# pm4ngs

*Release 0.0.1*

Roberto Vera Alvarez

May 28, 2020

# CONTENTS

PM4NGS was designed to generate a standard organizational structure for Next Generation Sequencing (ngs) data analysis. It includes a directory structure for the project, several Jupyter notebooks for data management and CWL workflows for pipeline execution.

Our work was inspired by a manuscript by Prof. William Noble in 2009: A Quick Guide to Organizing Computational Biology Projects. We recommend reading this paper for a better understanding of the guiding principles of our project.

The project is composed of three main parts.

1. a project organizational structure which define a standard files and directories for the project
2. Jupyter Notebooks as user interfaces for data management and visualization
3. CWL workflows that execute the data analysis

**PM4NGS** source code includes the templates used by **cookiecutter** to generate the project organizational structure and the Jupyter notebooks. The CWL workflows are defined in a separate GitHub project named: cwl-ngs-workflows-cbb.

All projects generated from these templates follow the same design principles explained in the Background Information.

# CHAPTER ONE

# FEATURES

- NGS data integration, management and analysis uses Jupyter notebooks, CWL workflows and cookiecutter project templates
- Easy installation and use with a minimum command line interaction
- Data analysis CWL workflows executed from the Jupyter notebook with automatic failing detection and can be validated with published data
- CWL workflows and Jupyter Notebooks are ready for cloud computing
- Project reports and dynamic content creation after data processing using CWL workflows are included
- Optional use of Docker/Biocontainers or Conda/Bioconda for Bioinformatics tool installations and managements are also included

CHAPTER

TWO

# LINKS TO AVAILABLE DATA ANALYSIS WORKFLOWS

## 2.1 Differential Gene expression from RNA-Seq data

**Warning:** Read the Background Information before proceeding with these steps

**Warning:** Read the *Project Templates Installation* notes to have the **cookiecutter** available in you shell depending on the execution environment you will be using.

### 2.1.1 Samples description file

A TSV file named **factors.txt** is the main file for the projects and workflow. This file should be created before any project creation. It is the base of the workflow and should be copied to the folder **data/{{dataset_name}}** just after creating the project structure.

The initial sample names, file name prefixes and metadata are specified on it.

It should have the following columns:

| id | SampleID | condition | replicate |
|---|---|---|---|
| classical01 | SRR4053795 | classical | 1 |
| classical01 | SRR4053796 | classical | 2 |
| nonclassical01 | SRR4053802 | nonclassical | 1 |
| nonclassical01 | SRR4053803 | nonclassical | 2 |

**Warning:** Columns names are required and are case sensitive.

**Columns**

- **id**: Sample names. It can be different of sample file name.
- **SampleID**: This is the prefix of the sample file name.

  For single-end data the prefix ends in the file extension. In this case, for the first column, a file name named **SRR4053795.fastq.gz** must exist.

  For paired-end data the files **SRR4053795_1.fastq.gz** and **SRR4053795_2.fastq.gz** must exist.

The data files should be copied to the folder **data/{{dataset_name}}/**.

- **condition**: Conditions to analyze or group the samples. Avoid using non alphanumeric characters.

  For RNASeq projects the differential gene expression will be generated comparing these conditions. If there are multiple conditions all comparisons will be generated. It must be at least two conditions.

  For ChIPSeq projects differential binding events will be detected comparing these conditions. If there are multiple conditions all comparisons will be generated. It must be at least two conditions.

  For ChIPexo projects the samples of the same condition will be grouped for the peak calling with MACE.

- **replicate**: Replicate number for samples.

## 2.1.2 Installation

### RNA-Seq workflow with Conda/Bioconda

The RNA-Seq project structure is created using the conda environment named **templates**.

First step is to activate the **templates** environment:

```
localhost:~> conda activate templates
```

Then, a YAML file (for this example I will call this file: **rnaseq-sra-paired.yaml**) with your project detail should be created.

```yaml
default_context:
  author_name: "Roberto Vera Alvarez"
  user_email: "veraalva@ncbi.nlm.nih.gov"
  project_name: "rnaseq-sra-paired"
  dataset_name: "PRJNA290924"
  is_data_in_SRA: "y"
  ngs_data_type: "RNA-Seq"
  sequencing_technology: "paired-end"
  create_demo: "y"
  number_spots: "1000000"
  organism: "human"
  genome_dir: "/gfs/data/genomes/igenomes/Homo_sapiens/UCSC/hg38"
  genome_name: "hg38"
  aligner_index_dir: "{{ cookiecutter.genome_dir}}/STAR"
  genome_fasta: "{{ cookiecutter.genome_dir}}/genome.fa"
  genome_gtf: "{{ cookiecutter.genome_dir}}/genes.gtf"
  genome_gff: "{{ cookiecutter.genome_dir}}/genes.gff"
  genome_gff3: "{{ cookiecutter.genome_dir}}/genes.gff3"
  genome_bed: "{{ cookiecutter.genome_dir}}/genes.bed"
  genome_chromsizes: "{{ cookiecutter.genome_dir}}/chrom.sizes"
  genome_mappable_size: "hg38"
  genome_blacklist: "{{ cookiecutter.genome_dir}}/hg38-blacklist.bed"
  fold_change: "2.0"
  fdr: "0.05"
  use_docker: "n"
  pull_images: "n"
  use_conda: "y"
  cwl_runner: "cwl-runner"
  cwl_workflow_repo: "https://github.com/ncbi/cwl-ngs-workflows-cbb"
  create_virtualenv: "n"
```

(continues on next page)

(continued from previous page)

```
  use_gnu_parallel: "y"
  max_number_threads: "16"
```

A full description of this parameters are *here*.

After the **rnaseq-sra-paired.yaml** is created the project structure should be created using this command obtaining the following output.

```
localhost:~> cookiecutter --no-input --config-file rnaseq-sra-paired.yaml https://
↪github.com/ncbi/pm4ngs.git
Checking RNA-Seq workflow dependencies .......... Done
localhost:~>
```

This process should create a project organizational structure like this:

```
localhost:~> tree rnaseq-sra-paired
rnaseq-sra-paired
├── bin
│   ├── bioconda (This directory include a conda envs for all bioinfo tools)
│   ├── cwl-ngs-workflows-cbb (CWL workflow repo cloned here)
│   └── jupyter (This directory include a conda envs for Jupyter notebooks)
├── config
│   └── init.py
├── data
│   └── PRJNA290924
├── doc
├── index.html
├── LICENSE
├── notebooks
│   ├── 00 - Project Report.ipynb
│   ├── 01 - Pre-processing QC.ipynb
│   ├── 02 - Samples trimming.ipynb
│   ├── 03 - Alignments.ipynb
│   ├── 04 - Quantification.ipynb
│   ├── 05 - DGA.ipynb
│   └── 06 - GO enrichment.ipynb
├── README.md
├── requirements
│   └── python.txt
├── results
│   └── PRJNA290924
├── src
└── tmp

14 directories, 12 files
```

Now you should copied the **factors.txt** file to the folder: **data/PRJNA290924**.

After this process, **cookiecutter** should have created create two virtual environment for this workflow.

The first one is for running the Jupyter notebooks which require Python 3.6+ and it is named: **jupyter**. It can be manually installed as described in here.

The second environment is be used to install all Bioinformatics tools required by the workflow and it will be named: **bioconda**.

You can verify the environments running this command:

```
localhost:~> conda env list
# conda environments:
#
base                  *  /gfs/conda
tempates                 /gfs/conda/envs/templates
                         /home/veraalva/rnaseq-sra-paired/bin/bioconda
                         /home/veraalva/rnaseq-sra-paired/bin/jupyter

localhost:~>
```

Please, note that the Conda prefix **/gfs/conda** will be different in you host. Also, note that the **bioconda** and **jupyter** envs are inside the **bin** directory of your project keeping them static inside the project organizational structure.

## RNA-Seq workflow usage with Conda/Bioconda

For start using the workflow you need to activate the conda environments **bioconda** and **jupyter**.

```
localhost:~> conda activate /home/veraalva/rnaseq-sra-paired/bin/bioconda
localhost:~> conda activate --stack /home/veraalva/rnaseq-sra-paired/bin/jupyter
```

Note the **–stack** option to have both environment working at the same time. Also, the order is important, **bioconda** should be activated before **jupyter**.

Test the conda envs:

```
localhost:~> which fastqc
/home/veraalva/rnaseq-sra-paired/bin/bioconda/bin/fastqc
localhost:~> which jupyter
/home/veraalva/rnaseq-sra-paired/bin/jupyter/bin/jupyter
```

Note that the **fastqc** tools is installed in the **bioconda** env and the **jupyter** command is installed in the **jupyter** env.

Then, you can start the jupyter notebooks.

```
localhost:~> jupyter notebook
```

If the workflow is deployed in a remote machine using SSH access the correct way to start the notebooks is:

```
localhost:~> jupyter notebook --no-browser --ip='0.0.0.0'
```

In this case the option **–ip='0.0.0.0'** will server the Jupyter notebook on all network interfaces and you can access them from your desktop browser using the port returned by the Jupyter server.

Finally, you should navegate in your browser to the **notebooks** folder and start executing all notebooks by their order leaving the **00 - Project Report.ipynb** to the end.

## RNA-Seq workflow with Docker

In this case, the RNA-Seq project structure is created using the Python virtual environment as described *here*

First step is to activate the Python virtual environment.

```
localhost:~> source venv-templates/bin/activate
```

Then, a YAML file (for this example I will call this file: **rnaseq-sra-paired.yaml**) with your project detail should be created.

```yaml
default_context:
  author_name: "Roberto Vera Alvarez"
  user_email: "veraalva@ncbi.nlm.nih.gov"
  project_name: "rnaseq-sra-paired"
  dataset_name: "PRJNA290924"
  is_data_in_SRA: "y"
  ngs_data_type: "RNA-Seq"
  sequencing_technology: "paired-end"
  create_demo: "y"
  number_spots: "1000000"
  organism: "human"
  genome_dir: "/gfs/data/genomes/igenomes/Homo_sapiens/UCSC/hg38"
  genome_name: "hg38"
  aligner_index_dir: "{{ cookiecutter.genome_dir}}/STAR"
  genome_fasta: "{{ cookiecutter.genome_dir}}/genome.fa"
  genome_gtf: "{{ cookiecutter.genome_dir}}/genes.gtf"
  genome_gff: "{{ cookiecutter.genome_dir}}/genes.gff"
  genome_gff3: "{{ cookiecutter.genome_dir}}/genes.gff3"
  genome_bed: "{{ cookiecutter.genome_dir}}/genes.bed"
  genome_chromsizes: "{{ cookiecutter.genome_dir}}/chrom.sizes"
  genome_mappable_size: "hg38"
  genome_blacklist: "{{ cookiecutter.genome_dir}}/hg38-blacklist.bed"
  fold_change: "2.0"
  fdr: "0.05"
  use_docker: "y"
  pull_images: "y"
  use_conda: "n"
  cwl_runner: "cwl-runner"
  cwl_workflow_repo: "https://github.com/ncbi/cwl-ngs-workflows-cbb"
  create_virtualenv: "y"
  use_gnu_parallel: "y"
  max_number_threads: "16"
```

A full description of this parameters are *here*.

After the **rnaseq-sra-paired.yaml** is created the project structure should be created using this command obtaining the following output.

```
localhost:~> cookiecutter --no-input --config-file rnaseq-sra-paired.yaml https://
↪github.com/ncbi/pm4ngs.git
Cloning Git repo: https://github.com/ncbi/cwl-ngs-workflows-cbb to /home/veraalva/
↪rnaseq-sra-paired/bin/cwl-ngs-workflows-cbb
Creating a Python3.7 virtualenv
Installing packages in: /home/veraalva/rnaseq-sra-paired/venv using file /home/
↪veraalva/rnaseq-sra-paired/requirements/python.txt
Checking RNA-Seq workflow dependencies .
    Pulling image: quay.io/biocontainers/fastqc:0.11.8--1 . Done .
    Pulling image: quay.io/biocontainers/trimmomatic:0.39--1 . Done .
    Pulling image: quay.io/biocontainers/star:2.7.1a--0 . Done .
    Pulling image: quay.io/biocontainers/samtools:1.9--h91753b0_8 . Done .
    Pulling image: quay.io/biocontainers/rseqc:3.0.0--py_3 . Done .
    Pulling image: quay.io/biocontainers/tpmcalculator:0.0.3--hdbb99b9_0 . Done .
    Pulling image: quay.io/biocontainers/igvtools:2.5.3--0 . Done .
    Pulling image: quay.io/biocontainers/sra-tools:2.9.6--hf484d3e_0 . Done .
    Pulling image: ubuntu:18.04 . Done
    Building image: r-3.5_ubuntu-18.04 . Done  Done
localhost:~>
```

This process should create a project organizational structure like this:

```
localhost:~> tree rnaseq-sra-paired
rnaseq-sra-paired
├── bin
│   └── cwl-ngs-workflows-cbb (CWL workflow repo cloned here)
├── config
│   └── init.py
├── data
│   └── PRJNA290924
├── doc
├── index.html
├── LICENSE
├── notebooks
│   ├── 00 - Project Report.ipynb
│   ├── 01 - Pre-processing QC.ipynb
│   ├── 02 - Samples trimming.ipynb
│   ├── 03 - Alignments.ipynb
│   ├── 04 - Quantification.ipynb
│   ├── 05 - DGA.ipynb
│   └── 06 - GO enrichment.ipynb
├── README.md
├── requirements
│   └── python.txt
├── results
│   └── PRJNA290924
├── src
├── tmp
└── venv
    ├── bin
    ├── etc
    ├── include
    ├── lib
    ├── locale
    ├── README.rst
    └── share

19 directories, 13 files
```

Now you should copied the **factors.txt** file to the directory: **data/PRJNA238004**.

After this process, **cookiecutter** should have pulled all docker images require by the project.

