

Introduction to workflow with Snakemake

Memo

Official documentation: <https://snakemake.readthedocs.io/en/stable/> To date (2022/11/28, more than 100 standardized workflows are available: <https://snakemake.github.io/snakemake-workflow-catalog/>

Pre-requisites

Data

Input data to run the workflow example are reduced RNASeq reads files from *Ostreococcus tauri* green algae ((with a focus on chr18) from runs SRR3099585-87 & SRR3105697-99, Bioproject PRJNA304086). To download data from zenodo [here](#), the easiest way is to click on the download icon, but it would download data on our local computer. However, we wish to use these data on Cloud. So we just collect the link to data by doing a right-click. The link is ["https://zenodo.org/record/3997237/files/FAIR_Bioinfo_data.tar.gz?download=1"](https://zenodo.org/record/3997237/files/FAIR_Bioinfo_data.tar.gz?download=1)

Snakemake

For the run of snakemake, deploy a BioPipes VM from Biosphere IFB Cloud (<https://biosphere.france-bioinformatique.fr/cloud/>)

```
$ ssh ubuntu@my.IP
The authenticity of host '193.49.167.73 (193.49.167.73)' can't be established.
ED25519 key fingerprint is SHA256:ek4Sqedg2Uhliet4wBE9WmsoqRBuxVTirRm2lEaMGrY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?

Type "yes"
```

Note : Replace "my.IP" by the IP address provided on your biosphere VM

Activate the environment to use Snakemake and check that the 'snakemake' program works by asking for the version used.

```
$ conda activate snakemake
(snakemake)$ snakemake --version
7.18.2
```

The architecture of your working directory

Create directories to get this architecture:

```
|— data -> /ifb/data
|— envs
|— logs
|— results
|— snakemake
```

Note: On Ubuntu, `mkdir` command does the job.

At the end of the practice, the architecture should look like this:

```

├─ data-> /ifb/data
  └─ data
    └─ mydatalocal
      └─ Data
        ├── SRR3099585_chr18.fastq.gz
        ├── SRR3099586_chr18.fastq.gz
        ├── ...
        ├── SRR3105699_chr18.fastq.gz
        ├── 0.tauri_annotation.gff
        └── 0.tauri_genome.fna
      └─ public
    └─ public
  └─ envs
    └─ qc.yaml
  └─ logs
  └─ results
  └─ snakemake
    ├── Snakefile.ex1.smk
    ├── Snakefile.ex2.smk
    └── ...

```

Conda environment based on the qc.yaml file:

```

name: qc # conda environment name
channels:
  - bioconda
dependencies:
  - fastqc=0.11.9 # quality check of fastq data (java)
  - multiqc=1.13 # reports aggregation (R package)

```

Data management

```

# Go to data/mydatalocal
$ cd ~/data/mydatalocal
$ wget "https://zenodo.org/record/3997237/files/FAIR_Bioinfo_data.tar.gz?download=1"
$ mv 'FAIR_Bioinfo_data.tar.gz?download=1' FAIR_Bioinfo_data.tar.gz
$ tar xvzf FAIR_Bioinfo_data.tar.gz
$ ls Data
0.tauri_annotation.gff  SRR3099585_chr18.fastq.gz  SRR3099587_chr18.fastq.gz
SRR3105698_chr18.fastq.gz
0.tauri_genome.fna      SRR3099586_chr18.fastq.gz  SRR3105697_chr18.fastq.gz
SRR3105699_chr18.fastq.gz

```

Workflow definition

The Snakefile example

The final objective is to create a snakefile to manage a small workflow with 2 steps: i) fastqc ii) multiqc. These two tools belonging to the bioinformatics domain allow to check the quality of high throughput sequence data. They are accessible via a Conda environment, qc.yaml

Hint: If you have already run this notebook, you may need to run:

```
$ rm -rf results/FastQC results/multiqc/multiqc_report.html
```

Objective 1: The rule concept

Create a Snakefile named Snakefile.ex1.smk including the first step of the RNAseq workflow (the reads quality checking thanks to the fastqc tool) on one of the RNAseq files.

fastqc access: through a conda environment (see prerequisites on top) fastqc command: `fastqc --outdir results/FastQC inputFile` `inputFile: SRR3099585_chr18.fastq.gz` in the Data directory outputfiles produced in outdirectory: The 2 result files (*_fastqc.zip & *_fastqc.html) will be located in your outdir and named based on the prefix of input file (eg. SRR3099585_chr18_fastqc.zip)

```
$ cd ~/snakemake
$ touch Snakefile
```

Edit the file with `nano` editor for example:

```
rule fastqc:
    input:
        "data/mydatalocal/Data/SRR3099585_chr18.fastq.gz"
    output:
        "results/FastQC/SRR3099585_chr18_fastqc.zip",
        "results/FastQC/SRR3099585_chr18_fastqc.html"
    conda:
        envs/qc.yaml
    shell: "fastqc --outdir results/FastQC/ {input}"
```

Then execute snakemake:

```
$ pwd
/home/ubuntu
$ snakemake --snakefile snakemake/Snakefile.ex1.smk --cores 1 --use-conda
```

Building DAG of jobs...

