.1 Common mutations . . . . . 41
13.2 Structural variants . . . . . 41
13.3 Genome rearrengments . . . . . 41
13.4 Read mapping algorithms and programs . . . . . 41
13.5 Identifying mutations . . . . . 41

**IV Structural bioinformatics** **43**

**14 Introduction to structural bioinformatics** **45**
14.1 Protein structures . . . . . 45
14.2 Identifying or predicting protein structures . . . . . 46
14.3 PDB database introduction . . . . . 46

**References** **49**

# Preface

> **Danger**
>
> This book is a work in progress

We started this book with the aim of compiling the lectures of the course Fundamentals of Computational Biology offered at Universidad EAFIT for undergrad students in Biology. The course has been taught from different perspectives from its creation, yet the last iteration was divided into three modules. i) introduction to Unix (4 lectures) ii) introduction to sequence analysis and genomics (7 lectures) and iii) principles of structural biology (4 lectures).

Lectures are focused on a theoretical-practical approach where basic concepts from biology, bioinformatics and computer science and interleave with the practice to solve challenges. Excercised or challenges are designed to improve stundents abilities that are likely to be involved in real-life problems in computational biology.

## Learning features

> **Note**
>
> Sometimes other fields might add interested value to the understanding of the computational biology area. This feauture remarks some of them and aim to explain these intersections.

> **Tip**
>
> As you move forward in the computational biology field you will find that there are several tips and tricks (mainly from the command line) as well as some random CLI programs that can leverage your daily workflow as a researcher. Using this feature we highlight some of those that appeared

to linger on the field.

**!** Important

To help you consolidate your understanding we end most chapters with important messages or concepts that help you evaluate yourself as you move forward on the lessons.

Danger

# Caution

When experimenting with the CLI and many other computational tools it is common to face several known errors and drawbacks. Then, we present some of them and how to sort them out.

Warning

# Challenges

Since focused on a competences learning approach we have highlighted several real-life (but basic) *challenges* a researcher faces when approaching computational biology problem (from tool selection, usage and result analysis). Therefore the book section *challenges* presents a selection of these problems that will later be apporached by a computational biology strategy (mainly from the CLI).

# File format

As many analysis specialize on data analysis, many formats arise that optimize the processing steps or the data storing steps. Some of these formats are keystones of bioinformatic analyses. We present examples of some formats an describe its main elements.

# Introduction

Here we present a course centered book of the Fundamentals of Computational Biology. We will cover several topics, from using the unix tools, the importance of package manager systems (such as homebrew and conda), sequencing technologies, sequence alignments, molecular phylogenetics, genome assembly and annotation, and variant calling analysis.

Inlcude a section, maybe an appendix about how to handle errors Include s section about the windows subsistem for linux WSL and the ease of use for windows users

# Part I

# The command line

# Chapter 1

# The command line

In this chapter we will explore the fundamentals of the command line interface (aka CLI). We will distinguish the differences between Unix, CLI, Bash and Terminal and other concepts from the computer sciences.

As you will see the CLI is composed of several programs enabling the interaction with the machine, we will discuss some of the basics to navigate your machine, and some advance one that enable complex operations and automating tasks.

## 1.1 Getting started with the command line

Before landing into the CLI let us consider the Unix concept. The first question that comes in this section is: what is Unix? It simply is an operating system (OS). In other words, it is a set of programs that inter-operate with each other to let you communicate with the machine. A very important variant (or clone) of Unix is the very well known OS Linux, which was created by Linus Torvalds from scratch. The most important idea behind Unix based systems is the idea that we can use it to access information and hardware programmatically. Other main feature from Unix-like OS systems is the fact that data is usually stored as text files and the interface by which users communicate with the machine is also text-based (TUI: text user interface as opposed to GUI, graphic user interface).

Almost every computer has a way to interact with or access to the inner elements of the computer. Such interface is called the the command-line-interface Fig. 1.1.

