Development of Snakemake Workflows designed for ChIP-seq analysis Integration in a Virtual Machine on the IFB cloud

Claire Rioualen[1,2], Lucie Khamvongsa[1,2], Christophe Blanchet[3], Jacques van Helden[1,2]
Contact : Jacques.van-Helden@univ-amu.fr, claire.rioualen@inserm.fr

1. INSERM, U1090 TAGC, Marseille F-13288, France.
2. Aix-Marseille Université, U1090 TAGC, Marseille F-13288, France.
3. CNRS, UMS 3601, Institut Français de Bioinformatique, IFB-core, Gif-sur-Yvette F-91190 , France.

WP 2.6 : Gene expression regulation

Workpackages were launched by France Génomique[1] in order to better coordinate scientists efforts in the development of protocoles and pipelines, as well as capitalize on knowledge and good practices. WP 2.6 focuses on gene expression regulation, and includes a section about ChIP-seq analyses which we have been putting efforts in.

We have focused on developing workflows using the Snakemake environment[2], for its scalability and ease of implementation. Then we proceeded with building a catalogue of tools for ChIP-seq analysis, that were implemented in our project. Finally, this work was integrated in a stand-alone appliance, runnable on the cloud of the French Bioinformatics Institute (IFB)[3].

Snakemake, a flexible environment for workflow development

From rules to targets: Snakemake concepts

Snakemake (Köster & Rahmann, 2012) is a python library made for building workflows. Based on the python language, it inherits concepts from the **GNU make** sofware: an ensemble of **rules** that can complete a number of operations characterized by their inputs and outputs, and **target** files to be generated through these operations.

Rule "all" defines **targets** to be completed.

Rule "peak-calling" can produce this target.

Rules are linked together by **dependencies**. In order to generate the target, rule "peak-calling" needs rule "mapping" to run, which needs "import_fastq" to run as well.

Several languages can be used in order to achieve this:

R
shell
python

workflow.py

```
"""Workflow example.
Author: Claire Rioualen.
Shell command: snakemake -s workflow.py
"""

configfile: "workflow.yml"

# Data
READS = config["dir"]["reads"]
RESULTS = config["dir"]["results"]

# Sample IDs
SAMPLES = ["sample1", "sample2"]
CONTROL = ["sample1"]
TREATMENT = ["sample2"]

rule all:
    input: expand("{results_directory}/{treatment}_vs_{control}.bed", \
        results_directory=RESULTS, treatment=TREATMENT, control=CONTROL)

rule peak_calling:
    input: control="{control}.sam", treatment="{treatment}.sam"
    output: "{treatment}_vs_{control}.bed"
    params: fdr=config["homer"]["fdr"]
    run: R("findPeaks {input.treatment} -i {input.control} -fdr {params.fdr} -o {output}")

rule mapping:
    input: "{samples}.fastq"
    output: "{samples}.sam"
    params: genome=config["genome"]["file"]
    shell: "bwa mem {params.genome} {input} > {output}"

rule import_fastq:
    input: "{reads_directory}/{samples}.sra"
    output: "{results_directory}/{samples}.fastq"
    params: outdir=config["dir"]["results"]
    run: os.system("fastq-dump --outdir {params.outdir} {input}")
```

workflow.yml

```
---
  description: |
    Config file designed to work with workflow.py

  genome:
    organism: "Arabidopsis thaliana"
    version: "TAIR10"
    file: "/project/genome/TAIR10.fasta"

  dir:
    reads: "/project/raw_data/"
    results: "/project/results/"

  homer:    fdr: "0.01"
```

Snakemake include many features

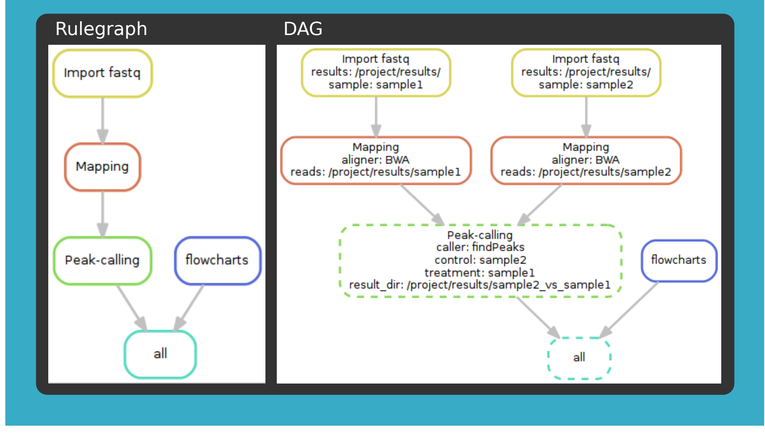
Configuration with JSON or YAML files

Automatic management of **parallelization** (via qsub)