## RNA-Seq workflow usage with Docker

For start using the workflow you need to activate the Python environment inside the project.

```
localhost:~> source venv/bin/activate
```

Then, you can start the jupyter notebooks now.

```
localhost:~> jupyter notebook
```

If the workflow is deployed in a remote machine using SSH access the correct way to start the notebooks is:

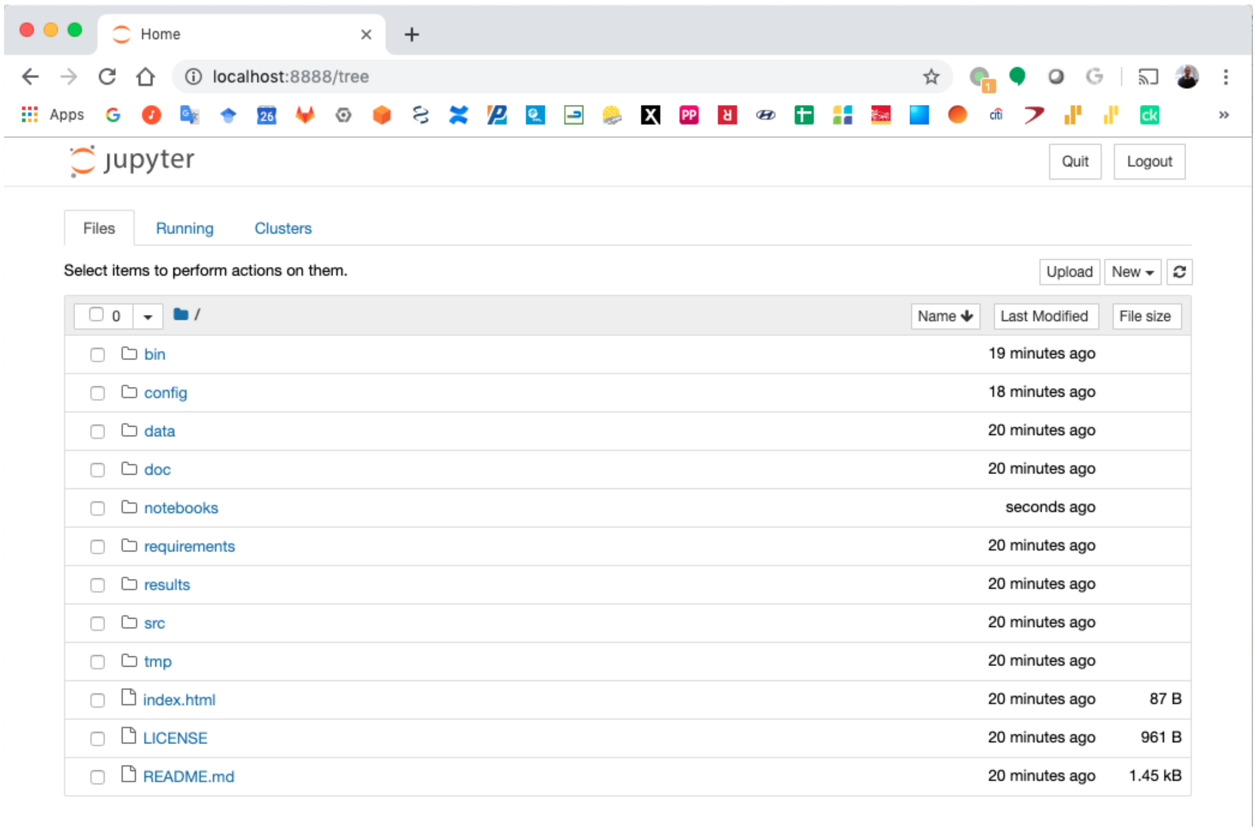
```
localhost:~> jupyter notebook --no-browser --ip='0.0.0.0'
```

In this case the option **–ip='0.0.0.0'** will server the Jupyter notebook on all network interfaces and you can access them from your desktop browser using the port returned by the Jupyter server.

Finally, you should navigate in your browser to the **notebooks** directory and start executing all notebooks by their order leaving the **00 - Project Report.ipynb** to the end.

### 2.1.3 Jupyter Notebook Server

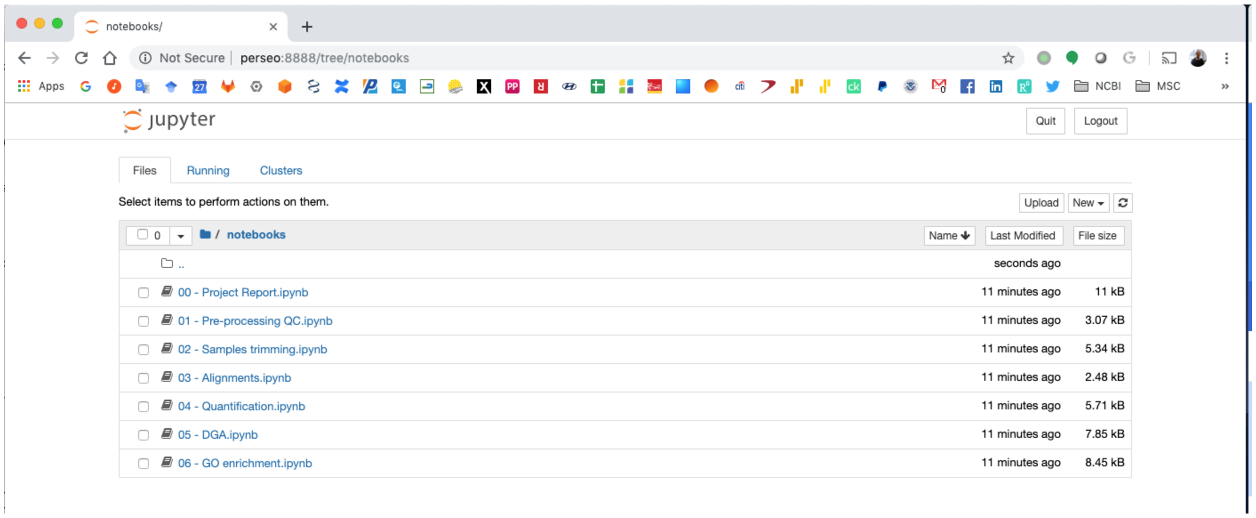
#### Top-level directories from the Jupyter server viewed in a web browser



Select items to perform actions on them.

| Name | Last Modified | File size |
|---|---|---|
| bin | 19 minutes ago | |
| config | 18 minutes ago | |
| data | 20 minutes ago | |
| doc | 20 minutes ago | |
| notebooks | seconds ago | |
| requirements | 20 minutes ago | |
| results | 20 minutes ago | |
| src | 20 minutes ago | |
| tmp | 20 minutes ago | |
| index.html | 20 minutes ago | 87 B |
| LICENSE | 20 minutes ago | 961 B |
| README.md | 20 minutes ago | 1.45 kB |

### Notebook generated fro the Chip-exo data analysis


notebooks/

Not Secure | perseo:8888/tree/notebooks

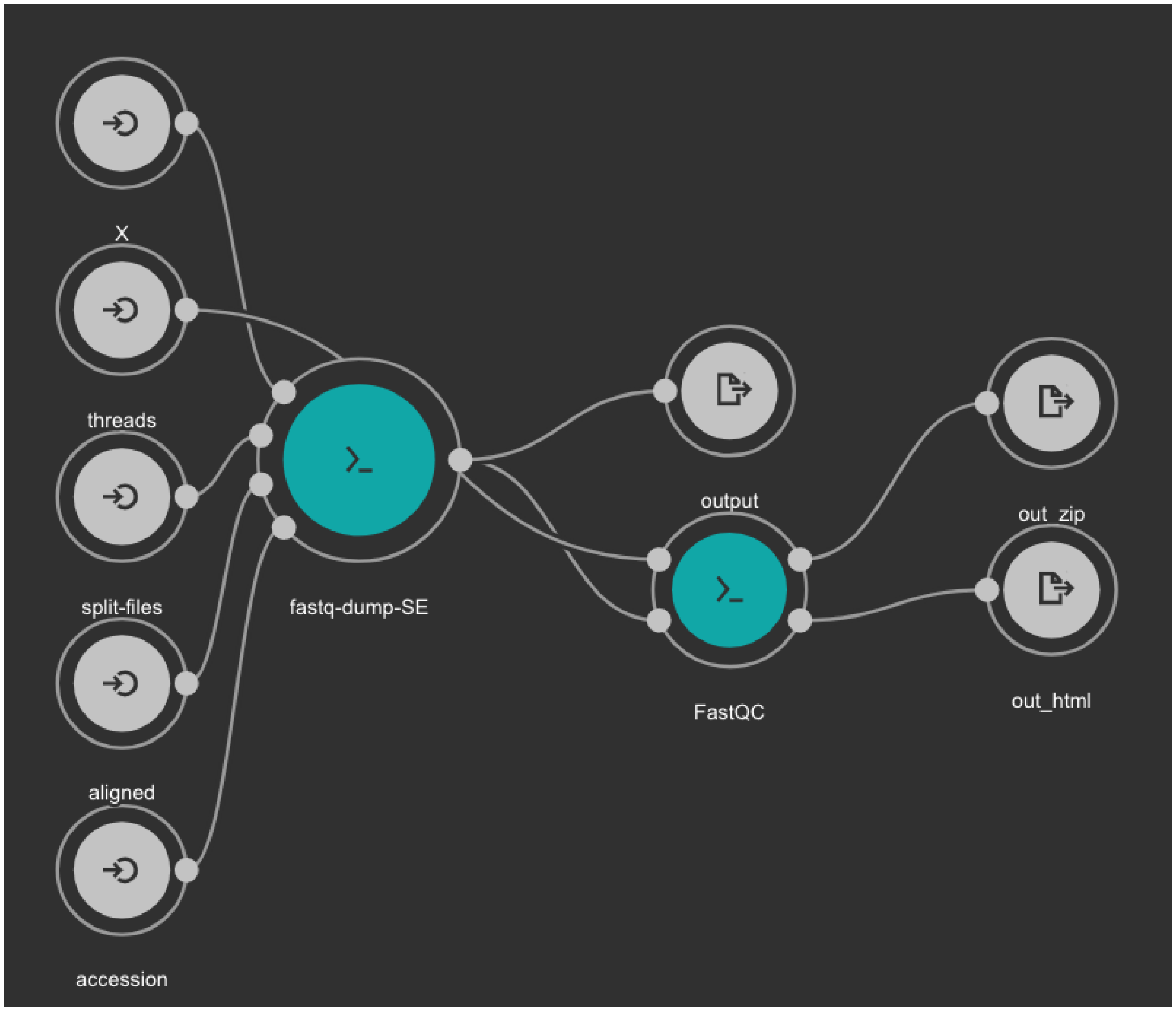
Select items to perform actions on them.

| Name | Last Modified | File size |
|---|---|---|
| .. | seconds ago | |
| 00 - Project Report.ipynb | 11 minutes ago | 11 kB |
| 01 - Pre-processing QC.ipynb | 11 minutes ago | 3.07 kB |
| 02 - Samples trimming.ipynb | 11 minutes ago | 5.34 kB |
| 03 - Alignments.ipynb | 11 minutes ago | 2.48 kB |
| 04 - Quantification.ipynb | 11 minutes ago | 5.71 kB |
| 05 - DGA.ipynb | 11 minutes ago | 7.85 kB |
| 06 - GO enrichment.ipynb | 11 minutes ago | 8.45 kB |

## 2.1.4 CWL workflows

### SRA download and QC workflow

This CWL workflow is designed to download FASTQ files from the NCBI SRA database using fastq-dump and then, execute fastqc generating a quality control report of the sample.



**Inputs**

- **accession**: SRA accession ID. Type: string. Required.
- **aligned**: Used it to download only aligned reads. Type: boolean. Optional.
- **split-files**: Dump each read into separate file. Files will receive suffix corresponding to read number. Type: boolean. Optional.
- **threads**: Number of threads. Type: int. Default: 1. Optional.
- **X**: Maximum spot id. Optional.

**Outputs**

- **output**: Fastq files downloaded. Type: File[]
- **out_zip**: FastQC report ZIP file. Type: File[]

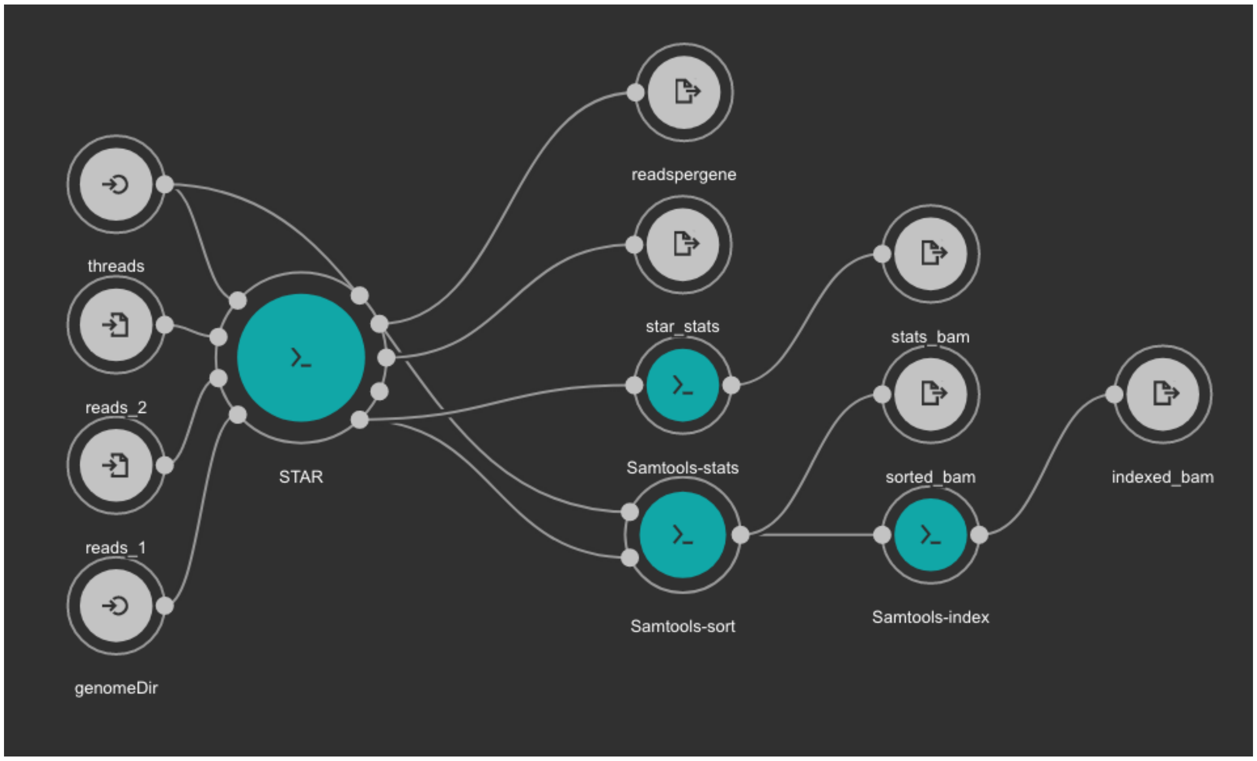
- **out_html**: FastQC report HTML. Type: File[]

### Samples Trimming

Our workflows uses Trimmomatic for read trimming. The Jupyter notebooks uses some basic Trimmomatic options that need to be modified depending on the FastQC quality control report generated for the sample.

