SnakeChunks Documentation

Release 3.0

Claire Rioualen, Jacques van Helden

Sep 07, 2019

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

3.1 Getting started

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

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

3.1.1 Download SnakeChunks

```
cd ${HOME}
wget --no-clobber https://github.com/SnakeChunks/SnakeChunks/archive/4.0.tar.gz
tar xvzf 4.0.tar.gz
SNAKECHUNKS_PATH=${HOME}/SnakeChunks-4.0
```

3.1.2 Clone SnakeChunks

```
cd ${HOME}
git clone https://github.com/SnakeChunks/SnakeChunks.git
```

3.1.3 Run SnakeChunks

Please refer yourself to the Tutorials section.

3.2 SnakeChunks library

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

You can find a description of these hereafter.

3.2.1 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

import_from_sra

quality_control

ChIP-seq

RNA-seq_DEG

integration_ChIP_RNA

Rules (.rules)

This section describes briefly each rule and its parameters. For details on the tools and how to install them, please check this section.

annotate_peaks

This rule runs a program that is part of the HOMER tools suite. It outputs a list of gene identifiers using a bed file, a fasta file and a gtf file. The bed file can be a peak file produced by any peak-calling rule.

More: http://homer.salk.edu/homer/ngs/annotation.html

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["fasta_file"]
- config["genome"]["gtf_file"]

bam_by_name

Sort aligned reads (in bam format) by name using 'samtools sort'.

Required parameters:

- config["qsub"]

bam_by_pos

Sort aligned reads (in bam format) by positions, using 'samtools sort'.

Required parameters:

- config["qsub"]

bam_stats

Computes mapping statistics using the 'samtools flagstat' tool.

Requires samtools 1.3+ version (not in apt-get repository as of 2016-03).

Required parameters:

- config["qsub"]

bam_to_bed

Converts bam files into bed files using 'bedtools bamtobed'.

Required parameters:

- config["qsub"]

bbduk

Performs trimming of raw reads using bbduk of the bbmap suite. Currently only handling single-end data.

Required parameters:

- config["qsub"]
- config["metadata"]["seq_type"]

Optional parameters:

- config["bbduk"]["length_threshold"]
- config["bbduk"]["qual_threshold"]
- config["metadata"]["strands"]

bed_to_fasta

Get a fasta file from a bedfile using the 'fetch-sequences' tool from the RSAT suite.

Fetch sequences from UCSC to obtain a fasta file from a set of genomic coordinates described in a bed file. Requires RSAT installation.

Alternative is to use getfasta.rules .

Example:

```
mkdir -p test/fetch_seq; cd test/fetch_seq;
wget http://pedagogix-tagc.univ-mrs.fr/rsat/demo_files/fetch-sequences_Schmidt_2011_
↪mm9_CEBPA_SWEMBL_R0.12_702peaks.bed;
cd -
snakemake --snakefile ${RSAT}/snakemake_files/chip-seq_motifs.py test/fetch_seq/fetch-
↪sequences_Schmidt_2011_mm9_CEBPA_SWEMBL_R0.12_702peaks.fasta
```

Required parameters:

- config["qsub"]

bedgraph_to_bigwig

Convert bedgraph to bigWig format using Deeptools.

Required parameters:

- config["qsub"]

bedgraph_to_tdf

Convert bedgraph to TDF format, which is recommended to load coverage data in IGV.

The conversion relies on igvtools.

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["fasta_file"]

bedtools_closest

Bedtools closest searches for overlapping features in two coordinate files. In the event that no feature in B overlaps the current feature in A, closest will report the nearest (that is, least genomic distance from the start or end of A) feature in B.

Usage: bedtools closest [OPTIONS] -a <FILE> -b <FILE1, FILE2, . . . , FILEN>

More: http://bedtools.readthedocs.io/en/latest/content/tools/closest.html

Required parameters:

- config["qsub"]
- config["dir"]["genome"]

- config["genome"]["gff3_file"]

bedtools_intersect

Bedtools intersect allows one to screen for overlaps between two sets of genomic features. Moreover, it allows one to have fine control as to how the intersections are reported. bedtools intersect works with both BED/GFF/VCF and BAM files as input.

More: http://bedtools.readthedocs.io/en/latest/content/tools/intersect.html

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["gff3_file"]

bedtools_window

Similar to bedtools intersect, window searches for overlapping features in A and B. However, window adds a specified number (1000, by default) of base pairs upstream and downstream of each feature in A. In effect, this allows features in B that are near features in A to be detected.

More: http://bedtools.readthedocs.io/en/latest/content/tools/window.html

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["gff3_file"]

Opional parameters:

- config["bedtools"]["window"]

blast_formatdb

Run the formatdb program of the BLAST suite in order to index all k-mers of the reference database. This has to be done only once, then the DB can be used for multiple searches with blastall.

Required parameters:

- config["qsub"]

blastall

Where {blast_program} should be replaced by one of the supported program options in blastall: blastp, blastn, blastx, tblastn.

Output file name: {query}_{blast_program}_hits.txt

Required parameters:

- config["qsub"]
- config["blastall"]["db"]

Optional parameters:

- config[“blastall”][“matrix”]
- config[“blastall”][“expect”]
- config[“blastall”][“view”]

bowtie_index

Rule for the creation of Bowtie 1 index. Has to be done only once. The output file is used to test whether the index already exists when aligning.

Required parameters:

- config[“qsub”]
- config[“dir”][“genome”]
- config[“genome”][“version”]
- config[“genome”][“fasta_file”]

bowtie

Read mapping using bowtie. Requires the indexing to have previously been done (using the rule bowtie_index).

Required parameters:

- config[“genome”][“version”]
- config[“genome”][“fasta_file”]
- config[“qsub”]
- config[“dir”][“fastq”]
- config[“dir”][“samples”]

Optional parameters:

- config[“bowtie”][“max_mismatches”]
- config[“bowtie”][“threads”]

bowtie2_index

Rule for the creation of Bowtie 2 index. Has to be done only once. The output file is used to test whether the index already exists when aligning.

Required parameters:

- config[“qsub”]
- config[“dir”][“genome”]
- config[“genome”][“fasta_file”]

bowtie2

Read mapping using Bowtie2. Requires the indexing to have previously been done (using the rule bowtie2_index).

Required parameters:

- config["genome"]["version"]
- config["genome"]["fasta_file"]
- config["qsub"]
- config["dir"]["fastq"]
- config["dir"]["samples"]

Optional parameters:

- config["bowtie2"]["threads"]
- config["bowtie2"]["max_mismatches"]

bPeaks

bPeaks is a peak-calling tool running in R. It was specifically designed for small eucaryotic organisms, such as the yeast. It is thus not recommanded for bigger genomes, as it could be very slow. You should choose parameters carefully. Input in bam, output in bed.

Required parameters:

- config["qsub"]
- config["dir"]["samples"]
- config["dir"]["peaks"]

Optional parameters:

- config["bPeaks"]["IPcoeff"]
- config["bPeaks"]["controlCoeff"]
- config["bPeaks"]["log2FC"]
- config["bPeaks"]["averageQuantiles"]
- config["bPeaks"]["windowSize"]
- config["bPeaks"]["windowOverlap"]

bwa_index

Rule for the creation of BWA index. Has to be done only once. The output file is used to test whether the index already exists when aligning.

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["version"]
- config["genome"]["fasta_file"]

bwa

Requires the indexing to have previously been done (using the rule bwa_index).

Required parameters:

- config[“dir”][“fastq”]
- config[“dir”][“samples”]
- config[“genome”][“version”]
- config[“genome”][“fasta_file”]
- config[“qsub”]

Optional parameters:

- config[“bwa”][“dir”]
- config[“bwa”][“threads”]

count_reads

A set of rules to count the number of reads in NGS files with different formats.

Includes:

- rule count_reads_fastq

Count number of reads in a fastq-formatted file (unaligned reads).

- rule count_reads_fastq_gz

Count number of reads in a gzipped fastq-formatted file (unaligned reads).

- rule count_reads_bam

Count number of reads in a bam-formatted file (binary alignment map, compressed sam).

-rule count_reads_sam

Count number of reads in a bam-formatted file (binary alignment map, compressed sam).

-rule count_features_bed

Count number of features in a bed-formatted file.

cufflinks

Annotate a genome (infer transcript regions) based on RNA-seq data. Assemble transcripts from RNA-seq data and produce a file with the location of detected transcripts.

Required parameters:

- config[“qsub”]
- config[“dir”][“genome”]
- config[“genome”][“gtf_file”]

Optional parameters:

- config[“cufflinks”][“libtype”]

- config[“cufflinks”][“threads”]

cutadapt

Trimming of raw read files using the tool cutadapt and the wrapper Trim Galore.

Currently only working with single end data.

Required parameters:

- config[“qsub”]
- config[“metadata”][“seq_type”]

Optional parameters:

- config[“cutadapt”][“length_threshold”]
- config[“cutadapt”][“qual_threshold”]
- config[“metadata”][“strands”]

dot_graph

This rule generates dot files for snakemake’s DAG and rulegraph.

Required parameter:

- config[“metadata”][“configfile”]

dot_to_image

Following rule dot_graph, this rule creates png, pdf and svg outputs for dot graphs from snakemake.

fastqc

Check the quality of the reads in a fastq or a bam file using the program fastQC (quality control). Results are stored in folder named ‘{reads}_fastqc’.

Custom parameters specified in the configuration file with the option config[“fastqc”][“other options”] will be passed to fastQC.

Required parameters:

- config[‘qsub’]

Optional parameters:

- config[‘fastqc’][‘other_options’]

featnb_from_bed

Count number of features in a bed-formatted file.

Required parameters:

- config[“qsub”]

genome_coverage_bedgraph_strands

Compute two strand-specific genome coverage files from a bam-formatted file with aligned reads. The coverage files are in bedgraph format, which can be loaded in the genome viewer IGV.

Required parameters:

- config["qsub"]

genome_coverage_bedgraph

Compute genome coverage from a bam-formatted file with aligned reads. The coverage file is in bedgraph format (with extension .bedgraph), which can be loaded in the genome viewer IGV.

Note however that IGV issues a warning when bedgraph files are given in input, and recommends to use the tdf format instead. We implemented hereafter rules to convert bedgraph to tdf.

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["fasta_file"]

genome_coverage_bigwig_normalized

Uses bamCompare tool from the deepTools suite.

Required parameters:

- config["qsub"]

genome_coverage_bigwig

Compute genome coverage from a bam-formatted file with aligned reads and produce a bigWig file. Uses bamCoverage tool from the deepTools suite.

Required parameters:

- config["qsub"]

genome_coverage_dz

Compute coverage (reads per position) for each position of a genome, from a bam-formatted file with aligned reads.

BEWARE: this rule is useful for small genomes (Bacteria, Fungi) but would produce a very big file for Metazoan or Plant genomes.

Required parameters:

- config["qsub"]

genome_coverage_wig

Compute genome coverage from a bam-formatted file with aligned reads and produce a wig file, the recommended format to upload coverage-type data as UCSC tracks.

Required parameters:

- config["qsub"]

get_chrom_sizes

This rule generates a file containg the chromosome sizes, with file extension *.genome. This file is required by a number of bedtools utilities.

Required parameters:

- config["qsub"]

getfasta

Get fasta from bed file using the bedtools suite.

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["fasta_file"]

gunzip

Uncompress a file with the gunzip program. The rule is very simple, but is convenient to use in a workflow: it can be used to fix some dependencies on.gz extensions, and/or to send compression jobs to a queue.

Required parameters:

- config["qsub"]

gzip

Uncompress a file with the gunzip program. The rule is very simple, but is convenient to use in a workflow: it can be used to fix some dependencies on.gz extensions, and/or to send compression jobs to a queue.

Required parameters:

- config["qsub"]

homer

Peak-calling with HOMER software, findPeaks algorithm. Input formats: .sam, .bam, .bed (bam input requires samtools to be installed)

The genome parameter can be either the code of a genome installed in Homer (eg HG18, dm3...) or a fasta file (see http://homer.salk.edu/homer-fdr{fdr}_peaks/introduction/update.html)

Required parameters:

- config[“qsub”]
- config[“dir”][“samples”]
- config[“dir”][“peaks”]
- config[“genome”][“fasta_file”]
- config[“genome”][“size”]

Optional parameters:

- config[“homer”][“style”]
- config[“homer”][“L”]
- config[“homer”][“F”]
- config[“homer”][“P”]
- config[“homer”][“fdr”]

index_bam

Index a bam file by creating a .bai file with Samtools The input bam MUST be sorted by position (rule bam_by_pos).

Required parameters:

- config[“qsub”]

index_fasta

Index a fasta file by creating an .fai file with Samtools.

Required parameters:

- config[“qsub”]

macs14

Peak-calling with macs14. Input: bed (others supported) Output: bed

Required parameters:

- config[“genome”][“size”]
- config[“qsub”]
- config[“dir”][“samples”]
- config[“dir”][“peaks”]

Optional parameters:

- config[“macs14”][“pval”]
- config[“macs14”][“keep_dup”]
- config[“macs14”][“bandwidth”]
- config[“macs14”][“mfold”]

- config["macs14"]["other_options"]

macs2

Peak-calling with MACS2. Input: bed Output: bed

Required parameters:

- config["dir"]["samples"]
- config["dir"]["peaks"]
- config["genome"]["size"]
- config["qsub"]

Optional parameters:

- config["macs2"]["qval"]
- config["macs2"]["keep_dup"]
- config["macs2"]["band_width"]
- config["macs2"]["mfold_min"]
- config["macs2"]["mfold_max"]
- config["macs2"]["other_options"]
- config["macs2"]["type"]

matrix_clustering

This rule is currently incomplete.

Motif discovery using the peak-motifs pipeline.

Documentation of tool here.

Required parameters:

- config["qsub"]

Optional parameters:

matrix_quality

Measuring peak enrichment for motifs. Requires at least 2 sets of peaks and 1 motif database.

Documentation of tool here.

Required parameters:

- config["qsub"]
- config["matrix-quality"]["background"]

Optional parameters:

md5sum

Compute the md5sum signature for a given file, which enables to check the consistency of its content after transfer.

Note: md5sum is recommended for submitting NGS data to GEO.

Usage: integrate in the targets of a workflow. Alternatively, can be called directly on the command line with find.

Example: find all the fastq files in a directory (named fastq) and compute one md5sum file for each, and assign the takss to 20 jobs in the scheduler.

```
find fastq/ -name '*.fastq'  \
   | awk '{print $1".md5sum"}' \
   | xargs snakemake  -j 20 -p \
       -s gene-regulation/scripts/snakefiles/rules/md5sum.rules \
       --configfile metadata/Glossina_palpalis.yml
```

Required parameters:

- config["qsub"]

merge_lanes

Concatenate multiple fastq files to produce a merged fastq file.

This rule typically serves to merge raw reads (fastq) corresponding to multiple sequencing lanes for the same sample into a single fastq file per sample.

Since the file naming conventions are highly dependent on the sequencing platform, the file grouping is read from a user-provided text file with tab-separated values (extension .tsv). This file must have been specified in the config file, as config["metadata"]["lane_merging"].

This file must contain at least two columns with this precise header:

- source_file
- merged_file

Additional columns can be included but will be ignored.

There should be a N to 1 correspondence from source file to merge file (each source file should in principle be assigned to a single merged file).

Source files are supposed to be compressed fastq sequence files (.fastq.gz).

The output file is an uncompressed fastq file, because bowtie version 1 does not support gzipped files as input.

Required parameters:

- config["metadata"]["lane_merging"] file indicating the source/merged file names
- config["dir"]["fastq"] base of the directory containing the fastq files

mosaics

Peak-calling with mosaics R package. Input: bed Output: bed

Required parameters:

- config["dir"]["samples"]
- config["qsub"]

Optional parameters:

- config["mosaics"]["FDR"]
- config["mosaics"]["frag_len"]
- config["mosaics"]["bin_size"]
- config["mosaics"]["type"]

peak_motifs

Motif discovery using the peak-motifs pipeline from RSAT.

Documentation.

Required parameters:

- config["qsub"]
- config["peak-motifs"]["motif_db"]

Optional parameters:

- config["peak-motifs"]["tasks"]
- config["peak-motifs"]["disco"]

readnb_from_bam

Count number of reads in a bam-formatted file (binary alignment map, compressed sam).

Required parameters:

- config["qsub"]

readnb_from_fastq

Count number of reads in a fastq-formatted file (unaligned reads).

Required parameters:

- config["qsub"]

readnb_from_sam

Count number of reads in a bam-formatted file (binary alignment map, compressed sam).

Required parameters:

- config["qsub"]

sam_to_bam

Convert reads from SAM (sequence alignment map) to BAM (binary alignment map) format.

Required parameters:

- config["qsub"]

sartools_DESeq2

This rule is designed to perform differential expression analysis of RNA-seq data with DESeq2, using the R package SARTools.

It requires replicated data to run properly.

Required parameters:

- config["qsub"]
- config["author"]
- config["dir"]["samples"]
- config["dir"]["diffexpr"]

Optional parameters:

- config["DESeq2"]["featuresToRemove"]
- config["DESeq2"]["varInt"]
- config["DESeq2"]["condRef"]
- config["DESeq2"]["batch"]
- config["DESeq2"]["alpha"]
- config["DESeq2"]["pAdjustMethod"]
- config["DESeq2"]["fitType"]
- config["DESeq2"]["cooksCutoff"]
- config["DESeq2"]["independentFiltering"]
- config["DESeq2"]["typeTrans"]
- config["DESeq2"]["locfunc"]

sartools_edgeR

This rule is designed to perform differential expression analysis of RNA-seq data with edgeR, using the R package SARTools.

It requires replicated data to run properly.

Required parameters:

- config["qsub"]
- config["author"]
- config["dir"]["samples"]
- config["dir"]["diffexpr"]

Optional parameters:

- config["edgeR"]["featuresToRemove"]
- config["edgeR"]["varInt"]
- config["edgeR"]["condRef"]
- config["edgeR"]["batch"]

- config["edgeR"]["alpha"]
- config["edgeR"]["pAdjustMethod"]
- config["edgeR"]["fitType"]
- config["edgeR"]["cpmCutoff"]
- config["edgeR"]["gene_selection"]
- config["edgeR"]["normalizationMethod"]

sartools_targetfile

This rule creates a so-called "targetfile", which is required by SARTools to run differential expression analyses with rules sartools_DESeq2 and sartools_edgeR.

Required parameters:

- config["qsub"]
- config["dir"]["samples"]
- config["dir"]["diffexpr"]

sickle

Trimming raw reads with sickle.

Required parameters:

- config["qsub"]
- config["metadata"]["seq_type"]

Optional parameters:

- config["sickle"]["qual_threshold"]
- config["sickle"]["length_threshold"]
- config["sickle"]["qual_type"]
- config["metadata"]["strands"]

split_bam_by_strands

Split a bam file in two separate bam files containing respectively the reads on the plus and minus strand.

Required parameters:

- config["qsub"]

spp

Peak-calling with SPP (R package). Input: bam Output: narrowPeak + bed format

Required parameters:

- config["qsub"]

- config[“dir”][“samples”]
- config[“dir”][“peaks”]

Optional parameters:

- config[“spp”][“fdr”]

sra_to_fastq_split

Converts SRA files in separate fastq files for paired-end data with SRA toolkit.

Required parameters:

- config[“qsub”]
- config[“dir”][“reads_source”]
- config[“dir”][“fastq”]

Optional parameters:

- config[“fastq_dump”][“options”]

Usage example :

IMPORT = expand(FASTQ_DIR + “/{samples}/{samples}.fastq”, samples=SAMPLE_IDS)

sra_to_fastq

Converts SRA files in fastq format with SRA toolkit (songle-end data only).

