GeneRegulation Documentation

Release 3.0

Claire Rioualen, Jacques van Helden

Apr 05, 2017

Contents

This git repository holds shared code for the analysis of Next Generation Sequencing data related to gene regulation: ChIP-seq, RNA-seq, and related technologies.

It was developed as part of the France Genomique Workpackage 2.6: Gene expression regulation.

One of the key goals is to ensure portability and re-usability of the code.

We have chosen to use the Snakemake workflow management system[1] in order to build reproducible and flexible NGS analysis pipelines.

CHAPTER 1

Contact

- Claire Rioualen claire.rioualen@inserm.fr
- Jacques van Helden Jacques.van-helden@univ-amu.fr

CHAPTER 2

References

1. Köster, Johannes and Rahmann, Sven. “Snakemake - A scalable bioinformatics workflow engine”. Bioinformatics 2012.

CHAPTER 3

User guide and reference

Getting started

The above commands and tutorials are designed for a Unix environment, and were tested under the OS Linux Mint Debian Edition (LMDE).

They assume that the gene-regulation library is downloaded or cloned into your home directory. You can specify any other preferred directory.

Download Gene-regulation

```
cd ${HOME}
wget --no-clobber https://github.com/rioualen/gene-regulation/archive/4.0.tar.gz
↪#archive doesn't exist yet
tar xvzf 4.0.tar.gz
GENE_REG_PATH=${HOME}/gene-regulation-4.0
```

Clone Gene-regulation

```
cd ${HOME}
git clone https://github.com/rioualen/gene-regulation.git
```

Run Gene-regulation

Please refer yourself to the Tutorials section.

Gene-regulation library

The library contains a variety of files, including scripts and configuration files.

You can find a description of these hereafter.

Snakemake files (snakefiles)

Snakefiles are based on the scripting language Python 3, and use a specific syntax.

For organization purpose, we have distinguished two types of Snakefiles:

- Rules are typically "bricks" to build workflows with. Each rule corresponds to a specific operation.
- Workflows are combinations of rules that serve a specific purpose: quality check of sequencing data, ChIP-seq peaks analysis...

Workflows (.wf)

File extension: *.wf

todo

Rules (.rules)

todo

Python scripts (.py)

todo

R scripts

todo

Configuration files (yaml)

todo

R markdown files (.Rmd)

todo

Tabulated files (.tab)

We use tabulated files in order to define and describe the samples to be processed in the workflows.

Examples of these files are available in the *examples* folder of the library.

Sample description files (samples.tab)

todo

Experimental design files (design.tab)

todo

Tutorials

Initial setup

Gene-regulation library

For each study presented here we're creating a link to the gene-regulation library, previously downloaded in section "Quick start".

Note: if you're using a clone of the library, you might want to make a copy of it, in order to ensure consistency for later analyses.

Genome directory

We chose to define a permanent location for genome downloads, then create symlinks for study cases.

```
GENOME_DIR=$HOME/genome
mkdir ${GENOME_DIR}
```

ChIP-seq study case in *S. cerevisiae*

Presentation

Reference

Preti M, Ribeyre C, Pascali C, Bosio MC et al. The telomere-binding protein Tbf1 demarcates snoRNA gene promoters in Saccharomyces cerevisiae. Mol Cell 2010 May 28;38(4):614-20. PMID: 20513435

Abstract

Small nucleolar RNAs (snoRNAs) play a key role in ribosomal RNA biogenesis, yet factors controlling their expression are unknown. We found that the majority of Saccharomyces snoRNA promoters display an aRCCCTaa sequence motif at the upstream border of a TATA-containing nucleosome-free region. Genome-wide ChIP-seq analysis showed that these motifs are bound by Tbf1, a telomere-binding protein known to recognize mammalian-like T(2)AG(3) repeats at subtelomeric regions. Tbf1 has over 100 additional promoter targets, including several other genes involved in ribosome biogenesis and the TBF1 gene itself. Tbf1 is required for full snoRNA expression, yet it does not influence nucleosome positioning at snoRNA promoters. In contrast, Tbf1 contributes to nucleosome exclusion at non-snoRNA promoters, where it selectively colocalizes with the Tbf1-interacting zinc-finger proteins Vid22 and Ygr071c. Our data show that, besides the ribosomal protein gene regulator Rap1, a second telomere-binding protein also functions as a transcriptional regulator linked to yeast ribosome biogenesis.

Access link

- GEO series: GSE20870

Download reference genome & annotations

```
mkdir ${GENOME_DIR}/sacCer2
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/fasta/saccharomyces_
↪cerevisiae/dna/Saccharomyces_cerevisiae.R64-1-1.30.dna.genome.fa.gz -P${GENOME_DIR}/
↪sacCer2
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/gff3/saccharomyces_
↪cerevisiae/Saccharomyces_cerevisiae.R64-1-1.30.gff3.gz -P ${GENOME_DIR}/sacCer2
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/gtf/saccharomyces_
↪cerevisiae/Saccharomyces_cerevisiae.R64-1-1.30.gtf.gz -P ${GENOME_DIR}/sacCer2
gunzip ${GENOME_DIR}/sacCer2/*.gz
```

Setup analysis environment

```
ANALYSIS_DIR=$HOME/ChIP-seq_GSE20870
mkdir -p ${ANALYSIS_DIR}
ln -s ${GENE_REG_PATH} gene-regulation
ln -s ${GENOME_DIR}/sacCer2 genome
CONFIG=${ANALYSIS_DIR}/gene-regulation/examples/ChIP-seq_SE_GSE41187/config.yml
```

Download data

```
mkdir -p ${ANALYSIS_DIR}/data/GSM521934 ${ANALYSIS_DIR}/data/GSM521935
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX021%2FSRX021358/SRR051929/SRR051929.sra -P ${ANALYSIS_DIR}/data/GSM521934
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX021%2FSRX021359/SRR051930/SRR051930.sra -P ${ANALYSIS_DIR}/data/GSM521935
```

show file arborescence

Workflow 'import_from_sra'

The purpose of this workflow is to convert .sra files to .fastq files. The .sra format (Short Read Archive) is used by the GEO database, but for downstream analyses we need to dispose of fastq-formatted files. *insert link to glossary section about file formats*

It order to run it, you must have followed sections "Setup analysis environment" and "Download data" for the dataset GSE20870.

Workflow execution

```
snakemake -s ${ANALYSIS_DIR}/gene-regulation/scripts/snakefiles/workflows/import_from_
↪sra.wf -p --configfile ${CONFIG}
```

show file arborescence

Workflow 'quality_control'

This workflow can be run after the workflow 'import_from_sra', or directly on properly-organized fastq files (see following section if you dispose of your own data).

The purpose of this workflow is to perform quality check with FastQC (*link to website and wiki*).

Optionally, trimming can be performed using the tool Sickle (*link to website and wiki*).

```
cd ${ANALYSIS_DIR}
```

show file arborescence

Workflow execution

```
snakemake -s ${ANALYSIS_DIR}/gene-regulation/scripts/snakefiles/workflows/quality_
↪control.wf -p --configfile ${CONFIG}
```

show arborescence and/or FastQC screencaps

Workflow 'ChIP-seq'

This workflows performs:

- mapping with various algorithms;
- genome coverage in different formats (*see glossary*);
- peak-calling with various algorithms;
- motifs search with suite RSAT (*ref*).

It order to run it, you must have followed sections "Setup analysis environment" and "Download data", and "Download genome and annotation" for the dataset GSE20870.

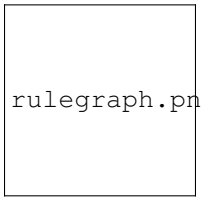
You must have run at least the workflow "import_from_sra', and optionally the workflow "quality_control".

```
cd ${ANALYSIS_DIR}
```

Workflow execution

```
snakemake -s ${ANALYSIS_DIR}/gene-regulation/scripts/snakefiles/workflows/ChIP-seq.wf␣
↪-p --configfile ${CONFIG}
```

add figure



Genome-scale analysis of *Escherichia coli* FNR

Presentation

Description

Note: this dataset should be replaced soon by a smaller one

Reference

Myers KS, Yan H, Ong IM, Chung D et al. Genome-scale analysis of Escherichia coli FNR reveals complex features of transcription factor binding. PLoS Genet 2013 Jun;9(6):e1003565. PMID: 23818864

GEO series

- ChIP-seq: GSE41187
- RNA-seq: GSE41190

Download reference genome & annotations

```
mkdir ${GENOME_DIR}/Ecoli-K12
wget -nc ftp://ftp.ensemblgenomes.org/pub/release-21/bacteria/fasta/bacteria_22_
↪collection/escherichia_coli_str_k_12_substr_mg1655/dna/Escherichia_coli_str_k_12_
↪substr_mg1655.GCA_000005845.1.21.dna.genome.fa.gz -P ${GENOME_DIR}/Ecoli-K12
wget -nc ftp://ftp.ensemblgenomes.org/pub/release-21/bacteria/gff3/bacteria_22_
↪collection/escherichia_coli_str_k_12_substr_mg1655/Escherichia_coli_str_k_12_substr_
↪mg1655.GCA_000005845.1.21.gff3.gz -P ${GENOME_DIR}/Ecoli-K12
wget -nc ftp://ftp.ensemblgenomes.org/pub/release-21/bacteria/gtf/bacteria_22_
↪collection/escherichia_coli_str_k_12_substr_mg1655/Escherichia_coli_str_k_12_substr_
↪mg1655.GCA_000005845.1.21.gtf.gz -P ${GENOME_DIR}/Ecoli-K12
gunzip ${GENOME_DIR}/Ecoli-K12/*.gz
```

Setup analysis environment

```
ANALYSIS_DIR=${HOME}/Integrated_analysis
```

Workflow 'ChIP-seq'

Setup analysis environment

```
ANALYSIS_DIR_CHIP=${ANALYSIS_DIR}/ChIP-seq_GSE41187
mkdir -p ${ANALYSIS_DIR_CHIP}
ln -s ${GENE_REG_PATH} ${ANALYSIS_DIR_CHIP}/gene-regulation
ln -s ${GENOME_DIR} ${ANALYSIS_DIR_CHIP}/genome
CONFIG_CHIP=${ANALYSIS_DIR_CHIP}/gene-regulation/examples/ChIP-seq_SE_GSE41187/config.
↪yml
```

Download ChIP-seq data

```
mkdir -p ${ANALYSIS_DIR_CHIP}/data/GSM1010224 ${ANALYSIS_DIR_CHIP}/data/GSM1010219 $
↪{ANALYSIS_DIR_CHIP}/data/GSM1010220
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX189%2FSRX189778/SRR576938/SRR576938.sra -P ${ANALYSIS_DIR_CHIP}/data/
↪GSM1010224
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX189%2FSRX189773/SRR576933/SRR576933.sra -P ${ANALYSIS_DIR_CHIP}/data/
↪GSM1010219
```

```
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX189/SRX189774/SRR576934/SRR576934.sra -P ${ANALYSIS_DIR_CHIP}/data/GSM1010220
```

Workflow execution

```
snakemake -s ${ANALYSIS_DIR_CHIP}/gene-regulation/scripts/snakefiles/workflows/import_
↪to_fastq.wf -p --configfile ${CONFIG_CHIP}
snakemake -s ${ANALYSIS_DIR_CHIP}/gene-regulation/scripts/snakefiles/workflows/
↪quality_control.wf -p --configfile ${CONFIG_CHIP}
snakemake -s ${ANALYSIS_DIR_CHIP}/gene-regulation/scripts/snakefiles/workflows/ChIP-
↪seq.wf -p --configfile ${CONFIG_CHIP}
```