### STAR based alignment and sorting

This workflows use STAR for alignning RNA-Seq reads to a genome. The obtained BAM file is sorted using SAMtools. Statistics outputs from STAR and SAMtools are returned as well.



**Inputs**

- **genomeDir**: Aligner indexes directory. Type: Directory. Required. Variable ALIGNER_INDEX in the Jupyter Notebooks.
- **threads**: Number of threads. Type: int. Default: 1. Optional.
- **reads_1**: FastQ file to be processed for paired-end reads _1. Type: File. Required.
- **reads_2**: FastQ file to be processed for paired-end reads _2. Type: File. Required.

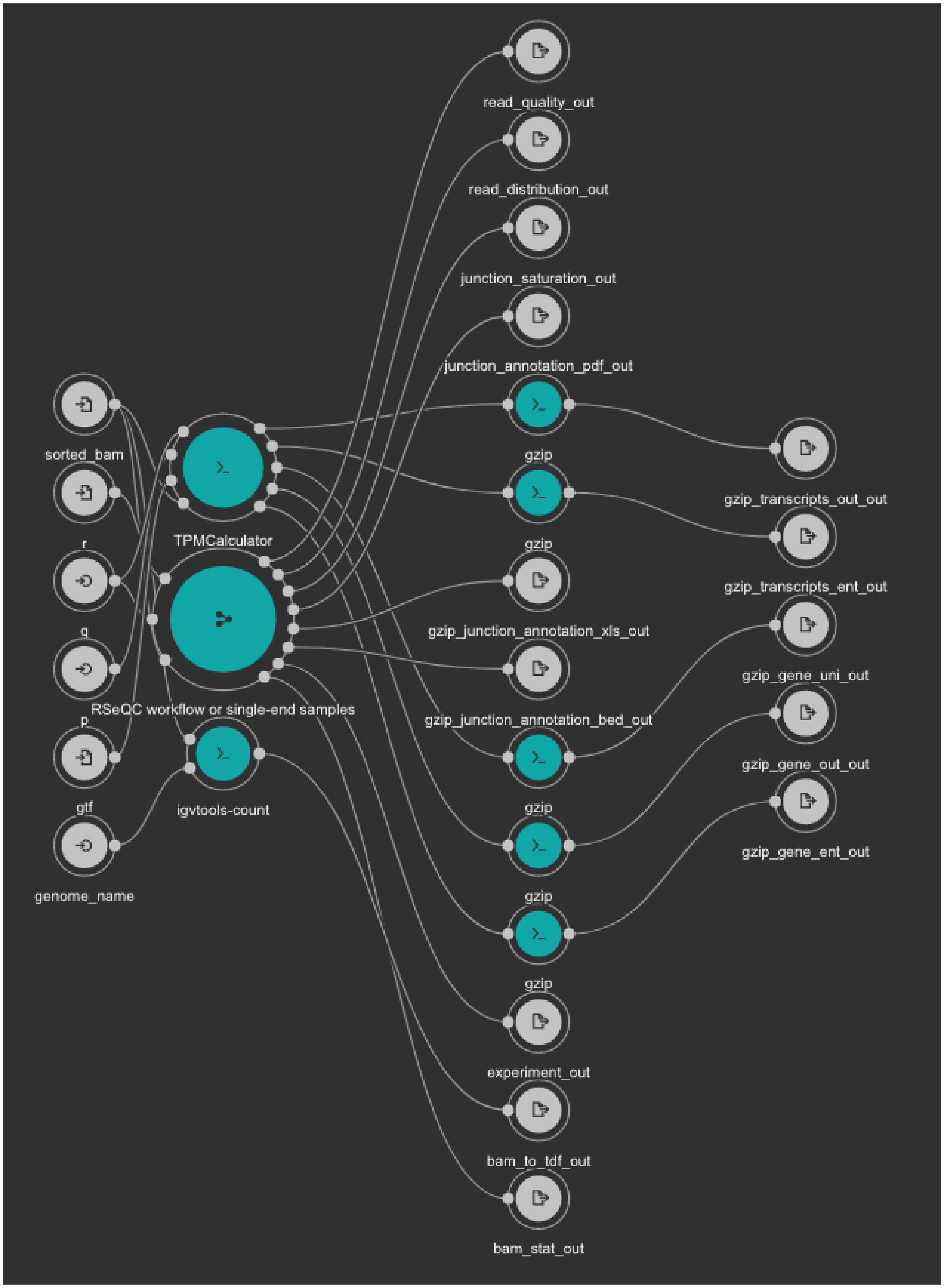
**Outputs**

- **sorted_bam**: Final BAM file filtered and sorted. Type: File.

- **indexed_bam**: BAM index file. Type: File.
- **star_stats**: STAR alignment statistics. Type: File.
- **readspergene**: STAR reads per gene output. Type: File.
- **stats_bam**: SAMtools stats output: Type: File.

### RNA-Seq quantification and QC workflow using TPMCalculator

This workflow uses TPMCalculator to quantify the abundance of genes and transcripts from the sorted BAM file. Additionally, RSeQC is executed to generate multiple quality control outputs from the sorted BAM file. At the end, a TDF file is generated using igvtools from the BAM file for a quick visualization.


read_quality_out
read_distribution_out
junction_saturation_out
junction_annotation_pdf_out
sorted_bam
gzip
TPMCalculator
gzip_transcripts_out_out
r
gzip
gzip_transcripts_ent_out
q
gzip_junction_annotation_xls_out
gzip_gene_uni_out
RSeQC workflow or single-end samples
gzip_junction_annotation_bed_out
p
gzip_gene_out_out
gtf
igvtools-count
gzip
gzip_gene_ent_out
genome_name
gzip
gzip
experiment_out
bam_to_tdf_out
bam_stat_out

**Inputs**

- **gtf**: Genome GTF file. Variable GENOME_GTF in the Jupyter Notebooks. Type: File. Required.
- **genome_name**: Genome name as defined in IGV for TDF conversion. Type: string. Required.
- **q**: Minimum MAPQ value to use reads. We recommend 255. Type: int. Required.
- **r**: Reference Genome in BED format used by RSeQC. Variable GENOME_BED in the Jupyter Notebooks. Type: File. Required.
- **sorted_bam**: Sorted BAM file to quantify. Type: File. Required.

**Outputs**

- **bam_to_tdf_out**: TDF file created with igvtools from the BAM file for quick visualization. Type: File.
- **gzip_gene_ent_out**: TPMCalculator gene ENT output gzipped. Type: File.
- **gzip_gene_out_out**: TPMCalculator gene OUT output gzipped. Type: File.
- **gzip_gene_uni_out**: TPMCalculator gene UNI output gzipped. Type: File.
- **gzip_transcripts_ent_out**: TPMCalculator transcript ENT output gzipped. Type: File.
- **gzip_transcripts_out_out**: TPMCalculator transcript OUT output gzipped. Type: File.
- **bam_stat_out**: RSeQC BAM stats output. Type: File.
- **experiment_out**: RSeQC experiment output. Type: File.
- **gzip_junction_annotation_bed_out**: RSeQC junction annotation bed. Type: File.
- **gzip_junction_annotation_xls_out**: RSeQC junction annotation xls. Type: File.
- **junction_annotation_pdf_out**: RSeQC junction annotation PDF figure. Type: File.
- **junction_saturation_out**: RSeQC junction saturation output. Type: File.
- **read_distribution_out**: RSeQC read distribution output. Type: File.
- **read_quality_out**: RSeQC read quality output. Type: File.

### Differential Gene Expression analysis from RNA-Seq data

Our notebooks are designed to execute a Differential Gene Expression analysis using two available tools: DESeq2 and EdgeR. Also, the results for the interception of both tools output is reported with volcano plots, heatmaps and PCA plots.

The workflow use the **factors.txt** file to generate an array with all combinations of **conditions**. The code to generate this array is very simple and can be found in the cell number 3 in the **05 - DGA.ipynb** notebook.

```
comparisons = []
for s in itertools.combinations(factors['condition'].unique(), 2):
    comparisons.append(list(s))
```

Let's suppose we have a **factors.txt** file with three conditions: **cond1**, **cond2** and **cond3**. The **comparisons** array will look like:

```
comparisons = [
    ['cond1', 'cond2'],
    ['cond1', 'cond3'],
    ['cond2', 'cond3']
]
```

To avoid this behavior and execute the comparison just in a set of conditions, you should remove the code in the cell number 3 in the **05 - DGA.ipynb** notebook and manually create the array of combinations to be compared as:

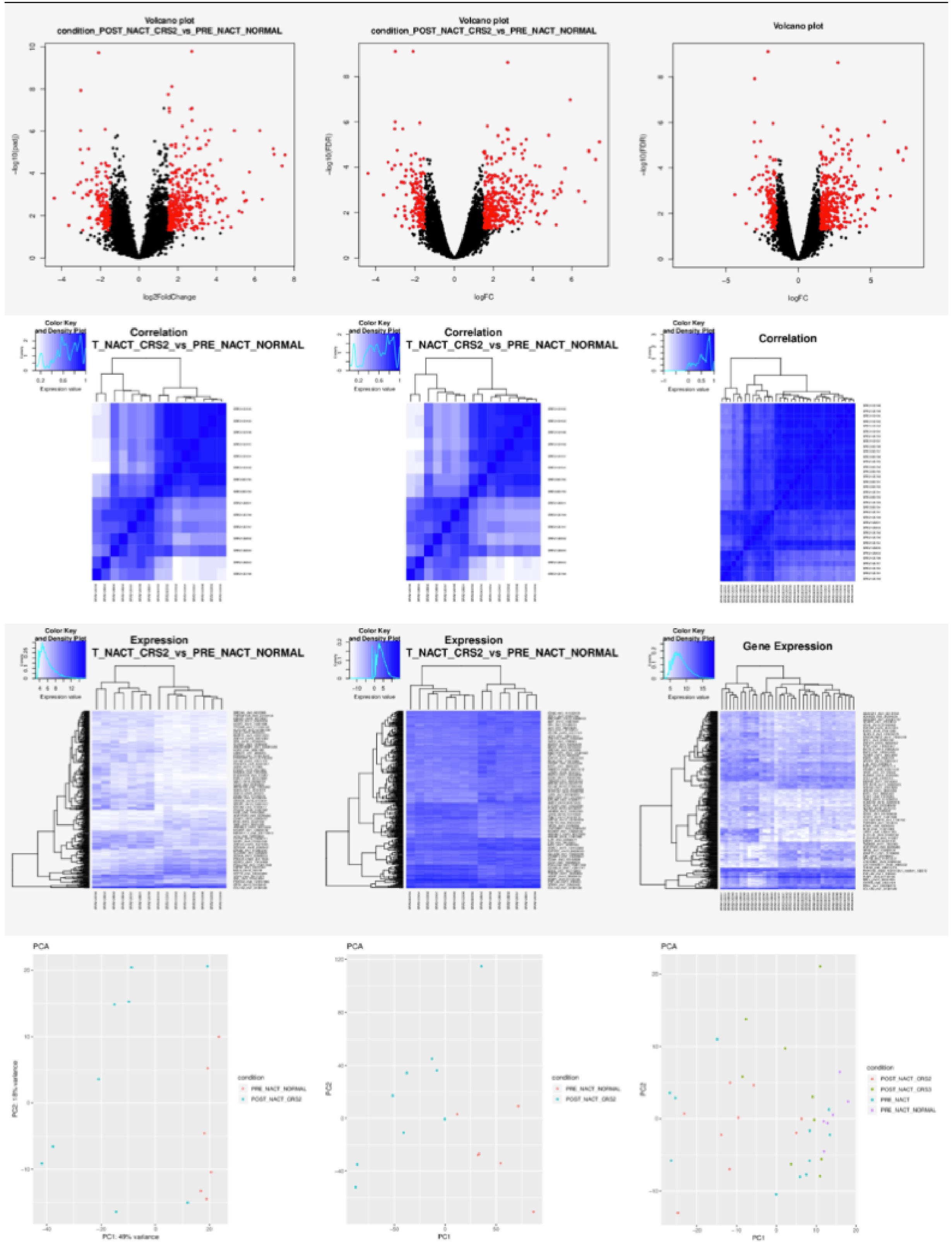
```
comparisons = [
    ['cond1', 'cond3'],
]
```

The R code used for running DESeq2 is embedded in deseq2-2conditions.cwl from line 14 to line 178. The R code used for running EdgeR is embedded in edgeR-2conditions.cwl from line 14 to line 165.