Required parameters:

- config[“qsub”]
- config[“dir”][“reads_source”]
- config[“dir”][“fastq”]

subread_align

Align each sample with the R-package subread.

To align each sample on the reference genome the R-package subread first needs to build a index with the function builindex(). The alignment is then executed with the function align(), which calls the tool read mapping tool Subread.

Reference: Liao Y, Smyth GK and Shi W (2013). The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, 41(10):e108

Required parameters:

- config[“dir”][“fastq”]
- config[“dir”][“samples”]
- config[“genome”][“version”]
- config[“genome”][“fasta_file”]
- config[“qsub”]
- config[“metadata”][“seq_type”]

Optional parameters:

- config[“subread-align”][“dir”]
- config[“subread-align”][“threads”]
- config[“subread-align”][“max_mismatches”]
- config[“subread-align”][“align_options”]
- config[“subread-align”][“seq_data”]

subread_featureCounts_all

This rule computes bam files and produces a tab file containing feature counts for all samples, using featureCounts from the subread toolkit.

Required parameters:

- config[“qsub”]
- config[“dir”][“genome”]
- config[“genome”][“gtf_file”]

Optional parameters:

- config[“subread-featureCounts”][“attr_type”]
- config[“subread-featureCounts”][“feature_type”]
- config[“subread-featureCounts”][“multi_mapping”]
- config[“subread-featureCounts”][“strand_specificity”]

Usage:

```
featureCounts [options] -a <annotation_file> -o <output_file> input_file1 [input_
↪file2] ...
```

subread_featureCounts

This rule computes bam files and produces tab files containing feature counts for every sample separately, using featureCounts from the subread toolkit.

Required parameters:

- config[“qsub”]
- config[“dir”][“genome”]
- config[“genome”][“gtf_file”]

Optional parameters:

- config[“subread-featureCounts”][“attr_type”]
- config[“subread-featureCounts”][“feature_type”]
- config[“subread-featureCounts”][“multi_mapping”]
- config[“subread-featureCounts”][“strand_specificity”]

Usage:

```
featureCounts [options] -a <annotation_file> -o <output_file> input_file1 [input_
↪file2] ...
```

subread_index

Rule for the creation of subread index. Has to be done only once. The output file is used to test whether the index already exists when aligning.

Reference: Liao Y, Smyth GK and Shi W (2013). The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, 41(10):e108

Required parameters:

- config["qsub"]
- config["dir"]["genome"]
- config["genome"]["version"]
- config["genome"]["fasta_file"]

swembl

Peak-calling with SWEMBL.

Beware: for SWEMBL the peaks MUST be sorted by position, other wise SWEMBL runs indefinitely. Usually by default we sort all bam files by position after alignment (with rule bam_by_pos).

Required parameters:

- config["qsub"]
- config["dir"]["samples"]
- config["dir"]["peaks"]

Optional parameters:

- config["swembl"]["x"]
- config["swembl"]["R"]
- config["swembl"]["N"]

tophat

Read mapping for single or paired end data using Tophat. Requires the indexing to have previously been done (using the rule bowtie2_index).

Required parameters:

- config["dir"]["fastq"]
- config["dir"]["samples"]
- config["metadata"]["seq_type"]
- config["dir"]["genome"]
- config["genome"]["version"]

- config["genome"]["fasta_file"]
- config["qsub"]

Optional parameters:

- config["tophat"]["max_mismatches"]
- config["tophat"]["threads"]

3.2.2 Python scripts (.py)

todo

3.2.3 R scripts

todo

3.2.4 Configuration files (yaml)

todo

3.2.5 R markdown files (.Rmd)

todo

3.2.6 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

3.3 Dependencies

These manuals aim at helping you install programs and dependencies used in the SnakeChunks library.

Some of them are mandatory, and some are optional, depending on the Snakemake workflows you need to run.

They were tested under Ubuntu 14.04.

3.3.1 Manual installation

This manual is organized in sections, so you can cherry-pick the programs you want to manually install. For "all inclusive" solutions, please refer yourself to the following sections.

General

Generic tools

nano

Nano is a simple command-line text editor.

```
sudo apt-get install nano
```

rsync

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

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

git

Git rsync is a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people.

```
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

Unix package required by several tools, including Sickle and Bamtools.

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

Java

Java is required by several tools using GUIs, such as FastQC or IGV.

It seems java 9 causes issues with IGV, so we chose to use java 8 here.

```
echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-
↪selections
echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-
↪selections
sudo add-apt-repository -y ppa:webupd8team/java
sudo apt-get update
sudo apt-get -y install oracle-java8-installer
```

Check installation:

```
java -version
```

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
```

Graphviz

Snakemake can generate useful graphviz outputs.

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

Python

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

```
python --version
python3 --version
```

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

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

Install python package managers and devel libraries.

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

Pandas library

Python Data Analysis Library is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

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

The package rpy2 alloàws to access R from within Python code.

```
sudo apt-get install python-matplotlib
pip3 install "rpy2<2.3.10"
```

R

You can fetch a CRAN mirror here.

```
sudo sh -c "echo 'deb <your mirror> trusty/' >> /etc/apt/sources.list"
↪          ## Repository for Ubuntu 14.04 Trusty Tahr
#sudo sh -c "echo 'deb http://ftp.igh.cnrs.fr/pub/CRAN/ trusty/' >> /etc/apt/sources.
↪list"      ## Mirror in Montpellier, France
sudo apt-get -y update
sudo apt-get -y install r-base r-base-dev libcurl4-openssl-dev libxml2-dev
echo "r <- getOption('repos'); r['CRAN'] <- 'http://cran.us.r-project.org';
↪options(repos = r);" >> ~/.Rprofile
```

Check installation:

```
R --version
```

Snakemake

"Snakemake is a workflow engine that provides a readable Python-based workflow definition language and a powerful execution environment that scales from single-core workstations to compute clusters without modifying the workflow.

It is the first system to support the use of automatically inferred multiple named wildcards (or variables) in input and output filenames."

(Köster and Rahman, 2012)

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

NB: Python 3 and pip3 are required (see this section).

```
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}"
```

```
touch $HOME/hello.py
nano $HOME/hello.py             ## copy/paste script above and save
```

Execute the workflow; two files should be created: `hello.txt` and `bye.txt`.

```
cd ; snakemake -s hello.py
```

In case you need to upgrade snakemake:

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

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

```
pip3 install docutils
```

Quality control

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
```

NB: FastQC requires to have Java installed (even for commandline use). See dedicated section to install it.

Check installation:

```
fastqc --version
```

MultiQC

MultiQC searches a given directory for analysis logs and compiles a HTML report. It's a general use tool, perfect for summarising the output from numerous bioinformatics tools.

```
sudo pip install multiqc
```

NB: a bug can appear depending on versions:

Command python setup.py egg_info failed with error code 1 in /tmp/pip_build_root/matplotlib Storing debug log for failure in /home/gr/.pip/pip.log

If so, it can be avoided by installing ubuntu dependencies, then reinstalling multiqc:

```
sudo apt-get install libfreetype6-dev python-matplotlib
sudo pip install multiqc
```

Check installation:

```
multiqc --version
```

Trimming

The quality of the reads generated by high-throughput sequencing technologies tends 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
```

Check installation:

```
sickle --version
```

Cutadapt

Cutadapt finds and removes adapter sequences, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads.

```
pip install --user --upgrade cutadapt
mv /root/.local/bin/cutadapt $HOME/bin
```

Check installation:

```
cutadapt --version
```

TrimGalore

In our workflows we use TrimGalore, a wrapper around Cutadapt and FastQC. It should be installed if you want to run cutadapt.

```
cutadapt --version                                  # Check that cutadapt is installed
fastqc -v                                           # Check that FastQC is installed

cd $HOME/app_sources
```

(continues on next page)

(continued from previous page)

```
curl -fsSL https://github.com/FelixKrueger/TrimGalore/archive/0.4.3.tar.gz -o trim_
↪galore.tar.gz
tar xvzf trim_galore.tar.gz
mv TrimGalore-0.4.3/trim_galore $HOME/bin
```

Check installation:

```
trim_galore --version
```

BBDuk

- SeqAnswers
- SourceForge

```
cd $HOME/app_sources
wget https://sourceforge.net/projects/bbmap/files/BBMap_37.31.tar.gz
tar xvzf BBMap_37.31.tar.gz
cp `find bbmap/ -maxdepth 1 -executable -type f` $HOME/bin
```

Alignment/mapping

The point of mapping is to replace the reads obtained from the sequencing step onto a reference genome. When the read is long enough, it can be mapped on the genome with a pretty good confidence, by tolerating a certain amount of so-called mismatches. However, genomes can contain repeated regions that are harder to deal with.

We call "sequencing depth" the average number of reads mapped at each position of the genome. The bigger the sequencing depth, the better the quality of the alignment, and the better the downstream analyses.

BWA

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

- 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
```

Check installation:

```
bwa
```

Bowtie

Bowtie performs ungapped alignment, and is therefore not suitable for certain types of data, like RNA-seq data.

```
cd $HOME/app_sources
wget --no-clobber http://downloads.sourceforge.net/project/bowtie-bio/bowtie/1.2.2/
↪bowtie-1.2.2-linux-x86_64.zip
unzip bowtie-1.2.2-linux-x86_64.zip
cp `find bowtie-1.2.2-linux-x86_64/ -maxdepth 1 -executable -type f` $HOME/bin
```

Check installation:

```
bowtie --version
```

Bowtie2

"Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index (based on the Burrows-Wheeler Transform or BWT) to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 gigabytes of RAM. Bowtie 2 supports gapped, local, and paired-end alignment modes. Multiple processors can be used simultaneously to achieve greater alignment speed. Bowtie 2 outputs alignments in SAM format, enabling interoperation with a large number of other tools (e.g. SAMtools, GATK) that use SAM. Bowtie 2 is distributed under the GPLv3 license, and it runs on the command line under Windows, Mac OS X and Linux."

General documentation

Instructions

Downloads

Reference:

Langmead B, Trapnell C, Pop M, L Salzberg S. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 200910:R25. DOI: 10.1186/gb-2009-10-3-r25

```
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
cp `find bowtie2-2.2.6/ -maxdepth 1 -executable -type f` $HOME/bin
```

Check installation:

```
bowtie2 --version
```

Subread-align

The Subread package comprises a suite of software programs for processing next-gen sequencing read data including:

> Subread: a general-purpose read aligner which can align both genomic DNA-seq and RNA-seq reads. It can also be used to discover genomic mutations including short indels and structural variants. Subjunc: a read aligner developed for aligning RNA-seq reads and for the detection of exon-exon junctions. Gene fusion events can be detected as well. featureCounts: a software program developed for counting reads to genomic features such as genes, exons, promoters and genomic bins. exactSNP: a SNP caller that discovers SNPs by testing signals against local background noises

Reference:

Liao Y, Smyth GK and Shi W. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, 41(10):e108, 2013

```
cd $HOME/app_sources
wget -nc https://sourceforge.net/projects/subread/files/subread-1.5.2/subread-1.5.2-
↪source.tar.gz
tar zxvf subread-1.5.2-source.tar.gz
cd subread-1.5.2-source/src
make -f Makefile.Linux
cd ../bin
cp `find * -executable -type f` $HOME/bin
```

Check installation:

```
subread-align --version
```

Tophat

TopHat is a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads to mammalian-sized genomes using the ultra high-throughput short read aligner Bowtie, and then analyzes the mapping results to identify splice junctions between exons.

```
cd $HOME/app_sources
wget --no-clobber https://ccb.jhu.edu/software/tophat/downloads/tophat-2.0.14.Linux_
↪x86_64.tar.gz
tar xvfz tophat-2.0.14.Linux_x86_64.tar.gz
cd tophat-2.0.14.Linux_x86_64
rm -Rf AUTHORS LICENSE README intervaltree/ sortedcontainers/
mv ./* $HOME/bin
cd ..; rm -Rf tophat-2.0.14.Linux_x86_64*
```

Check installation:

```
tophat --version
```

Peak-calling

The following tools can be used to perform ChIP-seq peak-calling.

Homer

Required in order to run the tutorials.

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).

However, Homer can also work with custom genomes in FASTA format and gene annotations in GTF format. Thus the SnakeChunks workflows don't require to install any genome.

Check installation:

```
findMotifs.pl
```

MACS 1.4

Required in order to run the demo workflow "ChIP-seq" on dataset GSE20870 (in the tutorials section).

- Documentation
- Installation manual

```
cd $HOME/app_sources
wget "https://github.com/downloads/taoliu/MACS/MACS-1.4.2.tar.gz"
tar -xvzf MACS-1.4.2.tar.gz
cd MACS-1.4.2
sudo python setup.py install
```

Check installaiton:

```
macs14 --version
```

MACS 2

Required in order to run the tutorials.

- Webpage

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

Check installation:

```
macs2 --version
```

bPeaks

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

It is currently not used in demo workflows, and is therefore not mandatory to run the tutorials.

Available as an R package.

Web page

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

SPP

This peak-caller is used in the ChIP-seq study case GSE20870.

- Method 1: git

See github page.

Commands in R:

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

- Method 2: Bioconductor [deprecated]

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

- Method 3: commandline [deprecated]

```
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
```

- Method 4: the ultimate protocol for Ubuntu 14.04

In unix shell:

```
# unix libraries
sudo apt-get update
sudo apt-get -y install r-base
sudo 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
cd $HOME/app_sources
wget http://compbio.med.harvard.edu/Supplements/ChIP-seq/spp_1.11.tar.gz
sudo R CMD INSTALL spp_1.11.tar.gz
```

Check installation in R:

```
library(spp)
```

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!

Mosaics

This peak-caller is used in the ChIP-seq study case GSE20870.

Installation in R from Bioconductor:

```
source("https://bioconductor.org/biocLite.R")
biocLite("mosaics")
```

SWEMBL

This peak-caller is used in the ChIP-seq study case GSE20870.

- SWEMBL beginner's manual

```
git clone https://github.com/stevenwilder/SWEMBL.git
cd SWEMBL
make
cp SWEMBL $(BIN_DIR)
```

Deprecated method

```
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

# This method require a manual fix of C flags in makefile
# gcc main.c IO.c calc.c stack.c summit.c refcalc.c wiggle.c overlap.c -o SWEMBL -lz -
↪lm
```

Motif discovery, motif analysis

These software can be useful for the analysis of ChIP-seq peaks.

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.

RNA-seq

featureCounts

Liao Y, Smyth GK and Shi W. featureCounts: an efficient general-purpose program for assigning sequence reads to genomic features. Bioinformatics, 30(7):923-30, 2014

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
- ...

3.3.2 Makefile

The SnakeChunks library comprises a makefile that can install most of the dependencies described in the previous section. It is recommended when you're setting up a virtual environments, as described in these tutorials.

If you want to run the workflows on your personal computer or on a server, you should follow the manual installation, or contact a sysadmin.

The makefile currently allows running the following workflows:

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

It is not yet handling al the RNA-seq dependencies.

```
# it is assumed that you have defined a global variable with the path to the␣
↪SnakeChunks library
cd ${SNAKECHUNKS_PATH}
make -f scripts/makefiles/install_tools_and_libs.mk all
source ~/.bashrc
```

3.3.3 Conda

A number of dependencies of SnakeChunks can be installed through a Conda environment. This list is not exhaustive.

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

3.4 Tutorials

3.4.1 Demo: ChIP-seq analysis

Case: ChIP-seq study of Tbf1 in *S. cerevisiae*

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

Setup analysis environment

Here, we create a directory that will contain the raw data, the SnakeChunks library, the reference genome data and the results of the workflow(s) used.

We are going to use a global variable: $ANALYSIS_DIR.

```
ANALYSIS_DIR=$HOME/ChIP-seq_GSE20870
mkdir -p ${ANALYSIS_DIR}
cd ${ANALYSIS_DIR}
```

Download SnakeChunks

We are going to download the SnakeChunks library in the analysis directory. Another possibility would be to download SnakeChunks in a fixed place, and create a symlink to the analysis directory.

```
wget --no-clobber https://github.com/SnakeChunks/SnakeChunks/archive/4.1.2.tar.gz
tar xvzf 4.1.2.tar.gz
ln -s SnakeChunks-4.1.2 SnakeChunks
```

Download reference genome & annotations

Here, we are going to download the genome sequence and annotation files in the analysis directory. It is also possible to define a fixed location to store genomes and then create a symlink to it.

It can be useful to store all the genomes in one place, in order to avoid duplication of big files. Also, most mapping algorithms need to index the genome before proceeding with the alignment. This index needs only be done once, but it takes time and storage space, so it's better to avoid duplicating it.

```
mkdir ${ANALYSIS_DIR}/genome
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
```

Download raw 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/ByRun/sra/
↪SRR/SRR051/SRR051929/SRR051929.sra -P ${ANALYSIS_DIR}/data/GSM521934
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR051/SRR051930/SRR051930.sra -P ${ANALYSIS_DIR}/data/GSM521935
```

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. You can check out the glossary to find out more about file formats.

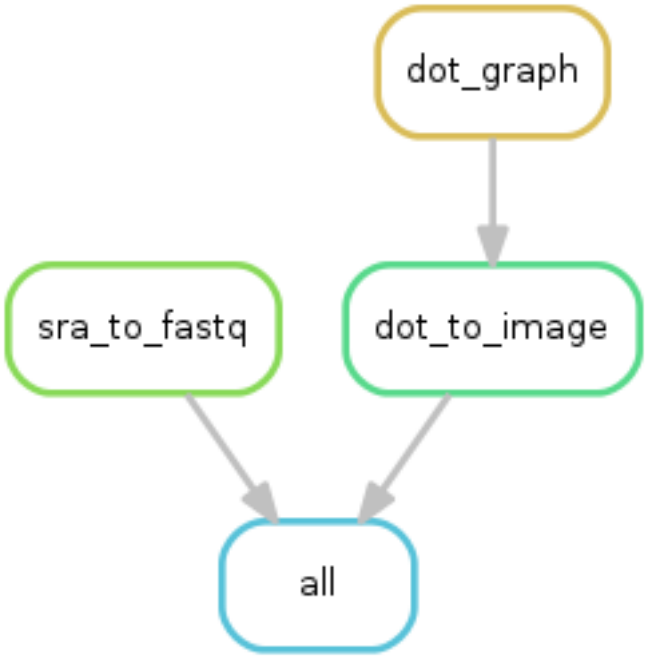
Workflow execution

If you have followed the previous steps, you have a file organization that looks like this:

```
root@vm0026:~/mydisk/ChIP-seq_SE_GSE20870# ls -lR
.:
total 8
lrwxrwxrwx 1 root root   21 Jul  4 11:02 SnakeChunks -> /root/SnakeChunks-4.0
drwxr-xr-x 4 root root 4096 Jul  4 11:02 data
drwxr-xr-x 2 root root 4096 Jul  4 11:03 genome

./data:
total 8
drwxr-xr-x 2 root root 4096 Jul  4 11:02 GSM521934
drwxr-xr-x 2 root root 4096 Jul  4 11:02 GSM521935

./data/GSM521934:
total 81184
-rw-r--r-- 1 root root 83043426 Jul  4 11:02 SRR051929.sra

./data/GSM521935:
total 94116
-rw-r--r-- 1 root root 96275406 Jul  4 11:03 SRR051930.sra

./genome:
total 29644
-rw-r--r-- 1 root root 12360636 Jul  4 11:03 Saccharomyces_cerevisiae.R64-1-1.30.dna.genome.fa
-rw-r--r-- 1 root root  6673011 Jul  4 11:03 Saccharomyces_cerevisiae.R64-1-1.30.gff3
-rw-r--r-- 1 root root 11268860 Jul  4 11:03 Saccharomyces_cerevisiae.R64-1-1.30.gtf
```

You should then be able to run the following command:

```
cd ${ANALYSIS_DIR}
snakemake -s SnakeChunks/scripts/snakefiles/workflows/import_from_sra.wf -p --
↪configfile SnakeChunks/examples/ChIP-seq_SE_GSE20870/config.yml --use-conda
```



Workflow 'quality_control'

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

The purpose of this workflow is to perform quality check with FastQC.