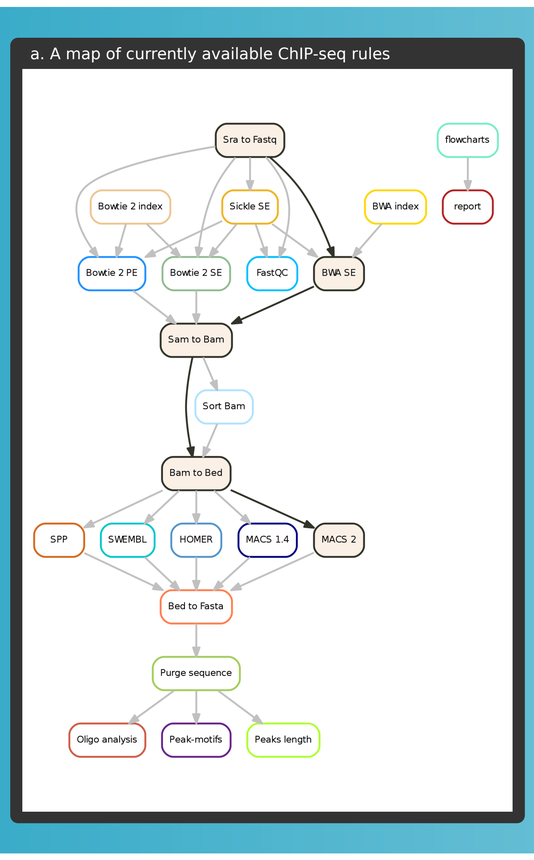
Benchmarking with alternative tools and parameters

Automatic generation of flowcharts:

- graph of rule dependencies (**rulegraph**)
- direct acyclic graph of jobs (**dag**)



A collaborative workflow designed for ChIP-seq analyses



We implemented a catalogue of shared rules, that can be used like "bricks" to build custom workflows, by picking up and linking rules of interest.

This approach shows many advantages:

Modularity: one can pick up their own rules in order to build a custom pipeline.

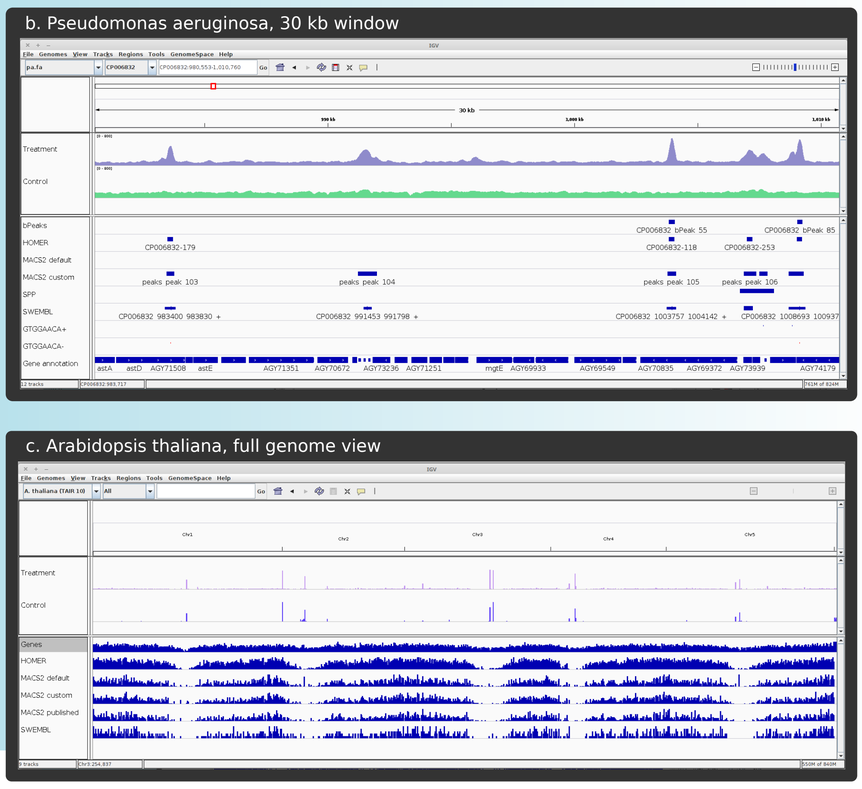
Recycling: a given rule can be used in several workflows, by several people.

Collaboration: several developpers can share their work, knowledge and tools are accumulated.

Portability: pipelines were already run on several servers, personal computers and virtual machines. Moreover, all the rules and a number of standard workflows are maintained in a git repository hosted by the Renater platform[4].

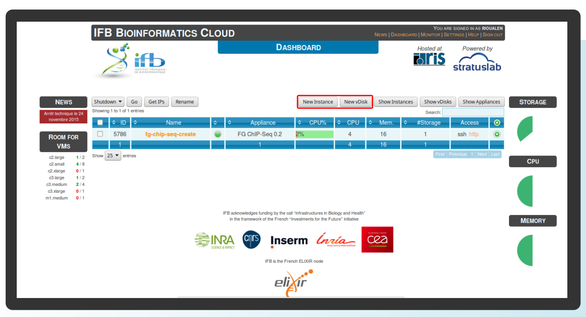
Flexibility: the config file allows the easy tuning of parameters, so we can cater workflows to different species (bacteria, plants, mammalians...).