A table with DGA plots is generated for each condition in the **00 - Project Report.ipynb** as shown next.

**Condition: POST NACT CRS2 vs PRE NACT NORMAL**

| Deseq2 | EdgeR | Union |
|---|---|---|



### GO enrichment from RNA-Seq data

The GO enrichment analysis is executed with an *in-house* developed python package named goenrichment. This tools uses the hypergeometric distribution test to estimate the probability of successes in selecting GO terms from a list of differentially expressed genes. The GO terms are represented as a network using the python library NetworkX.

The tool uses a pre-computed database, currently available for human and mouse, at https://ftp.ncbi.nlm.nih.gov/pub/goenrichment/. However, the project web page describe how to create your own database from a set of reference databases.

The workflow uses the **factors.txt** file to generate an array with all combinations of **conditions**. The code to generate this array is very simple and can be found in the cell number 3 in the **06 - GO enrichment.ipynb** notebook.

```
comparisons = []
for s in itertools.combinations(factors['condition'].unique(), 2):
    comparisons.append(list(s))
```

Let's suppose we have a **factors.txt** file with three conditions: **cond1**, **cond2** and **cond3**. The **comparisons** array will look like:

```
comparisons = [
    ['cond1', 'cond2'],
    ['cond1', 'cond3'],
    ['cond2', 'cond3']
]
```

To avoid this behavior and execute the comparison just in a set of conditions, you should remove the code in the cell number 3 in the **06 - GO enrichment.ipynb** notebook and manually create the array of combinations to be compared as:

```
comparisons = [
    ['cond1', 'cond3'],
]
```

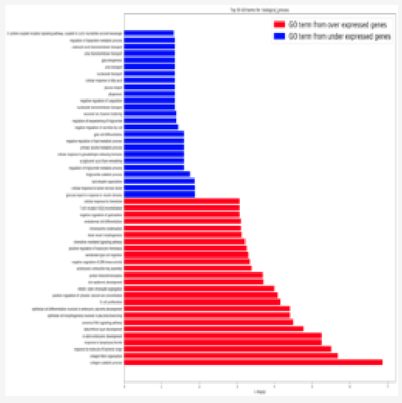
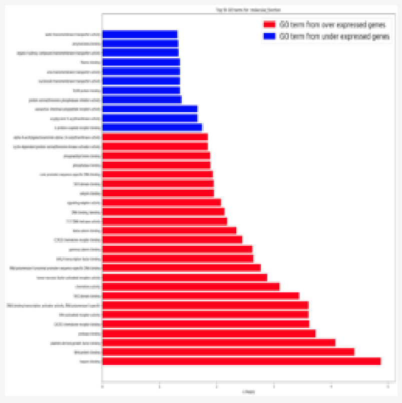
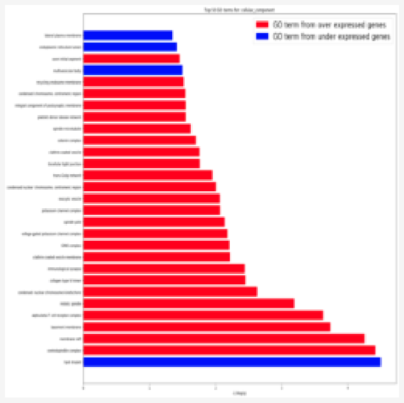
Additionally, the workflow requires three cutoff that are defined in the cell number 5 of the same notebook.

```
min_category_depth=4
min_category_size=3
max_category_size=500
```

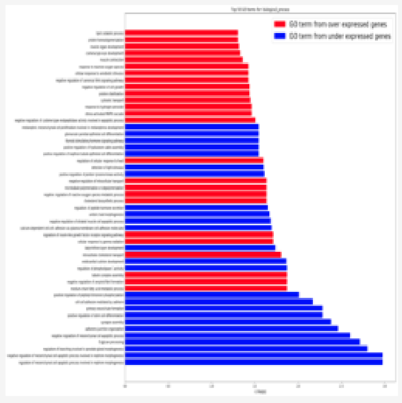
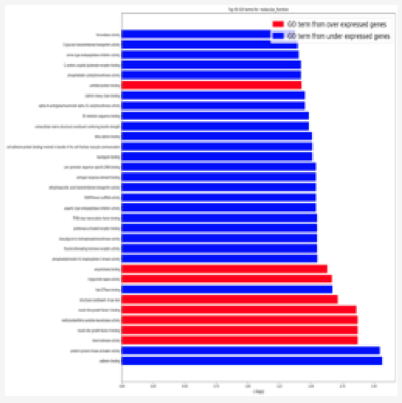
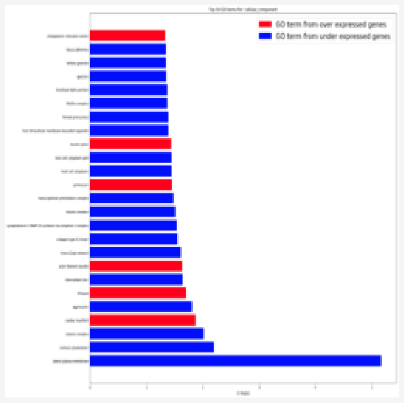
**Cutoffs definition**

- min_category_depth: Min GO term graph depth to include in the report. Default: 4
- min_category_size: Min number of gene in a GO term to include in the report. Default: 3
- max_category_size: Max number of gene in a GO term to include in the report. Default: 500

A table with GO terms plots is generated for each condition in the **00 - Project Report.ipynb** as shown next. In these plots the red bars are for GO terms selected from the over expressed genes and the blue bars are for GO terms selected from the under expressed genes. It is important to clarify that the two sets of GO terms don't overlap each other.

| Biological process | Molecular Function | Cellular Component |
|---|---|---|
|  |  |  |

**Condition: POST_NACT_CRS3 vs POST_NACT_CRS2**

| Biological process | Molecular Function | Cellular Component |
|---|---|---|
|  |  |  |

### 2.1.5 Test Project

A test project is available (read-only) at https://ftp.ncbi.nlm.nih.gov/pub/pm4ngs/examples/rnaseq-sra-paired

### 2.1.6 Extra requirements

#### Creating STAR indexes

This workflow uses STAR for sequence alignment. The STAR index creation is not included in the workflow, that's why we are including an small section here to describe how the STAR indexes can be created.

The **genome.fa** and **genes.gtf** files should be copied to the genome directory.

```
localhost:~> conda activate /home/veraalva/rnaseq-sra-paired/bin/bioconda
localhost:~> conda activate --stack /home/veraalva/rnaseq-sra-paired/bin/jupyter
localhost:~> cd rnaseq-sra-paired/data
localhost:~> mkdir genome
localhost:~> cd genome
localhost:~> mkdir STAR
localhost:~> cd STAR
localhost:~> cwl-runner --no-container ../../../bin/cwl-ngs-workflows-cbb/tools/STAR/
↪star-index.cwl --runThreadN 16 --genomeDir . --genomeFastaFiles ../genome.fa  --
↪sjdbGTFfile ../genes.gtf
```

(continues on next page)

(continued from previous page)

```
localhost:~> cd ..
localhost:~> tree
.
├── genes.gtf
├── genome.fa
└── STAR
    ├── chrLength.txt
    ├── chrNameLength.txt
    ├── chrName.txt
    ├── chrStart.txt
    ├── exonGeTrInfo.tab
    ├── exonInfo.tab
    ├── geneInfo.tab
    ├── Genome
    ├── genomeParameters.txt
    ├── Log.out
    ├── SA
    ├── SAindex
    ├── sjdbInfo.txt
    ├── sjdbList.fromGTF.out.tab
    ├── sjdbList.out.tab
    └── transcriptInfo.tab

1 directory, 18 files
```

Here all files inside the directory **STAR** are created by the workflow.

### Creating BED files from GTF

For generating a BED file from a GTF.

The **genes.gtf** file should be copied to the genome directory.

```
localhost:~> conda activate /home/veraalva/rnaseq-sra-paired/bin/bioconda
localhost:~> conda activate --stack /home/veraalva/rnaseq-sra-paired/bin/jupyter
localhost:~> cd rnaseq-sra-paired/data
localhost:~> mkdir genome
localhost:~> cd genome
localhost:~> cwl-runner --no-container ../../bin/cwl-ngs-workflows-cbb/workflows/UCSC/
↪gtftobed.cwl --gtf genes.gtf
localhost:~> tree
.
├── genes.bed
├── genes.genePred
├── genes.gtf
└── genome.fa

0 directory, 4 files
```

Here the files **genes.bed** and **genes.genePred** are created from the workflow.

## 2.2 Differential Binding detection from ChIP-Seq data

**Warning:** Read the Background Information before proceeding with these steps

**Warning:** Read the *Project Templates Installation* notes to have the **cookiecutter** available in you shell depending on the execution environment you will be using.

### 2.2.1 Samples description file

A TSV file named **factors.txt** is the main file for the projects and workflow. This file should be created before any project creation. It is the base of the workflow and should be copied to the folder **data/{{dataset_name}}** just after creating the project structure.

The initial sample names, file name prefixes and metadata are specified on it.

It should have the following columns:

| id | SampleID | condition | replicate |
|---|---|---|---|
| classical01 | SRR4053795 | classical | 1 |
| classical01 | SRR4053796 | classical | 2 |
| nonclassical01 | SRR4053802 | nonclassical | 1 |
| nonclassical01 | SRR4053803 | nonclassical | 2 |

**Warning:** Columns names are required and are case sensitive.

**Columns**

- **id**: Sample names. It can be different of sample file name.
- **SampleID**: This is the prefix of the sample file name.

  For single-end data the prefix ends in the file extension. In this case, for the first column, a file name named **SRR4053795.fastq.gz** must exist.

  For paired-end data the files **SRR4053795_1.fastq.gz** and **SRR4053795_2.fastq.gz** must exist.

  The data files should be copied to the folder **data/{{dataset_name}}/**.
- **condition**: Conditions to analyze or group the samples. Avoid using non alphanumeric characters.

  For RNASeq projects the differential gene expression will be generated comparing these conditions. If there are multiple conditions all comparisons will be generated. It must be at least two conditions.

  For ChIPSeq projects differential binding events will be detected comparing these conditions. If there are multiple conditions all comparisons will be generated. It must be at least two conditions.

  For ChIPexo projects the samples of the same condition will be grouped for the peak calling with MACE.
- **replicate**: Replicate number for samples.

### 2.2.2 Installation

#### ChIP-Seq workflow with Conda/Bioconda

The ChIP-Seq project structure is created using the conda environment named **templates**.

First step is to activate the **templates** environment:

```
localhost:~> conda activate templates
```

Then, a YAML file (for this example I will call this file: **chipseq-hmgn1.yaml**) with your project detail should be created.

```yaml
default_context:
  author_name: "Roberto Vera Alvarez"
  user_email: "veraalva@ncbi.nlm.nih.gov"
  project_name: "chipseq-hmgn1"
  dataset_name: "PRJNA481982"
  is_data_in_SRA: "y"
  ngs_data_type: "ChIP-Seq"
  sequencing_technology: "paired-end"
  create_demo: "y"
  number_spots: "2000000"
  organism: "mouse"
  genome_dir: "/gfs/data/genomes/igenomes/Homo_sapiens/UCSC/Mus_musculus/mm9"
  genome_name: "mm9"
  aligner_index_dir: "{{ cookiecutter.genome_dir}}/BWA"
  genome_fasta: "{{ cookiecutter.genome_dir}}/genome.fa"
  genome_gtf: "{{ cookiecutter.genome_dir}}/genes.gtf"
  genome_gff: "{{ cookiecutter.genome_dir}}/genes.gff"
  genome_gff3: "{{ cookiecutter.genome_dir}}/genes.gff3"
  genome_bed: "{{ cookiecutter.genome_dir}}/genes.bed"
  genome_chromsizes: "{{ cookiecutter.genome_dir}}/mm9.chrom.sizes"
  genome_mappable_size: "mm9"
  genome_blacklist: "{{ cookiecutter.genome_dir}}/mm9-blacklist.bed"
  fold_change: "2.0"
  fdr: "0.05"
  use_docker: "n"
  pull_images: "n"
  use_conda: "y"
  cwl_runner: "cwl-runner"
  cwl_workflow_repo: "https://github.com/ncbi/cwl-ngs-workflows-cbb"
  create_virtualenv: "n"
  use_gnu_parallel: "y"
  max_number_threads: "16"
```

A full description of this parameters are *here*.

After the **chipseq-hmgn1.yaml** is created the project structure should be created using this command obtaining the following output.

```
localhost:~> cookiecutter --no-input --config-file chipseq-hmgn1.yaml https://github.
↪com/ncbi/pm4ngs.git
Checking ChIP-Seq workflow dependencies .......... Done
localhost:~>
```

This process should create a project organizational structure like this:

```
localhost:~> tree chipseq-hmgn1
chipseq-hmgn1
├── bin
│   ├── bioconda (This directory include a conda envs for all bioinfo tools)
│   ├── cwl-ngs-workflows-cbb (CWL workflow repo cloned here)
│   └── jupyter (This directory include a conda envs for Jupyter notebooks)
├── config
│   └── init.py
├── data
│   └── PRJNA481982
├── doc
├── index.html
├── LICENSE
├── notebooks
│   ├── 00\ -\ Project\ Report.ipynb
│   ├── 01\ -\ Pre-processing\ QC.ipynb
│   ├── 02\ -\ Samples\ trimming.ipynb
│   ├── 03\ -\ Alignments.ipynb
│   └── 04\ -\ Peak\ Calling.ipynb
├── README.md
├── requirements
│   └── python.txt
├── results
│   └── PRJNA481982
├── src
└── tmp

12 directories, 9 files
```

Now you should copied the **factors.txt** file to the folder: **data/PRJNA481982**.

After this process, **cookiecutter** should have created create two virtual environment for this workflow.

The first one is for running the Jupyter notebooks which require Python 3.6+ and it is named: **jupyter**. It can be manually installed as described in here.

