FAIR_bioinfo : Open Science and FAIR principles in a bioinformatics project

How to make a bioinformatics project more reproducible

C. Hernandez[1] T. Denecker[2] J. Sellier[2] G. Le Corguillé[2]
C. Toffano-Nioche[1]

[1]Institute for Integrative Biology of the Cell (I2BC)
UMR 9198, Université Paris-Sud, CNRS, CEA
91190 - Gif-sur-Yvette, France

[2]IFB Core Cluster taskforce

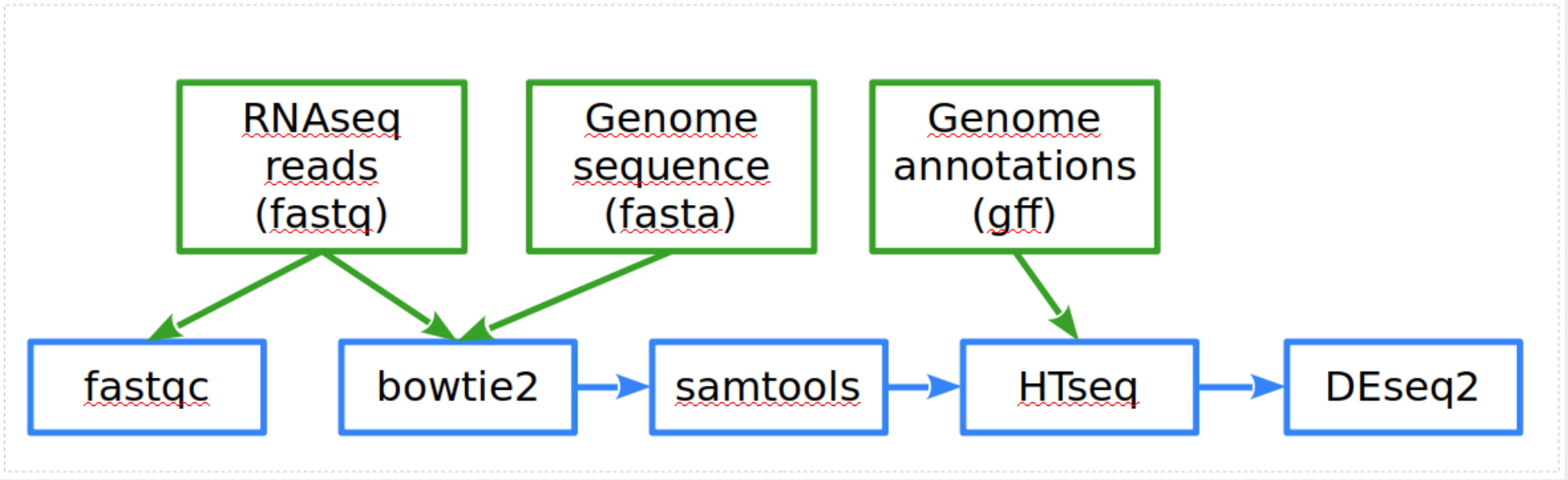
June 2021

Bonus

From a bash script to a complete snakefile

A classical RNAseq analysis

Analysis workflow



green=input, blue=tool

fastqc	control quality of the input reads
bowtie2	reads mapping on the genome sequence
samtools	mapped reads selection & formatting
HTseq	count table of mapped reads on genes (annotations)
DEseq2	statistical analysis: genes list having differential expression

Data, a bash script and its command line

Data

-g genome sequence acces (including extention .fna, .fasta)
-a genome annotation acces (inluding extention .gff)
-d RNAseq sample prefix
next args: RNAseq sample prefix, no .fastq.gz extention

Bash command line

```
FAIR_initial_script.sh -g ../0.tauri_genome.fna -a ../0.
    tauri_annotation.gff -d ../ SRR3099585_chr18 S*86_chr18
    S*87_chr18 S*97_chr18 S*98_chr18 S*99_chr18
```

Script in 3 main blocks

```
1) while getops do ... done
2) for sample in $* ; do ... done
3) creation of the result file, counts.txt, with paste, awk,
     and sed bash commands
```

Complete bash script, 1/3

getops block

```
while getopts g:a:d: flag do
        case $flag in
            g)  genome=$OPTARG
                echo genome is $genome ;;
            a)  annots=$OPTARG
                echo annotation is $annots ;;
            d)  rnadir=$OPTARG
                echo RNAseq path is $rnadir ;;
            :)  echo "L'option $OPTARG requiert un argument"
                exit 1 ;;
            \?) echo "$OPTARG : option invalide"
                exit 1 ;;
        esac
   done
shift $(( OPTIND - 1 ))  # shift past the last flag or
    argument
echo samples are $*
```