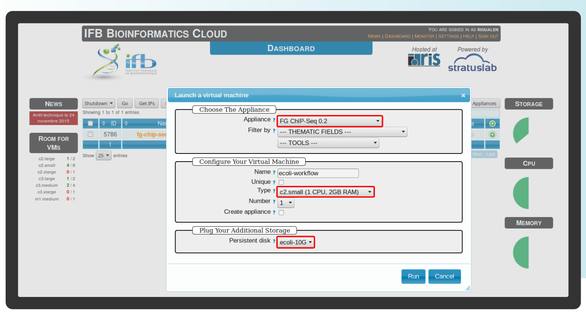
Benchmarking: a number of alternative software are already available for benchmarking the peak-calling and the mapping steps.



Figures
a. Map of the currently available rules and their interconnections. An example of custom workflow is highlighted.
b. The peak-calling step can prove to produce very different outcomes, depending on the algorithm choice and its parameters[5,7].
c. Global view of the peaks in A. thaliana show an interesting view on its chromosomes and centromeres[6,7].

Creation of an appliance on the IFB cloud






```
rg@vm0061:~/workspace/fg-chip-seq$ snakemake -s workflow.py -p
Provided cores: 1
Rules claiming more threads will be scaled down.
Job counts:
        count   jobs
        1       all
        2       import_fastq
        2       mapping
        1       peak_calling
        6
rule import_fastq:
        input: sample2.sra
        output: sample2.fastq
cp sample2.sra sample2.fastq
1 of 6 steps (17%) done
rule import_fastq:
        input: sample1.sra
        output: sample1.fastq
cp sample1.sra sample1.fastq
2 of 6 steps (33%) done
rule mapping:
        input: sample2.fastq
        output: sample2.sam
cp sample2.fastq sample2.sam
3 of 6 steps (50%) done
rule mapping:
        input: sample1.fastq
        output: sample1.sam
cp sample1.fastq sample1.sam
4 of 6 steps (67%) done
rule peak_calling:
        input: sample2.sam, sample1.sam
        output: sample2_vs_sample1.bed
touch sample2_vs_sample1.bed
5 of 6 steps (83%) done
localrule all:
        input: sample2_vs_sample1.bed
6 of 6 steps (100%) done
rg@vm0061:~/workspace/fg-chip-seq$ snakemake -s workflow.py -p
Nothing to be done.
rg@vm0061:~/workspace/fg-chip-seq$
```

The **IFB**[3] is a national gathering of platforms, which provides services and infrastructures in bioinformatics for people working in the field of life sciences.

The **IFB cloud** currently proposes resources amounting to 2 To of RAM and 50 To of storage, and is constantly increasing. We developed an **appliance** that anyone holding a user acccount on the cloud can run on their own data.

The **virtual machine** is made of 4 principal components:

- Operating system (Ubuntu 14.04)
- Programming tools (snakemake, python, libraries...)
- NGS tools (mapping, peak-calling, file conversion...)
- FG-ChIP-seq git repository (rules, workflows, configfiles)

In order to **run a workflow**, main steps are:

- Creating a virtual disk (vDisk) for data and results storage
- Running an instance (curr. up to 8 CPUs/32Gob RAM) with the vDisk mounted
- Loading data on the vDisk
- Running the pipeline in command line

We developed a set of workflows that can apply to a number of ChIP-seq analyses, and that can be shared easily between scientists.

The workflows created were tested on a **variety of organisms**: bacteria (E.coli, P.aeruginosa), yeast (S.cerevisiae, C.albicans), metazoa (D.melanogaster, C.elegans) and plants (A.thaliana).

Though it was so far designed only for **transcription factor marks**, it should be soon adapted to **histone marks**, as well as replication origins. We will also include the **functional analysis of peaks**, and their enrichment in gene ontologies. These tools will allow us to realize the benchmarking of ChIP-seq procedures, as prescribed by the WP 2.6.

A collaboration with colleagues also gave rise to a workflow for **differential analysis of RNA-seq** data.

References
1. France Génomique: www.france-genomique.org
2. Snakemake - a scalable bioinformatics workflow engine. Johannes Köster and Sven Rahmann. Bioinformatics (2012) 28 (19): 2520-2522. doi: 10.1093/bioinformatics/bts480
3. Institut Français de Bioinformatique: www.france-bioinformatique.fr
4. SourceSup - Renater: http://sourcesup.renater.fr/projects/fg-chip-seq/
5. Unpublished data. With the courtesy of Valentine Lagage, I2BC (CEA, CNRS, Université Paris Sud).
6. Transcriptional repression by MYB3R proteins regulates plant organ growth. Kobayashi et al. EMBO J. 2015 Aug 4;34(15):1992-2007. doi: 10.15252/embj.201490899.
Data: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM1482283
7. Integrative Genomics Viewer. Robinson, James T. et al. Nature Biotechnology 29.1 (2011): 24–26. PMC.

Acknowledgments
Valentine Lagage, Bruno Contreras Moreira. This work is funded by France Génomique.