Workflow 'RNA-seq' DEG

Setup analysis environment

```
ANALYSIS_DIR_RNA=${ANALYSIS_DIR}/RNA-seq_GSE41190
mkdir ${ANALYSIS_DIR_RNA}
ln -s ${GENE_REG_PATH} ${ANALYSIS_DIR_RNA}/gene-regulation
ln -s ${GENOME_DIR} ${ANALYSIS_DIR_RNA}/genome
CONFIG_RNA=${ANALYSIS_DIR_RNA}/gene-regulation/examples/RNA-seq_PE_GSE41190/config.yml
```

Download RNA-seq data

```
mkdir -p ${ANALYSIS_DIR_RNA}/data/GSM1010244 ${ANALYSIS_DIR_RNA}/data/GSM1010245 $
↪{ANALYSIS_DIR_RNA}/data/GSM1010246 ${ANALYSIS_DIR_RNA}/data/GSM1010247
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX264/SRX2641374/SRR5344681/SRR5344681.sra -P ${ANALYSIS_DIR_RNA}/data/
↪GSM1010244
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX264/SRX2641375/SRR5344682/SRR5344682.sra -P ${ANALYSIS_DIR_RNA}/data/
↪GSM1010245
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX264/SRX2641376/SRR5344683/SRR5344683.sra -P ${ANALYSIS_DIR_RNA}/data/
↪GSM1010246
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX264/SRX2641377/SRR5344684/SRR5344684.sra -P ${ANALYSIS_DIR_RNA}/data/
↪GSM1010247
```

Workflow execution

```
snakemake -s ${ANALYSIS_DIR_RNA}/gene-regulation/scripts/snakefiles/workflows/import_
↪to_fastq.wf -p --configfile ${CONFIG_RNA}
snakemake -s ${ANALYSIS_DIR_RNA}/gene-regulation/scripts/snakefiles/workflows/quality_
↪control.wf -p --configfile ${CONFIG_RNA}
snakemake -s ${ANALYSIS_DIR_RNA}/gene-regulation/scripts/snakefiles/workflows/RNA-seq_
↪workflow_PE.py -p --configfile ${CONFIG_RNA}
```

Workflow 'integrated_ChIP_RNA'

todo

Study case yet to find

Workflow alternative transcripts

Study case yet to find

Workflow orthologs

todo after we revise the Glossine dataset analysis

Running Gene-regulation workflows on your own data

Requirements

Assuming you have followed section 3.1 "Initial setup", you should have defined a location for the genome files and the Gene-regulation library.

Besides, you shoud dispose of you own fastq files.

TODO

Fastq files organization

Metadata

samples.tab

design.tab

config.yml

workflow.wf

Dependencies

Note: this section needs to be refreshed

Manual installation

This manual aims at helping you install the necessary programs and dependencies in order to have the snakemake workflows work. It was designed for Unix-running computers (Ubuntu, Debian).

General requirements

Generic tools

ssh

```
sudo apt-get install ssh
ssh-keygen
```

rsync

rsync is an open source utility that provides fast incremental file transfer.

```
sudo apt-get install rsync
```

git

- Install git on your machine.

```
sudo apt-get install git
```

Optional:

- Create an account on GitHub.
- Add your ssh public key to your GitHub account settings (account > settings > SSH keys > add SSH key).

```
less ~/.ssh/id_rsa.pub
```

zlib

Several tools require this dependency (e.g. sickle, bamtools...).

```
sudo apt-get install libz-dev
```

qsub

Create bin/ and app_sources/ (optional)

While some programs will be installed completely automatically, others will not. Here we create a directory that will be used for manual installations.

```
mkdir $HOME/bin
mkdir $HOME/app_sources
```

You might then have to edit your `$PATH` manually (see next section).

Edit `$PATH`

In order to use manually installed programs and make them executable, you may have to update your `$PATH` environment variable. You can do so by editing the `~/.profile` file.

```
nano ~/.profile
```

Fetch this paragraph and add the path to manually installed executables:

```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

Execute the file to validate the change.

```
source ~/.profile
```

Snakemake workflows basic requirements

Python

Snakemake requires to have Python 3.3+ installed. You can check this by issuing the following commands in a terminal:

```
python --version # usually the default python version is 2.7+
python3 --version
```

If you don't have python 3 you should install it.

```
sudo apt-get install python3
```

Install pip and pip3.

```
sudo apt-get install python-pip
sudo apt-get install python3-pip
```

Not installed natively?

```
apt-get install python-dev
apt-get install python3.4-dev
```

Pandas library

This library is used in order to read tab-delimited files used in the workflows (see files `samples.tab` and `design.tab`).

```
pip3 install pandas
```

Package rpy2

```
pip3 install "rpy2<2.3.10"
```

R

todo

Snakemake

- Documentation
- FAQ
- Forum
- See also Snakemake section for tutorials.

Now you have installed Python 3 and pip3 (see previous section), you can install snakemake safely.

```
pip3 install snakemake
```

You can check that snakemake works properly with this basic script:

```
"""Snakefile to test basic functions of snakemake.
"""
rule all:
    input: expand("bye.txt")

rule hello:
    """Write HELLO in a text file named hello.txt.
    """
    output: "hello.txt"
    message: "Generating {output} file."
    shell: "echo HELLO > {output}"

rule bye:
    """Write BYE in a text file named bye.txt.
    """
    input: "hello.txt"
    output: "bye.txt"
    message: "Generating {output} file."
    shell: "echo BYE > {output}"
```

- Save it to `~/workspace/hello.py`.
- Issue the command `cd ~/workspace ; snakemake -s hello.py`.
- 2 files should be created: `hello.txt` and `bye.txt`.

As of December 2015, you need snakemake version 3.4+.

```
pip3 install snakemake --upgrade
```

If you want to use Snakemake reports function (optional):

```
pip3 install docutils
```

Graphviz

Snakemake can generate useful graphviz outputs.

```
sudo apt-get install graphviz
```

NGS analysis software & tools

Quality assessment

FastQC

FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

The main functions of FastQC are:

- Import of data from BAM, SAM or FastQ files (any variant)
- Providing a quick overview to tell you in which areas there may be problems
- Summary graphs and tables to quickly assess your data
- Export of results to an HTML based permanent report
- Offline operation to allow automated generation of reports without running the interactive application

Links:

- QC Fail Sequencing
- FastQC results interpretation

FastQC is available from linux repositories:

```
sudo apt-get install fastqc
```

However, since it's an older version, it can cause problems of dependencies.

We recommend installing it manually:

```
cd $HOME/app_sources
wget --no-clobber http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.
↪11.5.zip
unzip -o fastqc_v0.11.5.zip
chmod +x FastQC/fastqc
ln -s -f $HOME/app_sources/FastQC/fastqc $HOME/bin/fastqc
```

Trimming

The quality of the reads generated by high-throughput sequencing technologies tend to decrease at their ends. Trimming consists in cutting out theses ends, and thus better the quality of reads before the mapping.

Sickle

Sickle is a trimming tool which better the quality of NGS reads.

Sickle uses sliding windows computing sequencing quality along the reads. When the quality falls below a chose q-value threshold, the reads is cut. If the size of the remaining read is too short, it is completely removed. Sickle takes into account three different types of read quality: Illumina, Solexa, Sanger.

- Pre-requisite: install `zlib` (*link to section*).
- Clone the git repository into your bin (*link to section*) and run `make`.

```
cd $HOME/app_sources
git clone https://github.com/najoshi/sickle.git
cd sickle
make
cp sickle $HOME/bin
```

Alignment/mapping

Le but de l'alignement est de replacer les *reads* issus du séquençage à leur emplacement sur un génome de référence. Lorsque le *read* est suffisamment long, il peut généralement être *mappé* sur le génome avec une bonne certitude, en tolérant une certain quantité de *mismatches*, c'est-à-dire de nucléotides mal appariés. Néanmoins certaines séquences répétées du génome peuvent s'avérer plus difficiles à aligner. On désigne par l'expression "profondeur de séquençage" (ou *sequencing depth*) le nombre moyen de *reads* alignés par position sur le génome. Plus cette profondeur est importante, meilleure est la qualité de l'alignement, et plus les analyses ultérieures seront de qualité.

BWA

BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome.

- Manual
- Publication

Li H. and Durbin R. (2009). Fast and accurate short read alignment with Burrows-Wheeler Transform. Bioinformatics, 25:1754-60.

```
sudo apt-get install bwa
```

Bowtie

```
cd $HOME/app_sources
wget --no-clobber http://downloads.sourceforge.net/project/bowtie-bio/bowtie/
↪$(BOWTIE1_VER)/bowtie-$(BOWTIE1_VER)-linux-x86_64.zip
unzip bowtie-$(BOWTIE1_VER)-linux-x86_64.zip
cp `find bowtie-$(BOWTIE1_VER)/ -maxdepth 1 -executable -type f` $HOME/bin
```

Bowtie2

General documentation

Instructions

Downloads

```
cd $HOME/app_sources
wget http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.2.6/bowtie2-2.2.6-
↪linux-x86_64.zip
unzip bowtie2-2.2.6-linux-x86_64.zip
p `find bowtie2-$(BOWTIE2_VER)/ -maxdepth 1 -executable -type f` $HOME/bin
```

Peak-calling

bPeaks

Web page

Peak-caller developped specifically for yeast, can be useful in order to process small genomes only.

Available as an R library.

```
install.packages("bPeaks")
library(bPeaks)
```

HOMER

Web page

Install instructions

```
mkdir $HOME/app_sources/homer
cd $HOME/app_sources/homer
wget "http://homer.salk.edu/homer/configureHomer.pl"
perl configureHomer.pl -install homer
cp `find $HOME/app_sources/homer/bin -maxdepth 1 -executable -type f` $HOME/bin
```

The basic Homer installation does not contain any sequence data. To download sequences for use with HOMER, use the configureHomer.pl script. To get a list of available packages:

```
perl $HOME/bin/HOMER/configureHomer.pl -list
```

To install packages, simply use the -install option and the name(s) of the package(s).