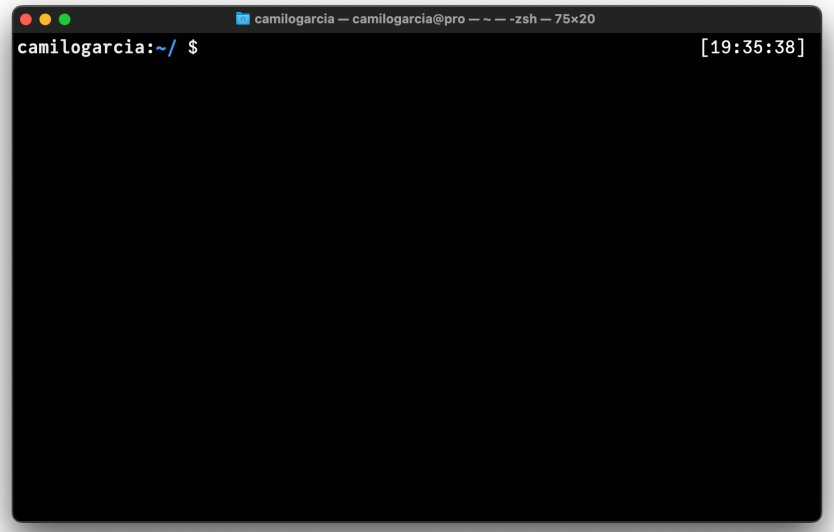



Figure 1.1: A **terminal** app displaying common features of the command line interface

## 1.2 File paths

Programs, files and directories on every machine (with Unix-like OS) display hierarchical paths (routes), starting out from the **root** (represented by the back-slash character `/`). The **root** represents the beginning of all the software installed in the machine. And many other files are nested from there forming a tree-like structure for the paths Fig. 1.2

> **Tip**
>
> You can inspect the paths of a nested directory tree using `tree` command in you cli:
>
> ```
> tree -d -L 1
> ```

There are basically **two** ways to explore or navigate your file system. If you always reprent it from the root, then you are presenting an **absolute path**. For instance the absoulte path to my desktop is (`/Users/camilogarcia/Desktop`).

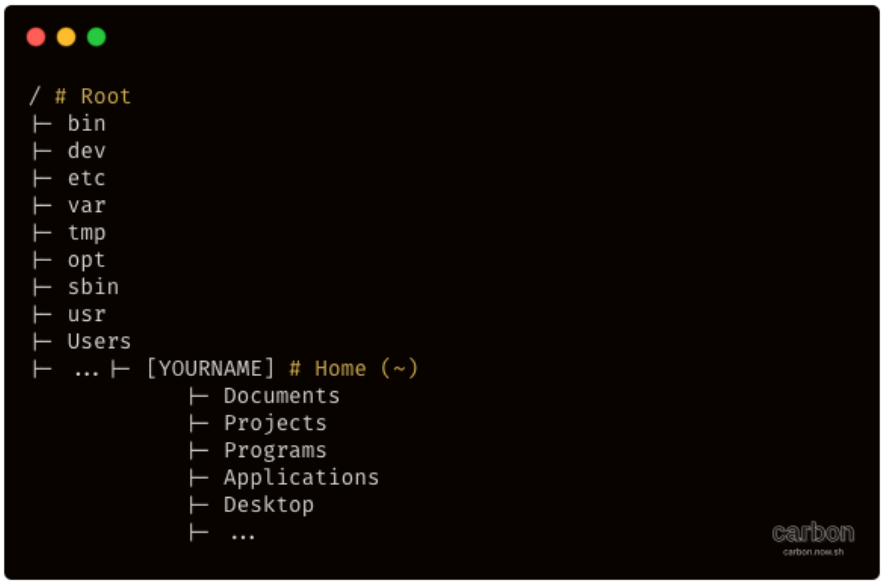



Figure 1.2: A figure displaying tree-like structure of the programs in a machine with macOS

## 1.3 Basic Unix commands

Given that the vast majority of file systems are orginzed in file paths, the first question when starting with the CLI is "Where am I?". So Unix tool system is equiped with a bunch of commands but its basic ones are pretty much oriented to answer that question and navigating this text-based interface of files. The following three commands (`pwd`, `cd`, `ls`) will help you conquer the CLI.