Optionally, trimming can be performed using the tools Sickle or Cutadapt.

Workflow execution

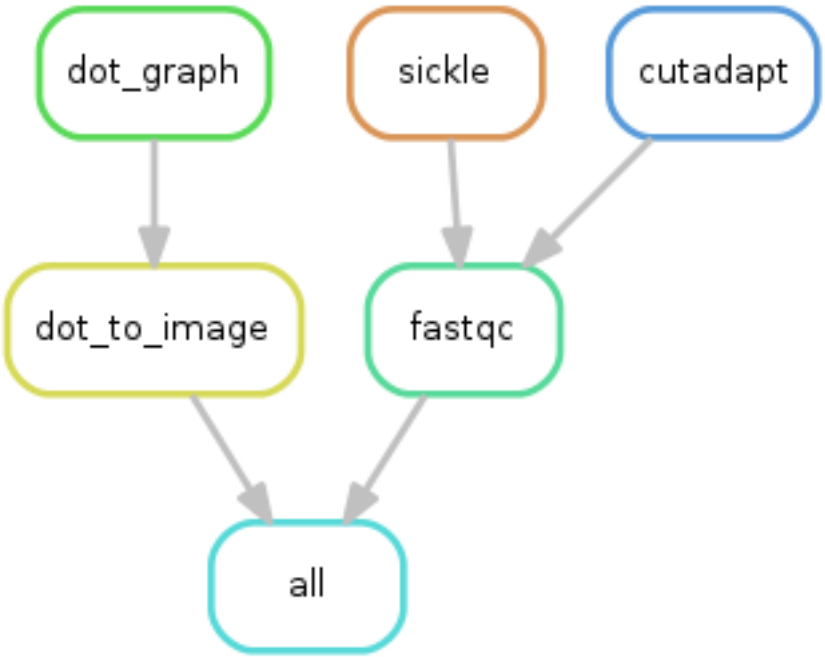
```
cd ${ANALYSIS_DIR}
snakemake -s SnakeChunks/scripts/snakefiles/workflows/quality_control.wf -p --
↪configfile SnakeChunks/examples/ChIP-seq_SE_GSE20870/config.yml --use-conda
```

Workflow 'ChIP-seq'

This workflows performs:

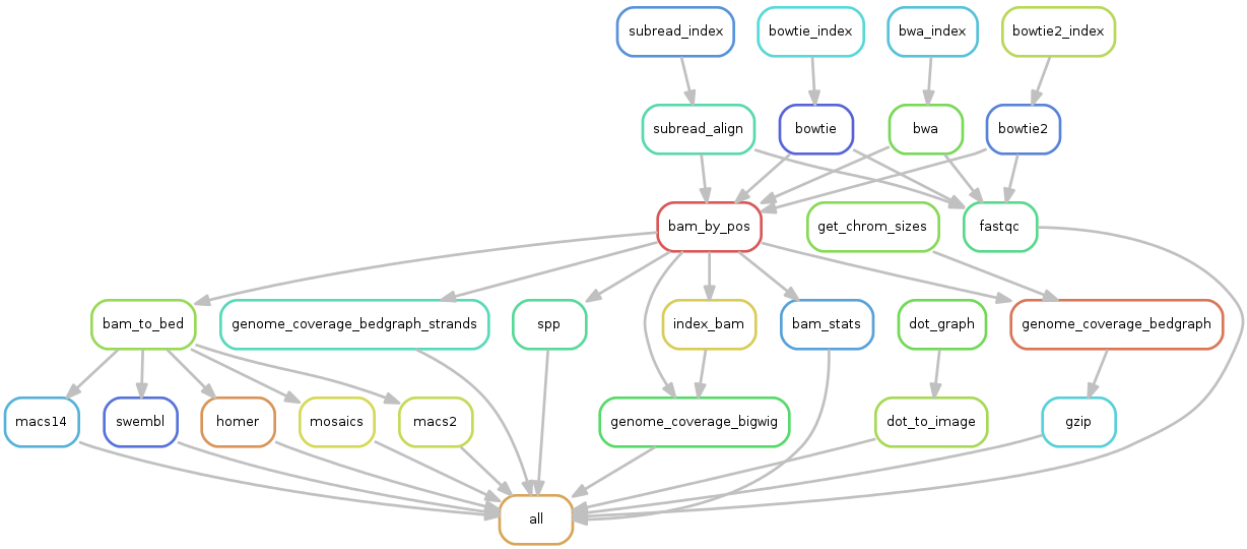
- mapping with various algorithms
- genome coverage in different formats (check out our glossary)
- peak-calling with various algorithms
- motifs search using the RSAT suite

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



Workflow execution

```
cd ${ANALYSIS_DIR}
snakemake -s SnakeChunks/scripts/snakefiles/workflows/ChIP-seq_peak-calling.wf -p --
↪configfile SnakeChunks/examples/ChIP-seq_SE_GSE20870/config.yml --use-conda
```



3.4.2 Demo: ChIP-seq and RNA-seq integration

Case: Genomic analysis of the scc2-4 mutant in budding yeast

Reference

Genomic analysis of the scc2-4 mutant in budding yeast

Musinu Zakari

GEO series

- ChIP-seq: GSE55357
- RNA-seq: GSE55316

Workflow 'ChIP-seq'

```
ANALYSIS_DIR=$HOME/ChIP-seq_GSE55357
mkdir ${ANALYSIS_DIR}
cd ${ANALYSIS_DIR}
```

Download the SnakeChunks library

```
wget --no-clobber https://github.com/SnakeChunks/SnakeChunks/archive/4.1.2.tar.gz
tar xvzf 4.1.2.tar.gz
ln -s SnakeChunks-4.1.2 SnakeChunks
```

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
```

Download ChIP-seq data

```
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176905/SRR1176905.sra -P ${ANALYSIS_DIR}/data/GSM1334674
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176907/SRR1176907.sra -P ${ANALYSIS_DIR}/data/GSM1334676
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176908/SRR1176908.sra -P ${ANALYSIS_DIR}/data/GSM1334679
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176910/SRR1176910.sra -P ${ANALYSIS_DIR}/data/GSM1334677
```

Workflow execution

Your directory should now look like this:

And you should be able to execute it like this:

```
rioualen@nlitb:~/ChIP-seq_GSE55357$ ls -l
total 7408
-rw-rw-r-- 1 rioualen rioualen 7569612 Aug  3 12:05 4.0.tar.gz
lrwxrwxrwx 1 rioualen rioualen      15 Aug  3 12:05 SnakeChunks -> SnakeChunks-4.0
drwxrwxr-x 6 rioualen rioualen    4096 Jun 12 19:05 SnakeChunks-4.0
drwxrwxr-x 6 rioualen rioualen    4096 Aug  3 12:09 data
drwxrwxr-x 2 rioualen rioualen    4096 Aug  3 12:06 genome
rioualen@nlitb:~/ChIP-seq_GSE55357$ ls -lR data genome
data:
total 16
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 12:06 GSM1334674
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 12:07 GSM1334676
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 12:09 GSM1334677
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 12:08 GSM1334679

data/GSM1334674:
total 221340
-rw-rw-r-- 1 rioualen rioualen 226644238 Aug  3 12:07 SRR1176905.sra

data/GSM1334676:
total 226152
-rw-rw-r-- 1 rioualen rioualen 231578633 Aug  3 12:08 SRR1176907.sra

data/GSM1334677:
total 253292
-rw-rw-r-- 1 rioualen rioualen 259370088 Aug  3 12:09 SRR1176910.sra

data/GSM1334679:
total 309352
-rw-rw-r-- 1 rioualen rioualen 316773150 Aug  3 12:09 SRR1176908.sra

genome:
total 29600
-rw-rw-r-- 1 rioualen rioualen 12360636 Aug  3 12:06 Saccharomyces_cerevisiae.R64-1-1.30.dna.genome.fa
-rw-rw-r-- 1 rioualen rioualen  6673011 Aug  3 12:06 Saccharomyces_cerevisiae.R64-1-1.30.gff3
-rw-rw-r-- 1 rioualen rioualen 11268860 Aug  3 12:06 Saccharomyces_cerevisiae.R64-1-1.30.gtf
```

```
cd ${ANALYSIS_DIR}
snakemake -s SnakeChunks/scripts/snakefiles/workflows/import_from_sra.wf -p --
↪configfile SnakeChunks/examples/ChIP-seq_GSE55357/config.yml
snakemake -s SnakeChunks/scripts/snakefiles/workflows/quality_control.wf -p --
↪configfile SnakeChunks/examples/ChIP-seq_GSE55357/config.yml
snakemake -s SnakeChunks/scripts/snakefiles/workflows/ChIP-seq_peak-calling.wf -p --
↪configfile SnakeChunks/examples/ChIP-seq_GSE55357/config.yml
```

Workflow 'RNA-seq' for differential expression analysis

```
ANALYSIS_DIR=$HOME/RNA-seq_GSE55316
mkdir ${ANALYSIS_DIR}
cd ${ANALYSIS_DIR}
```

Download the SnakeChunks library

```
wget --no-clobber https://github.com/SnakeChunks/SnakeChunks/archive/4.1.2.tar.gz
tar xvzf 4.1.2.tar.gz
ln -s SnakeChunks-4.1.2 SnakeChunks
```

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
```

Download RNA-seq data

```
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176894/SRR1176894.sra -P ${ANALYSIS_DIR}/data/GSM1334027
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176896/SRR1176896.sra -P ${ANALYSIS_DIR}/data/GSM1334029
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176900/SRR1176900.sra -P ${ANALYSIS_DIR}/data/GSM1334033
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
↪SRR/SRR117/SRR1176901/SRR1176901.sra -P ${ANALYSIS_DIR}/data/GSM1334034
```

Workflow execution

Your directory should now look like this:

And you should be able to execute it like this:

```
cd ${ANALYSIS_DIR}
snakemake -s SnakeChunks/scripts/snakefiles/workflows/import_from_sra.wf -p --
↪configfile SnakeChunks/examples/RNA-seq_GSE55316/config.yml
snakemake -s SnakeChunks/scripts/snakefiles/workflows/RNA-seq_complete.wf -p --
↪configfile SnakeChunks/examples/RNA-seq_GSE55316/config.yml
```

Workflow 'integration_ChIP_RNA'

coming soon

3.4.3 Demo: alternative transcripts with RNA-seq

Case: C.elegans

Setup workdir

```
ANALYSIS_DIR=$HOME/GSE59705_RNA-seq_splicing
mkdir ${ANALYSIS_DIR}
cd ${ANALYSIS_DIR}
```

```
rioualen@nlitb:~/RNA-seq_GSE55316$ ls -l
total 7408
-rw-rw-r-- 1 rioualen rioualen 7569612 Aug  3 13:13 4.0.tar.gz
lrwxrwxrwx 1 rioualen rioualen      15 Aug  3 13:13 SnakeChunks -> SnakeChunks-4.0
drwxrwxr-x 6 rioualen rioualen    4096 Jun 12 19:05 SnakeChunks-4.0
drwxrwxr-x 6 rioualen rioualen    4096 Aug  3 13:11 data
drwxrwxr-x 2 rioualen rioualen    4096 Aug  3 13:09 genome
rioualen@nlitb:~/RNA-seq_GSE55316$ ls -lR data genome
data:
total 16
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 13:09 GSM1334027
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 13:10 GSM1334029
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 13:10 GSM1334033
drwxrwxr-x 2 rioualen rioualen 4096 Aug  3 13:11 GSM1334034

data/GSM1334027:
total 509152
-rw-rw-r-- 1 rioualen rioualen 521366228 Aug  3 13:10 SRR1176894.sra

data/GSM1334029:
total 602260
-rw-rw-r-- 1 rioualen rioualen 616708999 Aug  3 13:10 SRR1176896.sra

data/GSM1334033:
total 619640
-rw-rw-r-- 1 rioualen rioualen 634504350 Aug  3 13:11 SRR1176900.sra

data/GSM1334034:
total 738392
-rw-rw-r-- 1 rioualen rioualen 756106659 Aug  3 13:11 SRR1176901.sra

genome:
total 29600
-rw-rw-r-- 1 rioualen rioualen 12360636 Aug  3 13:09 Saccharomyces_cerevisiae.R64-1-1.30.dna.genome.fa
-rw-rw-r-- 1 rioualen rioualen  6673011 Aug  3 13:09 Saccharomyces_cerevisiae.R64-1-1.30.gff3
-rw-rw-r-- 1 rioualen rioualen 11268860 Aug  3 13:09 Saccharomyces_cerevisiae.R64-1-1.30.gtf
```

Download the SnakeChunks library

```
wget --no-clobber https://github.com/SnakeChunks/SnakeChunks/archive/4.1.2.tar.gz
tar xvzf 4.1.2.tar.gz
ln -s SnakeChunks-4.1.2 SnakeChunks
```

Download reference genome & annotations

```
wget -nc ftp://ftp.wormbase.org/pub/wormbase/releases/WS220/species/c_elegans/c_
↪elegans.WS220.genomic.fa.gz -P ${ANALYSIS_DIR}/genome
wget -nc ftp://ftp.wormbase.org/pub/wormbase/releases/WS220/species/c_elegans/c_
↪elegans.WS220.annotations.gff3.gz -P ${ANALYSIS_DIR}/genome
wget -nc ftp://ftp.wormbase.org/pub/wormbase/releases/WS253/species/c_elegans/
↪PRJNA13758/c_elegans.PRJNA13758.WS253.canonical_geneset.gtf.gz -P ${ANALYSIS_DIR}/
↪genome
gunzip ${ANALYSIS_DIR}/genome/*.gz
```

```
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/001/SRR1523361/SRR1523361_2.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443914
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/001/SRR1523361/SRR1523361_1.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443914
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/002/SRR1523362/SRR1523362_2.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443915
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/002/SRR1523362/SRR1523362_1.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443915
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/003/SRR1523363/SRR1523363_2.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443916
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/003/SRR1523363/SRR1523363_1.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443916
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/004/SRR1523364/SRR1523364_2.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443917
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR152/004/SRR1523364/SRR1523364_1.fastq.
↪gz -P ${ANALYSIS_DIR}/fastq/GSM1443917
gunzip ${ANALYSIS_DIR}/fastq/*/*.gz
```

- rename

```
snakemake -s SnakeChunks/scripts/snakefiles/workflows/RNA-seq_transcripts.wf -p --
↪configfile SnakeChunks/examples/RNA-seq_GSE59705/config.yml -n
```

3.4.4 Running SnakeChunks workflows on your own data

SnakeChunks library & genome data

Hereafter is a suggestion for the organization of your files.

```
ANALYSIS_DIR=$HOME/my_analysis
mkdir -p ${ANALYSIS_DIR}
cd ${ANALYSIS_DIR}
```

```
# Download the SnakeChunks library
wget --no-clobber https://github.com/SnakeChunks/SnakeChunks/archive/4.1.2.tar.gz
```

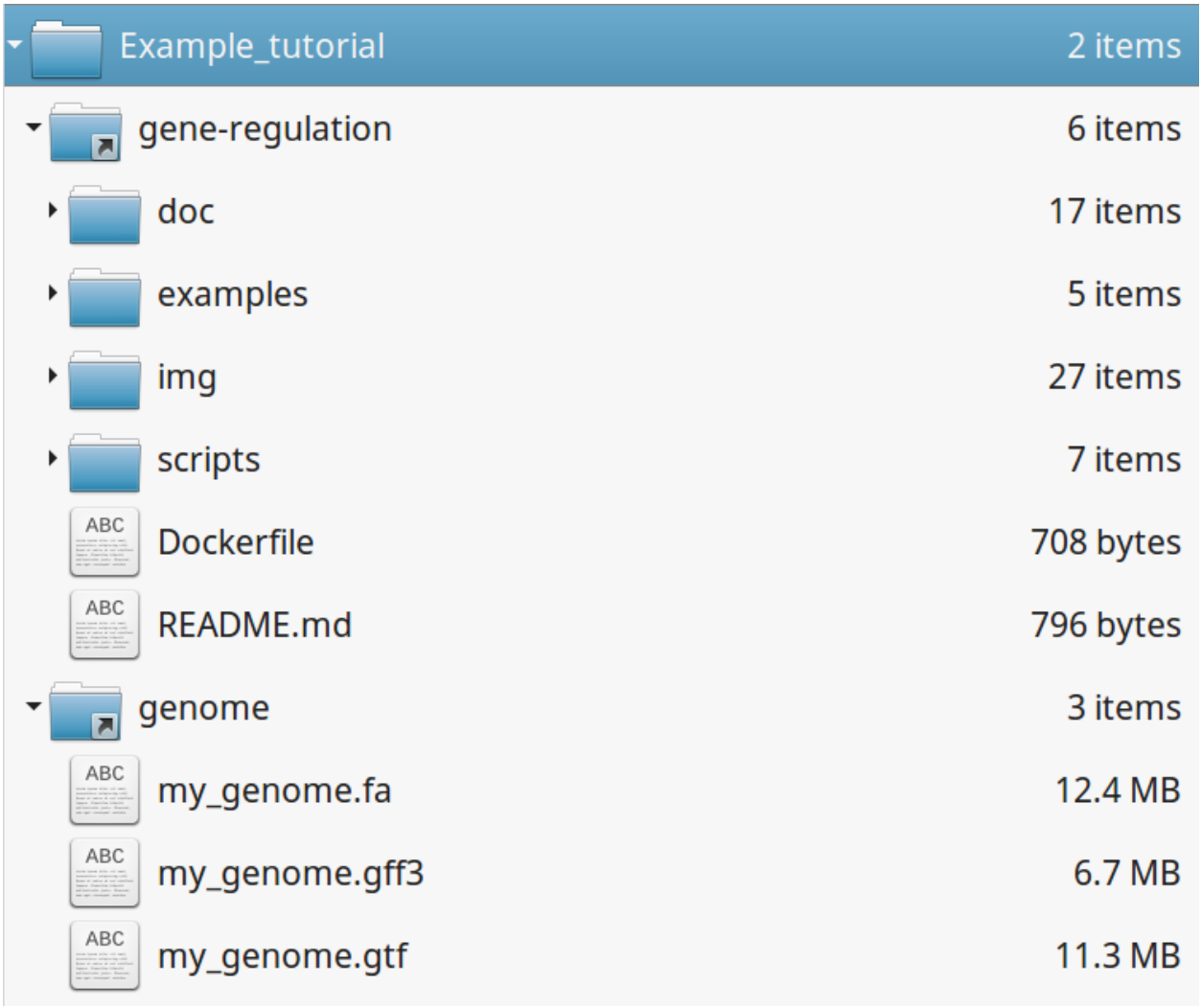
(continues on next page)

(continued from previous page)

```
tar xvzf 4.1.2.tar.gz
ln -s SnakeChunks-4.1.2 SnakeChunks
```

```
# Download genome data
wget -nc <URL_to_my_genome.fa.gz> -P ${ANALYSIS_DIR}/genome
wget -nc <URL_to_my_genome.gff3.gz> -P ${ANALYSIS_DIR}/genome
wget -nc <URL_to_my_genome.gtf.gz> -P ${ANALYSIS_DIR}/genome
gunzip ${ANALYSIS_DIR}/genome/*.gz
```

Your directory should look like this:

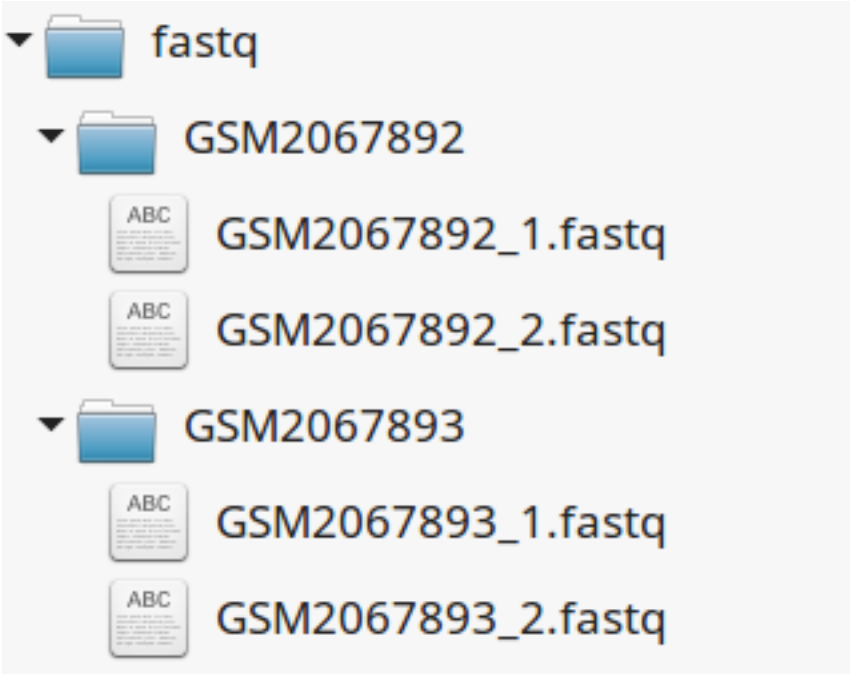


Fastq files organization

This tutorial assumes you dispose of your own fastq files. We recommend that your organize your samples in separate folders, and name both fastq files and their parent directories accordingly.