```
perl  $HOME/bin/HOMER/configureHomer.pl -install mouse # (to download the mouse␣
↪promoter set)
perl  $HOME/bin/HOMER/configureHomer.pl -install mm8   # (to download the mm8 version␣
↪of the mouse genome)
perl  $HOME/bin/HOMER/configureHomer.pl -install hg19  # (to download the hg19␣
↪version of the human genome)
```

Supported organisms:

Organism	Assembly
Human	hg17, hg18, hg19
Mouse	mm8, mm9, mm10
Rat	rn4, rn5
Frog	xenTro2, xenTro3
Zebrafish	danRer7
Drosophila	dm3
3. elegans	ce6, ce10
19. cerevisiae	sacCer2, sacCer3
19. pombe	ASM294v1
Arabidopsis	tair10
Rice	msu6

HOMER can also work with custom genomes in FASTA format and gene annotations in GTF format.

MACS 1.4

- Documentation
- Installation manual

```
cd $HOME/app_sources
wget "https://github.com/downloads/taoliu/MACS/MACS-1.4.3.tar.gz"
tar -xvzf MACS-1.4.3.tar.gz
cd MACS-1.4.3
sudo python setup.py install
macs14 --version
```

MACS2

- Webpage

```
sudo apt-get install python-numpy
sudo pip install MACS2
```

SPP R package

This one might be a little but tricky (euphemism).

Several possibilities, none of which have I had the courage to retry lately.

- In R

```
source("http://bioconductor.org/biocLite.R")
biocLite("spp")
install.packages("caTools")
install.packages("spp")
```

- In commandline

```
apt-get install libboost-all-dev
cd $HOME/app_sources
wget -nc http://compbio.med.harvard.edu/Supplements/ChIP-seq/spp_1.11.tar.gz
sudo R CMD INSTALL spp_1.11.tar.gz
```

- Using git (I haven't tried this one but it looks more recent) (see github page)

```
require(devtools)
devtools::install_github('hms-dbmi/spp', build_vignettes = FALSE)
```

I also wrote a little protocol a while ago. Here's the procedure on Ubuntu 14.04, in this very order:

In unix shell:

```
# unix libraries
apt-get update
apt-get -y install r-base
apt-get -y install libboost-dev zlibc zlib1g-dev
```

In R shell:

```
# Rsamtools
source("http://bioconductor.org/biocLite.R")
biocLite("Rsamtools")
```

In unix shell:

```
# spp
wget http://compbio.med.harvard.edu/Supplements/ChIP-seq/spp_1.11.tar.gz
sudo R CMD INSTALL spp_1.11.tar.gz
```

A few links:

- Download page can be found here, better chose version `1.11`.
- SPP requires the Bioconductor library Rsamtools to be installed beforehand.
- Unix packages `gcc` and `libboost` (or equivalents) must be installed.
- You can find a few more notes here.
- Good luck!

SWEMBL

- SWEMBL beginner's manual

```
cd $HOME/app_sources
wget "http://www.ebi.ac.uk/~swilder/SWEMBL/SWEMBL.3.3.1.tar.bz2"
bunzip2 -f SWEMBL.3.3.1.tar.bz2
tar xvf SWEMBL.3.3.1.tar
rm SWEMBL.3.3.1.tar
chown -R ubuntu-user SWEMBL.3.3.1
cd SWEMBL.3.3.1
make
```

It seems there could be issues with C flags. To be investigated.

Motif discovery, motif analysis

Regulatory Sequence Analysis Tools (RSAT)

see dedicated section

Link

to translate

Suite logicielle spécialisée pour l'analyse de motifs cis-régulateurs, développée par les équipes de Morgane Thomas-Chollier (ENS, Paris) et Jacques van Helden (TAGC, Marseille). Inclut des outils spécifiques pour l'analyse de données de ChIP-seq.

MEME

Link

to translate

Suite logicielle spécialisée pour l'analyse de motifs cis-régulateurs, développée par l'équipe de Tim Bailey. Inclut des outils spécifiques pour l'analyse de données de ChIP-seq.

Miscellaneous

SRA toolkit

This toolkit includes a number of programs, allowing the conversion of `*.sra` files. `fastq-dump` translates `*.sra` files to `*.fastq` files.

- SRA format
- fastq-dump manual
- Installation manual

You can download last version here, or issue the following commands:

```
cd $HOME/app_sources
wget -nc http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/2.5.2/sratoolkit.2.5.2-ubuntu64.
↪tar.gz
tar xzf sratoolkit.2.5.2-ubuntu64.tar.gz
cp `find sratoolkit.2.5.2-ubuntu64/bin -maxdepth 1 -executable -type l` $HOME/bin
```

You can also install SRA toolkit simply by issuing this command, but likely it won't be the most recent release:

```
sudo apt-get install sra-toolkit
```

```
fastq-dump --version
  fastq-dump : 2.1.7
```

Samtools

SAM (Sequence Alignment/Map) format is a generic format for storing large nucleotide sequence alignments.

SAMtools provides several tools to process such files.

```
cd $HOME/app_sources
wget --no-clobber http://sourceforge.net/projects/samtools/files/samtools/1.3/
↪samtools-1.3.tar.bz2
bunzip2 -f samtools-1.3.tar.bz2
tar xvf samtools-1.3.tar
cd samtools-1.3
make
sudo make install
```

Bedtools

The bedtools utilities are a swiss-army knife of tools for a wide-range of genomics analysis tasks. For example, bedtools allows one to intersect, merge, count, complement, and shuffle genomic intervals from multiple files in widely-used genomic file formats such as BAM, BED, GFF/GTF, VCF.

```
sudo apt-get install bedtools
```

or get the latest version:

```
cd $HOME/app_sources
wget --no-clobber https://github.com/arq5x/bedtools2/releases/download/v2.24.0/
↪bedtools-2.24.0.tar.gz
tar xvfz bedtools-2.24.0.tar.gz
cd bedtools2
make
sudo make install
```

Bedops

```
cd $HOME/app_sources
wget -nc https://github.com/bedops/bedops/releases/download/v2.4.19/bedops_linux_x86_
↪64-v2.4.19.tar.bz2
tar jxvf bedops_linux_x86_64-v2.4.19.tar.bz2
mkdir bedops
mv bin bedops
cp bedops/bin/* $HOME/bin
```

Deeptools

```
cd $HOME/app_sources
git clone https://github.com/fidelram/deepTools
cd deepTools
python setup.py install
```

Picard tools

todo

Other

- MICSA: peak-calling & motifs discovery (publication).
- ChIPMunk: deep and wide digging for binding motifs in ChIP-Seq data (publication).
- HMCan: a method for detecting chromatin modifications in cancer samples using ChIP-seq data (publication).
- seqMINER
- Crunch project
- CSDeconv
- ...

Makefile

Has to be revised

The Gene-regulation library comprises a makefile that can install most of the dependencies described in the previous section.

It currently allows running the following workflows:

- import_from_sra.wf
- quality_control.wf
- ChIP-seq.wf

```
cd $GENE_REG_PATH
make -f gene-regulation/scripts/makefiles/install_tools_and_libs.mk all
source ~/.bashrc
```

Conda

This section has to be written

```
conda install -c bioconda sickle=0.5
conda install -c bioconda bowtie=1.2.0
conda install -c bioconda bowtie2=2.3.0
conda install -c bioconda subread=1.5.0.post3
conda install -c bioconda tophat=2.1.1
conda install -c bioconda bwa=0.7.15
conda install -c bioconda fastqc=0.11.5
conda install -c bioconda macs2=2.1.1.20160309
conda install -c bioconda homer=4.8.3
conda install -c bioconda bedtools=2.26.0
conda install -c bioconda samtools=1.3.1
conda install -c bioconda bamtools=2.4.0
```

Virtual environments

How to run Gene-regulation workflows in virtual environments: tutorials.

These protocols were developed on a Unix computer, with the OS LMDE and a 64-bit architecture. The virtual machines are developed with the Ubuntu 14.04 OS.

Based on gene-regulation v2.0

has to be refreshed

VirtualBox

Creating a virtual machine (VM)

Creating a VM under VirtualBox software

Requirements

Virtualbox software

We used VirtualBox 5.0.2, downloadable from https://www.virtualbox.org/ or to be installed manually:

```
sudo apt-get install virtualbox-5.0
```

VirtualBox extension pack can be requested (eg. for handling USB2.0, see 'errors' section).

```
wget http://download.virtualbox.org/virtualbox/5.0.2/Oracle_VM_VirtualBox_Extension_
↪Pack-5.0.2.vbox-extpack
```

Ubuntu image

In this tutorial we used Ubuntu 14.04.4, latest long-term supported version.

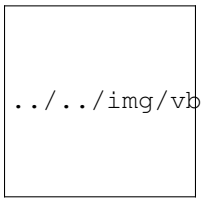
```
wget http://releases.ubuntu.com/14.04/ubuntu-14.04.4-desktop-amd64.iso
```

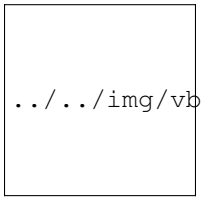
Virtual Box configuration

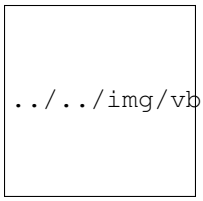
Before configuring the virtual machine, we need to tell VirtualBox how it will enable your local virtual machines to interact with their host (the operating system of the machine on which the VM is running).

1. Open *VirtualBox* > *File* > *Preferences...*
2. Open the tab *Network* > *Host-only Networks*
    - click on the "+" icon
    - this creates a network vboxnet0. Select this network, click on the screw driver icon (*edit host-only network*), and set the following options:
    - *Adapter* tab
        - IPv4 Address: 192.168.56.1
        - IPv4 Network Mask: 255.255.255.0
        - IPv6 Adress: blank
        - IPv6 Network Mask Length: 0
    - *DHCP Server* tab
        - Check *Enable Server*

  - *Server Address:* 192.168.56.100
  - *Server Mask:* 255.255.255.0
  - *Lower Address Bound:* 192.168.56.101
  - *Upper Address Bound:* 192.168.56.254







Creation of the virtual machine

1. Open VirtualBox
2. Click on the **New** button.
3. Parameters

- Name and operating system
  - Name: gene-regulation
  - Type: Linux
  - Version: Ubuntu (64 bits)
- Memory size: 2048 Mb (this can be modified afterwards).
- Hard drive: *Create a virtual hard drive now*.
- Hard drive file type: *VDI* (VirtualBox Disk Image).
- Storage on physical hard drive
  - Select *Dynamically allocated*
- File location and size
  - max size of virtual hard drive: 30GB
  - click on **Create** button

Note: you should adapt the virtual hard drive size to your needs. Be aware that it's difficult to extend later on, so you should aim larger than expected. Since the size is dynamically allocated, it won't take up too much space until you fill it.

At this stage, the VM has been created and needs to be configured before installing the operating system.

VM configuration

In the VirtualBox main window, select the newly created virtual machine, and click on the **Settings** button.

General

For the desktop version of Ubuntu, it is convenient to enable copy-paste between the guest and the host.

- Select the tab *Advanced*
- Set *Shared clipboard* to *Bidirectional*

Storage

Click on the **Empty** disc icon in the storage tree. Select the disc icon on the right and fetch the downloaded `.iso` image(see **Requirements**). Click on *OK*.

Network

VirtualBox offers many alternative ways to configure network communications between the virtual machine, the host machine, and the external network.

To get more information about network settings:

- VirtualBox manual page
- An excellent tutorial

We present here one possible way to configure your Virtual machine, but this should be adapted to the particular security/flexibility requirements of the network where the maching has to run.

In the VM settings, select tne *Network* tab. VirtualBox enables you to specify several adapters, each corresponding to one separate network access (e.g. using an ethernet card + wi-fi connection).

- click on the tab *Adapter 1*,
  - check *Enable Network Adapter*
  - Attached to: *Host-only Adapter*
  - Name: *vboxnet0* (this network must have been created beforehand, see section 1.2.3)
- click on the tab *Adapter 2*,
  - check *Enable Network Adapter*
  - Attached to : *NAT*
- click on the tab *Adapter 3*,
  - check *Enable Network Adapter*
  - Attached to : *Bridged Adapter*
  - Name: choose an option corresponding to the actual internet connection of the host machine (e.g. ethernet cable, Wi-Fi, ...).

**You can now start the VM. **

Operating system installation

- Welcome
  - check the language settings and click on *Install Ubuntu*.
- Preparing to install Ubuntu
  - leave all default parameters and click *Continue*.
- Installation type
  - (leave the default) Erase disk and install Ubuntu, click *Install Now*.
- Where are you (automatic)
  - Paris
- Keyboard layout
  - French - French
- Who are you ?
  - Your name: gene-regulation
  - Your computer's name: gene-regulation-virtual
  - Pick a username: gr
  - Choose a password: genereg
  - (Activate the option Log in automatically)

Restart once installation is completed.

Once on the desktop, go to the VM menu: select *Devices* then *Install Guest Additions CD image*. Run it.

The VirtualBox Guest Additions will provide closer integration between host and guest and improve the interactive performance of guest systems. Reboot again to see the new display.

Installing programs and dependencies

Once in the virtual machine, you can install the required programs from a terminal.

Get the `gene-regulation` repository

```
cd
wget --no-clobber https://github.com/rioualen/gene-regulation/archive/2.0.tar.gz -P
tar zvxf 2.0.tar.gz
```

```
cd
git clone https://github.com/rioualen/gene-regulation.git
```

Run makefile to install all required dependencies

This may take a while (30mn to 1h) & source the `.bashrc` (it's been updated with the `$PATH` for newly installed applications).