### 1.3.1 Printing your working directory

To know where you are you can see your current location, that is to *print your working directory* using the `pwd` command.

```
pwd
```

### 1.3.2 Change to other directory

```
cd test-dir
```

> **Tip**
>
> Some basic arguments to navigate across your terminal:
>
> ```
> cd .. # change backwards
> cd ~  # change to the home
> cd /  # change to the root
> cd -  # change to previous dir
> ```

### 1.3.3 Listing files

```
ls
```

> **Tip**
>
> You can navigate your executed commands by typing ↑ or ↓.

### 1.3.4 Making new directories

```
mkdir test-dir
```

### 1.3.5 Creating a file

A simple command to create any file inside your terminal is `touch` it just create a file, but do not allow any editing.

```
touch new-file.txt
```

The `new-file.txt` is empty and created on your current location unless you assign another path when creating it. We suggest to take a look at Allison Horst, especially on how to name files depending on the *case* see **?@fig-naming-files**

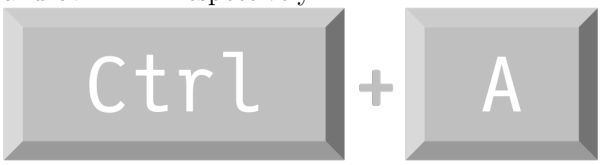
### 1.3.6 Printing files to the screen

```
cat new-file.txt
```

> **Tip**
>
> When using the CLI at first its common to feal quite slow. Then, a very useful tip to boost the productivity from the command line is the autocompletion of commands by hitting `<tab>` after the initial command.

### 1.3.7 Removing files or directories

```
rm
```

> **Tip**
>
> When having a long command, it becomes practically to go to the begining or to the end of it. To do so you can use the key combination `Ctrl + A` and `Ctrl + E` respectively.
>
> 
> 

```
rmdir
```

## 1.4 Anatomy of a command

There is still many conventions by wich the parts of a command line might be called, yet a very standard convention is presented in Figure 1.3

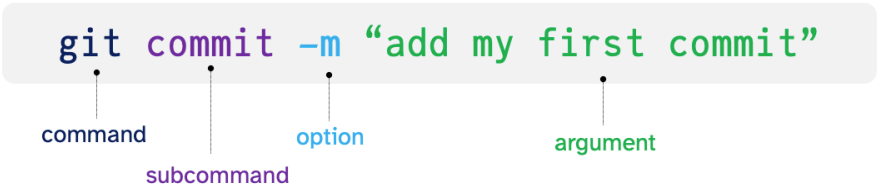



Figure 1.3: A simple command and a convention to call its main components

Some other for instance also tend to call the `option` as `flag`. This conventions are powerful becasue almost any command line interface display this structure (complex one add some other features and simple one tend to lack subcomands).

> **Challenge**
>
> Bacterial defense mechanisms to avoid bacteriophage infections are abundant. One of these is the resctriction-modification system (RM-System), which works by targeting a specific site called *motif*, shared by the phage and bacteria, with methylations. Motifs are commonly represented as a *motif logo* which is a probabilistic representation of the nucleotides in a given position. Find the number of times the motif from **?@fig-motif**

appears on *B. tequilensis* EA-CB0015 genome using a command. Assume that probabilities are equal when multiple bases appeared at one site.

## 1.5 Some great operators/ Metacharacteres in Bash

## 1.6 intermediate Unix commands

```
sed
```

```
grep
```

For more explanations on the basic commands in the command line we suggest to visit the first chapters of *Computing skills for biologist* from Allesina and Wilmes (2019)

# Chapter 2

# Introduction to the control of versions

## 2.1 Git basics

### 2.1.1 Git installation and configuration

### 2.1.2 The basics

### 2.1.3 The local workflow

### 2.1.4 Ignoring files

## 2.2 GitHub

### 2.2.1 Exploring GitHub

### 2.2.2 From local to remote

### 2.2.3 Branching and merging

# Chapter 3

# Package managers

## 3.1 What is the importance of package managers

## 3.2 Conda environments and other package managers

## 3.3 Creating enviroments with Conda

## 3.4 Package managers for OS

There are several package managers handling general purpose packages and apps. For MacOS the famous one is Homebrew and for Windowos several could be used such as Chocolatey and Scoop.