Complete bash script, 2/3

for block

```
1  nbs=0;
2  for sample in $* ; do
3      nbs=$(expr ${nbs} + 1)
4      echo traitement of sample ${sample}
5      # --------- quality control of reads
6      if [ ! -d FastQC ]; then
7          mkdir FastQC
8      fi
9      fastqc --outdir FastQC ${rnadir}${sample}.fastq.gz >
       FastQC/${sample}.log 2>&1
10     #---------- reads mapping
11     if [ ! -d Bwt2_index ]; then
12         mkdir Bwt2_index
13         bowtie2-build ${genome} Bwt2_index/tauri > Bwt2_index
       /Bwt2_index.log 2>&1
14     fi
```

Complete bash script, 3/3

for block, continuation

```
    bowtie2 -x Bwt2_index/tauri -U ${rnadir}${sample}.fastq.gz -S ${sample}.sam > ${sample}_bowtie2.log 2>&1
    #---------- selection and format modification
    samtools view -b ${sample}.sam -o ${sample}.bam
    samtools sort ${sample}.bam -o ${sample}_sort.bam
    samtools index ${sample}_sort.bam
    #---------- counting of mapped reads by gene
    featureCounts -t gene -g ID -a ${annots} -s 2 -o ${sample}_ftc.txt ${sample}_sort.bam > ${sample}_ftc.log 2>&1
done
```

Count table block

```
paste *_ftc.txt > ftc_tmp.txt
awk -v nb=${nbs} -v col=7 'BEGIN{FS="\t"}{ctmp=$1; for(i=col;i<=nb*col;i=i+col){count=sprintf("%s\t%s",ctmp,$i);ctmp=count};print count}' ftc_tmp.txt | sed 1d > counts.txt
```

Exercise 2

Continue the snakefile of the previous exercise in order to replace the bash script.
We will:

Objectives

- add a configuration file
- use a builtin snakemake function to get filenames of the input RNAseq data
- add rules to replace the mapping, formatting, counting, and counts aggregating steps of the bash script

ex2_o1.smk

```
1 cp ex1_o7.smk ex2_o1.smk
```

The getops block

Shell script

```
while getopts g:a:d: flag do
        case $flag in
            g)  genome=$OPTARG
            ...
```

We will use a configuration file:

Objective 1

Add a configuration file, named `RNAseq.yml`, containing both the genome sequence and the annotation files manes, and the access to the Data directory.
In the snakefile, change the configured variables (ex. replace `Data/` and `genome` by their `config[]` values). The Python strings concatenation is `+`
Then, run snakemake with the `--configfile` option.

Adding a configuration file

ex2_o1.yml

```
genome:
  O.tauri.fna
annots:
  O.tauri.gff
dataDir:
  Data/
```

ex2_o1.smk: "Data/..." in inputs replaced by a config call:

```
rule genome_bwt2_index:    config["dataDir"]+config["genome"]
rule fastqc:    config["dataDir"]+"{sample}.fastq.gz"
```

snakemake run:

```
rm -Rf FastQC/ Result/ Tmp/ Logs/ ; snakemake -s ex2_o1.smk
    --configfile ex2_o1.yml
```

The for block

Shell script

```
nbs=0;
for sample in $* ; do
    ...
done
```

To manage all *.fastq.gz files in a directory, use the `glob_wilcards()` function. In `ex2_o2.smk`, replace the SAMPLES definition by:

ex2_o2.smk

```
SAMPLES, = glob_wildcards(config["dataDir"]+"{sample}.fastq.
    gz")
```

and run snakemake.

Quality control, fastqc

```
if [ ! -d FastQC ]; then
    mkdir FastQC
fi
fastqc --outdir FastQC ${sample}.fastq.gz > FastQC/${sample}.log 2>&1
```

No more need to test the existence of a directory, it is created as needed.

rule fastqc:

This rule was already present in the snakefile

Reads mapping, bowtie2

```
if [ ! -d Bwt2_index ]; then
    mkdir Bwt2_index
    bowtie2-build ${genome} Bwt2_index/tauri > Bwt2_index/
     Bwt2_index.log 2>&1
fi
bowtie2 -x Bwt2_index/tauri -U ${sample}.fastq.gz -S ${
    sample}.sam > ${sample}_bowtie2.log 2>&1
```

2 rules: genome_bwt2_index (cf. previous ex.) and bwt2_mapping

ex2_o3.smk, rule bwt2_mapping (no run):

```
  output: "results/{sample}.sam"
  input: config["dataDir"]+"{sample}.fastq.gz",
         expand("Tmp/Otauri.{ext}.bt2", ext=BIDX)
  log: "Logs/{sample}_bwt2_mapping.log"
  shell: "bowtie2 -x Tmp/Otauri -U {input[0]} -S {output} 2>
    {log} "
```

Reads mapping, bowtie2

Why some troubles?