```
cd
#make -f gene-regulation-2.0/scripts/makefiles/install_tools_and_libs.mk all
make -f gene-regulation/scripts/makefiles/install_tools_and_libs.mk all
source ~/.bashrc
```

Executing snakemake workflow example

```
## Create a base directory for the analysis

export ANALYSIS_DIR="$HOME/ChIP-seq_SE_GSM20870"
mkdir $ANALYSIS_DIR
```

```
## Download source data

mkdir -p ${ANALYSIS_DIR}/data/GSM521934 ${ANALYSIS_DIR}/data/GSM521935
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX021%2FSRX021358/SRR051929/SRR051929.sra -P ${ANALYSIS_DIR}/data/GSM521934
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX021%2FSRX021359/SRR051930/SRR051930.sra -P ${ANALYSIS_DIR}/data/GSM521935
```

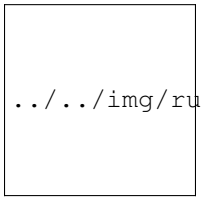
```
## Download reference genome & annotations

wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/fasta/saccharomyces_
↪cerevisiae/dna/Saccharomyces_cerevisiae.R64-1-1.30.dna.genome.fa.gz -P ${ANALYSIS_
↪DIR}/genome
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/gff3/saccharomyces_
↪cerevisiae/Saccharomyces_cerevisiae.R64-1-1.30.gff3.gz -P ${ANALYSIS_DIR}/genome
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/gtf/saccharomyces_
↪cerevisiae/Saccharomyces_cerevisiae.R64-1-1.30.gtf.gz -P ${ANALYSIS_DIR}/genome
gunzip ${ANALYSIS_DIR}/genome/*.gz
```

```
## Execute workflow

cd ${ANALYSIS_DIR}
ln -s  $HOME/gene-regulation
snakemake -p -s gene-regulation/scripts/snakefiles/workflows/ChIP-seq_workflow_SE.py -
↪-configfile gene-regulation/examples/ChIP-seq_SE_GSE20870/config.yml
```

Congratulations! You just executed this wonderful workflow:



Visualizing results

FastQC

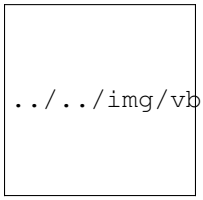
You can visualize the FastQC results using firefox or any other navigator. Fetch the `html` files located in the sample directories.

- Before trimming:

```
firefox ~/GSE20870-analysis/results/samples/GSM521934/GSM521934_fastqc/GSM521934_
↪fastqc.html
firefox ~/GSE20870-analysis/results/samples/GSM521935/GSM521935_fastqc/GSM521935_
↪fastqc.html
```

- After trimming:

```
firefox ~/GSE20870-analysis/results/samples/GSM521934/GSM521934_sickle-se-q20_
↪fastqc/GSM521934_sickle-se-q20_fastqc.html
firefox ~/GSE20870-analysis/results/samples/GSM521935/GSM521935_sickle-se-q20_
↪fastqc/GSM521935_sickle-se-q20_fastqc.html
```

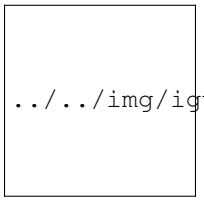


IGV

You can visualize the peaks by running IGV from the terminal.

```
igv
```

- Click "File" > "Open session..." and chose the file `~/GSE20870-analysis/results/peaks/igv_session.xml`.
- You may need to adjust the panel sizes.



Export appliance (todo)

The virtual machine created with VirtualBox can be exported and saved as an appliance.

- Shut down the VM.
- In VirtualBox, open *File -> Export Appliance ...*
- Select the VM `gene-regulation`
- *Next >*

- Save as: gene-regulation-[YYMMDD].ova
- Format: OVF 1.0
- Write Manifest File: check
- *Next* >
- Appliance Settings
  - Name: gene-regulation-[YYMMDD]
  - Product: Regulatory Genomics Pipeline
  - Product-URL: -
  - Vendor: Claire Rioualen, Jacques van Helden
  - Version: YYYY-MM-DD
  - Description: Regulatory Genomics Pipeline using Snakemake, installed on an Ubuntu 14.04 Virtual Machine.
  - License: Free of use for academic users, non-commercial and non-military usage.
- *Export*

The appliance saved can be re-imported later on, on another computer if needed.

Import appliance (todo)

In VirtualBox, click menu File > Import appliance > fetch OVA file.

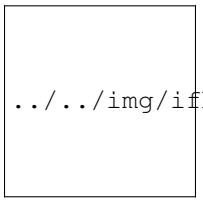
Note: there is apparently a bug with the export of VMs under VirtualBox 5.0. If you get this error when launching the imported file:

> A new node couldn't be inserted because one with the same name exists. (VERR_CFGM_NODE_EXISTS).

There is a workaround: go to the imported VM settings, to the USB tab, and untick "enable USB Controller". You should now be able to start the VM.

IFB cloud

IFB cloud utilities



The French Bioinformatics Institute (IFB) cloud provides users with a number of bioinformatics facilities, under the form of ready-to-use *appliances*. A cloud appliance is a template or a virtual machine (VM) built with a bundle of scientific or utility software components that are already configured. Several appliances are dedicated to special fields of bioinformatics, such as proteomics, genomics... Some of them come with an HTML interface, such as Galaxy or RSAT.

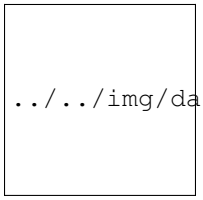
The cloud also provides "basic" Ubuntu or CentOS appliances. Provided you hold a developper account, it allows you to instantiate a virtual machine, setup your own tools, and register it as a new appliance to be used again later on and even shared with other cloud users.

The official website is still under development. However, here are a few useful links:

- The IFB
- IFB cloud
- Cloud usage
- Documentation

User account creation & configuration

- Using the IFB cloud facilities requires to have a user account. Register here.
- Once your account has been validated, you can login.
- In order to be able to access your instances through SSH, you should register your SSH public key in your account settings, through the dashboard.

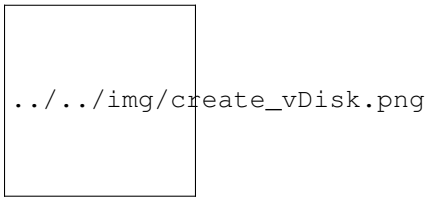


Using an existing appliance

Virtual disk creation

Appliances usually have a limited amount of disk space (up to 10, 20Go). If the instance to be run necessitates disk space, you have to create a virtual disk (vDisk) prior to launching it. By default, the capacity of storage granted to a user is 250Go, which can be divided into as many vDisks as necessary. When instantiating an appliance, you can chose to attach one of these vDisks to the virtual machine. You'll be able to access data on this disk through SSH.

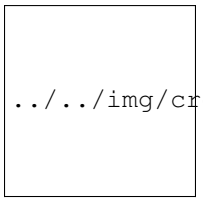
1. Click *New vDisk* button.
2. Enter a size (whole number equating to the amount of Go needed).
3. Name it.



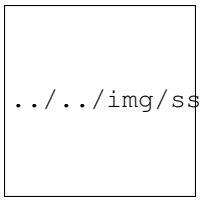
Creation of an instance

1. Click *New Instance* button.

2. Choose an appliance in the drop-down menu. You may use the filter menu in order to look for a specific tool.
3. Name your VM.
4. Choose the amount of CPU and RAM to grant the VM (up to 8 CPU, 32 GB RAM).
5. Attach the vDisk.
6. Click *Run*.



7. After a few seconds, you may refresh the page until the newly created instance shows up on the dashboard. Clicking on the ssh mention in the *Access* column will give you the commands to access your virtual machine.



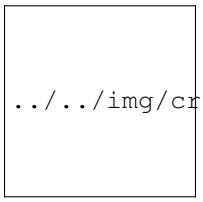
8. If the appliance has an HTTP interface, a link will also be provided in the *Access* column.

Creation of an appliance

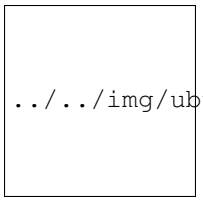
Creation

Creating your own appliance can be as simple as instantiating an existing one.

1. Click *New Instance* button.
2. Choose the appliance **Ubuntu 14.04 IFB-10G (2015-10)** or **CentOS 6.7 IFB-20G (2016-01)**.
3. Name your instance.
4. Check **Create appliance**.
5. Choose the amount of CPU and RAM to grant the VM (up to 8 CPU, 32 GB RAM).
6. Attach the vDisk.
7. Click *Run*.



8. Refresh the page. Your instance should appear in orange because of the creation mode you selected. You can now click on the **ssh** column to see the ssh parameters. It should look like this:



9. Connect to your VM by commandline.

```
ssh -A -p 22 root@192.54.201.111
```

Configuration (optional)

User account

Create user account and grant it sudo privileges (followed procedure here).

Shell coloring

```
nano ~/.bashrc
```

Fetch following paragraph and uncomment command `force-color`.

