Hackathon reproductibilité, M2 AMI2B 2025

Release 3.0

Frederic Lemoine, Thomas Cokelaer

Oct 10, 2025

CONTENTS

CHAPTER

ONE

NEXTFLOW

1.1 Practical

1.1.1 Run a workflow

Try to run the workflow presented in section "1.2.4 Example of a Nextflow workflow", with the following fasta file:

```
>seq1
ACGACTATCGATGATCATCG
>seq2
ACGATCACGACTGACTGACG
>seq3
GCATCGACAGCACTAGCATG
```

- Write the script + config file
- Connect to the cloud and upload the script + config file
- Execute the script using nextflow

1.1.2 Write a workflow

- Create a Nextflow workflow with the following steps:
  - Download the following phylogenetic tree with wget https://booster.pasteur.fr/static/files/primates/ref.nw.gz
  - List the names of the tips with the command *gotree stats tips -i <file>*
- You can use the following docker containers:
  - evolbioinfo/ubuntu:v16.04
  - evolbioinfo/gotree:v0.4.1a
- Write your script locally, push it on a github repo, and pull it from the cloud VM
- Execute the script!

1.2 Useful information

1.2.1 Nextflow documentation

https://www.nextflow.io/docs/latest/

1.2.2 Install Nextflow

- Connect to the VM on the IFB cloud
- Uninstall any java (if needed...): `sudo apt-get remove openjdk-21-*`
- Install Java: `sudo apt-get install openjdk-21-jdk`
- `mkdir bin && cd bin`
- Install nextflow: `curl -s https://get.nextflow.io | bash`
- Test nextflow: `./nextflow run hello`

1.2.3 Install Apptainer (Reminder)

On the IFB cloud:

```
$ sudo apt update
$ sudo apt install -y wget
$ cd /tmp
$ wget https://github.com/apptainer/apptainer/releases/download/v1.4.3/apptainer_1.4.3_
↪amd64.deb
$ sudo apt install -y ./apptainer_1.4.3_amd64.deb
$ rm ./apptainer_1.4.3_amd64.deb
```

1.2.4 Example of a Nextflow workflow

`main.nf`:

```
process reformatFasta {
  input:
  file sequences

  output:
  file "*.phylip"

  script:
  """
  goalign reformat phylip -i ${sequences} -o ${sequences.baseName}.phylip
  """
}

workflow {
    fastaFiles = Channel.fromPath("/home/user/fasta/*.fasta")
```

(continues on next page)

(continued from previous page)

```
    phylipFiles = reformatFasta(fastaFiles)
}
```

`nextflow.config`:

1.2.5 Run the workflow

`nextflow run main.nf`

Nextflow options:

- `-resume` : Restarts an analysis without reexecuting tasks that were succesful or that are not impacted by the workflow update
- `-w <dir>` : Changes the default work directory (default: `work/`)
- `-C <file>`: Changes the default configuration file (default: `nextflow.config`)

CHAPTER

TWO

SNAKEMAKE

In the following, we will play with two small FastQ files. Let us download them:

1. Measles R1 data set
2. Measles R2 data set

Download the two fastq files and unzip them for the next exercice.

2.1 Exercice: execution

Copy this code in a file called Snakefile

```
samples = ['A', 'B']

rule all:
  input:
    expand("stats/{sample}.txt", sample=samples)

rule count:
  input: "{sample}.fastq"
  output: "stats/{sample}.txt"
  run:
    count = 0
    with open(input[0], 'r') as fin:
        for line in fin.readlines():
            count += 1
    with open(output[0], "w") as fout:
        N = int(count / 4)
        fout.write("{}".format(N))
```

Then execute the pipeline and get a flavor of what is going on.

```
snakemake --cores 1
```

Read the error. Adapt the workflow to analyse the two files you have just downloaded.

2.2 Exercice (deal with error)

For this exercice, the files must be compressed. Use **gzip** command to compress them. Copy this code in a file called Snakefile:

```
samples = ['measles_R1_', 'measles_R2_']

rule all:
    input:
        expand("fastqc/{sample}_fastqc.html", sample=samples)

rule fastqc:
    input: "{sample}.fastq.gz"
    output:
        html="fastqc/{sample}_fastqc.html",
        zip="fastqc/{sample}_fastqc.zip"
    log: "logs/{sample}.log"
    shell: "fastqc {input} >{log} 2>&1"
```

Execute the code. What is wrong with the workflow ?

You have two main solutions. The first solution consists in checking the fastqc documentation (--help), and change the pipeline to fix it. A second solution consists in hacking the output.

2.3 Exercice: use singularity/apptainer

Consider the example below. Adapt as in the example above

```
samples = ['measles_R1_', 'measles_R2_']
rule all:
    input:
        expand("seqkit/{sample}.txt", sample=samples)

rule fastqc:
    input: "{sample}.fastq.gz"
    output:
        txt="seqkit/{sample}.txt"
    log: "logs/{sample}.log"
    shell: """seqkit {input} >{log} 2>&1 """
```

Check that the pipeline works using:

```
snakemake -s Snakefile -c 1 --forceall
```

Here we add --forceall just to make sure we are indeed executing the entire pipeline (in case it was already run).

In principle the pipeline fails because the required software **seqkit** is not installed. We do not want to install it. Instead, we will use a container.

Let us add the container keyword. We want to use a container that provide seqkit. You need to add a field **container** and a valid container. It can be an URL or a docker file. If you want to use a URL, **seqkit** is available in the damona.readthedocs.io project.