If you have paired-ends samples, they should be in the same directory and distinguished using a suffix of any sort.


fastq
GSM521934
GSM521934.fastq
GSM521935
GSM521935.fastq


fastq
GSM2067892
GSM2067892_1.fastq
GSM2067892_2.fastq
GSM2067893
GSM2067893_1.fastq
GSM2067893_2.fastq

Metadata

Running the workflows provided by the SnakeChunks library requires the use of three configuration files.

samples.tab

This file should contain, at least, one column named "ID", that should contain sample names matching those defined in the previous section. In the case of an RNA-seq analysis, it should also contain a column "Condition", which will define groups of comparison (see design file in the section below).

All the samples will be processed in the same manner. You can prevent certain samples from being processed by commenting the corresponding lines with a ";" at the beginning of the line.

RNA-seq sample groups should contain at least 2 samples.

You can add any other relevant information related to samples in other tab-separated columns.

```
1 ; Sample description file
2 ; Organism: Saccharomyces cerevisiae
3 ; Experiment type: ChIP-seq
4 ;
5 ID→Condition→ENA_experiment→Scan Name
6 GSM521934→input→SRX021359→SRR051930
7 GSM521935→chip→SRX021358→SRR051929
8
9
```

```
1 ; Experiment type: RNA-seq
2 ;
3 ID→Condition
4 GSM1010244→WT
5 GSM1010245→WT
6 GSM1010246→FNR
7 GSM1010247→FNR
```

design.tab

The purpose of this file is to determine which samples should be processed together. In a ChIP-seq analysis, it will be used to define which ChIP samples should be compared with which inputs. In an RNA-seq experiment, it defines the conditions to be compared against each other.

Column names should be respected.

```
1 ; Design for the peak-calling of ChIP-seq analysis
2 ;
3 treatment → control
4 GSM521935 → GSM521934
```

```
1 ; Experiment type: RNA-seq
2 ; Analysis: differential expression
3 ;
4 Reference → Test
5 WT → FNR
```

config.yml

You can find examples of configuration files in the examples section of the SnakeChunks directory.

Directories should be defined relative to the working directory defined in the beginning: genome, SnakeChunks, fastq, etc. Same goes for configuration files.

Genome filenames should be mentionned as they appear in the defined genome directory.

Genome size should be filled in, as well as the sequencing type: “se” for single-end data, and “pe” for paired-ends data. In the case of paired-ends data, suffixes (parameter “strands”) should be mentioned and should match the filenames (minus the “_”).

The minimum of configuration should look like this:

All the parameters related to the tools used are optional, and the default parameters of each program will be used when they’re not set in the configfile.

Running a workflow

If your directory now looks like this, you should be ready to run a worflow!

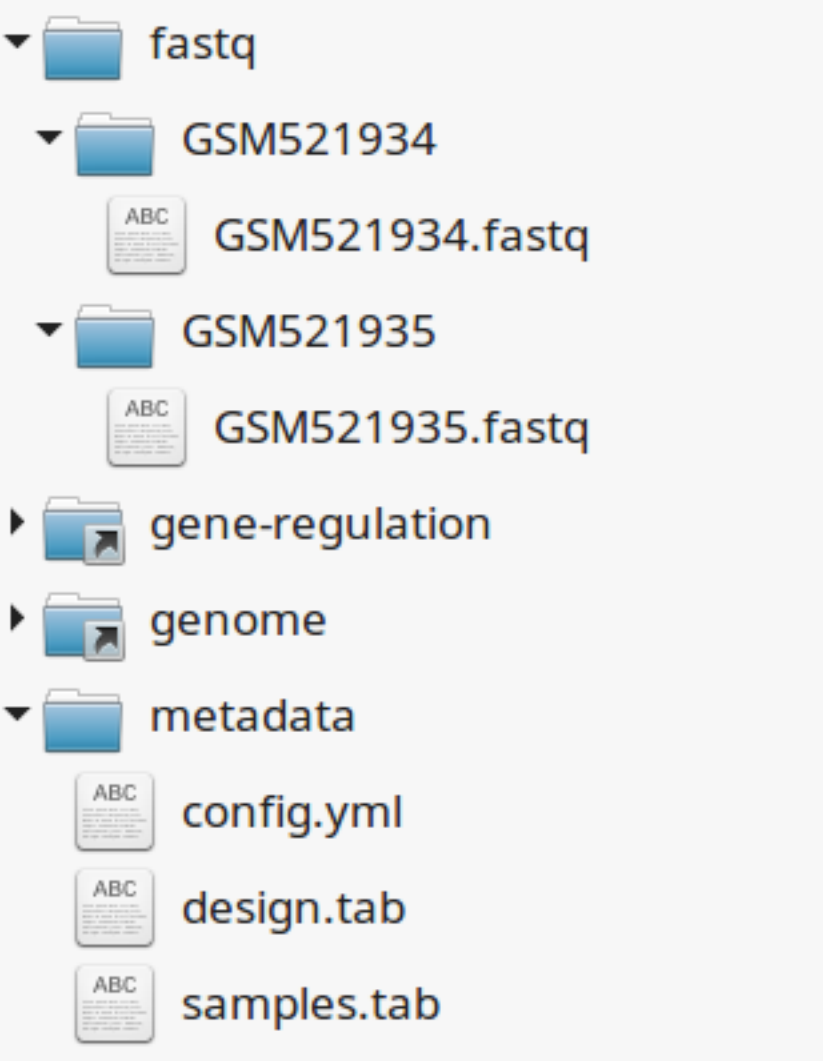
You can verify it by doing dry runs:

```
cd ${ANALYSIS_DIR}
# Run the quality check
snakemake -s SnakeChunks/scripts/snakefiles/workflows/quality_control.wf --config-
↪file metadata/config.yml -p -n
# Run the ChIP-seq workflow
snakemake -s SnakeChunks/scripts/snakefiles/workflows/ChIP-seq.wf --config-file␣
↪metadata/config.yml -p -n
# Run the RNA-seq workflow
snakemake -s SnakeChunks/scripts/snakefiles/workflows/RNA-seq.wf --config-file␣
↪metadata/config.yml -p -n
```

Just remove the *-n* option to actually run them.

```
  author: "Claire Rioualen"
  genome:
    organism: "Escherichia coli K12 MG1655"
    size: "4639221"
    fasta_file: "my_genome.fa"
    gff3_file: "my_genome.gff3"
    gtf_file: "my_genome.gtf"

  metadata:
    samples: "metadata/samples.tab"
    design: "metadata/design.tab"
    configfile: "metadata/config.yml"
    seq_type: "pe"
    strands: "1 2"

  dir:
    fastq: "fastq"
    genome: "genome"
    results: "results"
    gene_regulation: "gene-regulation"

  tools:
    trimming: "sickle"
    mapping: "bowtie2 subread-align"
    peakcalling: "homer macs2 spp"
```

```
  sickle:
    qual_threshold: "20"

  bowtie2:
    max_mismatches: "1"
    threads: "10"

  subread-align:
    threads: "10"
    seq_data: "0"
    max_mismatches: "3"
    align_options: "-d 0 -D 600"

  macs2:
    qval: "0.001"
    keep_dup: "all"
    mfold_min: "2"
    mfold_max: "50"
    other_options: "--call-summits"
```


fastq
GSM521934
GSM521934.fastq
GSM521935
GSM521935.fastq
gene-regulation
genome
metadata
config.yml
design.tab
samples.tab

3.5 Virtual environments

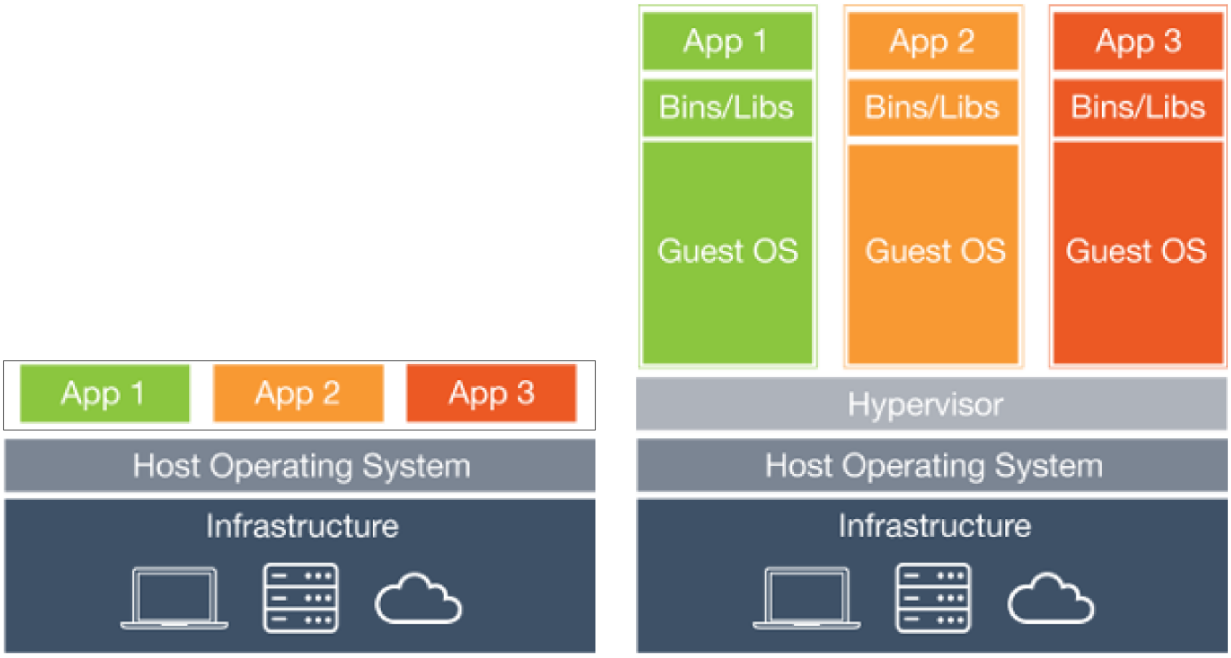
Tutorials on how to run SnakeChunks workflows in virtual environments or virtual machines (VM).

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.

Last update: 17/05/09

In computing, a **virtual machine** (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination. Source: Wikipedia

Diagram of a physical machine vs a virtual machine:



3.5.1 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:

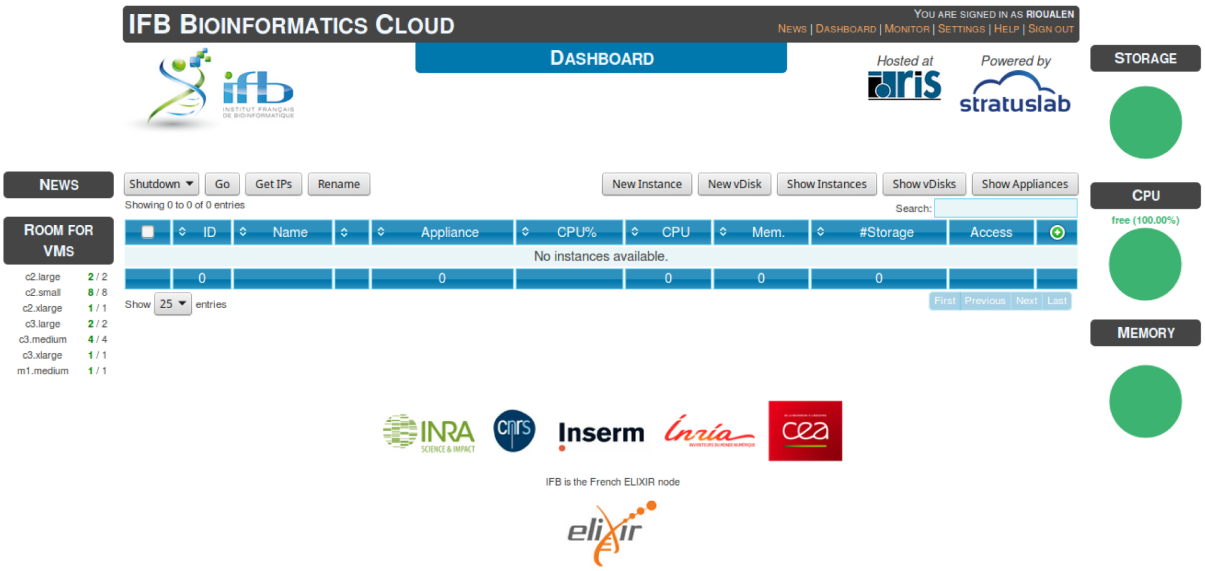
- The IFB
- IFB cloud
- Cloud usage
- Documentation

The first parts of this tutorial will explain you how to use the IFB cloud for general purposes.

For a specific use of the SnakeChunks appliance, you should refer yourself to this section.

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.



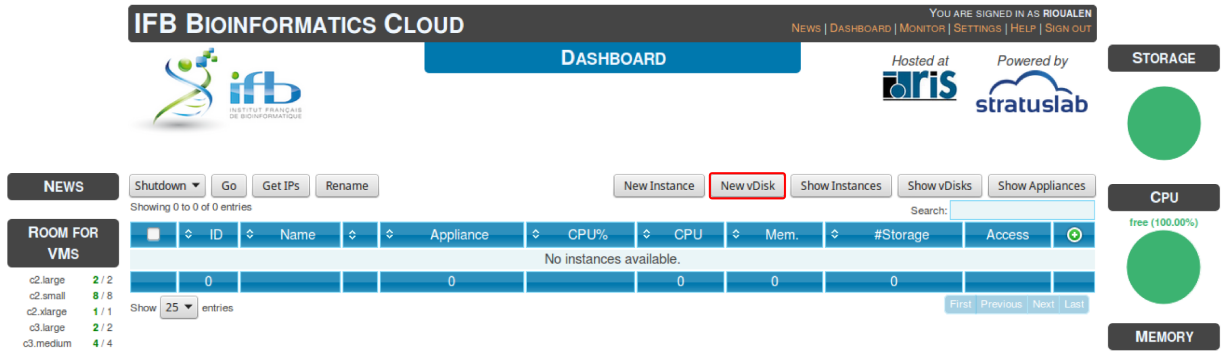
Virtual disk creation

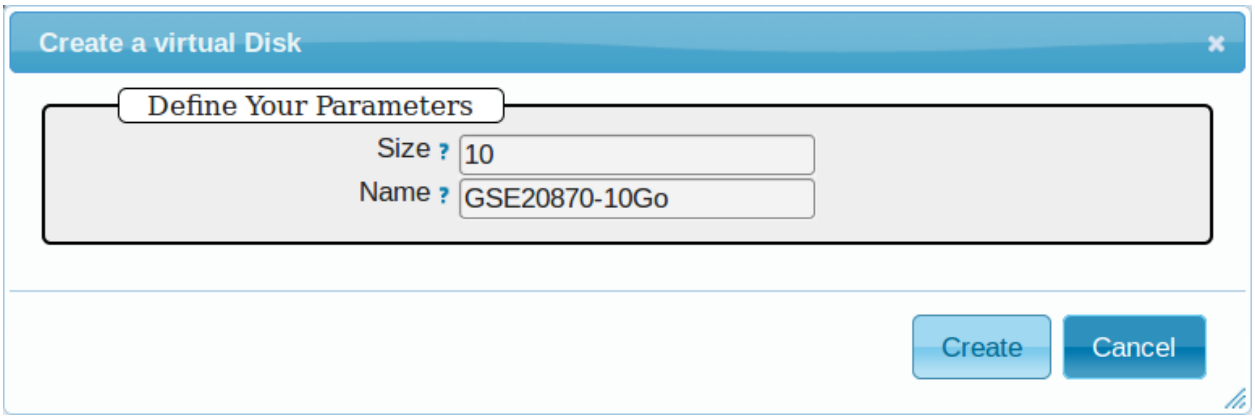
Appliances usually have a limited amount of disk space (up to 10 or 20Go). If the instance to be run necessitates disk space, you have to create a virtual disk (vDisk) prior to launching it. Depending on the type of account that you have, you'll have a certain amount of storage space available. This space can be divided into as many vDisks as you want.

When instantiating an appliance, you can chose to attach one of these vDisks to the virtual machine.