```
# uncomment for a colored prompt, if the terminal has the capability; turned
# off by default to not distract the user: the focus in a terminal window
# should be on the output of commands, not on the prompt
force_color_prompt=yes
```

```
source ~/.bashrc
```

Data management

Virtual disk

By default, if a vDisk has been attached to the VM, it is mounted under `/root/mydisk`.

Transfers

You can transfer data from your local computer to the VM using commands provided under *Access* > ssh:

```
scp -P 22 ${localfile} root@192.54.201.111:
sftp -oPort=22 root@192.54.201.111
```

Another way is to use rsync:

```
rsync -ruptvl ${localfile} root@192.54.201.177:/root/mydisk/
```

Then...

Software installation

Once you're connected to the VM through `ssh`, you can install any program just the way you would do it locally (see tutorials in this directory for instance).

Using the Gene-regulation appliance

Requirements

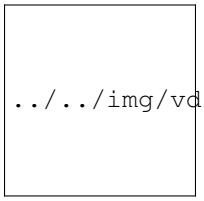
User account creation & configuration

- Using the IFB cloud facilities requires to have a user account. Register here.
- Once your account has been validated, you can login.
- In order to be able to access your instances through SSH, you should register your SSH public key in your account settings, through the dashboard.

Virtual disk creation

Appliances usually have a limited amount of disk space (up to 10, 20Go). If the instance to be run necessitates disk space, you have to create a virtual disk (vDisk) prior to launching it. By default, the capacity of storage granted to a user is 250Go, which can be divided into as many vDisks as necessary. When instantiating an appliance, you can chose to attach one of these vDIsks to the virtual machine. You'll be able to access data on this disk through SSH.

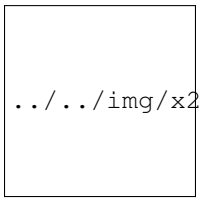
1. Click *New vDisk* button.
2. Enter a size (whole number equating to the amount of Go needed).
3. Name it (e.g. `GSE20870-10Gb`, the ID of the Gene Expression Omnibus series that will be stored on the virtual drive).



Creation of an instance

1. Click *New Instance* button.
2. Choose appliance "Gene regulation 2.0" in the drop-down menu.
3. Name your VM.
4. Choose the amount of CPU and RAM to grant the VM (up to 8 CPU, 32 GB RAM).
5. Attach the vDisk.
6. Click *Run*.

7. After a few seconds, you may refresh the page until the newly created instance shows up on the dashboard. Clicking on the ssh mention in the *Access* column will give you the commands to access your virtual machine.



Connection to the device

Open a terminal on your host computer and type in:

```
ssh -A -p 22 root@192.54.201.124
```

Download source data

On the IFB cloud VM, the vDisk is automatically attached and mounted by default under `/root/mydisk`, or `~/mydisk`.

Here we create a folder to store the source data files and download them .

```
ANALYSIS_DIR=${HOME}/mydisk/GSE20870-analysis
```

```
mkdir -p ${ANALYSIS_DIR}
cd ${ANALYSIS_DIR}
ln -s ${HOME}/gene-regulation-2.0 gene-regulation
```

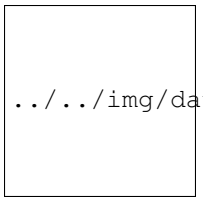
Download data

```
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX021%2FSRX021358/SRR051929/SRR051929.sra -P ${ANALYSIS_DIR}/data/GSM521934
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX
↪%2FSRX021%2FSRX021359/SRR051930/SRR051930.sra -P ${ANALYSIS_DIR}/data/GSM521935
```

Download reference genome & annotations

```
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/fasta/saccharomyces_
↪cerevisiae/dna/Saccharomyces_cerevisiae.R64-1-1.30.dna.genome.fa.gz -P ${ANALYSIS_
↪DIR}/genome
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/gff3/saccharomyces_
↪cerevisiae/Saccharomyces_cerevisiae.R64-1-1.30.gff3.gz -P ${ANALYSIS_DIR}/genome
wget -nc ftp://ftp.ensemblgenomes.org/pub/fungi/release-30/gtf/saccharomyces_
↪cerevisiae/Saccharomyces_cerevisiae.R64-1-1.30.gtf.gz -P ${ANALYSIS_DIR}/genome
gunzip ${ANALYSIS_DIR}/genome/*.gz
```

You should now have something like this:



Run the workflow

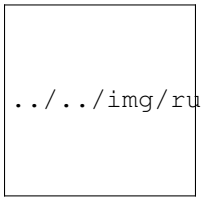
You can use the option `-n` to make a dry run.

```
cd  ${ANALYSIS_DIR}
snakemake -p -s gene-regulation/scripts/snakefiles/workflows/factor_workflow.py --
↪configfile gene-regulation/examples/GSE20870/GSE20870.yml -n
```

```
snakemake -p -s gene-regulation/scripts/snakefiles/workflows/factor_workflow.py --
↪configfile gene-regulation/examples/GSE20870/GSE20870.yml
```

Using 4CPU & 8Go of RAM, the workflow took about 12mn to complete.

Congratulations! You just executed this wonderful workflow:



Visualizing results

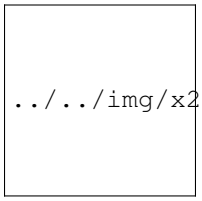
Install and run the X2Go client on your host computer

The Virtual Machine created on the IFB cloud doesn't have a graphical interface, but it contains the X2GO software. We're gonna use it to create a distant desktop to visualize the results from the host machine.

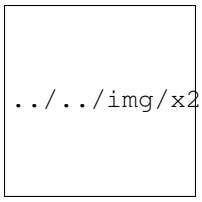
1. Install the x2go client and launch it from your local computer.

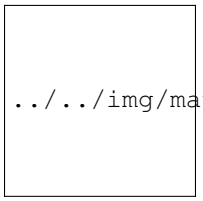
```
sudo apt-get install x2goclient
x2goclient
```

2. Create a new session using the Mate desktop.



3. The session now appears on the right panel. Just click it to lauch it!
4. You should be now on the virtual desktop!



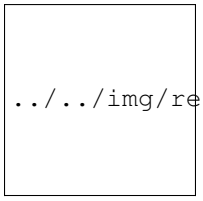


Note: you may need to change your keyboard settings

- Go to **System > Preferences > Keybords**
- Click on tab **Layouts**
- Add and/or remove desired keyboards

Visualize results

The result files should be organized like this:



FastQC

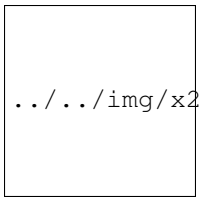
You can visualize the FastQC results using firefox or any other navigator. Fetch the `html` files located in the sample directories.

- Before trimming:

```
firefox /root/mydisk/GSE20870-analysis/results/samples/GSM521934/GSM521934_fastqc/
↪GSM521934_fastqc.html
firefox /root/mydisk/GSE20870-analysis/results/samples/GSM521935/GSM521935_fastqc/
↪GSM521935_fastqc.html
```

- After trimming:

```
firefox /root/mydisk/GSE20870-analysis/results/samples/GSM521934/GSM521934_sickle-
↪se-q20_fastqc/GSM521934_sickle-se-q20_fastqc.html
firefox /root/mydisk/GSE20870-analysis/results/samples/GSM521935/GSM521935_sickle-
↪se-q20_fastqc/GSM521935_sickle-se-q20_fastqc.html
```

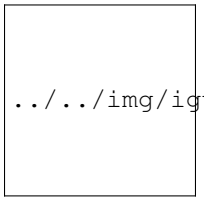


IGV

You can visualize the peaks by running IGV from the terminal.

```
igv
```

- Click "File" > "Open session..." and chose the file `/root/mydisk/GSE20870-analysis/reports/peaks/igv_session.xml`.
- You may need to adjust the panel sizes.



Create your own Gene-regulation appliance

Creating a new appliance from scratch is very similar to using one. You have to satisfy the requirements described in part 1.1.

If you want to manipulate data, you should also create a vDisk following step 1.2.

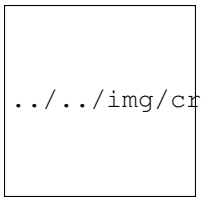
Creation of an instance

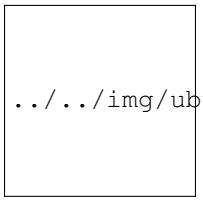
When creating a new instance choose a 10 Go Ubuntu appliance and check the "Create appliance" option:

1. Click *New Instance* button.
2. **Choose appliance "Ubuntu 14.04 IFB-X2GO-10GB" in the drop-down menu.**
3. Name your VM.
4. Choose the amount of CPU and RAM to grant the VM (up to 8 CPU, 32 GB RAM).
5. **Check the box *Create appliance*.**
6. Attach the vDisk.
7. Click *Run*.

The new instance should appear in orange bold fonts in the dashboard.

You can connect to the instance through `ssh` as shown in part 1.4.





Installing programs and dependencies

Once in the virtual machine, you can install the required programs.

Get the `gene-regulation` repository

```
wget https://github.com/rioualen/gene-regulation/archive/2.0.tar.gz
tar zvxf 2.0.tar.gz
```

Run makefile to install the dependencies

This may take a while (up to 30mn-1h) & source the `.bashrc` in order to update the `$PATH` accordingly.

```
make -f gene-regulation-2.0/scripts/makefiles/install_tools_and_libs.mk all
source ~/.bashrc
```

If you want to install the x2go server on the VM for visualization purposes:

```
make -f gene-regulation-2.0/scripts/makefiles/install_tools_and_libs.mk desktop_and_
↪x2go
```

You should now be able to execute the example workflow by following steps 1.5 and 1.6.

In order for your appliance to remain persistant and be available to other users on the IFB cloud, you should contact an admin: @?

Docker

Get started with Docker!

Create a Docker account

Instructions here.

Install Docker on your local host

Instructions for a linux install can be found here, along with mac and windows instructions. A useful script is available here for a debian install.

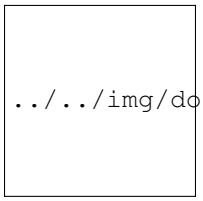
You can also install it on Ubuntu 14.04 (64bits) typing the following:

```
#sudo apt-get update
sudo apt-get -y install docker.io
sudo usermod -aG docker <username>
```

You should now log out and in again from your Ubuntu session. This short procedure was tested in a virtual machine under VirtualBox (see corresponding tutorial).

You can test whether docker works properly:

```
docker run hello-world
```



NB: it seems qwerty keyboard keeps popping up after docker install. Switch back to azerty:

```
setxkbmap fr
```

<!– Run the following command:

```
sudo apt-get --yes install docker
```

–>

Create shared repositories and download source data

In order to execute the study case GSE20870, you should enter the following commands:

```
export ANALYSIS_DIR=~/GSE20870-analysis
mkdir $ANALYSIS_DIR
cd $ANALYSIS_DIR
```

```
mkdir data/GSM521934
wget -nc ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX%2FSRX021
↪%2FSRX021358/SRR051929/SRR051929.sra -P data/GSM521934

mkdir data/GSM521935
wget -nc ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX%2FSRX021
↪%2FSRX021359/SRR051930/SRR051930.sra -P data/GSM521935
```

Fetch the Docker image and run it with shared folders

```
docker pull rioualen/gene-regulation:2.0
docker run -v $ANALYSIS_DIR:~/GSE20870-analysis -it rioualen/gene-regulation:2.0 /bin/
↪bash
```

You can share as many folders as desired, using this syntax: `-v /path/on/host/:/path/on/docker/`.

Execute the pipeline

```
snakemake -p -s gene-regulation/scripts/snakefiles/workflows/factor_workflow.py --
↪configfile gene-regulation/examples/GSE20870/GSE20870.yml
```

<!– # JVH / Mac

Quick tour

On Mac OSX

1. Install docker

```
https://docs.docker.com/engine/installation/mac/
```

2. Open the application Docker Quickstart Terminal. This open a new terminal window and launches the docker daemon.
3. Get the gene-regulation docker

docker pull rioualen/gene-regulation:0.3

4. Check the list of docker images available locally

docker images

5. Start the gene-regulation image. The option `-it` specifies the interactive mode, which is necessary to be able using this VM

```
docker run -it rioualen/gene-regulation:0.3 /bin/bash
```

You are now in a bash session of a gene-regulation docker. In this session, you are "root" user, i;e. you have all the administration rights. You can check this easily:

```
whoami
```

6. Check the disks available on this docker

```
df -h
```

Currently, your docker can only access its local disk, which comes with the VM. **Beware**: any data stored on this local disk will be lost when you shut down the gene-regulation docker.

7. Exit and get back to your gene-regulation container

If you exits your shell session, the docker will still be running.