Install damona and type:

```
damona search seqkit
```

once you have found a container, adapt the snakefile and execute the pipeline with singularity using:

```
snakemake -s Snakefile --use-singularity --forceall -c 1
```

CHAPTER

THREE

PROJECT

3.1 Introduction

The goal is to reproduce parts of the analysis described in this paper (to read):

- https://www.nature.com/articles/s41467-020-15966-7

We want to analyze the RNA-Seq data in order to find differentially expressed genes, i.e. genes that are more (or less) expressed in one condition (persisters) compared to another (control).

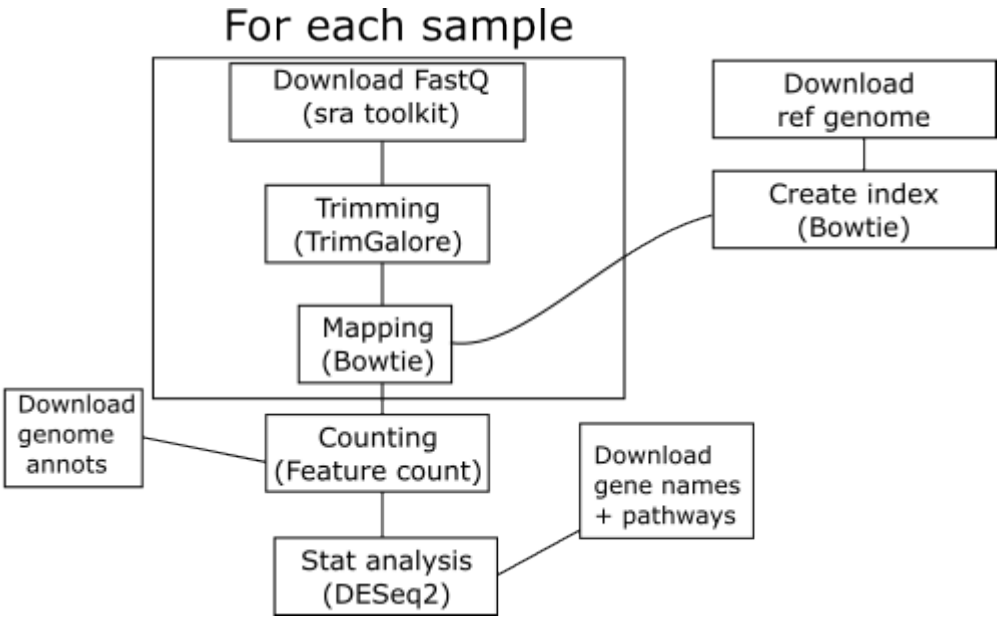
To do so you will design and implement a workflow (Snakemake or Nextflow) that must be reproducible. The work will be done in small groups (4 students).

1. This workflow must use containers that you will *build yourself* (Docker or Apptainer) to run the processes;
2. The code must be readable, commented, and documented;
3. We should be able to re-execute easily the workflow;

Deliverables:

1. Container recipes (Dockerfiles, Apptainer recipes)
2. Workflow code (Nextflow files + configuration or Snakemake files)
3. README.md + run.sh with all instructions for us to reproduce *easily* your analysis
4. **Report with the following parts:**
    a. Introduction (Reproducibility, biological topic of the papers to reproduce, etc.)
    b. Material and Methods (Tools used, and setup)
    c. Results (Workflow developed, results obtained after execution)
    d. Conclusion / perspectives (Interpretation of the results, and conclusion about reproducibility)
5. Oral presentation: Each group will make a presentation (**December 12th?**).
6. **Mid-project evaluation / progress report**
   [**November 7th**:]
    a. 10 minutes oral presentation
    b. **First version of the code (in the git repository) with:**
        - All the image recipes (working) and
        - First steps of the workflow (data download, genome download and indexing) (working)

3.2 Workflow to implement and execute



3.3 Downloading FASTQ files

- Tool: fastq dump
- Commands:

```
$ fasterq-dump --threads <#CPUS> --progress <SRAID>
$ gzip *.fastq
```

If you find better / quicker ways of downloading the data, feel free!

3.4 Trimming the reads

- Commands:

```
$ trim_galore -q 20 --phred33 --length 25  <FASTQ FILE>
```

3.5 Downloading reference genome

- Commands:

```
$ wget -q -O reference.fasta "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?
↪db=nuccore&id=CP000253.1&rettype=fasta"
```

3.6 Downloading reference genome annotations

- Commands:

```
$ wget -O reference.gff "https://www.ncbi.nlm.nih.gov/sviewer/viewer.cgi?db=nuccore&
↪report=gff3&id=CP000253.1"
```

3.7 Creating genome index

- Tool: Bowtie
- Commands:

```
$   bowtie-build <full genome fasta file> <index name>
```

3.8 Mapping FastQ files

- Tool: FastQC
- Commands:

```
$ bowtie -p <#cpus> -S <INDEX NAME> <(gunzip -c <GZIPED FASTQ FILE>) | \
    samtools sort -@ <#CPUS> > <NAME>.bam
$ samtools index <NAME>.bam
```

3.9 Counting reads

- Tool: subread
- Commands:

```
$ featureCounts --extraAttributes Name -t gene -g ID -F GTF -T <#CPUS> -a <GFF> -o␣
↪counts.txt <BAM FILES>
```

With options: #. **-t gene** indicates that counts should be done on gene. You may use other features such as **exon** #. **-g ID** selects the gene ID to store in the output file. This depends on your input GFF file .

3.10 Statistical analysis (differential gene expression)

- Tool: DESeq2
- Commands: You will have to find the method and write the script yourself.

3.11 Additional informations

- **Some steps are not highly described, for example:**
  - Getting the mapping between gene identifiers (in the GFF file) and gene names (needed in the final MA plot). This can be downloaded from AureoWiki.
  - Getting the list of genes involved in translation. This can be found in KEGG, using the REST api for example.