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

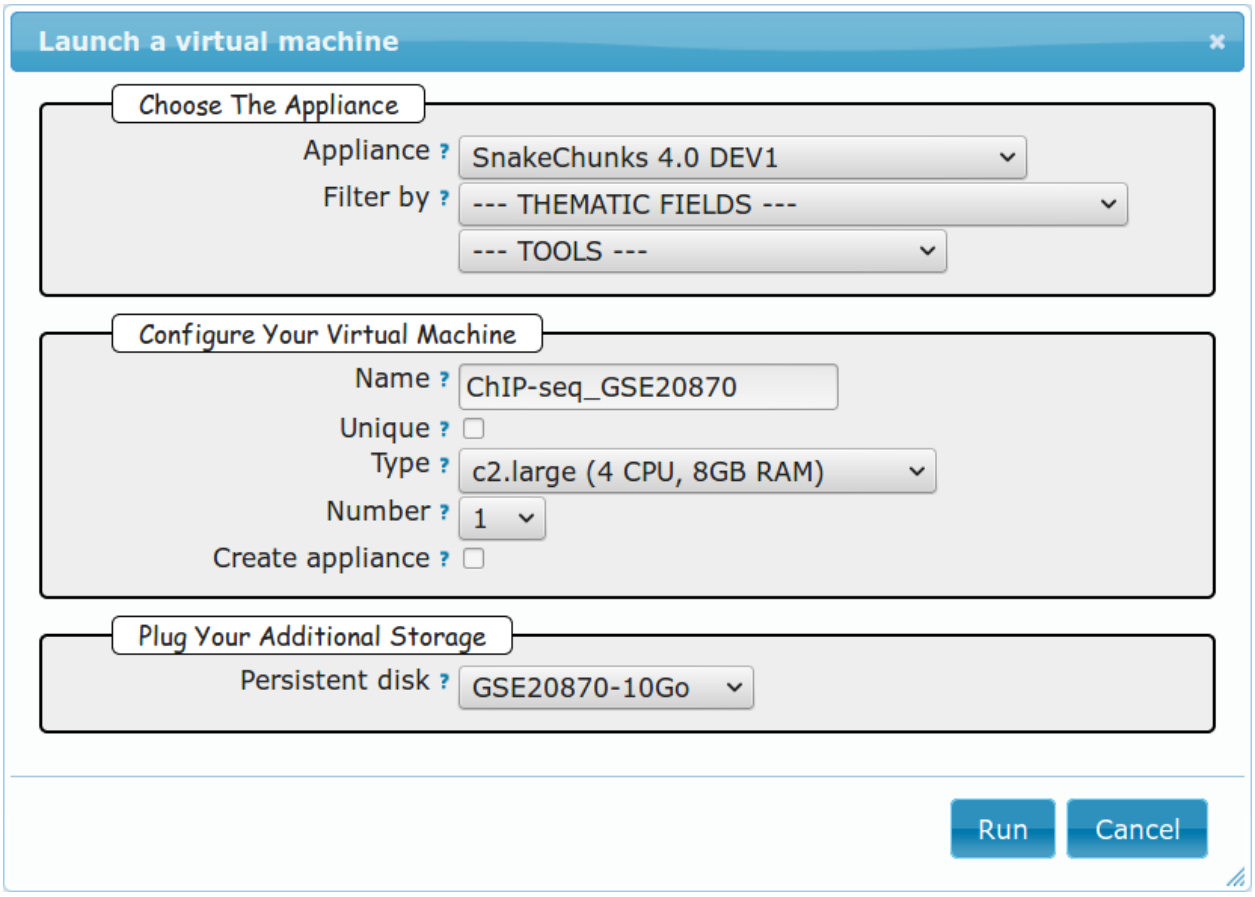
Creating an *instance* consists in *instanciating* an existing *appliance*. It creates a virtual environment that has the same contents as the appliance chosen, and that is accessible through ssh from your local host.

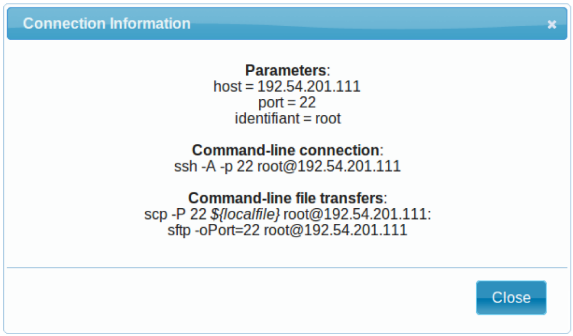
1. Click *New Instance* button.
2. Choose an appliance in the drop-down menu. You may use the filter menu in order to look for a specific tool.
3. Name your virtual machine.
4. Choose the amount of CPU and RAM to grant the VM.
5. Attach the vDisk.
6. Click *Run*.

change screencap

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.
9. Connect to your VM by commandline.

```
# Replace XXX by the IP of your instance
ssh -A -p 22 root@192.54.201.XXX
```

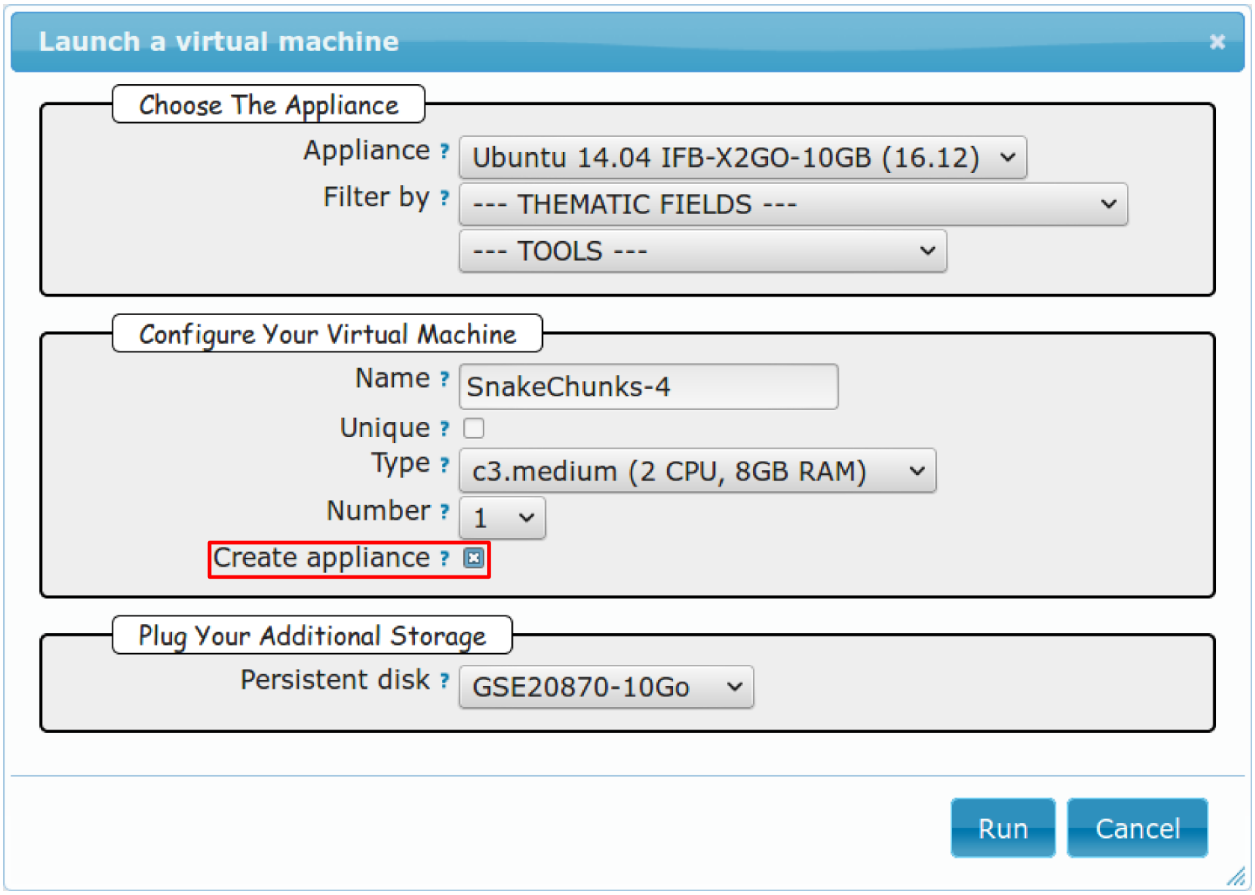

Launch a virtual machine
Choose The Appliance
Appliance ?
SnakeChunks 4.0 DEV1
Filter by ?
--- THEMATIC FIELDS ---
--- TOOLS ---
Configure Your Virtual Machine
Name ?
ChIP-seq_GSE20870
Unique ?
Type ?
c2.large (4 CPU, 8GB RAM)
Number ?
1
Create appliance ?
Plug Your Additional Storage
Persistent disk ?
GSE20870-10Go
Run
Cancel


Connection Information
Parameters:
host = 192.54.201.111
port = 22
identifiant = root
Command-line connection:
ssh -A -p 22 root@192.54.201.111
Command-line file transfers:
scp -P 22 ${localfile} root@192.54.201.111:
sftp -oPort=22 root@192.54.201.111
Close

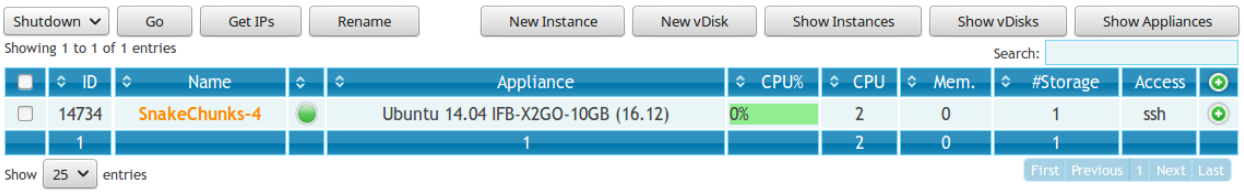
Creation of an appliance

Creating your own appliance can be as simple as instantiating an existing one. You just need to chose a "base" to build it on.

1. Click *New Instance* button.
2. Choose the appliance **Ubuntu 14.04 IFB (16-12)**.
3. Name your instance.
4. Check **Create appliance**.
5. Choose the amount of CPU and RAM to grant the VM.
6. Attach the vDisk.
7. Click *Run*.



8. Refresh the page. Your appliance 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.

```
# Replace XXX by the IP of your instance
ssh -A -p 22 root@192.54.201.XXX
```

Once you're connected to your appliance, you can install all the programs that you want. You can check out this section for a manual on how to install NGS tools. Beware that the amount of disk space of the appliance itself is limited!

Later on, you can ask an admin from the IFB cloud to register the appliance, in order for it to become available to other cloud users.

First connection to the instance

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:

```
# Replace XXX by the IP of your instance
scp -P 22 ${localfile} root@192.54.201.XXX:
sftp -oPort=22 root@192.54.201.XXX
```

Another way is to use rsync:

```
# Replace XXX by the IP of your instance
rsync -ruptvl ${localfile} root@192.54.201.XXX:/root/mydisk/
```

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).

Configuration

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
```

Using the SnakeChunks appliance

Requirements

User account creation & configuration

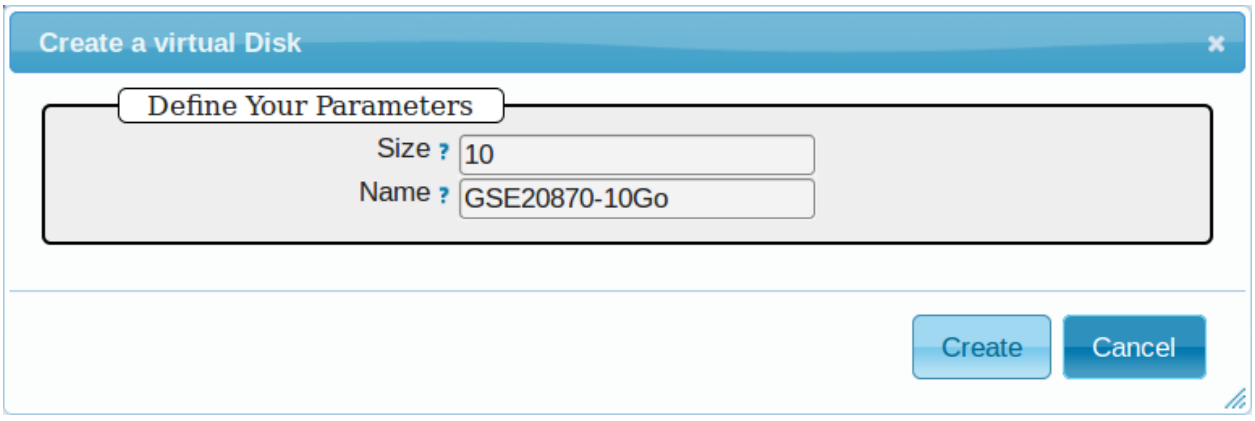
- Using the IFB cloud facilities requires to have a user account. Register here.
- Once your account has been validated, you can login.
- In order to be able to access your instances through SSH, you should register your SSH public key in your account settings, through the dashboard.

Virtual disk creation

Appliances usually have a limited amount of disk space (up to 10 or 20Go). If the instance to be run necessitates disk space, you have to create a virtual disk (vDisk) prior to launching it.

Check out this section for details.

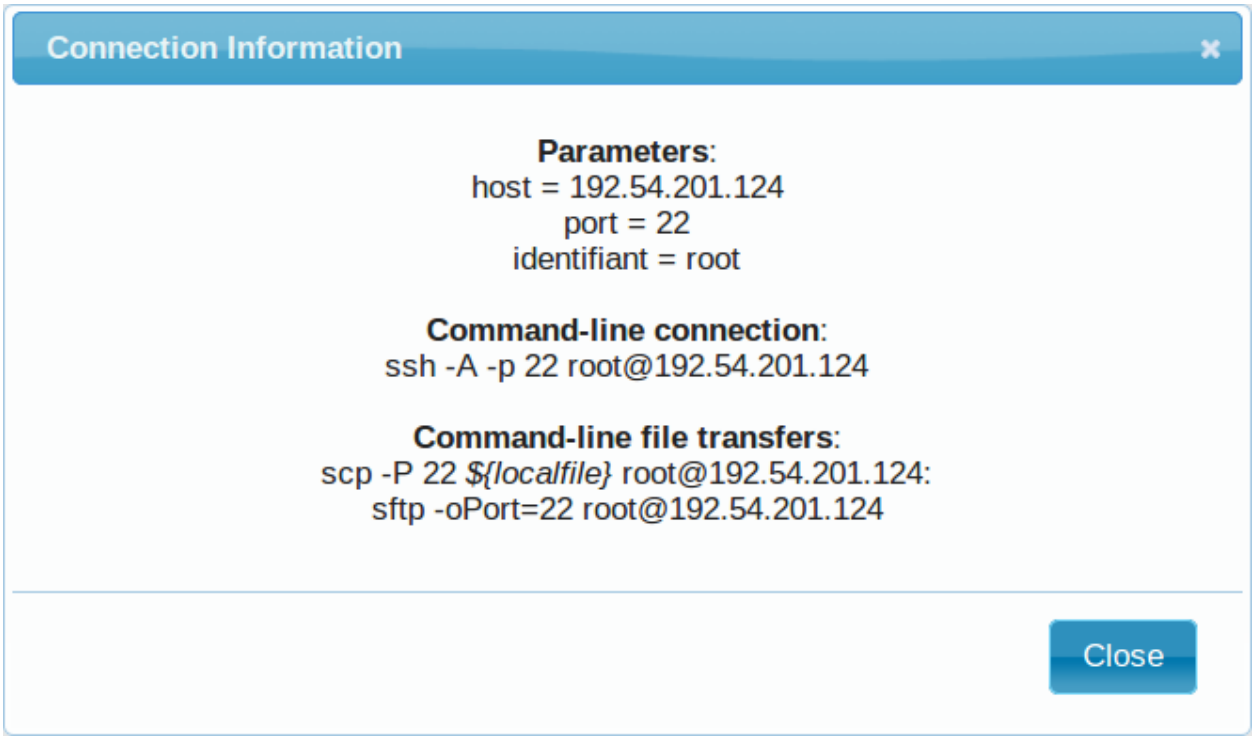
1. Click *New vDisk* button.
2. Enter a size (whole number equating to the amount of Go needed).
3. Name it (e.g. `GSE20870-10Gb`, the ID of the Gene Expression Omnibus series that will be stored on the virtual drive).



Creation of an instance

1. Click *New Instance* button.
2. Choose appliance **SnakeChunks 4.0** in the drop-down menu.

3. Name your VM.
4. Choose the amount of CPU and RAM to grant the VM.
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:

```
# Replace XXX by the IP of your instance
ssh -A -p 22 root@192.54.201.XXX
```

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 the files that will results from the execution of our workflow.

We also create a link to the SnakeChunks library.

```
ANALYSIS_DIR=${HOME}/mydisk/ChIP-seq_SE_GSE20870
mkdir -p ${ANALYSIS_DIR}
cd ${ANALYSIS_DIR}
ln -s ${HOME}/SnakeChunks SnakeChunks
```

Download data

The following commands will download the raw files from the GEO database, and create the folders to organize them properly.

```
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

The following commands will download the required genome files in a specific directory:

- fasta file of the reference genome
- gff3 annotation file
- gtf annotation file

```
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
```

Your file organization should now look like this:

```
root@vm0026:~/mydisk/ChIP-seq_SE_GSE20870# ls -lR
.:
total 8
lrwxrwxrwx 1 root root   21 Jul  4 11:02 SnakeChunks -> /root/SnakeChunks-4.0
drwxr-xr-x 4 root root 4096 Jul  4 11:02 data
drwxr-xr-x 2 root root 4096 Jul  4 11:03 genome

./data:
total 8
drwxr-xr-x 2 root root 4096 Jul  4 11:02 GSM521934
drwxr-xr-x 2 root root 4096 Jul  4 11:02 GSM521935

./data/GSM521934:
total 81184
-rw-r--r-- 1 root root 83043426 Jul  4 11:02 SRR051929.sra

./data/GSM521935:
total 94116
-rw-r--r-- 1 root root 96275406 Jul  4 11:03 SRR051930.sra

./genome:
total 29644
-rw-r--r-- 1 root root 12360636 Jul  4 11:03 Saccharomyces_cerevisiae.R64-1-1.30.dna.genome.fa
-rw-r--r-- 1 root root  6673011 Jul  4 11:03 Saccharomyces_cerevisiae.R64-1-1.30.gff3
-rw-r--r-- 1 root root 11268860 Jul  4 11:03 Saccharomyces_cerevisiae.R64-1-1.30.gtf
```

Run the workflow

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

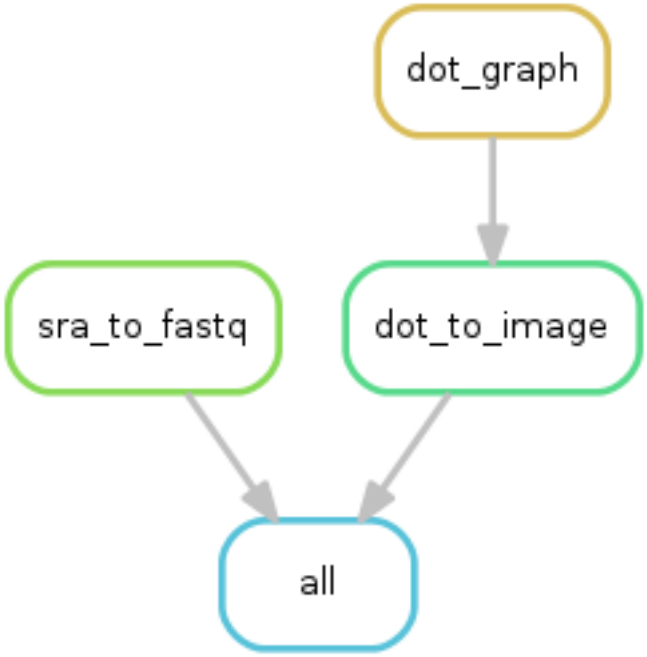
```
cd  ${ANALYSIS_DIR}
snakemake -p -s SnakeChunks/scripts/snakefiles/workflows/import_from_sra.wf --
↪configfile SnakeChunks/examples/ChIP-seq_SE_GSE20870/config.yml -n
```

If there is no error, you can procede with the analysis:

```
# This workflow extracts .fastq files from the .sra archives
snakemake -p -s SnakeChunks/scripts/snakefiles/workflows/import_from_sra.wf --
↪configfile SnakeChunks/examples/ChIP-seq_SE_GSE20870/config.yml
# This workflow performs quality check and trimming on the raw data
snakemake -p -s SnakeChunks/scripts/snakefiles/workflows/quality_control.wf --
↪configfile SnakeChunks/examples/ChIP-seq_SE_GSE20870/config.yml
# This workflow perform a classic ChIP-seq analysis, including mapping, peak-calling␣
↪and motif search, using 4 CPU
snakemake -p -j 4 -s SnakeChunks/scripts/snakefiles/workflows/ChIP-seq.wf --
↪configfile SnakeChunks/examples/ChIP-seq_SE_GSE20870/config.yml
```

Using 4CPU & 8Go of RAM, the workflows take ~30mn to complete.