```
exit
```

You are now back to the host terminal.

Check the currently active docker containers (processes).

```
docker ps -a
```

Note that you can run several containers of the same image. Each active container has a unique identifier which appears in the first column when you run `docker ps` (e.g. `faff5298ef95`). You can re-open a running container with the command

```
docker attach [CONTAINER_ID]
```

where `[CONTAINER_IDR]` must be replaced by the actual ID of the running docker container (e.g. `faff5298ef95`).

8. Shutting down the container

We will now shut down this image, and start a new one which will enable you to store persistent data.

```
docker stop [CONTAINER_ID]
```

9. Starting a docker container with a shared folder.

500 docker pull rioualen/gene-regulation:0.3 501 mkdir -p ~/gene-regulation_data/GSE20870/GSM521934 ~/gene-regulation_data/GSE20870/GSM521935 502 cd ~/gene-regulation_data/GSE20870/GSM521934 503 wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX%2FSRX021%2FSRX021358/SRR051929/SRR051929.sra 504 cd ~/gene-regulation_data/GSE20870/GSM521935 505 wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/SRX%2FSRX021%2FSRX021359/SRR051930/SRR051930.sra 506 mkdir ~/gene-regulation_data/results/GSE20870 507 mkdir -p ~/gene-regulation_data/results/GSE20870 508 docker pull rioualen/gene-regulation:0.3 509 docker run -v ~/gene-regulation_data:/data -it rioualen/gene-regulation:0.3 /bin/bash

10. Running the snakemake demo workflow on the docker container

```
ls /data
ls /data/GSE20870/
ls /data/GSE20870/GSM521934/
exit
ls /data
source ~/bin/ngs_bashrc
snakemake -s scripts/snakefiles/workflows/factor_workflow.py -np
history
snakemake -s scripts/snakefiles/workflows/factor_workflow.py -np
```

Questions

1. Quand on fait un login dans la vm gene–regulation, on entre dans un shell basique (pas bash). Est-il possible de configurer docker pour qu'on entre automatiquement en bash ?

Entry point /bin/bash

2. Il faut ajouter le bashrc dans le /etc du docker.

Conda

TODO

Snakemake

This tutorial was developed assuming a unix-like architecture (Ubuntu 14.04).

Introduction

Snakemake concepts

- Inspired by GNU Make: system of **rules** & **targets**
- A rule is the *recipe* for a target
- Rules are combined by *matching their inputs and outputs*

Installation

```
sudo apt-get -y install python3-pip
sudo pip3 install snakemake
```

Downloads for practical exercises

Ubuntu libraries

```
sudo apt-get -y install zlib1g-dev                          # samtools (1-6)
sudo apt-get -y install libncurses5-dev libncursesw5-dev    # samtools (1-6)

sudo apt-get -y install r-base-core                         # Rsamtools (4-6)
sudo pip3 install "rpy2<2.5.6"                              # Rsamtools (4-6)

sudo pip3 install pyyaml                                    # Config management (5-6)
```

Tuto material

```
wget https://github.com/rioualen/gene-regulation/archive/1.0.tar.gz
tar xvzf 1.0.tar.gz
cd gene-regulation-1.0/doc/snakemake_tutorial
```

Samtools

```
wget -nc http://sourceforge.net/projects/samtools/files/samtools/1.3/samtools-1.3.tar.
↪bz2
bunzip2 -f samtools-1.3.tar.bz2
tar xvf samtools-1.3.tar
cd samtools-1.3
make
sudo make install
cd gene-regulation-1.0/doc/snakemake_tutorial
```

Rsamtools

```
R
```

```
source("http://bioconductor.org/biocLite.R")
biocLite("Rsamtools")
quit()``
```

Demo workflows

Workflow 1: Rules and targets

- Only the first **rule** is executed by default
- Rule `all` defines the **target**
- Rule `sam_to_bam` automatically produces the target

```
# file: workflow1.py
rule all:
    input: "GSM521934.bam"

rule sam_to_bam:
    input: "GSM521934.sam"
    output: "GSM521934.bam"
    shell: "samtools view {input} > {output}"
```

In the terminal:

```
snakemake -s workflow1/workflow1.py
```

Workflow 2: Introducing wildcards

- **Wildcards** can replace variables
- Workflow applies to list of files or samples
- Use of the **expand** function

```
# file: workflow2.py
SAMPLES = ["GSM521934", "GSM521935"]

rule all:
    input: expand("{sample}.bam", sample = SAMPLES)

rule sam_to_bam:
    input: "{file}.sam"
    output: "{file}.bam"
    shell: "samtools view {input} > {output}"
```

In the terminal:

```
snakemake -s workflow2/workflow2.py
```

Workflow 3: Keywords

- Rules can use a variety of **keywords**
- An exhaustive list can be found here

```
# file: workflow3.py
SAMPLES = ["GSM521934", "GSM521935"]

rule all:
    input: expand("{sample}.bam", sample = SAMPLES)

rule sam_to_bam:
    input: "{file}.sam"
    output: "{file}.bam"
    params:
        threads = 2 log: "{file}.log"
    benchmark: "{file}.json"
    shell: "(samtools view -bS --threads {params.threads} {input} > {output}) > {log}"
```

In the terminal:

```
snakemake -s workflow3/workflow3.py
```

Workflow 4: Combining rules

- Dependencies are handled implicitly, by matching filenames
- Commands can be executed by keywords `run` or `shell`
- Several languages: `R`, `bash`, `python`

```
# file: workflow4.py
from snakemake.utils
import R

SAMPLES = ["GSM521934", "GSM521935"]

rule all:
    input: expand("{sample}_sorted.bam", sample = SAMPLES)

rule sam_to_bam:
    input: "{file}.sam"
    output: "{file}.bam"
    params:
        threads = 2 log: "{file}.log"
    benchmark: "{file}.json"
    shell: "(samtools view -bS --threads {params.threads} {input} > {output}) > {log}"

rule bam_sorted:
    input: "{file}.bam"
    output: "{file}\_sorted.bam"
    run:
        R("""
        library(Rsamtools)
        sortBam("{input}", "{output}")
        """)
```

In the terminal:

```
snakemake -s workflow4/workflow4.py
```

Workflow 5: Configuration file

- Can be in `json` or in `yml` format
- Acessible through the global variable **config**

```
# file: workflow5.py
from snakemake.utils
import R

configfile: "config.yml"

SAMPLES = config["samples"].split() OUTDIR = config["outdir"]

rule all:
    input: expand(OUTDIR + "{sample}_sorted.bam", sample = SAMPLES)

rule sam_to_bam:
    input: "{file}.sam"
    output: "{file}.bam"
    params:
        threads = config["samtools"]["threads"]
    log: "{file}.log"
    benchmark: "{file}.json"
    shell: "(samtools view -bS --threads {params.threads} {input} > {output}) > {log}"

rule bam_sorted:
    input: "{file}.bam"
    output: "{file}_sorted.bam"
    run:
        R("""
        library(Rsamtools)
        sortBam("{input}", "{output}")
        """)
```

```
# file: config.yml
samples: "GSM521934 GSM521935"
outdir: "gene-regulation-1.0/doc/snakemake_tutorial/results/"
samtools:
    threads: "2"
```

In the terminal:

```
snakemake -s workflow5/workflow5.py
```

Workflow 6: Separated files

- The keyword `include` is used to import rules

```
# file: workflow6.py

```

```
from snakemake.utils
import R

configfile: "config.yml"

SAMPLES = config["samples"].split()
OUTDIR = config["outdir"]

include: "sam_to_bam.rules"
include: "bam_sorted.rules"

rule all:
    input: expand(OUTDIR + "{sample}_sorted.bam", sample = SAMPLES)
```

```
# file: sam_to_bam.rules

rule sam_to_bam:
    input: "{file}.sam"
    output: "{file}.bam"
    params:
        threads = config["samtools"]["threads"]
    log: "{file}.log"
    benchmark: "{file}.json"
    shell: "(samtools view -bS --threads {params.threads} {input} > {output}) > {log}"
```

```
# file: bam_sorted.rules

rule bam_sorted:
    input: "{file}.bam"
    output: "{file}_sorted.bam"
    run:
        R("""
        library(Rsamtools)
        sortBam("{input}", "{output}")
        """)
```

In the terminal:

```
snakemake -s workflow6/workflow6.py
```

Workflow 7: The keyword Ruleorder todo

Workflow 8: Combining wildcards with zip

Workflow 9: Combining wildcards selectively

Workflow 10: Using regular expression in wildcards

Other

- temp()
- touch()
- target/all

Bonus: generating flowcharts

```
snakemake -s workflow6/workflow6.py --dag | dot -Tpng -o d.png
snakemake -s workflow6/workflow6.py --rulegraph | dot -Tpng -o r.png
```

include img

More on snakemake...

Documentation

- Manual
- FAQ
- Forum

Installation

```
apt-get install python3-pip
pip3 install snakemake
```

Reference

Köster, Johannes and Rahmann, Sven. "Snakemake - A scalable bioinformatics workflow engine". Bioinformatics 2012.

Other manuals (*to be updated*)

RSAT installation manual

NB: OLD, to be updated / hopefully replaced with apt-get

Unless stated otherwise, the following commands will be executed as root.

!!! Experimented users only !!!

First steps

Set the locales manually if needed.

```
export LANGUAGE=en_US.UTF-8
export LANG=en_US.UTF-8
export LC_ALL=en_US.UTF-8
locale-gen en_US.UTF-8
sudo dpkg-reconfigure locales
```

Check date & time, and adjust your timezone if needed.

```
date
```

```
dpkg-reconfigure tzdata
```

Create RSAT directory. It must be readable by all users (in particular by the apache user).

```
mkdir -p /packages
cd /packages
```

Check the installation device. This should give sda1 or vda1?

```
DEVICE=`df -h / | grep '/dev'| awk '{print $1}' | perl -pe 's/\/dev\///'`
echo ${DEVICE}
```

Packages installation

Update apt-get.

```
apt-get update
apt-get --quiet --assume-yes upgrade
```

Packages to be installed.

```
PACKAGES="
ssh
git
cvs
wget
zip
unzip
finger
screen
make
g++
apache2
php5
libapache2-mod-php5
libgd-tools
libgd-gd2-perl
ghostscript
gnuplot
graphviz
mysql-client
default-jre
python
python-pip
python-setuptools
python-numpy
python-scipy
python-matplotlib
python-suds
python3
python3-pip
python3-setuptools
python3-numpy
python3-scipy
python3-matplotlib
r-base
```

```
emacs
x11-apps
firefox
eog
ntp
curl
libcurl4-openssl-dev
"
```

Perl modules.

```
PACKAGES_PERL="perl-doc
pmtools
libyaml-perl
libemail-simple-perl
libemail-sender-perl
libemail-simple-creator-perl
libpostscript-simple-perl
libstatistics-distributions-perl
libio-all-perl
libobject-insideout-perl
libobject-insideout-perl
libsoap-lite-perl
libsoap-wsdl-perl
libxml-perl
libxml-simple-perl
libxml-compile-cache-perl
libdbi-perl
liblockfile-simple-perl
libobject-insideout-perl
libgd-perl
libdbd-mysql-perl
libjson-perl
libbio-perl-perl
libdigest-md5-file-perl
libnet-address-ip-local-perl
"
```

Install the apt-get libraries.

