Fundamentals of computational biology

Camilo García-Botero

2022-04-11

Table of contents

Preface

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 were basic concepts from biology, bioinformatics and computer science and interleave with the practice to solve challenges.

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.

Part I

Unix

1 Welcome to the command line

In this chapter we will explore the fundamentals of the command line. That is the concepts of Unix based systems the command line (CLI) and how we can use it to access information programmatically.

Part II

Sequence analysis

2 Introduction to sequence analysis

In this chapter we will discuss several about several points of view about bioinformatics and computational biology and how to get started with the command line being a biologist, we will further consider several biological concepts that appear central to understand the manipulation of biological data.

Endless debate: bioinformatics vs. computational biology

Getting started with the command line

The duality of DNA

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

Sequencing strategies

Sequencing over time

Some insights from sequencing genomes

3 Sanger analysis

This is a section about the first gen sequencing tech

Databases exploration

Sanger sequencing methods

The chain termination method

Sanger with capillary electrophoresis

Strengths and limitations of Sanger methods

Files from Sanger

Sanger processing workflow

The 16S rRNA and its relevance for sequencing

Part III

Challenges demostrations

4 Genome searching

In this chapter we will use several tools to download a genome from the command line. We will identify some features

Downloading a genome

```
ncbi-genome-download
```

Downloading from NCBI

The first step in this journey is to download a bunch of sequences programatically. To do so, we will use the program ncbi-genome-download.

You could inspect all the options it provides, now we will set our command as the following:

```
ngd --genera "Bacillus subtilis"\
    -s refseq\
    -l complete\
    -o Data\
    --flat-output\
    --format features\
    -n bacteria\
    | head -n 10
```

```
Considering the following 193 assemblies for download:
GCF_000772125.1 Bacillus subtilis   ATCC 13952
GCF_000772165.1 Bacillus subtilis   ATCC 19217
GCF_000772205.1 Bacillus subtilis   Bs-916
GCF_000782835.1 Bacillus subtilis   SG6
GCF_000789295.1 Bacillus subtilis   PS832
GCF_000952895.1 Bacillus subtilis   BS34A
GCF_000953615.1 Bacillus subtilis   BS49
GCF_001015095.1 Bacillus subtilis   UD1022
GCF_001037985.1 Bacillus subtilis   TO-A JPC
```

Listing files

```
ls Data | head -n 10
```

Decompressing using `gzip`

```
gzip -d *
```

. . .

Some files in our data dir

```
ls Data | head
```

Importing the files into R

```
library(tidyverse)
library(fs)

all_features <- dir_ls("Data/") %>%
  map_df(read_tsv)

all_features %>%
  head()
```

. . .

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --

v ggplot2 3.3.5     v purrr   0.3.4
v tibble  3.1.6     v dplyr   1.0.8
v tidyr   1.2.0     v stringr 1.4.0
v readr   2.1.2     v forcats 0.5.1
```

```
-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
library(fs)

all_features <- dir_ls("Data/") %>%
  map_df(read_tsv)

all_features %>%
  head()
```

```
# A tibble: 0 x 0
```

Data processing

```
all_features_grouped <- all_features %>%
  rename(feature = `# feature`) %>%
  select(assembly, feature) %>%
  group_by(assembly, feature) %>%  operations
  count() %>%
  pivot_wider(names_from = feature, values_from = n) %>%
  arrange(desc(CDS))

all_features_grouped %>%
  head()
```

create a new dataset that will group by features per accession. get read of the weird name of the column. Select these two columns. Group by these two columns to perform. count the numbers of rows based on the applied group. generate a wide dataset sending row names as columns. Arrange descending by the number of CDSs.

5 Sanger processing

Procesing a single `.ab1` pair

```
library(sangeranalyseR)

groEL <- SangerAlignment(
  ABIF_Directory = "~/Projects/Bacillus/Data/Sanger/Inter/groEL/",
  REGEX_SuffixForward = "_1_F.ab1",
  REGEX_SuffixReverse = "_2_R.ab1",
  TrimmingMethod = "M2",
  M2CutoffQualityScore = 33,
  M2SlidingWindowSize = 10
)

writeFasta(groEL,
           outputDir = "~/Documents/Teaching/BiologyCourses/BI0487/Demos/02-demo-sangeranalys
           selection = "contigs_unalignment",
           )

launchApp(groEL)
generateReport(groEL)
qualityBasePlot(groEL)
```

Processing a bulk of `.ab1` files

```
library(fs)
library(purrr)

dirs <- fs::dir_ls("~/Projects/Bacillus/Data/Sanger/Inter")

sanger_bulk <- function(dir) {
```

```
  SangerAlignment(
    ABIF_Directory = dir,
    REGEX_SuffixForward = "_1_F.ab1",
    REGEX_SuffixReverse = "_2_R.ab1"
  )
}

genes <- dirs %>%
  map(sanger_bulk)

launchApp(genes$`/Users/camilogarcia/Projects/Bacillus/Data/Sanger/Inter/gyrA`)

writeFasta(
  outputDir = "~/Documents/Teaching/BiologyCourses/BI0487/Demos/02-demo-sangeranalysis",
  selection = "contigs_unalignment"
)
```

6 Sequence alignment demo

Download sequences

Make sure to use the `--flat-output` avoiding download of multiple metadata