The second environment is be used to install all Bioinformatics tools required by the workflow and it will be named: **bioconda**.

You can verify the environments running this command:

```
localhost:~> conda env list
# conda environments:
#
base                  *  /gfs/conda
tempates                 /gfs/conda/envs/templates
                         /home/veraalva/chipseq-hmgn1/bin/bioconda
                         /home/veraalva/chipseq-hmgn1/bin/jupyter

localhost:~>
```

Please, note that the Conda prefix **/gfs/conda** will be different in you host. Also, note that the **bioconda** and **jupyter** envs are inside the **bin** directory of your project keeping them static inside the project organizational structure.

### ChIP-Seq workflow usage with Conda/Bioconda

For start using the workflow you need to activate the conda environments **bioconda** and **jupyter**.

```
localhost:~> conda activate /home/veraalva/chipseq-hmgn1/bin/bioconda
localhost:~> conda activate --stack /home/veraalva/chipseq-hmgn1/bin/jupyter
```

Note the **–stack** option to have both environment working at the same time. Also, the order is important, **bioconda** should be activated before **jupyter**.

Test the conda envs:

```
localhost:~> which fastqc
/home/veraalva/chipseq-hmgn1/bin/bioconda/bin/fastqc
localhost:~> which jupyter
/home/veraalva/chipseq-hmgn1/bin/jupyter/bin/jupyter
```

Note that the **fastqc** tools is installed in the **bioconda** env and the **jupyter** command is installed in the **jupyter** env.

Then, you can start the jupyter notebooks.

```
localhost:~> jupyter notebook
```

If the workflow is deployed in a remote machine using SSH access the correct way to start the notebooks is:

```
localhost:~> jupyter notebook --no-browser --ip='0.0.0.0'
```

In this case the option **–ip='0.0.0.0'** will server the Jupyter notebook on all network interfaces and you can access them from your desktop browser using the port returned by the Jupyter server.

Finally, you should navegate in your browser to the **notebooks** folder and start executing all notebooks by their order leaving the **00 - Project Report.ipynb** to the end.

## ChIP-Seq workflow with Docker

In this case, the ChIP-Seq project structure is created using the Python virtual environment as described *here*

First step is to activate the Python virtual environment.

```
localhost:~> source venv-templates/bin/activate
```

Then, a YAML file (for this example I will call this file: **chipseq-hmgn1.yaml**) with your project detail should be created.

```
default_context:
  author_name: "Roberto Vera Alvarez"
  user_email: "veraalva@ncbi.nlm.nih.gov"
  project_name: "chipseq-hmgn1"
  dataset_name: "PRJNA481982"
  is_data_in_SRA: "y"
  ngs_data_type: "ChIP-Seq"
  sequencing_technology: "paired-end"
  create_demo: "y"
  number_spots: "2000000"
  organism: "mouse"
  genome_dir: "/gfs/data/genomes/igenomes/Homo_sapiens/UCSC/Mus_musculus/mm9"
  genome_name: "mm9"
  aligner_index_dir: "{{ cookiecutter.genome_dir}}/BWA"
  genome_fasta: "{{ cookiecutter.genome_dir}}/genome.fa"
  genome_gtf: "{{ cookiecutter.genome_dir}}/genes.gtf"
  genome_gff: "{{ cookiecutter.genome_dir}}/genes.gff"
```

(continues on next page)

(continued from previous page)

```
  genome_gff3: "{{ cookiecutter.genome_dir}}/genes.gff3"
  genome_bed: "{{ cookiecutter.genome_dir}}/genes.bed"
  genome_chromsizes: "{{ cookiecutter.genome_dir}}/mm9.chrom.sizes"
  genome_mappable_size: "mm9"
  genome_blacklist: "{{ cookiecutter.genome_dir}}/mm9-blacklist.bed"
  fold_change: "2.0"
  fdr: "0.05"
  use_docker: "y"
  pull_images: "y"
  use_conda: "n"
  cwl_runner: "cwl-runner"
  cwl_workflow_repo: "https://github.com/ncbi/cwl-ngs-workflows-cbb"
  create_virtualenv: "y"
  use_gnu_parallel: "y"
  max_number_threads: "16"
```

A full description of this parameters are *here*.

After the **chipseq-hmgn1.yaml** is created the project structure should be created using this command obtaining the following output.

```
localhost:~>  cookiecutter --no-input --config-file chipseq-paired.yaml https://
↪github.com/ncbi/pm4ngs.git
Cloning Git repo: https://github.com/ncbi/cwl-ngs-workflows-cbb to /home/veraalva/
↪chipseq-hmgn1/bin/cwl-ngs-workflows-cbb
Creating a Python3.7 virtualenv
Installing packages in: /home/veraalva/chipseq-hmgn1/venv using file /home/veraalva/
↪chipseq-hmgn1/requirements/python.txt
Checking ChIP-Seq workflow dependencies .
        Pulling image: quay.io/biocontainers/fastqc:0.11.8--1 . Done .
        Pulling image: quay.io/biocontainers/trimmomatic:0.39--1 . Done .
        Pulling image: quay.io/biocontainers/bwa:0.7.17--h84994c4_5 . Done .
        Pulling image: quay.io/biocontainers/bedtools:2.28.0--hdf88d34_0 . Done .
        Pulling image: quay.io/biocontainers/bcftools:1.9--h5c2b69b_5 . Done .
        Pulling image: quay.io/biocontainers/phantompeakqualtools:1.2--1 . Done .
        Pulling image: quay.io/biocontainers/samtools:1.9--h91753b0_8 . Done .
        Pulling image: quay.io/biocontainers/rseqc:3.0.0--py_3 . Done .
        Pulling image: quay.io/biocontainers/sra-tools:2.9.6--hf484d3e_0 . Done .
        Pulling image: quay.io/biocontainers/igvtools:2.5.3--0 . Done .
        Pulling image: quay.io/biocontainers/macs2:2.1.2--py27r351h14c3975_1 . Done .
        Pulling image: quay.io/biocontainers/homer:4.10--pl526hc9558a2_0 . Done .
        Pulling image: ubuntu:18.04 . Done
        Building image: r-3.5_ubuntu-18.04 . Done  Done
localhost:~>
```

This process should create a project organizational structure like this:

```
localhost:~> tree chipseq-hmgn1
chipseq-hmgn1
.
├── bin
│   └── cwl-ngs-workflows-cbb (CWL workflow repo cloned here)
├── config
│   └── init.py
├── data
│   └── PRJNA481982
├── doc
```

(continues on next page)

(continued from previous page)

```
├── index.html
├── LICENSE
├── notebooks
│   ├── 00 - Project Report.ipynb
│   ├── 01 - Pre-processing QC.ipynb
│   ├── 02 - Samples trimming.ipynb
│   ├── 03 - Alignments.ipynb
│   └── 04 - Peak Calling.ipynb
├── README.md
├── requirements
│   └── python.txt
├── results
│   └── PRJNA481982
├── src
├── tmp
└── venv
    ├── bin
    ├── etc
    ├── include
    ├── lib
    ├── lib64 -> lib
    ├── LICENSE.txt
    ├── locale
    ├── README.md
    ├── README.rst
    ├── setup.cfg
    └── share

20 directories, 14 files
```

Now you should copied the **factors.txt** file to the directory: **data/PRJNA481982**.

After this process, **cookiecutter** should have pulled all docker images require by the project.

#### ChIP-Seq workflow usage with Docker

For start using the workflow you need to activate the Python environment inside the project.

```
localhost:~> source venv/bin/activate
```

Then, you can start the jupyter notebooks now.

```
localhost:~> jupyter notebook
```

If the workflow is deployed in a remote machine using SSH access the correct way to start the notebooks is:

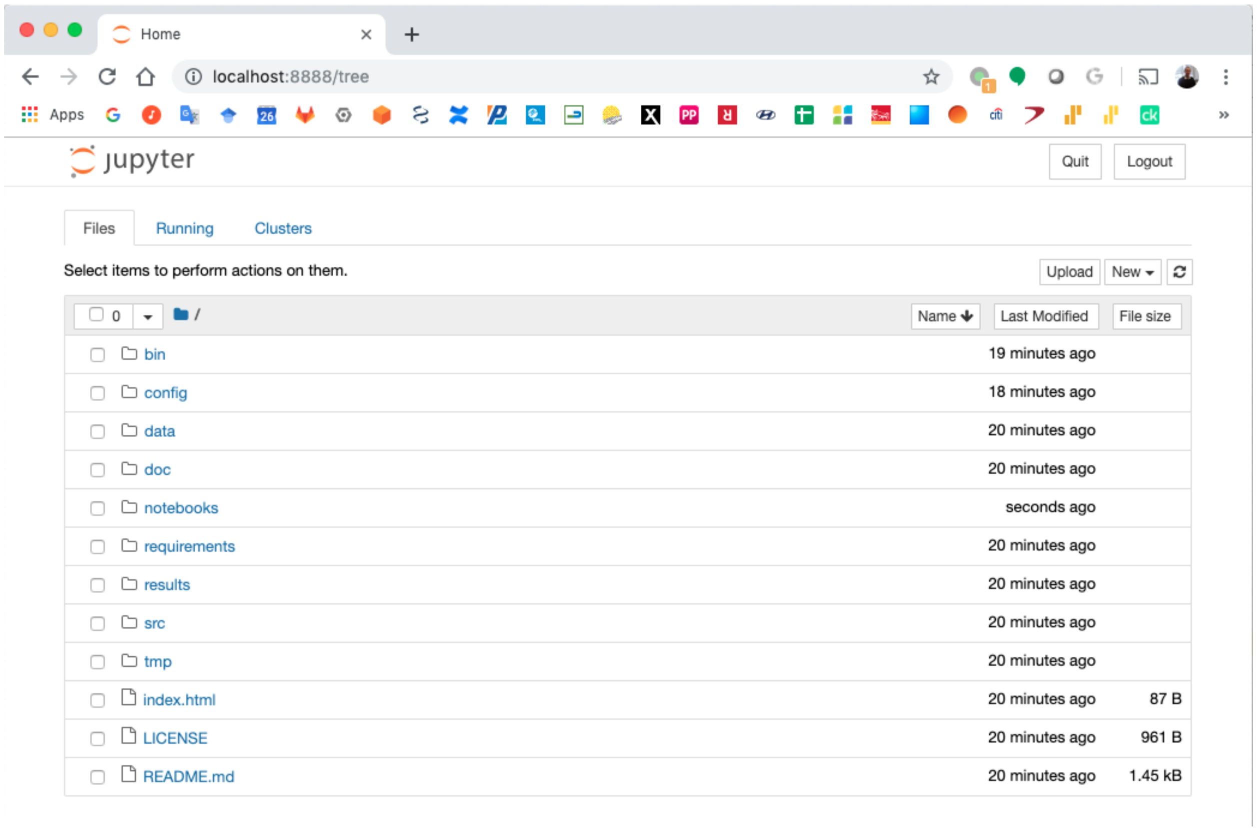
```
localhost:~> jupyter notebook --no-browser --ip='0.0.0.0'
```

In this case the option **–ip='0.0.0.0'** will server the Jupyter notebook on all network interfaces and you can access them from your desktop browser using the port returned by the Jupyter server.

Finally, you should navigate in your browser to the **notebooks** directory and start executing all notebooks by their order leaving the **00 - Project Report.ipynb** to the end.

### 2.2.3 Jupyter Notebook Server

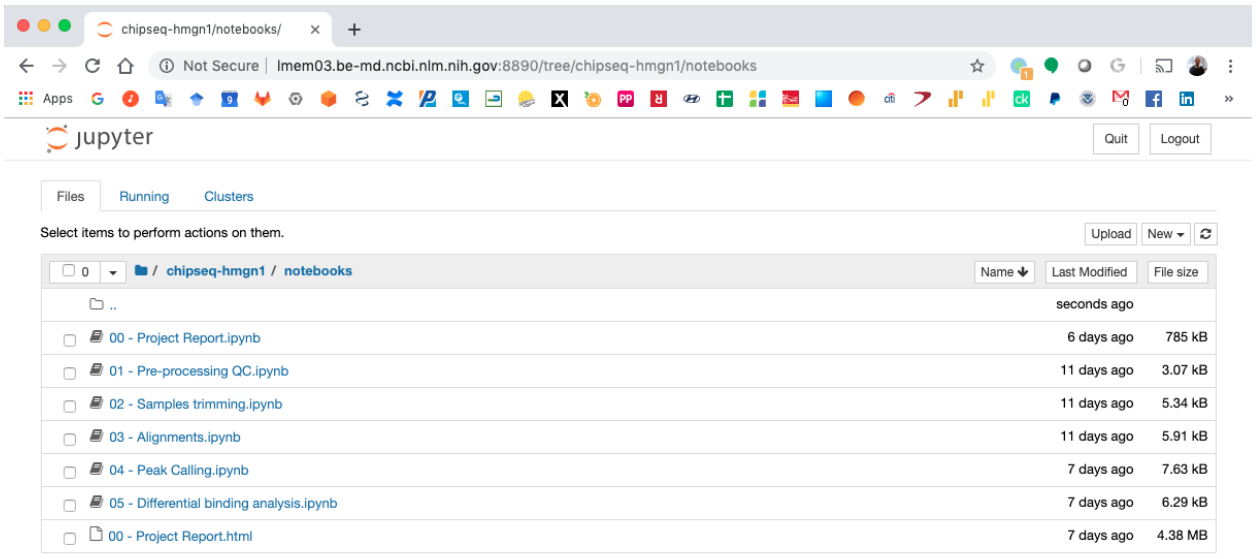
## Top-level directories from the Jupyter server viewed in a web browser



Select items to perform actions on them.