```
echo "Packages to be installed with apt-get --quiet --assume-yes"
echo "${PACKAGES}"
echo "Perl module packages to be installed with apt-get --quiet --assume-yes"
echo "${PACKAGES_PERL}"
for LIB in ${PACKAGES} ${PACKAGES_PERL}; \
do \
   echo "`date '+%Y/%m/%d %H:%M:%S'`  installing apt-get library ${LIB}" ; \
   sudo apt-get install --quiet --assume-yes ${LIB} ; \
done
```

Package to be installed in an interactive mode.

```
apt-get install --quiet --assume-yes console-data
```

- Options:
  - Select keymap from arch list
  - <Don't touch keymap> (default)

- Keep kernel keymap
- Select keymap from full list

Specific treatment for some Python libraries.

```
sudo apt-get --quiet --assume-yes build-dep python-numpy python-scipy
```

To free space, remove apt-get packages that are no longer required. /?\

```
apt-get --quiet --assume-yes  autoremove
apt-get --quiet --assume-yes  clean
```

Python libraries installation

```
pip install soappy
pip install fisher
pip install httplib2
```

Apache Web server configuration

/! Manual interventions needed here.

Activate CGI module.

```
nano /etc/apache2/sites-available/000-default.conf
```

Uncomment the following line: `Include conf-available/serve-cgi-bin.conf`.

To avoid puzzling warning at apache start, set ServerName globally.

```
nano /etc/apache2/apache2.conf
```

Add the following line at the end of the file: `ServerName localhost`.

Add CGI script.

```
nano /etc/apache2/mods-available/mime.conf
```

Uncomment the line `AddHandler cgi-script .cgi`.

Optional: associate a plain/text mime type to extensions for some classical bioinformatics files. `AddType text/plain .fasta` `AddType text/plain .bed`.

Adapt the PHP parameters.

```
nano /etc/php5/apache2/php.ini
```

Modify the following parameters: `post_max_size = 100M` and `upload_max_filesize=100M`.

Activate cgi scripts. Found here.

```
chmod 755 /usr/lib/cgi-bin
chown root.root /usr/lib/cgi-bin
a2enmod cgi
service apache2 restart
```

You can check whether apache server was successfully configured and started by opening a web connection to `http://{IP}`.

RSAT distribution

/!\ Note: The git distribution requires an account at the ENS git server, which is currently only possible for RSAT developing team. In the near future, we may use git also for the end-user distribution. For users who don't have an account on the RSAT git server, the code can be downloaded as a tar archive from the Web site.

Create RSAT directory.

```
mkdir -p /packages/rsat
cd /packages
export RSAT=/packages/rsat
```

Git repository cloning.

```
git clone git@depot.biologie.ens.fr:rsat
git config --global user.mail claire.rioualen@inserm.fr
git config --global user.name "reg-genomics VM user"
```

** OR **

Archive download.

```
export RSAT_DISTRIB=rsat_2016-11-06.tar.gz
export RSAT_DISTRIB_URL=http://pedagogix-tagc.univ-mrs.fr/download_rsat/${RSAT_
↪DISTRIB}
```

```
sudo wget ${RSAT_DISTRIB_URL}
sudo tar -xpzf ${RSAT_DISTRIB}
sudo rm -f ${RSAT_DISTRIB}
cd ~; ln -fs /packages/rsat rsat
```

RSAT configuration

Run the configuration script, to specify the environment variables.

```
cd $RSAT
sudo perl perl-scripts/configure_rsat.pl
```

Which options to specify?

Load the (updated) RSAT environment variables.

```
source RSAT_config.bashrc
```

Check that the RSAT environment variable has been properly configured.

```
echo ${RSAT}
```

Initialise RSAT folders

```
make -f makefiles/init_rsat.mk init
```

Perl modules for RSAT

```
cpan
```

```
cpan> install YAML
cpan> install CPAN
cpan> reload cpan
cpan> quit
```

Get the list of Perl modules to be installed.

```
make -f makefiles/install_rsat.mk  perl_modules_list
make -f makefiles/install_rsat.mk perl_modules_check
more check_perl_modules_eval.txt
grep Fail  check_perl_modules_eval.txt
grep -v '^OK'  check_perl_modules_eval.txt | grep -v '^;'
MISSING_PERL_MODULES=`grep -v '^OK'  check_perl_modules_eval.txt | grep -v '^;' | cut␣
↪-f 2 | xargs`
echo "Missing Perl modules:     ${MISSING_PERL_MODULES}"
```

Install the missing Perl modules.

```
make -f makefiles/install_rsat.mk perl_modules_install PERL_MODULES="${MISSING_PERL_
↪MODULES}"
```

Check once more if all required Perl modules have been correctly installed.

```
make -f makefiles/install_rsat.mk perl_modules_check
more check_perl_modules_eval.txt
```

Note: Object::InsideOut always displays “Fail”, whereas it is OK during installation.

Configure RSAT web server

```
cd ${RSAT}
sudo rsync -ruptvl RSAT_config.conf /etc/apache2/sites-enabled/rsat.conf
apache2ctl restart
```

RSAT Web server URL

```
echo $RSAT_WWW
```

If the value is “auto”, get the URL as follows:

```
export IP=`ifconfig eth0 | awk '/inet /{print $2}' | cut -f2 -d':'`
echo ${IP}
export RSAT_WWW=http://${IP}/rsat/
echo $RSAT_WWW
```

Other

compile RSAT programs written in C

```
make -f makefiles/init_rsat.mk compile_all
export INSTALL_ROOT_DIR=/packages/
```

Install some third-party programs required by some RSAT scripts.

```
make -f makefiles/install_software.mk install_ext_apps
```

Mkvtree licence / Vmatch

Get a licence here

Alternately, you can copy-paste from another RSAT device...

```
rsync -ruptvl /packages/rsat/bin/vmatch.lic root@<IP>:/packages/rsat/bin/
```

Data management

```
export RSAT_DATA_DIR=/root/mydisk/rsat_data
cd ${RSAT}/public_html
mv data/* ${RSAT_DATA_DIR}/
mv data/.htaccess ${RSAT_DATA_DIR}/
rmdir data
ln -s ${RSAT_DATA_DIR} data
cd $RSAT
```

Install model organisms, required for some of the Web tools.

```
download-organism -v 1 -org Saccharomyces_cerevisiae -org Escherichia_coli_K_12_
↪substr__MG1655_uid57779
download-organism -v 1 -org Drosophila_melanogaster
```

Get the list of organisms supported on your computer.

```
supported-organisms
```

Install selected R librairies

Packages required for some RSAT scripts.

```
cd $RSAT; make -f makefiles/install_rsat.mk install_r_packages
```

```
cd $RSAT; make -f makefiles/install_rsat.mk update ## install R packages + compile␣
↪the C programs
```

NB: second only if git repo

Testing RSAT & external programs

Test a simple Perl script that does not require for organisms to be installed.(OK)

```
which random-seq
random-seq -l 100
```

Test a simple python script that does not require organisms to be installed.(OK)

```
random-motif -l 10 -c 0.90
```

Test vmatch

```
random-seq -l 100 | purge-sequence
```

seqlogo

```
which seqlogo
seqlogo
```

weblogo 3

```
which weblogo
weblogo --help
```

ghostscript

```
which gs
gs --version
```

Check that the model genomes have been correctly installed

```
# Retrieve all the start codons and count oligonucleotide frequencies (most should be␣
↪ATG).
retrieve-seq -org Saccharomyces_cerevisiae -all -from 0 -to +2 | oligo-analysis -l 3 -
↪1str -return occ,freq -sort
```

Configure the SOAP/WSDL Web services

Check the URL of the web services (RSAT_WS). By default, the server addresses the WS requests to itself (http://localhost/rsat) because web services are used for multi-tierd architecture of some Web tools (retrieve-ensembl-seq, NeAT).

```
cd $RSAT
#echo $RSAT_WS
```

Get the current IP address

```
export IP=`/sbin/ifconfig eth0 | awk '/inet /{print $2}' | cut -f2 -d':'`
echo ${IP}
export  RSAT_WS=http://${IP}/rsat/
```

Initialize the Web services stub

```
make -f makefiles/init_rsat.mk ws_init RSAT_WS=${RSAT_WS}
```

After this, re-generate the web services stubb, with the following command

```
make -f makefiles/init_rsat.mk ws_stub RSAT_WS=${RSAT_WS}
```

Test the local web services OK

```
make -f makefiles/init_rsat.mk ws_stub_test
```

Test RSAT Web services (local and remote) without using the SOAP/WSDL stubb (direct parsing of the remote WSDL file)

```
make -f makefiles/init_rsat.mk ws_nostub_test
```

Test the program supported-organisms-server, which relies on Web services without stub

```
supported-organisms-server -url ${RSAT_WS} | wc
supported-organisms-server -url http://localhost/rsat/ | wc
supported-organisms-server -url http://rsat-tagc.univ-mrs.fr/ | wc
```

Tests on the Web site

Run the demo of the following tools (**to redo**)

- retrieve-seq to check the access to local genomes (at least Saccharomyces cerevisiae)
- feature-map to check the GD library
- retrieve-ensembl-seq to check the interface to Ensembl
- fetch-sequences to check the interface to UCSC
- some NeAT tools (they rely on web services)
- peak-motifs because it mobilises half of the RSAT tools -> a good control for the overall installation.
- footprint-discovery to check the tools depending on homology tables (blast tables).

Install the cluster management system (torque, qsub, ...)

Check the number of core (processors)

```
grep ^processor /proc/cpuinfo
```

Check RAM

```
grep MemTotal /proc/meminfo
```

Install Sun Grid Engine (SGE) job scheduler

Beware, before installing the grid engine we need to modify manually the file `/etc/hosts`

```
nano /etc/hosts
```

Initial config (problematic)

```
127.0.0.1       localhost       rsat-vm-2015-02
127.0.1.1      rsat-vm-2015-02
```

Config to obtain:

```
127.0.0.1       localhost       rsat-vm-2015-02
127.0.1.1      rsat-vm-2015-02
```

/?

```
apt-get install --quiet --assume-yes gridengine-client
apt-get install --quiet --assume-yes gridengine-exec
apt-get install --quiet --assume-yes gridengine-master
apt-get install --quiet --assume-yes gridengine-qmon
```

```
qconf -aq default  ## aggregate a new queue called "default"
qconf -mq default  ## modify the queue "default"
qconf -as localhost ## aggregate the localhost tho the list of submitters
```

Set the following values: `hostlist localhost`

Take all default parameters BUT for the SGE master parameter, type `localhost` (it must be the hostname)

Test that jobs can be sent to the job scheduler.

OPTIONAL

Install some software tools for NGS analysis.

```
cd ${RSAT}
make -f makefiles/install_software.mk install_meme
```

Ganglia: tool to monitor a cluster (or single machine)

Link.