Congratulations! You just executed these wonderful workflows:



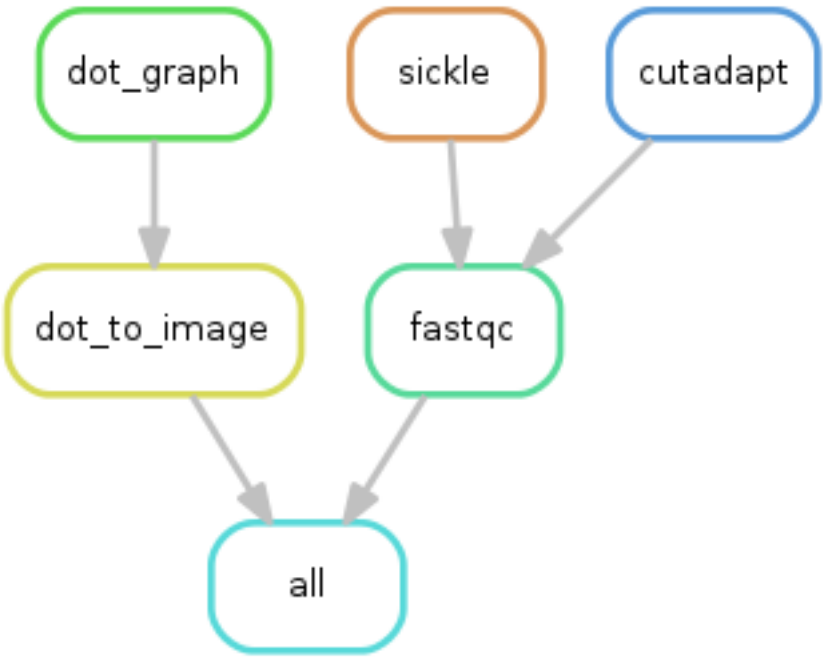
Visualizing results

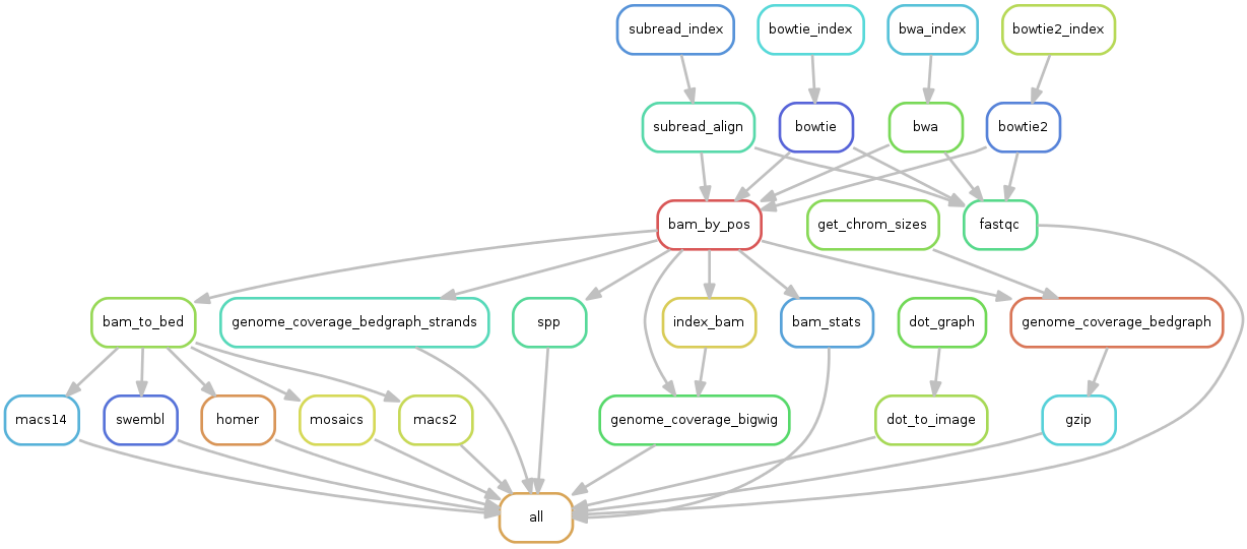
Install and run the X2Go client on your host computer

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.

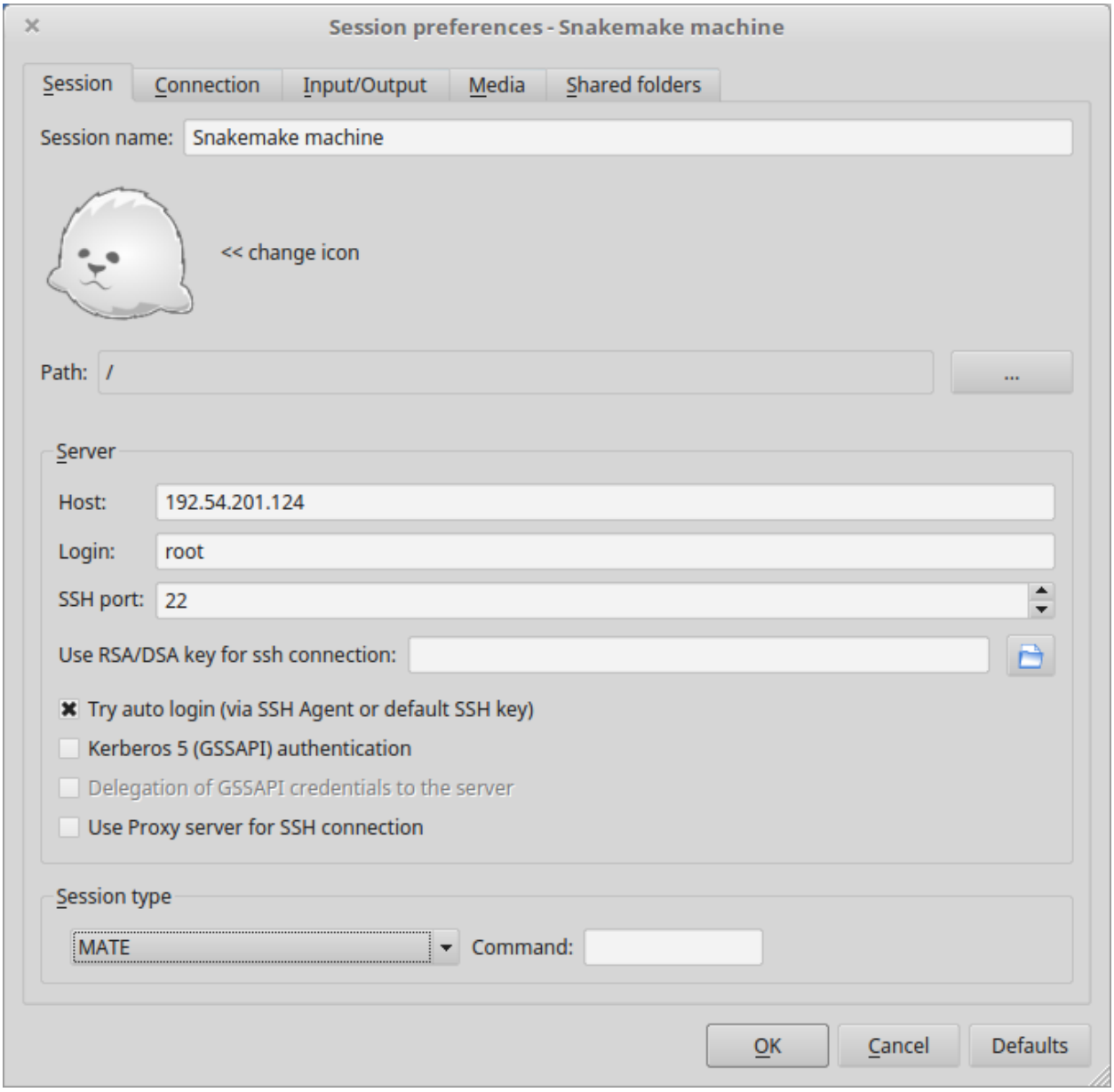
1. Install the x2go client and launch it from your local computer.

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


dot_graph
sickle
cutadapt
dot_to_image
fastqc
all


subread_index
bowtie_index
bwa_index
bowtie2_index
subread_align
bowtie
bwa
bowtie2
bam_by_pos
get_chrom_sizes
fastqc
bam_to_bed
genome_coverage_bedgraph_strands
spp
index_bam
bam_stats
dot_graph
genome_coverage_bedgraph
macs14
swembl
homer
mosaics
macs2
genome_coverage_bigwig
dot_to_image
gzip
all

2. Create a new session using the Mate desktop.



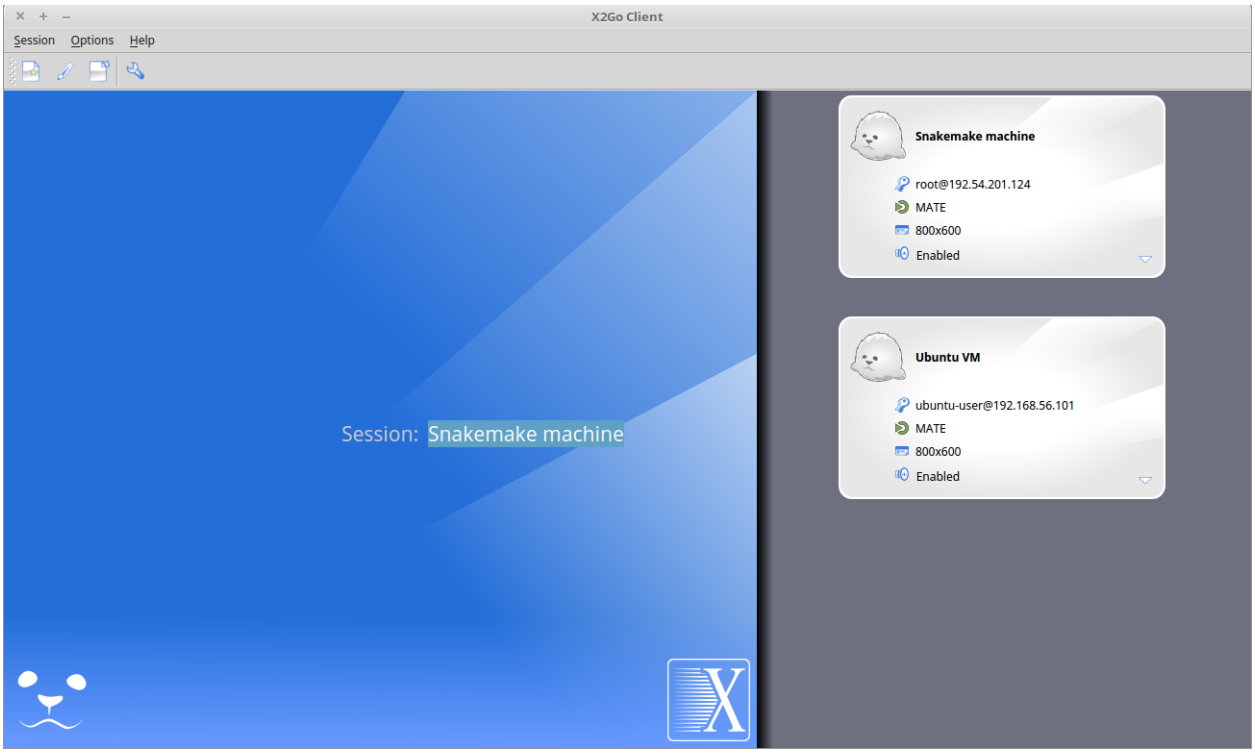
3. The session now appears on the right panel. Just click it to lauch it!
4. You should be now on the virtual desktop!

Note: you may need to change your keyboard settings

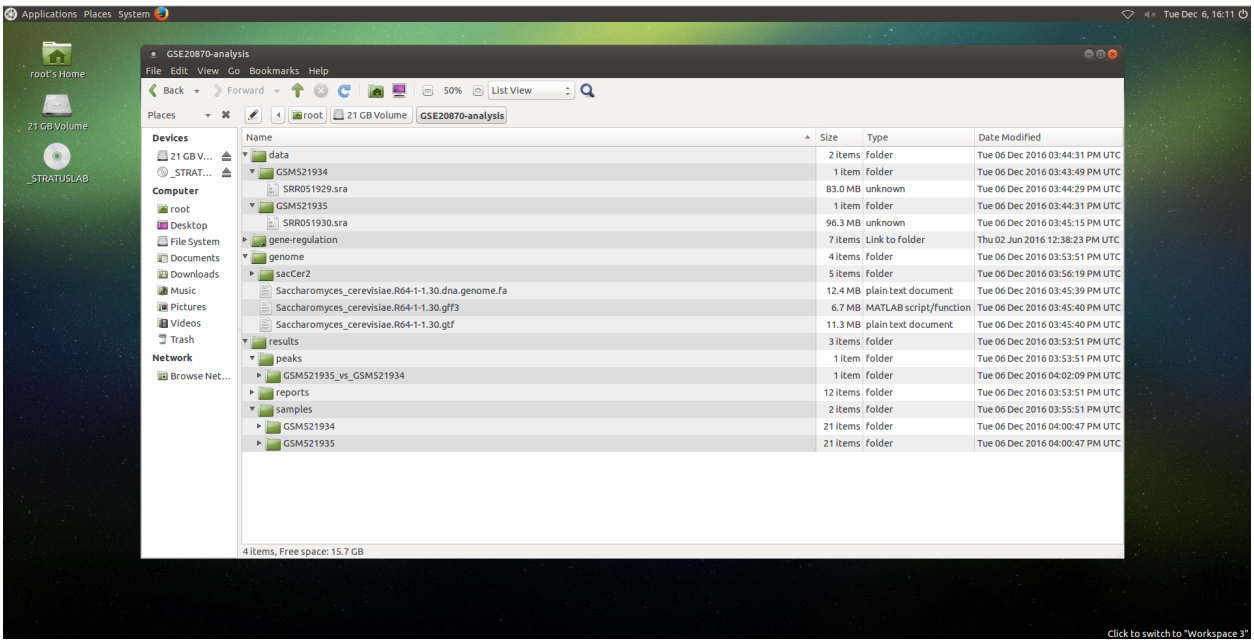
- Go to **System > Preferences > Keybords**
- Click on tab **Layouts**
- Add and/or remove desired keyboards

Visualize results

The result files should be organized like this:


X2Go Client
Session Options Help
Session: Snakemake machine
Snakemake machine
root@192.54.201.124
MATE
800x600
Enabled
Ubuntu VM
ubuntu-user@192.168.56.101
MATE
800x600
Enabled


session-12-04-IFB
Applications Places System
Tue Apr 12, 10:36
Accessories
Education
Graphics
Internet
Office
Programming
Sound & Video
System Tools
Universal Access
Caja
dconf Editor
Disk Usage Analyzer
Dolphin
GDebi Package Installer
K3b
KDE Partition Manager
Konsole
KSysGuard
KSystemLog
KWalletManager
Log File Viewer
MATE System Monitor
MATE Terminal
Muon Discover
Muon Update Manager
Nepomuk Cleaner
Oracle Java 9 Console
Oracle Java 9 VisualVM
Power Statistics
Startup Disk Creator
UXTerm
XTerm
Use the command line
ubuntu MATE



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/ChIP-seq_SE_GSE20870/fastq/GSM521934/GSM521934_fastqc/
↪GSM521934_fastqc.html
firefox /root/mydisk/ChIP-seq_SE_GSE20870/fastq/GSM521935/GSM521935_fastqc/
↪GSM521935_fastqc.html
```

- After trimming with Sickle:

```
firefox /root/mydisk/ChIP-seq_SE_GSE20870/fastq/GSM521934/GSM521934_sickle-se-q20_
↪fastqc/GSM521934_sickle-se-q20_fastqc.html
firefox /root/mydisk/ChIP-seq_SE_GSE20870/fastq/GSM521935/GSM521935_sickle-se-q20_
↪fastqc/GSM521935_sickle-se-q20_fastqc.html
```

IGV

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

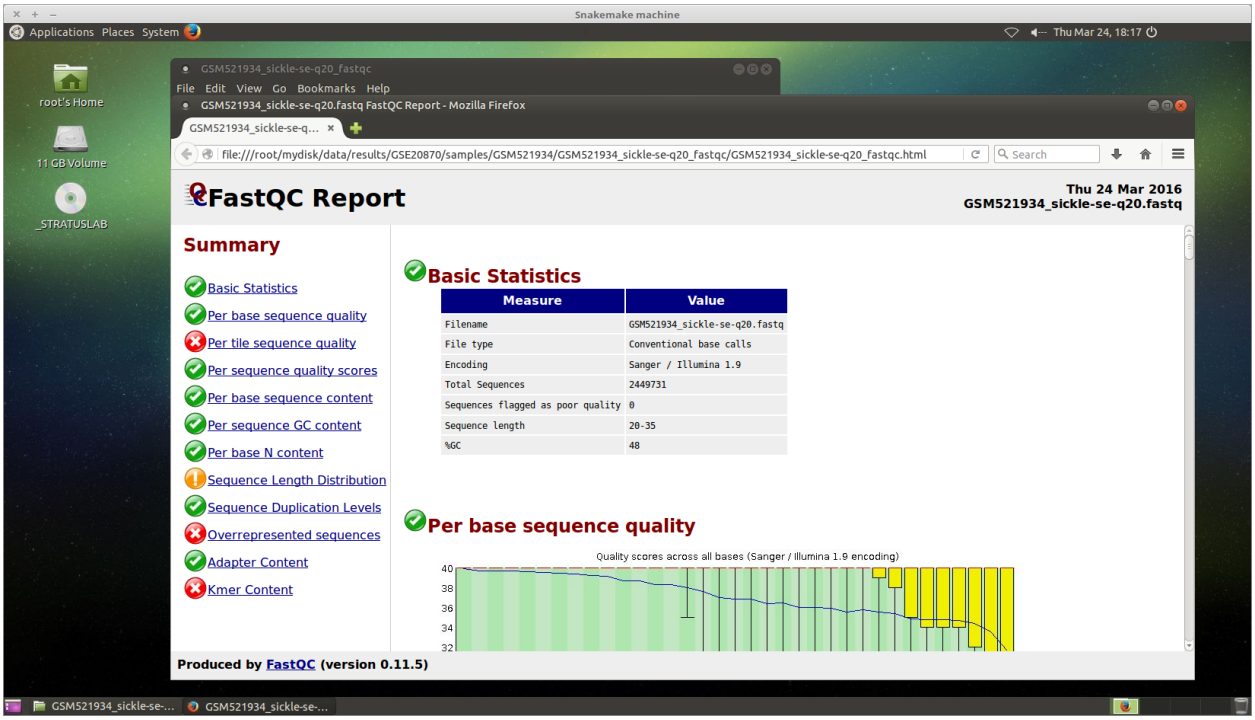
```
igv
```

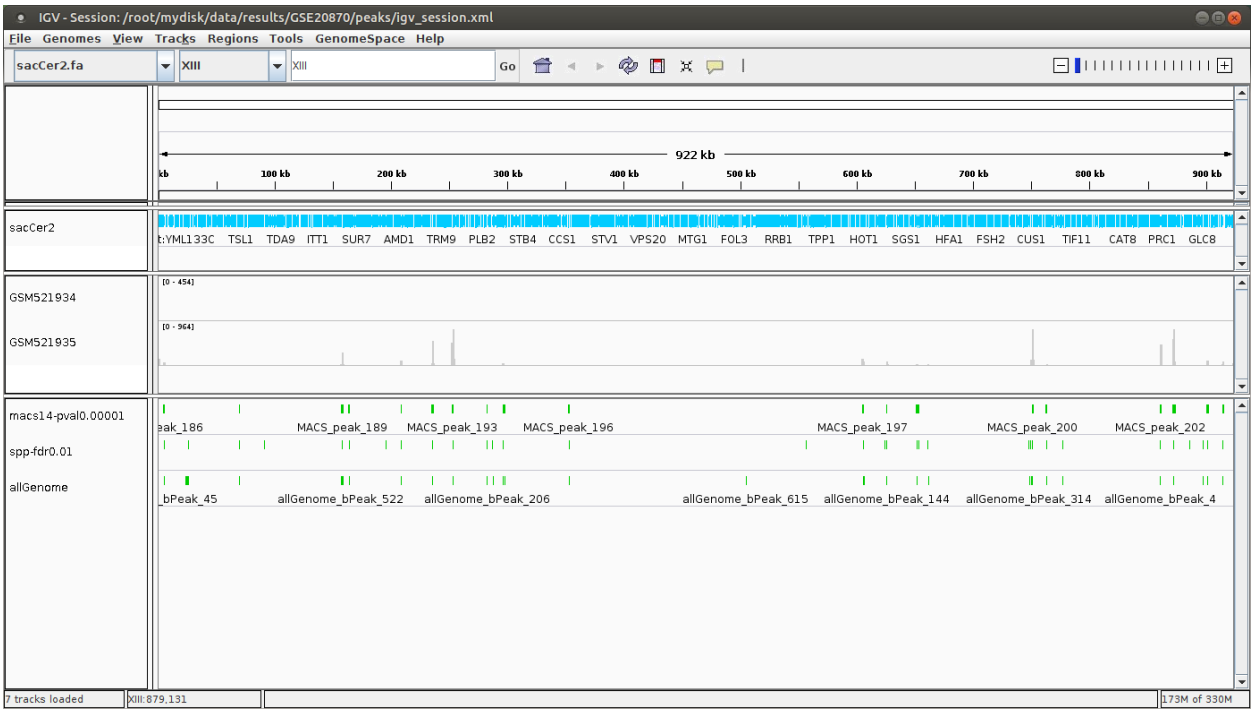
- Click "File" > "Open session..." and chose the file `/root/mydisk/ChIP-seq_SE_GSE20870/reports/peaks/igv_session.xml`.
- You may need to adjust the panel sizes.

Create your own SnakeChunks appliance

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

If you want to manipulate data, you should also create a vDisk following these instructions.

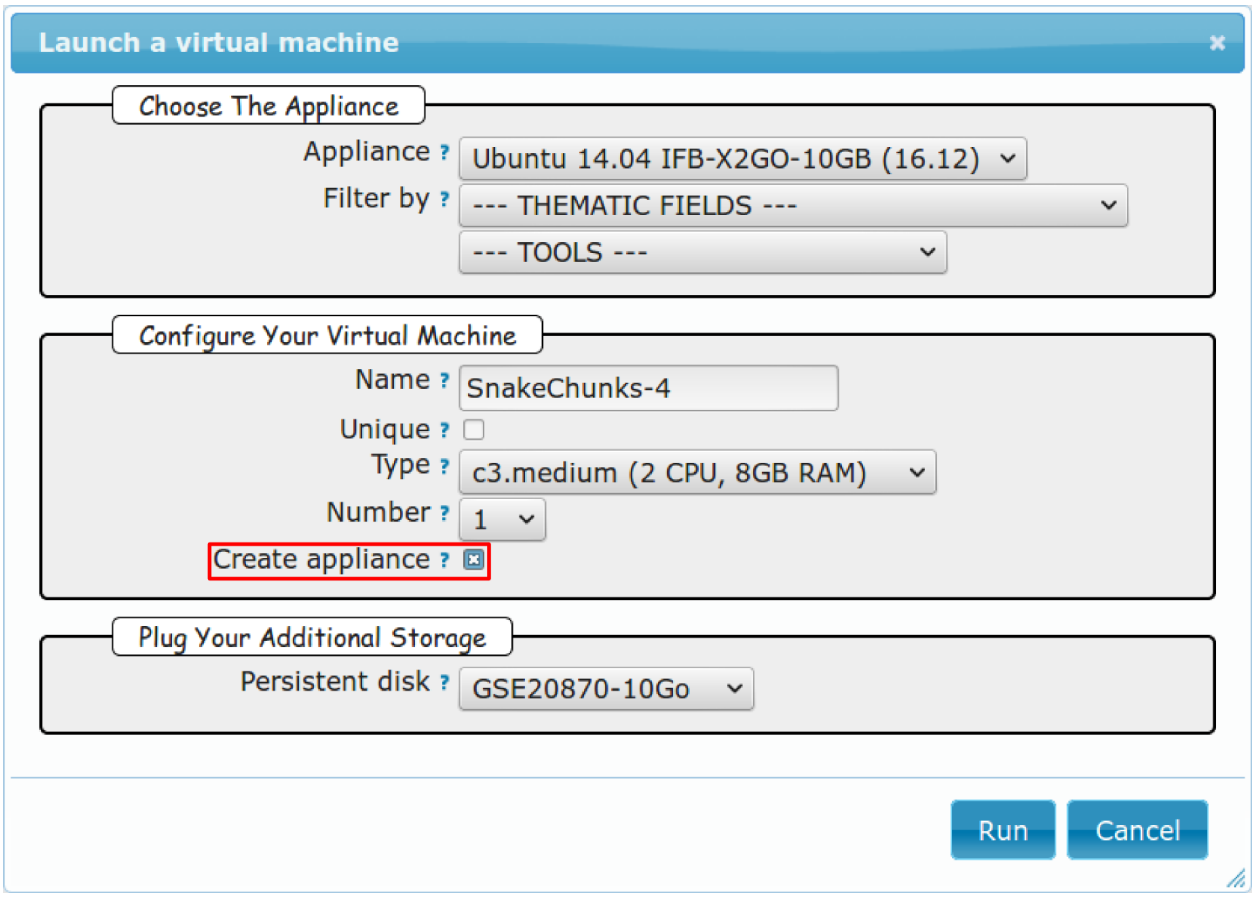

Snakemake machine
Applications Places System
Thu Mar 24, 18:17
root's Home
11 GB Volume
_STRATUSLAB
GSM521934_sickle-se-q20_fastqc
File Edit View Go Bookmarks Help
GSM521934_sickle-se-q20.fastq FastQC Report - Mozilla Firefox
file:///root/mydisk/data/results/GSE20870/samples/GSM521934/GSM521934_sickle-se-q20_fastqc/GSM521934_sickle-se-q20_fastqc.html
FastQC Report
Thu 24 Mar 2016
GSM521934_sickle-se-q20.fastq
Summary
Basic Statistics
Per base sequence quality
Per tile sequence quality
Per sequence quality scores
Per base sequence content
Per sequence GC content
Per base N content
Sequence Length Distribution
Sequence Duplication Levels
Overrepresented sequences
Adapter Content
Kmer Content
Basic Statistics
Measure
Value
Filename
GSM521934_sickle-se-q20.fastq
File type
Conventional base calls
Encoding
Sanger / Illumina 1.9
Total Sequences
2449731
Sequences flagged as poor quality
0
Sequence length
20-35
%GC
48
Per base sequence quality
Quality scores across all bases (Sanger / Illumina 1.9 encoding)
40
38
36
34
32
Produced by FastQC (version 0.11.5)


IGV - Session: /root/mydisk/data/results/GSE20870/peaks/igv_session.xml
File Genomes View Tracks Regions Tools GenomeSpace Help
sacCer2.fa
XIII
Go
922 kb
sacCer2
GSM521934
GSM521935
macs14-pval0.00001
spp-fdr0.01
allGenome
MACS_peak_189
MACS_peak_193
MACS_peak_196
MACS_peak_197
MACS_peak_200
MACS_peak_202
allGenome_bPeak_522
allGenome_bPeak_206
allGenome_bPeak_615
allGenome_bPeak_144
allGenome_bPeak_314
7 tracks loaded

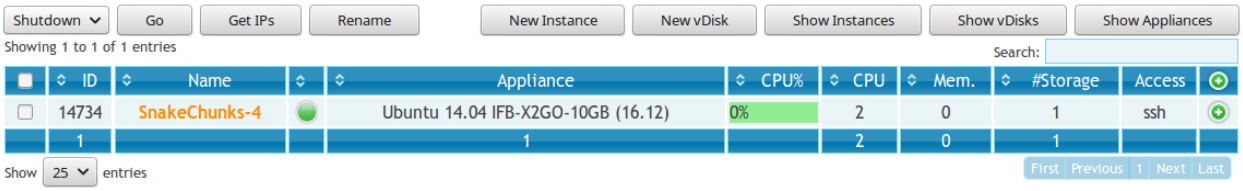
Creation of an *appliance*

When creating a new instance, choose a 10Go 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.
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 previous sections.

Get the SnakeChunks repository

```
wget -nc https://github.com/SnakeChunks/SnakeChunks/archive/4.0.tar.gz
tar zvxf 4.0.tar.gz
```

Run makefile to install the dependencies

The SnakeChunks library contains a makefile that installs most of the dependencies required to execute the snakemake workflows. You can also install tools manually, following these instructions.

The execution of the makefile may take a while (up to 30mn-1h), mostly because of the python libraries that are necessary to several NGS tools.

Then you should source the `.bashrc` in order to update the `$PATH` accordingly.

```
make -f SnakeChunks-4.0/scripts/makefiles/install_tools_and_libs.mk all
source ~/.bashrc
```

If you want to install the x2go server on the VM for visualization purposes, as shown here, you can also execute this rule:

```
make -f SnakeChunks-4.0/scripts/makefiles/install_tools_and_libs.mk add_repos desktop_
↪and_x2go
```

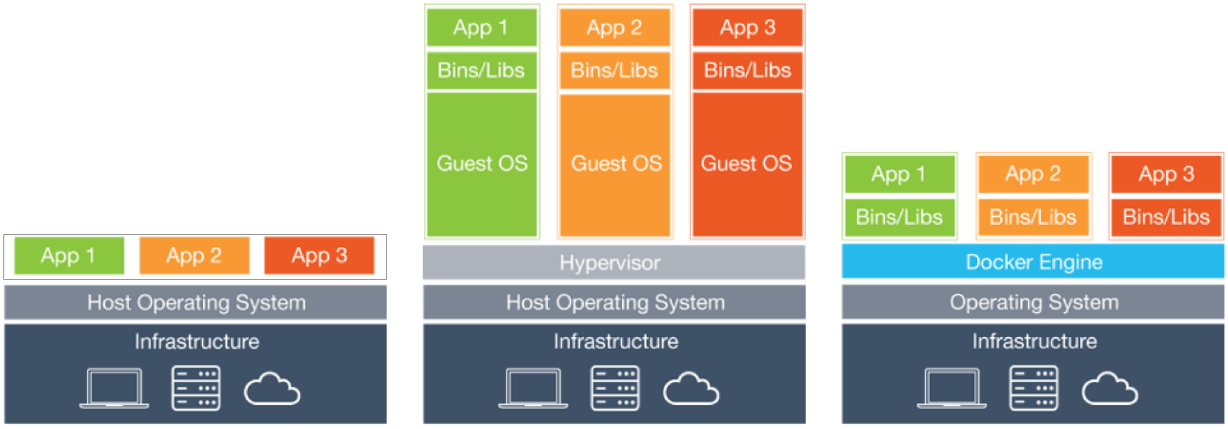
You should now be able to execute the example workflow by following instructions from here.

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

3.5.2 Docker

Docker is an open-source project that automates the deployment of applications inside software containers. Source: Wikipedia.

Diagram of Docker containers compared to physical and virtual machines.



Get started with Docker!

Create a Docker account

Instructions here (linux users).

Install Docker on your local host (Linux)

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
```

```
ubuntu-user@ubuntu-VM:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from hello-world

3f12c794407e: Pull complete
975b84d108f1: Pull complete
Digest: sha256:8be990ef2aeb16dbcb9271ddfe2610fa6658d13f6dfb8bc72074cc1ca36966a7
Status: Downloaded newer image for hello-world:latest

Hello from Docker.
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker Hub account:
 https://hub.docker.com

For more examples and ideas, visit:
 https://docs.docker.com/userguide/

ubuntu-user@ubuntu-VM:~$
```

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

```
setxkbmap fr
```

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
```

SnakeChunks with Docker

Create shared repositories and download source data

In order to execute the study case GSE55357, you should enter the following commands:

```
export ANALYSIS_DIR=~/ChIP-seq_GSE55357
mkdir $ANALYSIS_DIR
cd $ANALYSIS_DIR
```

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
```

Download ChIP-seq data

```
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX476/SRX476133/SRR1176905/SRR1176905.sra -P ${ANALYSIS_DIR}/data/GSM1334674
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX476/SRX476135/SRR1176907/SRR1176907.sra -P ${ANALYSIS_DIR}/data/GSM1334676
```

(continues on next page)

(continued from previous page)

```
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX476/SRX476136/SRR1176908/SRR1176908.sra -P ${ANALYSIS_DIR}/data/GSM1334679
wget --no-clobber ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByExp/sra/
↪SRX/SRX476/SRX476138/SRR1176910/SRR1176910.sra -P ${ANALYSIS_DIR}/data/GSM1334677
```

Fetch the Docker image and run it with shared folders

```
docker pull snakechunks/snakechunks:4.0
docker run -v $ANALYSIS_DIR:$HOME/ChIP-seq_GSE55357 -it snakechunks/snakechunks:4.0 /
↪bin/bash

docker pull snakechunks/snakechunks:latest
docker run -v $ANALYSIS_DIR:$HOME/ChIP-seq_GSE55357 -it snakechunks/
↪snakechunks:latest /bin/bash
```

docker run -v $ANALYSIS_DIR:$ANALYSIS_DIR -e ANALYSIS_DIR=$ANALYSIS_DIR -e HOME=$HOME -it snakechunks/snakechunks:latest /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 SnakeChunks/scripts/snakefiles/workflows/ChIP-seq.wf --configfile␣
↪SnakeChunks/examples/ChIP-seq_GSE55357/config.yml
```

3.5.3 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 14.04.5, latest long-term supported version.