| Name | Last Modified | File size |
|---|---|---|
| bin | 19 minutes ago | |
| config | 18 minutes ago | |
| data | 20 minutes ago | |
| doc | 20 minutes ago | |
| notebooks | seconds ago | |
| requirements | 20 minutes ago | |
| results | 20 minutes ago | |
| src | 20 minutes ago | |
| tmp | 20 minutes ago | |
| index.html | 20 minutes ago | 87 B |
| LICENSE | 20 minutes ago | 961 B |
| README.md | 20 minutes ago | 1.45 kB |

## Notebook generated fro the ChIP-Seq data analysis


chipseq-hmgn1/notebooks/
Not Secure | lmem03.be-md.ncbi.nlm.nih.gov:8890/tree/chipseq-hmgn1/notebooks
Apps
jupyter
Quit
Logout
Files
Running
Clusters
Upload
New
/ chipseq-hmgn1 / notebooks
Name
Last Modified
File size

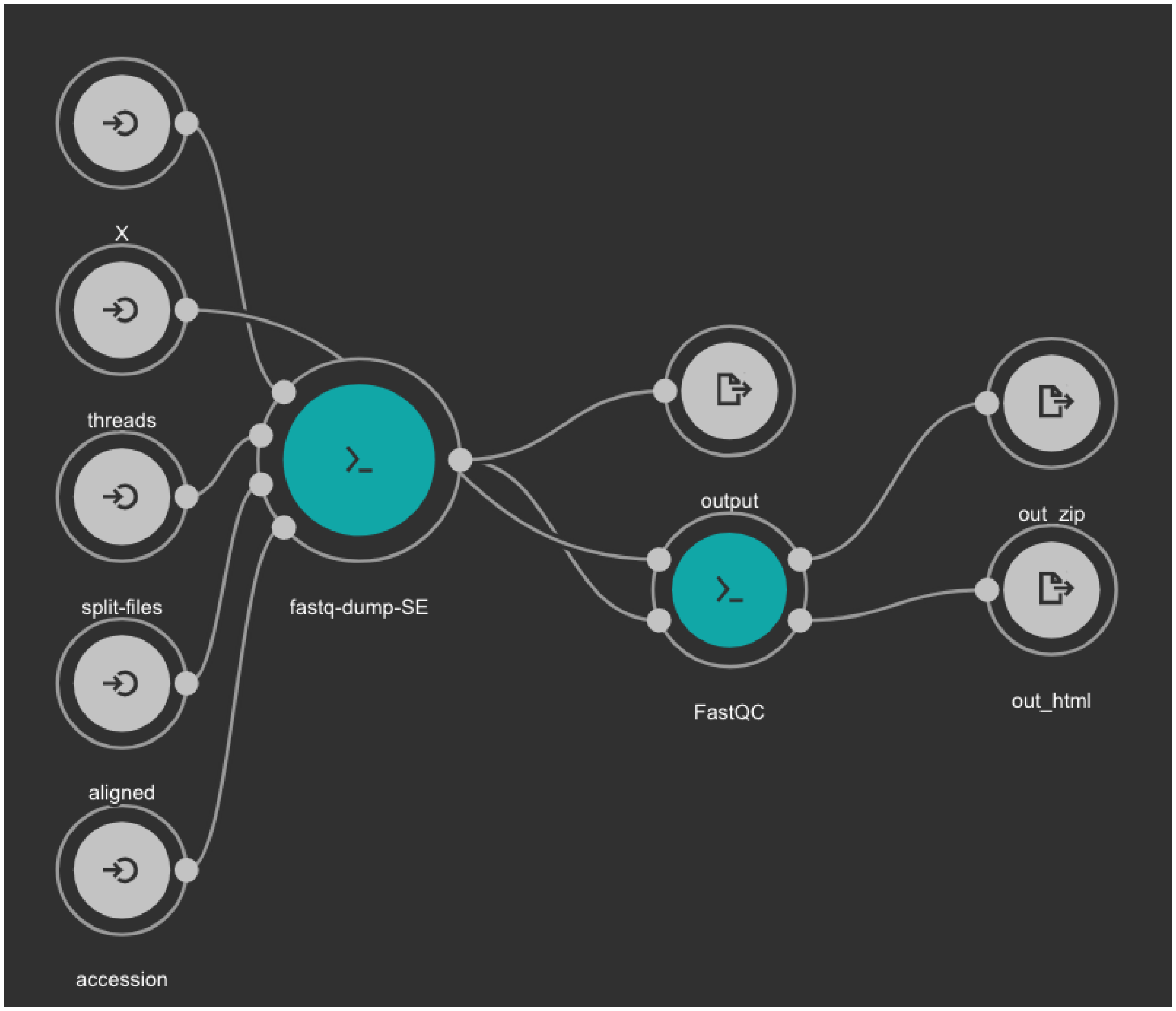
Select items to perform actions on them.

| Name | Last Modified | File size |
|---|---|---|
| .. | seconds ago | |
| 00 - Project Report.ipynb | 6 days ago | 785 kB |
| 01 - Pre-processing QC.ipynb | 11 days ago | 3.07 kB |
| 02 - Samples trimming.ipynb | 11 days ago | 5.34 kB |
| 03 - Alignments.ipynb | 11 days ago | 5.91 kB |
| 04 - Peak Calling.ipynb | 7 days ago | 7.63 kB |
| 05 - Differential binding analysis.ipynb | 7 days ago | 6.29 kB |
| 00 - Project Report.html | 7 days ago | 4.38 MB |

### 2.2.4 CWL workflows

#### SRA download and QC workflow

This CWL workflow is designed to download FASTQ files from the NCBI SRA database using fastq-dump and then, execute fastqc generating a quality control report of the sample.



**Inputs**

- **accession**: SRA accession ID. Type: string. Required.
- **aligned**: Used it to download only aligned reads. Type: boolean. Optional.
- **split-files**: Dump each read into separate file. Files will receive suffix corresponding to read number. Type: boolean. Optional.
- **threads**: Number of threads. Type: int. Default: 1. Optional.
- **X**: Maximum spot id. Optional.

**Outputs**

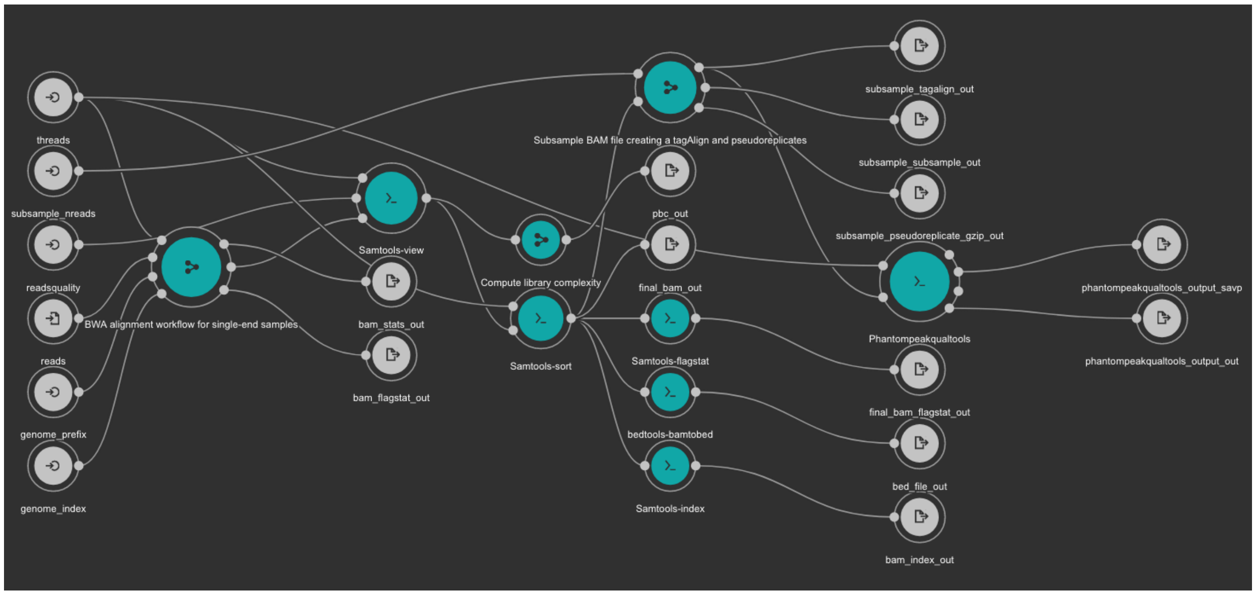
- **output**: Fastq files downloaded. Type: File[]
- **out_zip**: FastQC report ZIP file. Type: File[]
- **out_html**: FastQC report HTML. Type: File[]

### Samples Trimming

Our workflows uses Trimmomatic for read trimming. The Jupyter notebooks uses some basic Trimmomatic options that need to be modified depending on the FastQC quality control report generated for the sample.

### BWA based alignment and quality control workflow

This workflow use BWA as base aligner. It also use SAMtools and bedtools for file conversion and statistics report. Finally, Phantompeakqualtools is used to generate quality control report for the processed samples.



**Inputs**

- **genome_index**: Aligner indexes directory. Type: Directory. Required. Variable ALIGNER_INDEX in the Jupyter Notebooks.
- **genome_prefix**: Prefix of the aligner indexes. Generally, it is the name of the genome FASTA file. It can be used as os.path.basename(GENOME_FASTA) in the Jupyter Notebooks. Type: string. Required.
- **readsquality**: Minimum MAPQ value to use reads. We recommend for ChIP_exo dataa value of: 30. Type: int. Required.
- **threads**: Number of threads. Type: int. Default: 1. Optional.
- **subsample_nreads**: Number of reads to be used for the subsample. We recomend for ChIP_exo dataa value of: 500000. Type: int. Required.

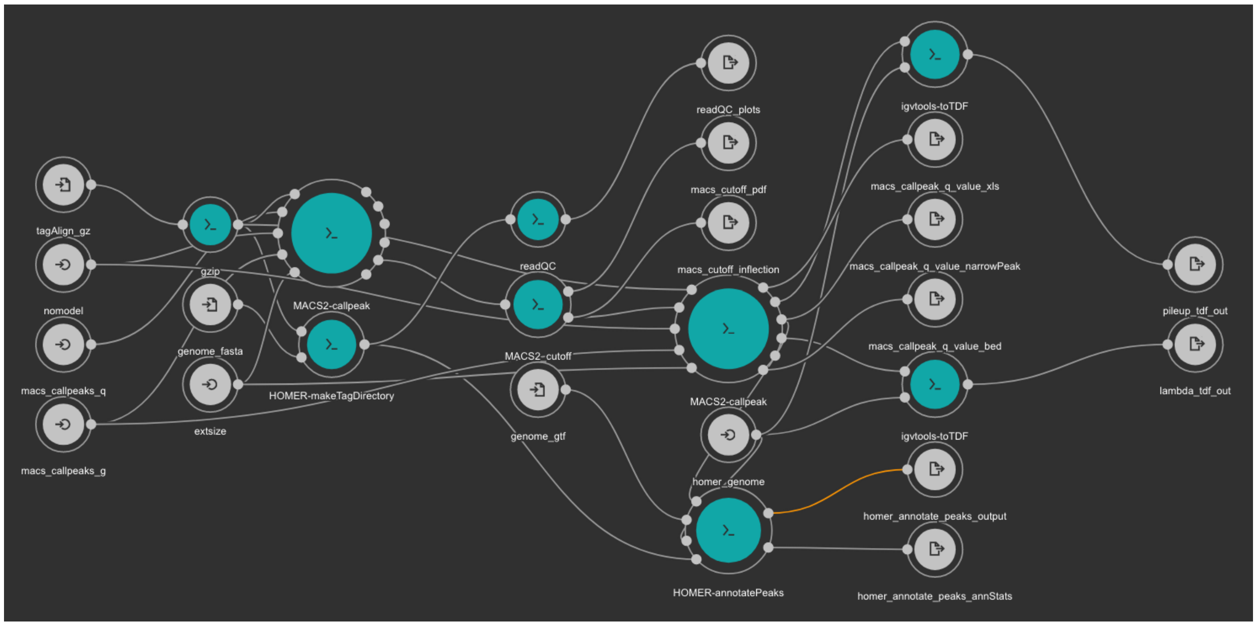
- **reads**: FastQ file to be processed. Type: File. Required.