# Chapter 4

# Notions of HPC

## 4.1 What is HPC

## 4.2 Some important concepts of the hardware

See Jakob's blog

## 4.3 Using the Apolo cluster wirh Slurm

# Part II

# Sequence analysis

# Chapter 5

# Introduction to sequence analysis

In this chapter we will discuss several about several points of view about bioinformatics and computational biology, we will further consider several biological concepts that appear central to understand the manipulation of biological data.

## 5.1 Endless debate: bioinformatics vs. computational biology

## 5.2 The duality of DNA

## 5.3 The central ~~dogma~~ theory of molecular biology extended

## 5.4 Sequencing strategies

## 5.5 Sequencing over time

## 5.6 Some insights from sequencing genomes

# Chapter 6

# Sanger analysis

This is a section about the first gen sequencing tech

## 6.1 Databases exploration

## 6.2 Sanger sequencing methods

### 6.2.1 The chain termination method

### 6.2.2 Sanger with capillary electrophoresis

### 6.2.3 Strengths and limitations of Sanger methods

## 6.3 Files from Sanger

## 6.4 Sanger processing workflow

## 6.5 The 16S rRNA and its relevance for sequencing

# Chapter 7

# Sequence alignments

## 7.1 Why do we align sequences?

In search of homology and identy

## 7.2 What is homology

## 7.3 Pairwise alignments algorihtms

### 7.3.1 Hamming distance

### 7.3.2 Edit distance

#### 7.3.2.1 Dynamic programming

### 7.3.3 Needleman-Wunsch (global alignment)

### 7.3.4 Smith-Waterman (local alignment)

## 7.4 The genetic code and Scoring matrices

## 7.5 BLAST and its families

psi-blast? true homologs, recurrent blast to polish scoring matrix during several generations to generate true homologs