```
ngd --flat-output -p 4 -s genbank -A genome-accessions.txt -F cds-fasta bacteria
```

In this case `cds-fasta` parameter will download the nucleotide sequences of the gene. Other alternatives could be useful such as blast search on a genome database or searching through the GENBANK annotation files (both files also could be downloaded using `ngd`).

Unwrapping FASTA records

NCBI registries came with an undesirable wrapping around the lines of sequencing which basically is inserting a return character after some established number of characters. Then a way to get rid of them is to use a command line utility from AstrobioMike (Mike Lee) which will give a line per sequence after the FASTA header. We can later assume the the first line after the header will be the entire sequence

```
for i in GCA_*; do
    N=$(basename $i .fna);
    bash bit-remove-wraps.sh ${i} > ${N}_unwrapped.fasta;
done
```

Gene search

A possible way to search throughout the file registries is by using the `grep` command, that recursively will search each file. Fine tuned it allow to search for the first match, but also for the "after-context" in terms of lines desired to be printed:

```
grep -h\
     -m 1\
     -A 1\
     -E "DNA gyrase, A| gyrase subunit A | gyrase alpha| gyrase \(subunit A\)| gyrA"  *.fasta
     sed "s/--//g" |\
     sed "/^$/d"
```

After finding the genes we could exclude some lines using `sed` avoid the "–" characters and the empty blank line using the appropriate regular expression (^$/d) . We are now with an almost clean multi sequence file, because header names are still and will be problematic. How do we programatically change the FASTA headers? We will see in the next step.

Renaming fasta headers

A simple but powerful script to do this is `bit-dedup-fasta-heades` it was developed by AstrobioMike (Mike Lee) and it simply parses the headers and substitutes by a simple encoder found en each of them:

```
python bit-dedupe-fasta-headers.py -i all_gyrA.fasta -o all_gyrA_renamed.fasta
```

Now the the files has files names that are simply to work with. Which will enable to asses better out sequence alignment matrix.

Sequence alignment

There are many programs that are suited for performed multiple sequence alignments. Perhaps the two most used are `MAFFT` and `MUSCLE` both specialized in multiple sequence alignment (that is: when having two or more than two sequences). The second tends to be more accurate when having large data-sets, but the first on is more versatile, fast and accurate on different kind of data-sets.

Both programas take as input a single file containing all the sequences concatenated horizontally (that is a multi-fasta file) careless of the extension but (MFA, FA, FASTA, FNA, etc). And generate a simple output (whether with the `-o` in `MUSCLE` or to the std output in `MAFFT`)

```
ginsi --preservecase --reorder all_gyrA_renamed.fasta > all_gyrA_renamed_ginsi.fasta # globa
einsi --preservecase --reorder all_gyrA_renamed.fasta > all_gyrA_renamed_einsi.fasta # gene-
linsi --preservecase --reorder all_gyrA_renamed.fasta > all_gyrA_renamed_linsi.fasta # local

muscle -i all_gyrA_renamed.fasta -o all_gyrA_renamed_muscle.fasta

famsa -t 8 all_gyrA_renamed.fasta > all_gyrA_renamed_famsa.fasta

kalign -i all_gyrA_renamed.fasta -o all_gyrA_renamed_kalign.fasta
```

Assesment of the alignment

Inspection of the alignment is there very first step for assesing its quality. A CDS tends to genereate a codon-like alignment starting with the methione codon (ATG,GTG) and finishing with a stop (TAA, TAG, etc.). Therefore finding this structure when aligning a complete genes is expected. If a middle fraction of the gene is being aligned ORF might not display any stop codon. Verifying a codon-like alignment shows a biological order on the sequences other that mere artifact of the alignment, that is an evolutionary behavior of the sequence. We can do it usin seqfu from the CLI or interactively with AliView.

A second step is to find the variability of the alignment. A simple way to find that is to calculate simpl stats from the alignment (sites, variable sites, As, Ts, etc.). A powerful cli program to do so is goalign

```
goalign stats -i all_gyrA_renamed_linsi.fasta
```

```
length  2508
nseqs   8
avgalleles  1.7400
variable sites  1202
char    nb  freq
-   273 0.013606
A   6418    0.319876
C   3633    0.181071
G   4755    0.236992
T   4985    0.248455
alphabet    nucleotide
```

An alternative approach using BLAST

```
ngd --flat-output -p 4 -s genbank -A genome-accessions.txt -F fasta --parallel 8 bacteria

for i in GCA_*; do cat ${i} >> all_genomes.fasta; done

makeblastdb -in all_genomes.fasta -parse_seqids -blastdb_version 5 -title "demo" -dbtype nucl

blastn -db all_genomes.fasta -query gyrA.fasta -outfmt "6 sseqid sseq" -word_size 5 -evalue 1
```

The alternative using the GCF

```
for i in *fna; do; goalign subset -e "gyrA" -i ${i} --unaligned;done | grep ">"
```

References