Objective 2: One rule, 2 input files

Add a second input RNAseq file to the rule.

Hint: second input file: `Data/SRR3099586_chr18.fastq.gz` Don't forget to add the cognate output files.

Create another Snakefile called Snakedile.ex2.smk:

```
rule fastqc:
    output:
        "FastQC/SRR3099585_chr18_fastqc.zip",
        "FastQC/SRR3099585_chr18_fastqc.html",
        "FastQC/SRR3099586_chr18_fastqc.zip",
        "FastQC/SRR3099586_chr18_fastqc.html"
    input:
```

```
"Data/SRR3099585_chr18.fastq.gz",
"Data/SRR3099586_chr18.fastq.gz"
shell: "fastqc --outdir FastQC/ {input}"
```

Then execute snakemake:

```
$ pwd
/home/ubuntu
$ snakemake --snakefile snakemake/Snakefile.ex2.smk --cores 1 --use-conda
```

Objective 3: Manage all the RNAseq files

Boring with writing all input and output file names? Use the `expand()` function to manage all the input RNAseq files at once.

Hint: Create a Python list at the beginning of the snakefile and containing all the basename of the input files (don't include the `.fastq.gz` suffix). Python list format: `list_name = ["item1", "item2", ..., "itemN"]` Replace the filename lists of the input and output directives by the `expand()` function.

Create another Snakefile called `Snakedile.ex3.smk`:

```
SAMPLES=["SRR3099585_chr18", "SRR3099586_chr18", "SRR3099587_chr18", "SRR3105697_chr18",
"SRR3105698_chr18", "SRR3105699_chr18"]

rule fastqc:
    output:
        expand("FastQC/{sample}_fastqc.zip", sample=SAMPLES),
        expand("FastQC/{sample}_fastqc.html", sample=SAMPLES)
    input:
        expand("data/{sample}.fastq.gz", sample=SAMPLES)
    shell: "fastqc --outdir FastQC {input}"
```

Then execute snakemake:

```
$ pwd
/home/ubuntu
$`snakemake --snakefile snakemake/Snakefile.ex3.smk --cores 1 --use-conda
```

Objective 4: Add a second step

With a second tool, it is a "real" analysis workflow! The second tool `multiqc` will aggregate all the fastqc results.

Hint: `multiqc` command: `multiqc *_fastqc.zip` `multiqc` inputs: the fastqc zip files 2
outputs of `multiqc`: a file `multiqc_report.html` & a repository `multiqc_data` (to manage with `directory("multiqc_data")`)

Create another Snakefile called `Snakedile.ex4.smk`:

```
SAMPLES=["SRR3099585_chr18", "SRR3099586_chr18", "SRR3099587_chr18",
"SRR3105697_chr18", "SRR3105698_chr18", "SRR3105699_chr18"]

rule all:
    input:
```

```

        expand("results/FastQC/{sample}_fastqc.html", sample=SAMPLES),
        "results/multiqc/multiqc_report.html"

rule fastqc:
    input:
        expand("data/mydatalocal/Data/{sample}.fastq.gz", sample=SAMPLES)
    output:
        expand("results/FastQC/{sample}_fastqc.zip", sample=SAMPLES),
        expand("results/FastQC/{sample}_fastqc.html", sample=SAMPLES)
    conda:
        "envs/qc.yaml"
    shell: "fastqc --outdir results/FastQC/ {input}"

rule multiqc:
    input:
        expand("results/FastQC/{sample}_fastqc.zip", sample = SAMPLES)
    output:
        "results/multiqc/multiqc_report.html"
    conda:
        "envs/qc.yaml"
    shell:
        "multiqc --outdir results/multiqc {input}"

```

Then execute snakemake:

```

$ pwd
/home/ubuntu
$ snakemake --snakefile snakemake/Snakefile.ex4.smk --cores 1 --use-conda

```

Extra objective, the log file

In Unix systems, the output of a command is usually sent to 2 separate streams: the expected output to Standard Out (stdout, or ">"), and the error messages to Standard Error (stderr, or "2>"). To integrate stderr and stdout into the same log, use "&>" (use it with care because output files are often printed to stdout).

Hint: Redirect the stdout and stderr streams of the fastqc and multiqc rules by adding a "log:" directive with two variables, out and err to separately redirect each streams.

```

log:
    std="Logs/{sample}_fastqc.std",
    err="Logs/{sample}_fastqc.err"
shell: "fastqc --outdir FastQC/ {input} 1>{log.std} 2>{log.err}"

log:
    std="Logs/multiqc.std",
    err="Logs/multiqc.err"
shell: "multiqc {input} 1>{log.std} 2>{log.err}"

```

A starting point is available in Snakefile.ex5.smk.

Enjoy !!