The snakemake launch probably didn't do what was expected. What have we forgotten?
We added a new rule to a snakemake but we didn't manage the rule tree, their is no input-output link to include the new rule to the workflow.
We will do that by completing the input directive of the target rule (caution to respect the Python "list" structure, coma-separated).

ex2_o2.smk, target rule:

```
1 rule all:
2   input:
3     expand("FastQC/{sample}_fastqc.html", sample=SAMPLES),
4     expand("Tmp/Otauri.{ext}.bt2", ext=BIDX),
5     expand("Tmp/{sample}.sam", sample=SAMPLES)
```

samtools

Shell script

```
samtools sort -O bam -o ${sample}_sort.bam ${sample}.sam
samtools index ${sample}_sort.bam
```

ex2_o4.smk, rule sam2bam_sort (no run):

```
  output:
    bam="Result/{sample}_sort.bam",
    bai="Result/{sample}_sort.bam.bai"
  input: "Tmp/{sample}.sam"
  log:
    sort="Logs/{sample}_sam2bam_sort.log",
    index="Logs/{sample}_bam2bai.log"
  shell:
    "samtools sort -O bam -o {output.bam} {input} 2> {log.
   sort} ;"
    "samtools index {output.bam} 2> {log.index}"
```

FeatureCount

Shell script

```
featureCounts -t gene -g ID -a ${annots} -s 2 -o ${sample}
    _ftc.txt ${sample}_sort.bam > ${sample}_ftc.log 2>&1
```

ex2_o5.smk, rule counting (params; no run):

```
  output: "Tmp/{sample}_ftc.txt"
  input:
    bam="Result/{sample}_sort.bam",
    annot=config["dataDir"]+config["annots"]
  params: t="gene", g="ID", s="2"
  log: "Logs/{sample}_counts.log"
  shell: "featureCounts -t {params.t} -g {params.g} -a {
    input.annot} -s {params.s} -o {output} {input.bam} &> {
    log}"
```

Counts matrix creation

Shell script

```
paste *_ftc.txt > counts_tmp.txt
awk -v nb=${nb_sample} 'BEGIN{FS="\t"}{count_tmp=$1; for(i=7;i<=nb*7;i=i+7){count=sprintf("%s\t%s",count_tmp,$i); count_tmp=count};print count}' counts_tmp.txt | sed 1d > counts.txt
```

Hint

Create 2 rules to manage some files aggregation to one result file:

- rule `extract_counts`: extract geneID and counts in individual files
- rule `matrix_counts`: paste these files

Counts matrix creation

ex2_o6.smk (2 rules, shell 3", copy, run):

```
rule matrix_counts:
  output: "Result/counts_matrix.txt"
  input: countfile=expand("Tmp/{sample}_ftc7.txt", sample=
    SAMPLES), geneID=expand("Tmp/{sample}_ftc1.txt", sample=
    SAMPLES)
  log: "Logs/matrix_counts.log"
  shell: """cp {input.geneID[0]} Tmp/ftc_geneID.txt > {log}
    ; paste Tmp/ftc_geneID.txt {input.countfile} > {output}
    > {log}"""

rule extract_counts:
  output: col7="Tmp/{sample}_ftc7.txt",
          col1="Tmp/{sample}_ftc1.txt"
  input: "Tmp/{sample}_ftc.txt"
  log: "Logs/{sample}_extract_counts.log"
  shell: """cut -f 7 {input} | sed 1d > {output.col7} > {log
    } ; cut -f 1 {input} | sed 1d > {output.col1} """
```

DESeq2

The DESeq2 step is the statistical analysis. From the count matrix, the statistical analysis is managed by a non parallelizable R script, DESeq2.

So, up to date, the workflow is complete and the only thing left is this DESeq2 statistical analysis. We will see this analysis through the notebooks session.

Bonus

Add a help rule

https://lachlandeer.github.io/snakemake-econ-r-tutorial/self-documenting-help.html#a-help-rule