**Outputs**

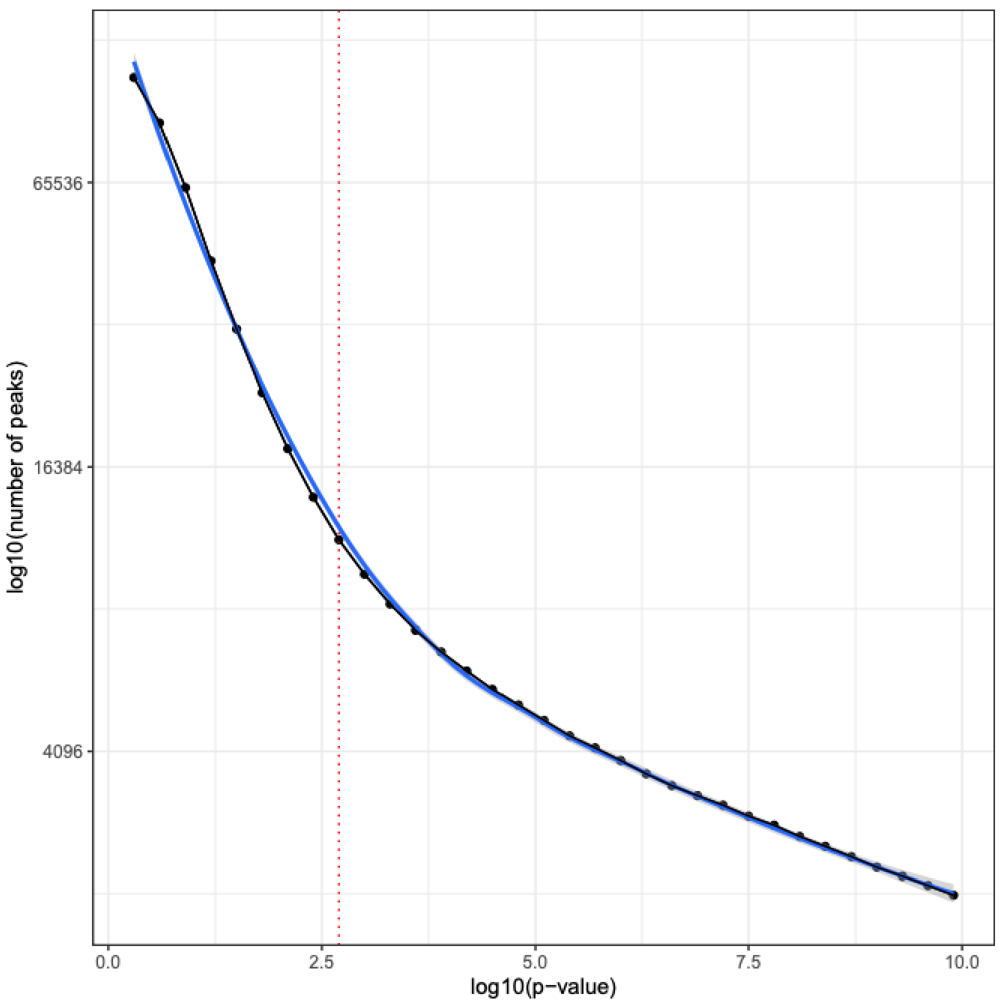
- **bam_flagstat_out**: SAMtools flagstats for unfiltered BAM file. Type: File.
- **bam_stats_out**: SAMtools stats for unfiltered BAM file. Type: File.
- **final_bam_flagstat_out**: SAMtools flagstats for filtered BAM file. Type: File.
- **bed_file_out**:: Aligned reads in BED format. Type: File.
- **final_bam_out**: Final BAM file filtered and sorted. Type: File.
- **bam_index_out**: BAM index file. Type: File.
- **pbc_out**: Library complexity report. Type: File.
- **phantompeakqualtools_output_out**: Phantompeakqualtools main output. Type: File.
- **phantompeakqualtools_output_savp**: Phantompeakqualtools SAVP output. Type: File.
- **subsample_pseudoreplicate_gzip_out**: Subsample pseudoreplicates tagAlign gzipped. Type: File[].
- **subsample_tagalign_out**: Subsample tagAlign gzipped. Type: File[].
- **subsample_subsample_out**: Subsample shuffled tagAlign gzipped. Type: File[].

### Peak caller workflow using MACS2

This workflow uses MACS2 as peak caller tool. The annotation is created using Homer and TDF files are created with igvtools.



MACS2 is executed two times. First, the **cutoff-analysis** option is used to execute a cutoff value analysis which is used to estimate a proper value for the p-value used by MACS2 (for more detailed explanation read this thread).



RSeQC is also executed for quality control.

**Inputs**

- **homer_genome**: Homer genome name. Type: string. Required.
- **genome_fasta** Genome FASTA file. Type: File. Required. Variable GENOME_FASTA in the Jupyter Notebooks.
- **genome_gtf**: Genome GTF file. Type: File. Required. Variable GENOME_GTF in the Jupyter Notebooks.
- **tagAlign_gz**: Tag aligned file created with the BWA based alignment and quality control workflow. Type: File. Required.
- **macs_callpeaks_g**: Genome mapeable size as defined in MACS2. Type: string. Required. Variable GENOME_MAPPABLE_SIZE in the Jupyter Notebooks.
- **macs_callpeaks_q**: MACS2 **q** option. Starting qvalue (minimum FDR) cutoff to call significant regions. Type: float. Required. Variable fdr in the Jupyter Notebooks.
- **nomodel**: MACS2 nomodel option. MACS will bypass building the shifting model. Type: boolean. Optional.
- **extsize**: MACS2 extsize option. MACS uses this parameter to extend reads in 5'->3' direction to fix-sized fragments. Type: int. Optional.

**Outputs**

- **readQC_plots**: RSeQC plots. Type: File[]
- **macs_cutoff_pdf** MACS2 cutoff analysis plot in PDF format. Type: File
- **macs_cutoff_inflection**: MACS2 inflection q value used for the second round. Type: File
- **macs_callpeak_q_value_narrowPeak**: Final MACS2 narrowpeak file. Type: File
- **macs_callpeak_q_value_xls**: Final MACS2 XLS file. Type: File
- **macs_callpeak_q_value_bed**: Final MACS2 BED file. Type: File
- **homer_annotate_peaks_output**: Homer annotated BED file. Type: File
- **homer_annotate_peaks_annStats**: Homer annotation statistics. Type: File
- **lambda_tdf_out**: MACS2 lambda file in TDF format. Type: File
- **pileup_tdf_out**: MACS2 pileup file in TDF format. Type: File

#### Differential binding analysis with Diffbind

Differential binding event is detected with Diffbind. Thsi tool will be executed for all comparisons added to the **comparisons** array. See cell number 3 in the notebook **05 - Differential binding analysis.ipynb** (ChIP-Seq workflow).

**Inputs**

- **bamDir**: Directory with the BAM files. Type: Directory. Required.
- **bedDir**: Directory with BED files created from MACS2 peak calling workflow Type: Directory. Required.

**Outputs**

- **outpng**: PNG files created from Diffbind. Type File[]
- **outxls**: XLS files created from Diffbind. Type: File[]
- **outbed** BED files created from Diffbind. Type: File[]

### 2.2.5 Test Project

A test project is available (read-only) at https://ftp.ncbi.nlm.nih.gov/pub/pm4ngs/examples/chipseq-hmgn1

### 2.2.6 Extra requirements

#### Configuring Homer databases

Homer needs organism, promoter and genome databases for the annotation process. These databases are not distributed with the default installation package.

The users need to install the specific databases for the organism analyzed in their projects. The next example is for mouse.

**Using Conda**

```
localhost:~> conda activate /home/veraalva/chipseq-hmgn1/bin/bioconda
localhost:~> perl /home/veraalva/chipseq-hmgn1/bin/bioconda/share/homer-4.10-0/
↪configureHomer.pl -install mouse-o mouse-p mm9
localhost:~> perl /home/veraalva/chipseq-hmgn1/bin/bioconda/share/homer-4.10-0/
↪configureHomer.pl -list | grep -v "^-"

    Current base directory for HOMER is /home/veraalva/chipseq-hmgn1/bin/bioconda/
↪share/homer-4.10-0/

--2019-08-30 12:05:27--  http://homer.ucsd.edu/homer/update.txt
Resolving homer.ucsd.edu (homer.ucsd.edu)... 169.228.63.226
Connecting to homer.ucsd.edu (homer.ucsd.edu)|169.228.63.226|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16187 (16K) [text/plain]
Saving to: '/home/veraalva/chipseq-hmgn1/bin/bioconda/share/homer-4.10-0//update.txt'

/gfs/projects/ngs_templates/cookiecutter/chips 100
↪%[==========================================================================================
↪]  15.81K  --.-KB/s    in 0.07s

2019-08-30 12:05:28 (211 KB/s) - '/home/veraalva/chipseq-hmgn1/bin/bioconda/share/
↪homer-4.10-0//update.txt' saved [16187/16187]

    Updating Settings...
Packages with name conflicts have a trailing -o, -p, or -g
Version Installed   Package Version Description
SOFTWARE
+   homer   v4.10.4 Code/Executables, ontologies, motifs for HOMER
ORGANISMS
+   mouse-o v6.0    Mus musculus (mouse) accession and ontology information
PROMOTERS
+   mouse-p v5.5    mouse promoters (mouse)
GENOMES
+   mm9     v6.0    mouse genome and annotation for UCSC mm9
SETTINGS
```

**Using Docker**

A directory named **data/homer** will be used to store all homer configuration and databases.

```
localhost:~> cd chipseq-hmgn1/data
localhost:~> mkdir -p homer
localhost:~> docker run -u `id -u`:`id -g` -i -t -v `pwd`/homer:/data quay.io/
↪biocontainers/homer:4.10--pl526hc9558a2_0 cp /usr/local/share/homer-4.10-0/config.
↪txt /data/
localhost:~> docker run -u `id -u`:`id -g` -i -t -v `pwd`/homer:/data quay.io/
↪biocontainers/homer:4.10--pl526hc9558a2_0 cp /usr/local/share/homer-4.10-0/update.
↪txt /data/
localhost:~> docker run -u `id -u`:`id -g` -i -t -v `pwd`/homer:/data quay.io/
↪biocontainers/homer:4.10--pl526hc9558a2_0 cp -rf /usr/local/share/homer-4.10-0/data␣
↪/data/
localhost:~> docker run -i -t -v `pwd`/homer/config.txt:/usr/local/share/homer-4.10-0/
↪config.txt -v `pwd`/homer/update.txt:/usr/local/share/homer-4.10-0/update.txt -v␣
↪`pwd`/homer/data:/usr/local/share/homer-4.10-0/data  quay.io/biocontainers/homer:4.
↪10--pl526hc9558a2_0 perl /usr/local/share/homer-4.10-0/configureHomer.pl -install␣
↪mouse-o mouse-p mm9
```

(continues on next page)

(continued from previous page)

```
localhost:~> docker run -i -t -v `pwd`/homer/config.txt:/usr/local/share/homer-4.10-0/
↪config.txt -v `pwd`/homer/update.txt:/usr/local/share/homer-4.10-0/update.txt -v␣
↪`pwd`/homer/data:/usr/local/share/homer-4.10-0/data  quay.io/biocontainers/homer:4.
↪10--pl526hc9558a2_0 perl /usr/local/share/homer-4.10-0/configureHomer.pl -list |␣
↪grep -v "^-"

    Current base directory for HOMER is /usr/local/share/homer-4.10-0/

Connecting to homer.ucsd.edu (169.228.63.226:80)
update.txt           100% |*******************************| 16187   0:00:00 ETA
    Updating Settings...
Packages with name conflicts have a trailing -o, -p, or -g
Version Installed   Package Version Description
SOFTWARE
+   homer   v4.10.4 Code/Executables, ontologies, motifs for HOMER
ORGANISMS
+   mouse-o v6.0    Mus musculus (mouse) accession and ontology information
PROMOTERS
+   mouse-p v5.5    mouse promoters (mouse)
GENOMES
+   mm9     v6.0    mouse genome and annotation for UCSC mm9
SETTINGS
```

## 2.3 Detection of binding events from ChIP-exo data

**Warning:** Read the Background Information before proceeding with these steps

**Warning:** Read the *Project Templates Installation* notes to have the **cookiecutter** available in you shell depending on the execution environment you will be using.

### 2.3.1 Samples description file

A TSV file named **factors.txt** is the main file for the projects and workflow. This file should be created before any project creation. It is the base of the workflow and should be copied to the folder **data/{{dataset_name}}** just after creating the project structure.

The initial sample names, file name prefixes and metadata are specified on it.

It should have the following columns:

| id | SampleID | condition | replicate |
|---|---|---|---|
| classical01 | SRR4053795 | classical | 1 |
| classical01 | SRR4053796 | classical | 2 |
| nonclassical01 | SRR4053802 | nonclassical | 1 |
| nonclassical01 | SRR4053803 | nonclassical | 2 |

**Warning:** Columns names are required and are case sensitive.

**Columns**

- **id**: Sample names. It can be different of sample file name.
- **SampleID**: This is the prefix of the sample file name.

  For single-end data the prefix ends in the file extension. In this case, for the first column, a file name named **SRR4053795.fastq.gz** must exist.

  For paired-end data the files **SRR4053795_1.fastq.gz** and **SRR4053795_2.fastq.gz** must exist.

  The data files should be copied to the folder **data/{{dataset_name}}/**.
- **condition**: Conditions to analyze or group the samples. Avoid using non alphanumeric characters.

  For RNASeq projects the differential gene expression will be generated comparing these conditions. If there are multiple conditions all comparisons will be generated. It must be at least two conditions.

  For ChIPSeq projects differential binding events will be detected comparing these conditions. If there are multiple conditions all comparisons will be generated. It must be at least two conditions.

  For ChIPexo projects the samples of the same condition will be grouped for the peak calling with MACE.
- **replicate**: Replicate number for samples.

## 2.3.2 Installation

### ChIP-exo workflow with Conda/Bioconda

The ChIP-exo project structure is created using the conda environment named **templates**.

First step is to activate the **templates** environment:

```
localhost:~> conda activate templates
```

Then, a YAML file (for this example I will call this file: **chipexo-single.yaml**) with your project detail should be created.

```yaml
default_context:
    author_name: "Roberto Vera Alvarez"
    user_email: "veraalva@ncbi.nlm.nih.gov"
    project_name: "chipexo-single"
    dataset_name: "PRJNA338159"
    is_data_in_SRA: "y"
    ngs_data_type: "ChIP-exo"
    sequencing_technology: "single-end"
    create_demo: "n"
    number_spots: "1000000"
    organism: "human"
    genome_dir: "/gfs/data/genomes/NCBI/Escherichia_coli/K-12/MG1655/"
    genome_name: "NC_000913.3"
    aligner_index_dir: "{{ cookiecutter.genome_dir}}/BWA"
    genome_fasta: "{{ cookiecutter.genome_dir}}/NC_000913.3.fa"
    genome_gtf: "{{ cookiecutter.genome_dir}}/NC_000913.3.gtf"
```

(continues on next page)

(continued from previous page)

```
    genome_gff: "{{ cookiecutter.genome_dir}}/NC_000913.3.gff"
    genome_gff3: "{{ cookiecutter.genome_dir}}/NC_000913.3.gff3"
    genome_bed: "{{ cookiecutter.genome_dir}}/NC_000913.3.bed"
    genome_chromsizes: "{{ cookiecutter.genome_dir}}/NC_000913.3.sizes"
    genome_mappable_size: "3714120"
    genome_blacklist: "{{ cookiecutter.genome_dir}}/NC_000913.3.bed"
    fold_change: "2.0"
    fdr: "0.05"
    use_docker: "n"
    pull_images: "n"
    use_conda: "y"
    cwl_runner: "cwl-runner"
    cwl_workflow_repo: "https://github.com/ncbi/cwl-ngs-workflows-cbb"
    create_virtualenv: "n"
    use_gnu_parallel: "y"
    max_number_threads: "16"
```

A full description of this parameters are *here*.