```
sudo apt-get install -y ganglia-monitor rrdtool gmetad ganglia-webfrontend
sudo cp /etc/ganglia-webfrontend/apache.conf /etc/apache2/sites-enabled/ganglia.conf
sudo apachectl restart
```

Galaxy server setup

Downloading Galaxy code

We followed the instructions from the Galaxy Web site:

- https://wiki.galaxyproject.org/Admin/GetGalaxy

‘‘‘{r eval=FALSE} ## get a git clone of galaxy git clone https://github.com/galaxyproject/galaxy/ cd galaxy ## Go th the galaxy directory

Check out the master branch, recommended for production server

git checkout -b master origin/master
git pull ## Just in case, we are already up-to-date ‘‘‘

Configure the Galaxy server (and get python modules if required)

We first edit the config file to chooe a specific port for Galaxy

`{r eval=FALSE} cp config/galaxy.ini.sample config/galaxy.ini`

We then edit this file by setting the port to 8082, because our 8080 is already used for other purposes.

We performed the following modifications.

admin_users=admin1@address.fr,admin2@univbazar.fr,admin3@gmail.com port = 8082 # The port on which to listen. host = 0.0.0.0 ## To enable access over the network allow_user_deletion = True

Configuring the Apache server on RSAT

Activate the Apache module rewrite.load

```
{r eval=FALSE} ln -s /etc/apache2/mods-available/rewrite.load /etc/apache2/
mods-enabled/rewrite.load
```

Create a file /etc/apache2/sites-enabled/galaxy.conf with the following content

```
<VirtualHost *:80>
ServerAdmin webmaster@localhost
ServerSignature Off

# Config pour galaxy ands http://mydomain.com/galaxy
RewriteEngine on
RewriteRule ^/galaxy$ /galaxy/ [R]
RewriteRule ^/galaxy/static/style/(.*) /home/galaxy/galaxy/static/june_2007_style/
↪blue/$1 [L]
RewriteRule ^/galaxy/static/scripts/(.*) /home/galaxy/galaxy/static/scripts/packed/$1␣
↪[L]
RewriteRule ^/galaxy/static/(.*) /home/galaxy/galaxy/static/$1 [L]
RewriteRule ^/galaxy/favicon.ico /home/galaxy/galaxy/static/favicon.ico [L]
RewriteRule ^/galaxy/robots.txt /home/galaxy/galaxy/static/robots.txt [L]
RewriteRule ^/galaxy(.*) http://localhost:8082$1 [P]
#RewriteRule ^/galaxy(.*) http://192.168.1.6:8082$1 [P]
</VirtualHost>
```

Restart the Apache server. `{r eval=FALSE} sudo service apache2 restart`

Starting the galaxy server

```
{r eval=FALSE} sh run.sh
```

On our internal network, the server becomes available at the address:

http://192.168.1.6:8082

Registrating

- open a connection to the Galaxy server
- In the Galaxy menu, run the command **User -> Register**. Enter the same email address as you declared as admin users.

Install Galaxy modules

Wiki NGS & Bioinformatics

To be completed

Glossary

We define hereafter a series of abbreviations, terms and concepts which appear recurrently in the litterature about NGS analysis. This document aims at providing a support for the interpretation of analysis reports.

Other resources:

- link

A

B

- **bam (file format):** compressed sam (more).
- **bed (file format):** genomic coordinates (more).
- **bedgraph (file format):** positions + density values (more).
- **bin:**
- **Bonferroni correction:** used in **multiple testing**. Consists in adapting the alpha threshold rather than correcting the **p-value**.

C

- **ChIP-exo:**
- **ChIP-seq:**
- **cigar:** alignment (**'more <>'__**).
- **Cloud:**
- **Copy number variation:**
- **Core:**

D

- **DEG**: Differentially Expressed Gene.

E

- **e-value (E):** indicates the number of false positives expected by chance, for a given threshold of **p-value**. It is a number that can exceed 1, it is thus not a probability, and thus, not a p-value.

E = <FP> = P . m

Where **m** is the number of tests (e.g. genes), FP the number of false positives, the notation < > denotes the random expectation, and P is the nominal p-value of the considered gene.

Note that the e-value is a positive number ranging from 0 to m (number of tests). It is thus not a p-value, since probabilities are by definition comprized between 0 and 1.

F

- **Family-wise error rate (FWER):** indicates the probability to observe at least one false positive among the multiple tests.

FWER = P(FP >= 1)

- **fastq (file format):** raw sequences + quality (more).
- **False discovery rate (FDR):** indicates the expected proportion of false positives *among the cases declared positive*. For example, if a differential analysis reports 200 differentially expressed genes with an FDR threshold of 0.05, we should expect to have 0.05 x 200=10 false positive among them.

G

- **genome (file format):**
- **genomic input:**
- **gff (file format):** genome feature file - annotations (more). See also `gtf`.
- **gtf (file format):** variant of GFF, with two fields for annotation (more).
- **gft2 (file format):** Gene annotation (more).
- **GSM:** Gene Expression Omnibus Sample identifier.
- **GSE:** Gene Expression Omnibus Series identifier (a collection of samples related to the same publication or thematics).

H

I

- **input:** Pour le peak-calling, le mot "input" est utilisé dans un sens tout à fait particulier, pour désigner un jeu de séquences servant à estimer les densités de reads attendues au hasard en fonction de la position génomique. Les méthodes typiques sont l'input génomique (actuellement le plus généralement utilisé) et le mock.

J

K

L

- **Library:** Terme utilisé de façon parfois ambiguë selon le contxte. Les biologistes se réfèrent à une librairie d'ADN pour désigner ... (à définir). Les bioinformaticiens parlent de librairie de séquences pour désigner l'nsemble des fragments de lectures provenant du séquençage d'un même échantillon. Les informaticiens appellent ""library"" (bibliothèque, librairies ?) des modules de code regroupant une série de fonctions et procédures.

M

- **m:** number of tests in a multiple-testing schema (e.g. number of genes in differential expression analysis).
- **Mapped read:**
- **Mapping:** Identifying genomic positions for the raw reads of a sequence library.

- **mock:** type of control for the peak-calling in ChIP-seq. It is an input obtained by using a non-specific antibody (eg. anti-GFP) for the immunoprecipitation. *afin d'estimer le taux de séquençage aspécifique pour chaque région génomique. L'intérêt du mock est qu'il constitue un contrôle réalisé dans les mêmes conditions que le ChIP-seq spécifique. La faiblesse est que les tailles de librairries sont parfois tellement faibles que l'estimation du backgroun est très peu robuste.
- **motif:**
- **Multiple testing:** the multiple testing problem arises from the application of a given statistical test to a large number of cases. For example, in differential expression analysis, each gene/transcript is submitted to a test of equality between two conditions. A single analysis thus typically involves several tens of thousands tests. The general problem of multiple testing is that the risk of false positive indicated by the nominal **p-value** will be challenged for each element. Various types of corrections for multiple testing have been defined (**Bonferroni**, **e-value**, **FWER**, **FDR**).

N

- **Negative control:**
- **NGS**: Next Generation Sequencing.

O

P

- **p-value (P):** the **nominal p-value** is the p-value attached to one particular element in a series of multiple tests. For example, in differential analysis, one nominal p-value is computed for each gene. This p-value indicates the risk to obtain an effect at least as important as our observation *under the null hypothesis*, i.e. in the absence of regulation.
- **padj (abbr.):** adjusted p-value. Statistics derived from the nominal **p-value** in order to correct for the effects of **multiple testing** (see **Bonferroni correction**, **e-value**).

The most usual correction is the FDR, which can be estimated in various ways.

- **Paired end:**
- **Peak:**
- **Peak-calling:**
- **pileup (file format):** base-pair information at each chromosomal position (more).

Q

- **q-value:**

R

- **RAM:**
- **Raw read:** non-aligned read.
- **Read:** short sequence (typically 25-75bp) obtained by high-throughput sequencing.
- **Region-calling:**
- **Replicate:** ... distinguer réplicat technique et réplicat biologique

- **RNA-seq:**

S

- **sam (file format):** aligned reads (more).
- **Single end:**
- **Single nucleotide polymorphism:**
- **SRA:** Sequence Read Archive (SRA). Database maintained by the NCBI.
- **SRX:** Short Read Experiment. See documentation.
- **SRR:** Short Read Run. See documentation.

T

U

V

- **vcf (file format):** variant call format (more).
- **Virtual machine:**

W

- **wig (file format):** coverage / density of some signal along genome (more).

X

Y

Z

Notes on multiple testing corrections

The problem with multiple tests

The **multiple testing** problem arises from the application of a given statistical test to a large number of cases. For example, in differential expression analysis, each gene/transcript is submitted to a test of equality between two conditions. A single analysis thus typically involves several tens of thousands tests.

The general problem of **multiple testing** is that the risk of false positive indicated by the nominal p-value will be challenged for each element.

P-value and derived multiple testing corrections

P-value (nominal p-value)

The **nominal p-value** is the p-value attached to one particular element in a series of multiple tests. For example, in differential analysis, one nominal p-value is computed for each gene. This p-value indicates the risk to obtain an effect at least as important as our observation *under the null hypothesis*, i.e. in the absence of regulation.

Bonferroni correction

E-value

The **e-value** indicates the number of false positives expected by chance, for a given threshold of p-value.

$E =< FP >= P \cdot m$

Where m is the number of tests (e.g. genes), FP the number of false positives, the notation $<>$ denotes the random expectation, and P is the nominal p-value of the considered gene.

Note that the e-value is a positive number ranging from 0 to m (number of tests). It is thus not a p-value, since probabilities are by definition comprized between 0 and 1.

Family-wise error rate (FWER)

The Family-Wise Error Rate (**FWER**) indicates the probability to observe at least one false positive among the multiple tests.

$FWER = P(FP >= 1)$

False Discovery Rate (FDR)

The **False Discovery Rate** (**FDR**) indicates the expected proportion of false positives *among the cases declared positive*. For example, if a differential analysis reports 200 differentially expressed genes with an FDR threshold of 0.05, we should expect to have $0.05 \cdot 200 = 10$ false positive among them.

What is an adjusted p-value?

An **adjusted p-value** is a statistics derived from the nominal p-value in order to correct for the effects of multiple testing.

Various types of corrections for multiple testing have been defined (Bonferoni, e-value, FWER, FDR). Note that some of these corrections are not actual "adjusted p-values".

- the original Bonferoni correction consists in adapting the α threshold rather than correcting the p-value.
- the e-value is a number that can exceed 1, it is thus not a probability, and thus, not a p-value.

The most usual correction is the FDR, which can be estimated in various ways.

Useful links

Versioning, code sharing

- GitHub
- SourceForge
- BitBucket
- SourcesSup Renater

Q & A sites

Specialized in bioformatics:

- SeqAnswers
- Biostars
- Biostars Galaxy

For questions related to computing problems:

- Stack Overflow
- Ask Ubuntu
- Super User

Miscellaneous

- QC Fail Sequencing
- FastQC results interpretation
- A Wikipedia list of sequence alignment software
- Genome sizes for common organisms
- A list of formats maintained by the UCSC
- The IFB cloud and its documentation
- A catalogue of NGS-related tools: Sequencing (OmicTools)
- Elixir's Tools and Data Services Registry.
- Wikipedia list of biological databases

Bibliography

ChIP-seq guidelines

- Bailey et al., 2013. Practical Guidelines for the Comprehensive Analysis of ChIP-seq Data.
- ENCODE & modENCODE consortia, 2012. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia.

Tutorials

French

- Thomas-Chollier et al. 2012. A complete workflow for the analysis of full-size ChIP-seq (and similar) data sets using peak-motifs.
- **TODO** add JvH & MTC tutos
- **TODO** Roscoff bioinformatics school: link
- RNA-seq tutorial

English

- Galaxy tutorial

CHAPTER 4

Indices and tables

- genindex
- modindex
- search