```
wget http://releases.ubuntu.com/14.04/ubuntu-14.04.5-desktop-amd64.iso
```

Virtual Box configuration

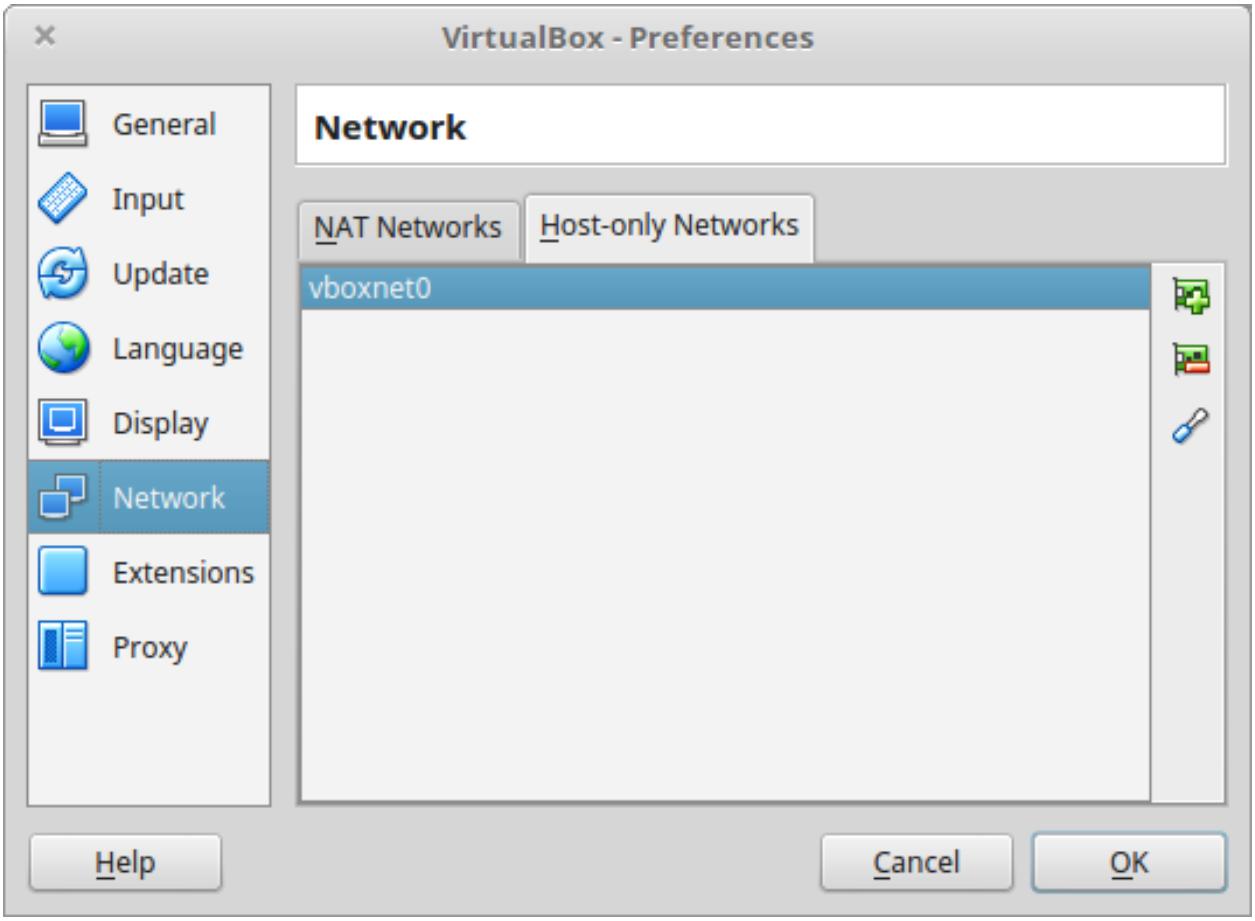
Before configuring the virtual machine, we need to tell VirtualBox how it will enable your local virtual machines to interact with their host (the operating system of the machine on which the VM is running).

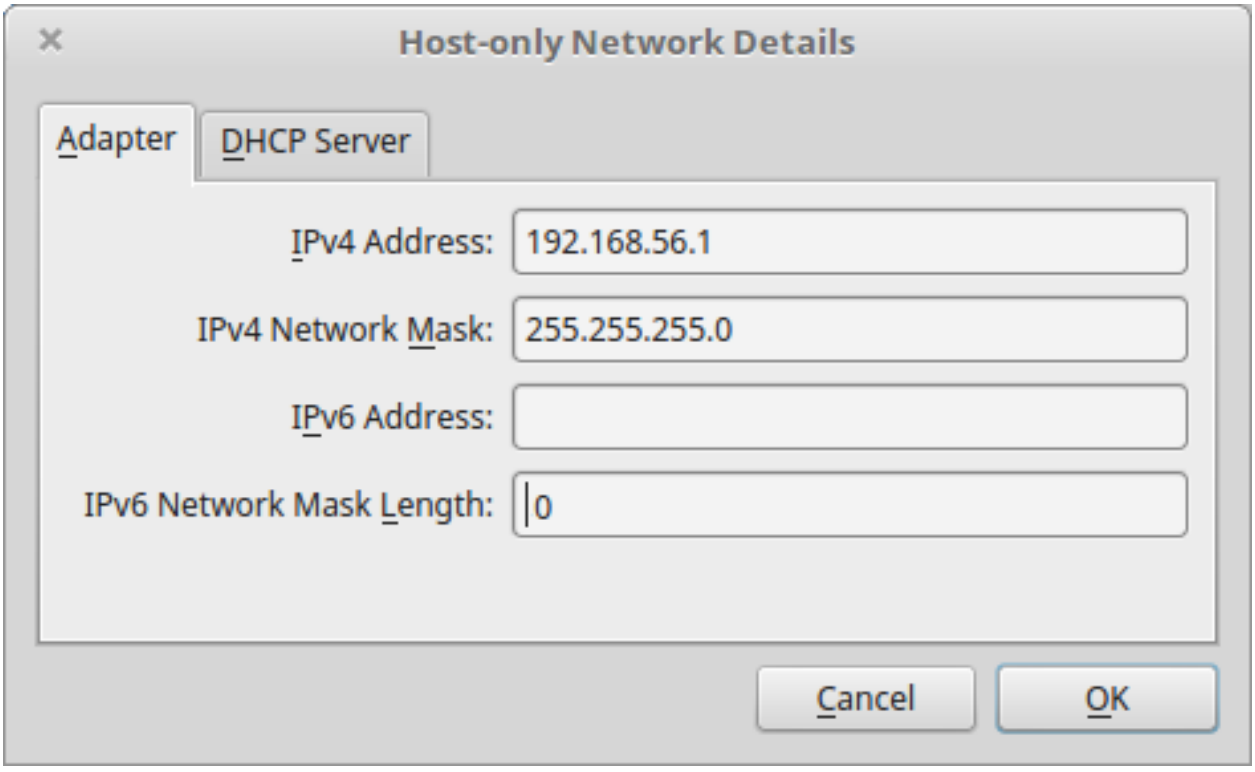
1. Open *VirtualBox* > *File* > *Preferences...*
2. Open the tab *Network* > *Host-only Networks*
    - click on the "+" icon
    - this creates a network vboxnet0. Select this network, click on the screw driver icon (*edit host-only network*), and set the following options:
    - *Adapter* tab
        - 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

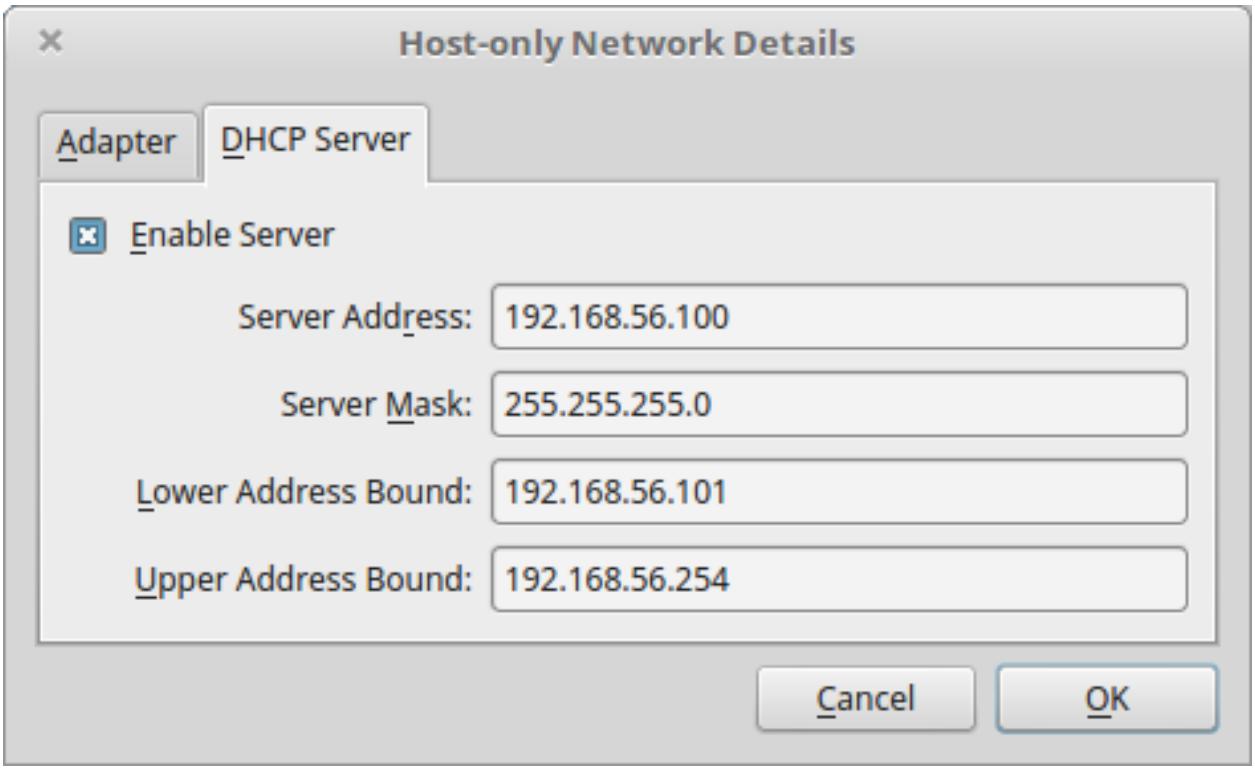
Creation of the virtual machine

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


VirtualBox - Preferences
General
Input
Update
Language
Display
Network
Extensions
Proxy
Network
NAT Networks
Host-only Networks
vboxnet0
Help
Cancel
OK


Host-only Network Details
Adapter
DHCP Server
IPv4 Address:
192.168.56.1
IPv4 Network Mask:
255.255.255.0
IPv6 Address:
IPv6 Network Mask Length:
0
Cancel
OK



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*
- Set *Drag'n Drop* 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 above)
- 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/4.0.tar.gz
tar zvxf 4.0.tar.gz
```

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
ln -s gene-regulation-4.0 gene-regulation
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}
```

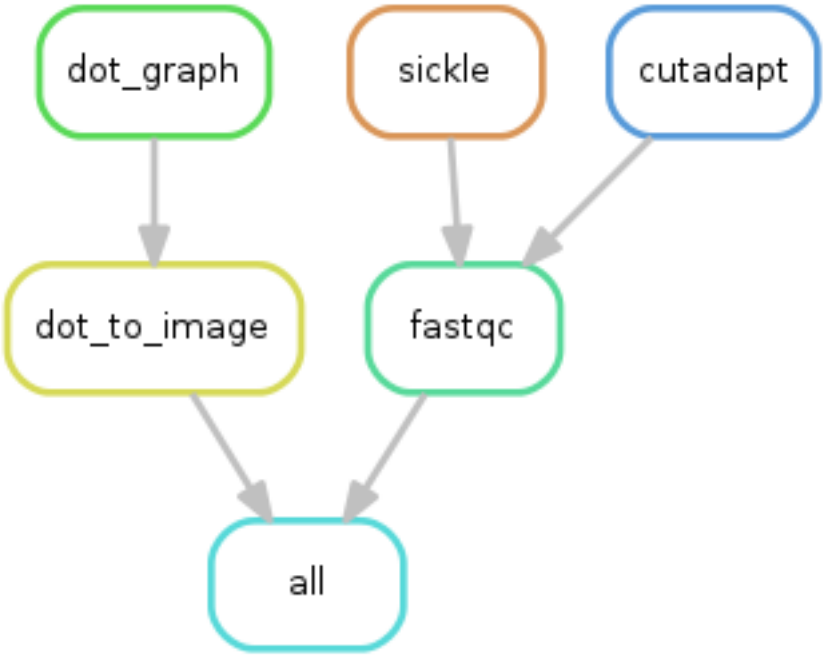
```
## Download source data

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 --configfile gene-regulation/examples/ChIP-seq_SE_GSE20870/config.yml -s␣
↪gene-regulation/scripts/snakefiles/workflows/import_from_sra.wf
snakemake -p --configfile gene-regulation/examples/ChIP-seq_SE_GSE20870/config.yml -s␣
↪gene-regulation/scripts/snakefiles/workflows/quality_control.wf
snakemake -p --configfile gene-regulation/examples/ChIP-seq_SE_GSE20870/config.yml -s␣
↪gene-regulation/scripts/snakefiles/workflows/ChIP-seq.wf
```

Congratulations! You just executed these wonderful workflows:

../img/import_to_fastq_rulegraph.png

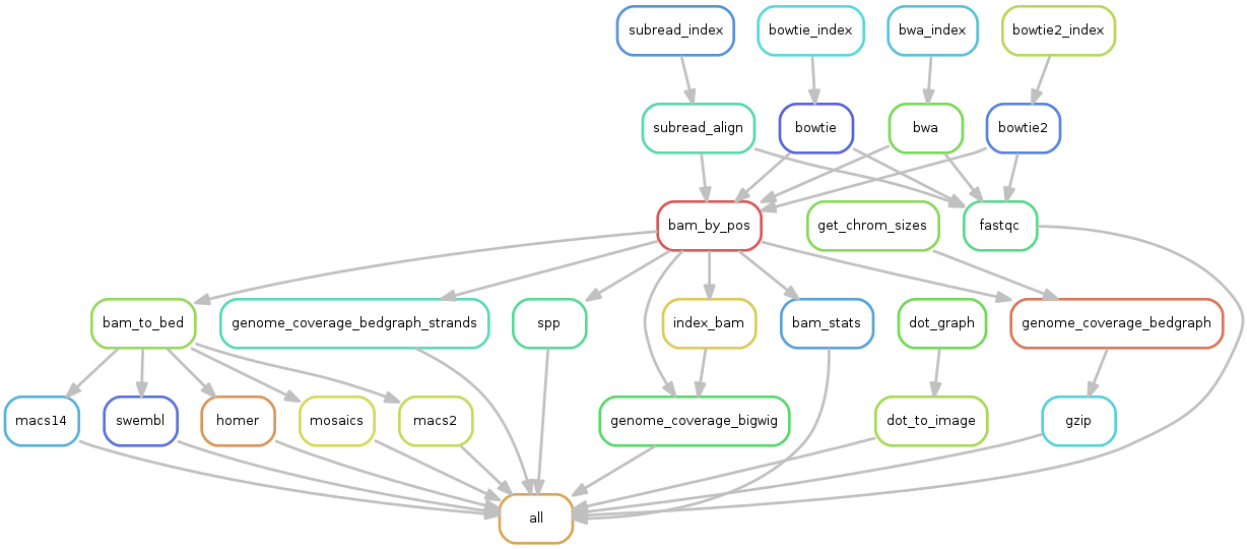


Visualizing results

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 ~/ChIP-seq_SE_GSE20870/fastq/GSM521934/GSM521934_fastqc/GSM521934_fastqc.
↪html
firefox ~/ChIP-seq_SE_GSE20870/fastq/GSM521935/GSM521935_fastqc/GSM521935_fastqc.
↪html
```

- After trimming:

```
firefox ~/ChIP-seq_SE_GSE20870/fastq/GSM521934/GSM521934_sickle-se-q20_fastqc/
↪GSM521934_sickle-se-q20_fastqc.html
firefox ~/ChIP-seq_SE_GSE20870/fastq/GSM521935/GSM521935_sickle-se-q20_fastqc/
↪GSM521935_sickle-se-q20_fastqc.html
```

IGV

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

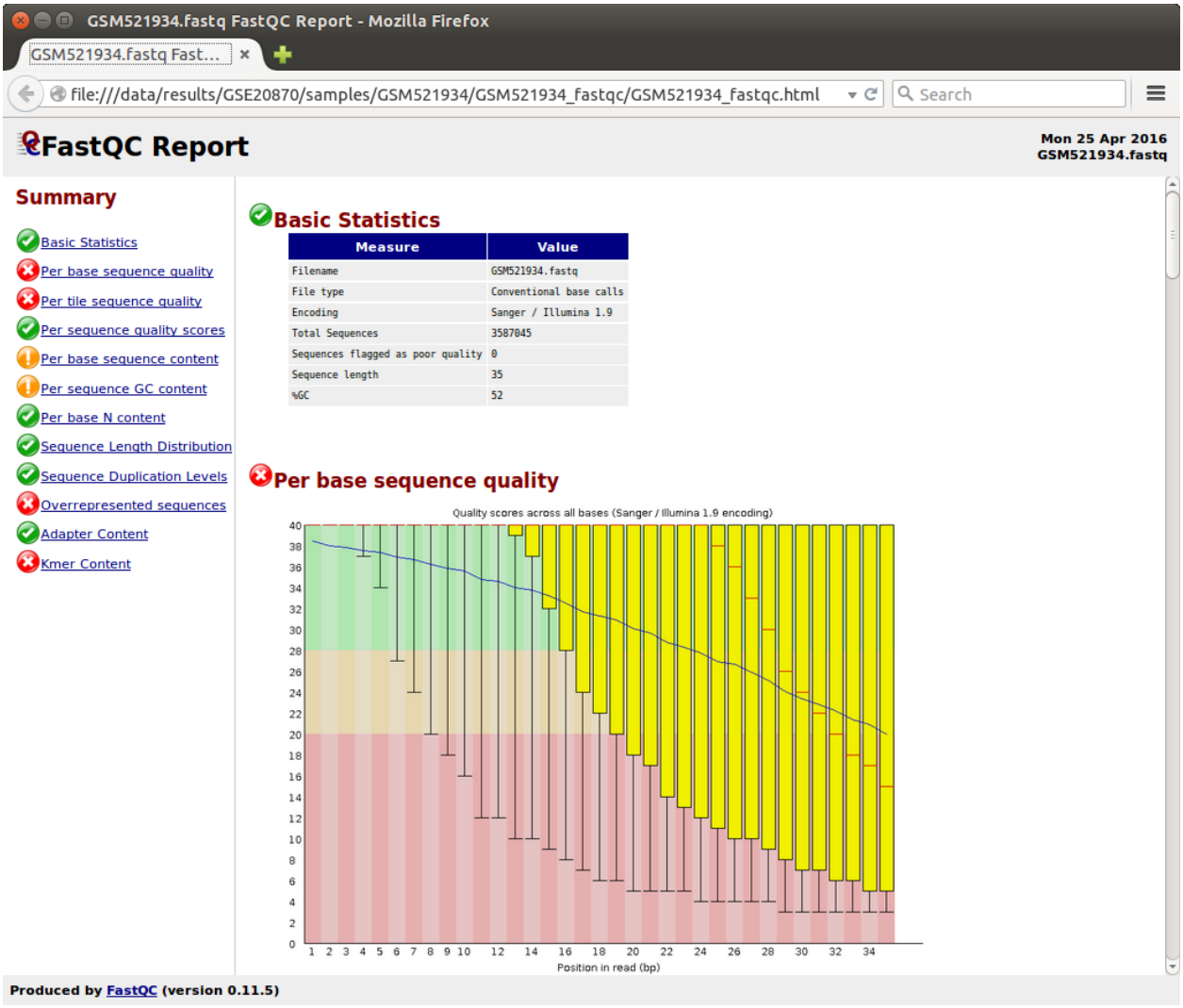
```
igv
```

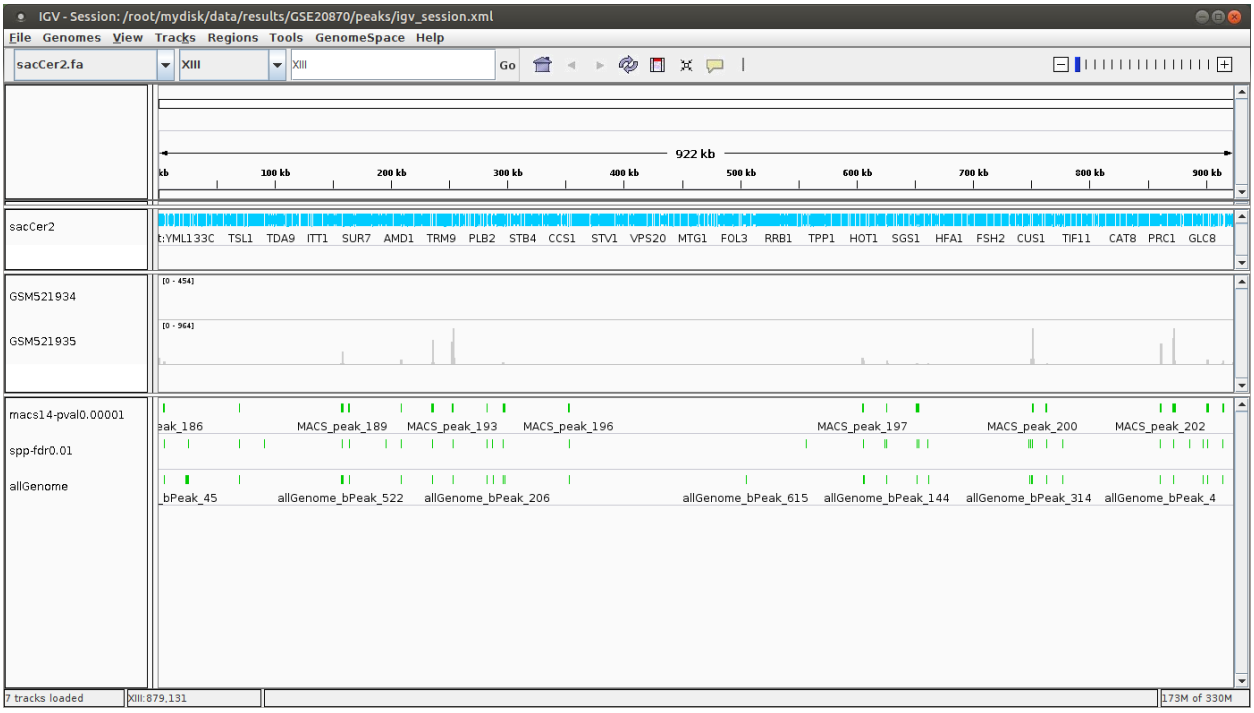
- Click "File" > "Open session..." and chose the file `~/ChIP-seq_SE_GSE20870/results/peaks/igv_session.xml`.
- You may need to adjust the panel sizes.

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


GSM521934.fastq FastQC Report - Mozilla Firefox
GSM521934.fastq Fast...
file:///data/results/GSE20870/samples/GSM521934/GSM521934_fastqc/GSM521934_fastqc.html
Search
FastQC Report
Mon 25 Apr 2016
GSM521934.fastq
Summary
Basic Statistics
Per base sequence quality
Per tile sequence quality
Per sequence quality scores
Per base sequence content
Per sequence GC content
Per base N content
Sequence Length Distribution
Sequence Duplication Levels
Overrepresented sequences
Adapter Content
Kmer Content
Basic Statistics
Measure
Value
Filename
GSM521934.fastq
File type
Conventional base calls
Encoding
Sanger / Illumina 1.9
Total Sequences
3587045
Sequences flagged as poor quality
0
Sequence length
35
%GC
52
Per base sequence quality
Quality scores across all bases (Sanger / Illumina 1.9 encoding)
Position in read (bp)
Produced by FastQC (version 0.11.5)



- 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.

3.5.4 Conda

TODO

3.6 Snakemake

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

3.6.1 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
```

3.6.2 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/SnakeChunks/archive/1.0.tar.gz
tar xvzf 1.0.tar.gz
cd SnakeChunks-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 SnakeChunks-1.0/doc/snakemake_tutorial
```

Rsamtools

```
R
```

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

3.6.3 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"]
```

(continues on next page)

(continued from previous page)

```
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"
```

(continues on next page)

(continued from previous page)

```
    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: "SnakeChunks-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

3.6.4 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

3.6.5 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.

3.7 Wiki NGS & Bioinformatics

To be completed

3.7.1 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

3.7.2 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.

3.7.3 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

3.7.4 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