## 7.6 Multiple sequence alignments

# Chapter 8

# Phylogenetics

## 8.1 What is a phylogenetic tree

## 8.2 Mehtods for phylogenetic reconstruction

## 8.3 Building a phylogenetic reconstruction

### 8.3.1 Evolutionary substitution model

### 8.3.2 Maximum likelihood

### 8.3.3 Bayesian inference

# Chapter 9

# NGS and TGS: principles

## 9.1 Platforms yields

## 9.2 Reads main differences

## 9.3 Illumina principle (sequencing by synthesis)

### 9.3.1 The fastq format

### 9.3.2 Quality assesment of Illumina

## 9.4 PacBio principle (sequencing by incorporation)

### 9.4.1 Throughput evolution

### 9.4.2 Quality assesment of PacBio

## 9.5 Oxford Nanopore Technology (ONT) principle

### 9.5.1 Platforms

### 9.5.2 The fast5 file format

# Part III

# Genomics

# Chapter 10

# Genome assembly

## 10.1 The problem of assembling genomes

## 10.2 Main algorithms for genome asssembly

### 10.2.1 Overlay, Layout, Consensus (OLS)

### 10.2.2 De Bruijn graphs

## 10.3 Main concepts of an assembly

### 10.3.1 Contigs, Unitigs, Scaffolds

## 10.4 A complete workflow for assembling genomes

## 10.5 Assessing genomes

### 10.5.1 Inspecting genome graphs

### 10.5.2 Genome completeness

## 10.6 Understanding genome difficulties

- End of chromosomes
- Erros
- Lack of coverage
- Heterozigozity

- repeats

# Chapter 11

# Genome annotation and visualizaiton

## 11.1 *ab initio* annotation

## 11.2 Homolgy annotation

## 11.3 Annotation files

### 11.3.1 the GBK and GBFF

### 11.3.2 The GFF specificaitons

## 11.4 Visualizing genomes and annotations

# Chapter 12

# Chapter 13

# Variant calling analysis

## 13.1 Common mutations

## 13.2 Structural variants

## 13.3 Genome rearrengments

## 13.4 Read mapping algorithms and programs

### 13.4.1 Burrow-Wheeler-Alignment

### 13.4.2 BWA-MEM2

### 13.4.3 Minimap2

### 13.4.4 SAM, BAM and CRAM formats

## 13.5 Identifying mutations

### 13.5.1 Freebayes and Snippy

### 13.5.2 The VCF file

# Part IV

# Structural bioinformatics

# Chapter 14

# Introduction to structural bioinformatics

Structural bioinformatics is a multidisciplinary area enriched by chemistry, physics, computer science and many others. Although, it could be focused on different biological macro-molecules, here the emphasis will be focused on proteins.

One of the first protein structure elucidated was myoglobin and it triggered the study of the role of the structure of proteins and its biological functions

Identify a protein related to your study that could be further analyzed.

## 14.1 Protein structures

Difference between the levels of protein structure primary structure is the basic linear representation of aminoacids. Natural aminoacids and modified or rare aminoacids display physico-chemical rich information and could be represented by letters. Therefore in the genetic code we could find the one-letter code.

Secondary structures result from the spatial arrangement of aminoacids that interact with each closer neighbors. There are some remarkable secondary structures such as $\beta$-sheets, $\alpha$-helix, coils (flexible) and others.

Tertiary structure informs about the structural disposition of the secondary structures that fold between each other due to hydrophobic interactions, disulfure bonds, and other chemical interactions forming a globular and dynamic structure. Thus, proteins could display multiple structural states depending on the physical and energy stability (see the Levinthal's paradox).

Quaternary structures result from interaction of multiple tertiary structures. *The structure, therefore dictates the protein function.* This basic concept have

triggered more recently a boom on the analysis of the structure of proteins.

## 14.2 Identifying or predicting protein structures

Xray crystallography, nuclear magnetic resonance (NMR) allows to encapsulated dynamic information of the protein in time Electron micrography (EM) and Cryo-EM. These experiment rely on highly specialized set ups and there are other drawbacks

To date, helium is scarce around the world, so labs all around are having trouble to get this element.

Modelling the structure whether *ab initio* or by *homology* also allow structure prediction. However these strategies

Recently AlphaFold

The protein database (PDB)

Protein topologies resulting from the folding: horshoe, beta-barrel and other could be identified

Structural classification of proteins (SCOP) when analyzing a new protein classification by class, architecture, topology (fold-family), homologous superfamily and sequence family

Importance of Gene Ontology

### 14.2.1 Secondary structure prediction

Secondary structures could also be represented in one letter (e.g.)

Functional domains could be predicted by sequence alignment and allow structural inference. Main predictions are based heavily on machine learning and are frequently accepted under a consensus of multiple tools.

There is a group of protein called intrinsically-disordered proteins that change in its tertiary structures quite frequently, therefore it prediction could be troublesome.

> ⚠ Exercise 01
>
> Submit 5lWM (JAK3) from the PDB on FASTA format on the JPRED and PSIPRED and compared with the experimentally predicted version of the protein. Analyse the predictions and tell are there differences between predictions? Which one is more accurate?

## 14.3 PDB database introduction

The PDB database is one of the most important and ancient open biological database where all new protein structures are submitted. It is an international consortium where several regions work together to curate information.

The PDB in Europe (PDBe) is not only for proteins but form many other *experimentally predicted* macro-molecules (protein-protein interactions, peptides, RNA and so on).

Protein structures are registered using a unique code of four characters.

X-ray crystallography: is a chemical state of the macro-molecule where it is immobilized, therefore information correspond to one state of the structure, then crystal protein is submitted to an x-ray beam to generate a diffraction pattern.

NMR spectroscopy: captures dynamic information of the protein, but generally resolves small proteins. The principle?

Electron microscopy adapted to cryo-preservation allow proteins visualization.

# References

Allesina, Stefano, and Madlen Wilmes. 2019. “Computing Skills for Biologists,” January. https://doi.org/10.2307/j.ctvc77jrc.