After the **chipexo-single.yaml** is created the project structure should be created using this command obtaining the following output.

```
localhost:~> cookiecutter --no-input --config-file chipexo-single.yaml https://github.
↪com/ncbi/pm4ngs.git
Checking ChIP-exo workflow dependencies .......... Done
localhost:~>
```

This process should create a project organizational structure like this:

```
localhost:~> tree chipexo-single
chipexo-single
├── bin
│   ├── bioconda (This directory include a conda envs for all bioinfo tools)
│   ├── cwl-ngs-workflows-cbb (CWL workflow repo cloned here)
│   └── jupyter  (This directory include a conda envs for Jupyter notebooks)
├── config
│   └── init.py
├── data
│   └── PRJNA338159
├── index.html
├── LICENSE
├── notebooks
│   ├── 00 - Project Report.ipynb
│   ├── 01 - Pre-processing QC.ipynb
│   ├── 02 - Samples trimming.ipynb
│   ├── 03 - Alignments.ipynb
│   ├── 04 - Peak Calling.ipynb
│   └── 05 - MEME Motif.ipynb
├── README.md
├── requirements
│   └── python.txt
├── results
│   └── PRJNA338159
├── src
└── tmp

10 directories, 11 files
```

Now you should copied the **factors.txt** file to the folder: **data/PRJNA338159**.

After this process, **cookiecutter** should have created create two virtual environment for this workflow.

The first one is for running the Jupyter notebooks which require Python 3.6+ and it is named: **jupyter**. It can be manually installed as described in here.

The second environment is be used to install all Bioinformatics tools required by the workflow and it will be named: **bioconda**.

You can verify the environments running this command:

```
localhost:~> conda env list
# conda environments:
#
base                  *  /gfs/conda
tempates                 /gfs/conda/envs/templates
                         /home/veraalva/chipexo-single/bin/bioconda
                         /home/veraalva/chipexo-single/bin/jupyter

localhost:~>
```

Please, note that the Conda prefix **/gfs/conda** will be different in you host. Also, note that the **bioconda** and **jupyter** envs are inside the **bin** directory of your project keeping them static inside the project organizational structure.

## ChIP-exo workflow usage with Conda/Bioconda

For start using the workflow you need to activate the conda environments **bioconda** and **jupyter**.

```
localhost:~> conda activate /home/veraalva/chipexo-single/bin/bioconda
localhost:~> conda activate --stack /home/veraalva/chipexo-single/bin/jupyter
```

Note the **–stack** option to have both environment working at the same time. Also, the order is important, **bioconda** should be activated before **jupyter**.

Test the conda envs:

```
localhost:~> which fastqc
/home/veraalva/chipexo-single/bin/bioconda/bin/fastqc
localhost:~> which jupyter
/home/veraalva/chipexo-single/bin/jupyter/bin/jupyter
```

Note that the **fastqc** tools is installed in the **bioconda** env and the **jupyter** command is installed in the **jupyter** env.

Then, you can start the jupyter notebooks.

```
localhost:~> jupyter notebook
```

If the workflow is deployed in a remote machine using SSH access the correct way to start the notebooks is:

```
localhost:~> jupyter notebook --no-browser --ip='0.0.0.0'
```

In this case the option **–ip='0.0.0.0'** will server the Jupyter notebook on all network interfaces and you can access them from your desktop browser using the port returned by the Jupyter server.

Finally, you should navegate in your browser to the **notebooks** folder and start executing all notebooks by their order leaving the **00 - Project Report.ipynb** to the end.

### ChIP-exo workflow with Docker

In this case, the ChIP-exo project structure is created using the Python virtual environment as described *here*

First step is to activate the Python virtual environment.

```
localhost:~> source venv-templates/bin/activate
```

Then, a YAML file (for this example I will call this file: **chipexo-single.yaml**) with your project detail should be created.

```
default_context:
    author_name: "Roberto Vera Alvarez"
    user_email: "veraalva@ncbi.nlm.nih.gov"
    project_name: "chipexo-single"
    dataset_name: "PRJNA338159"
    is_data_in_SRA: "y"
    ngs_data_type: "ChIP-exo"
    sequencing_technology: "single-end"
    create_demo: "n"
    number_spots: "1000000"
    organism: "human"
    genome_dir: "/gfs/data/genomes/NCBI/Escherichia_coli/K-12/MG1655/"
    genome_name: "NC_000913.3"
    aligner_index_dir: "{{ cookiecutter.genome_dir}}/BWA"
    genome_fasta: "{{ cookiecutter.genome_dir}}/NC_000913.3.fa"
    genome_gtf: "{{ cookiecutter.genome_dir}}/NC_000913.3.gtf"
    genome_gff: "{{ cookiecutter.genome_dir}}/NC_000913.3.gff"
    genome_gff3: "{{ cookiecutter.genome_dir}}/NC_000913.3.gff3"
    genome_bed: "{{ cookiecutter.genome_dir}}/NC_000913.3.bed"
    genome_chromsizes: "{{ cookiecutter.genome_dir}}/NC_000913.3.sizes"
    genome_mappable_size: "3714120"
    genome_blacklist: "{{ cookiecutter.genome_dir}}/NC_000913.3.bed"
    fold_change: "2.0"
    fdr: "0.05"
    use_docker: "y"
    pull_images: "y"
    use_conda: "n"
    cwl_runner: "cwl-runner"
    cwl_workflow_repo: "https://github.com/ncbi/cwl-ngs-workflows-cbb"
    create_virtualenv: "y"
    use_gnu_parallel: "y"
    max_number_threads: "16"
```

A full description of this parameters are *here*.

After the **chipexo-single.yaml** is created the project structure should be created using this command obtaining the following output.

```
localhost:~> cookiecutter --no-input --config-file chipexo-single.yaml https://github.
↪com/ncbi/pm4ngs.git
Checking ChIP-exo workflow dependencies .......... Done
localhost:~>
```

This process should create a project organizational structure like this:

```
localhost:~> tree chipexo-single
chipexo-single
```

(continues on next page)

(continued from previous page)

```
├── bin
├── config
│   └── init.py
├── data
│   └── PRJNA338159
├── index.html
├── LICENSE
├── notebooks
│   ├── 00 - Project Report.ipynb
│   ├── 01 - Pre-processing QC.ipynb
│   ├── 02 - Samples trimming.ipynb
│   ├── 03 - Alignments.ipynb
│   ├── 04 - Peak Calling.ipynb
│   └── 05 - MEME Motif.ipynb
├── README.md
├── requirements
│   └── python.txt
├── results
│   └── PRJNA338159
├── src
├── tmp
└── venv

11 directories, 11 files
```

Now you should copied the **factors.txt** file to the directory: **data/PRJNA338159**.

After this process, **cookiecutter** should have pulled all docker images require by the project.

### ChIP-exo workflow usage with Docker

For start using the workflow you need to activate the Python environment inside the project.

```
localhost:~> source venv/bin/activate
```

Then, you can start the jupyter notebooks now.

```
localhost:~> jupyter notebook
```

If the workflow is deployed in a remote machine using SSH access the correct way to start the notebooks is:

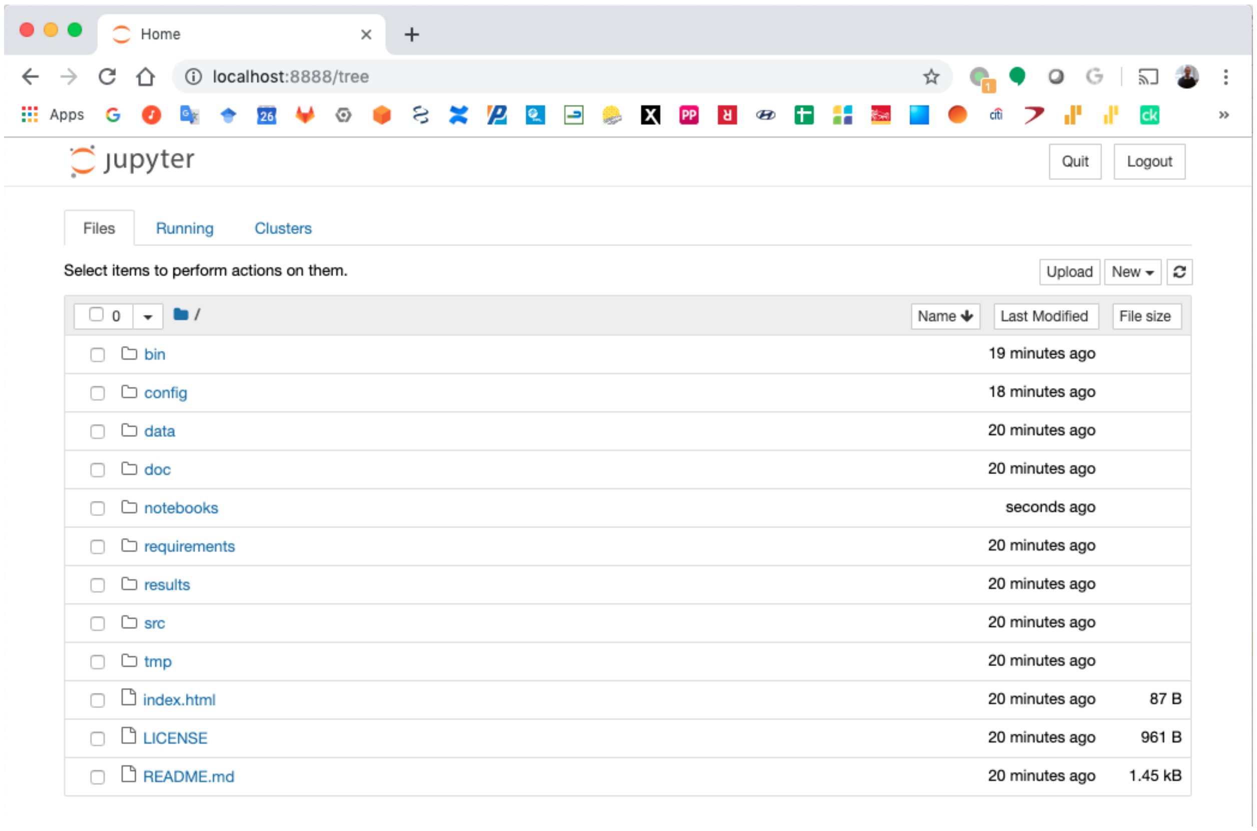
```
localhost:~> jupyter notebook --no-browser --ip='0.0.0.0'
```

In this case the option **–ip='0.0.0.0'** will server the Jupyter notebook on all network interfaces and you can access them from your desktop browser using the port returned by the Jupyter server.

Finally, you should navigate in your browser to the **notebooks** directory and start executing all notebooks by their order leaving the **00 - Project Report.ipynb** to the end.

## 2.3.3 Jupyter Notebook Server

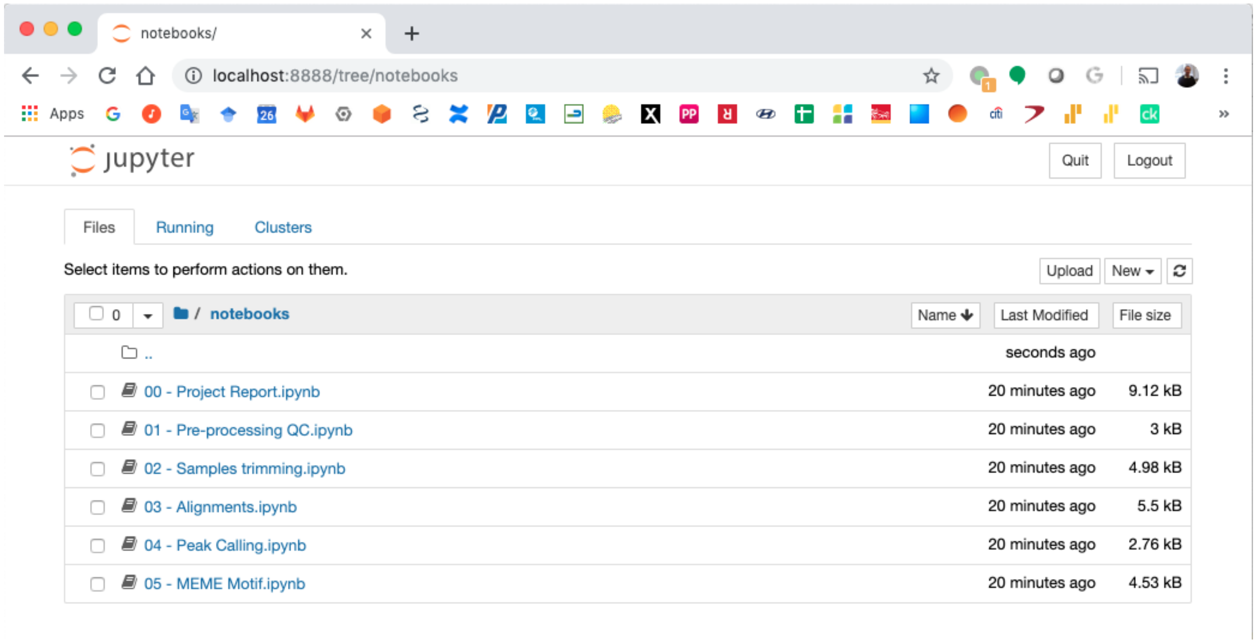
## Top-level directories from the Jupyter server viewed in a web browser



Select items to perform actions on them.

| Name | Last Modified | File size |
|---|---|---|
| bin | 19 minutes ago | |
| config | 18 minutes ago | |
| data | 20 minutes ago | |
| doc | 20 minutes ago | |
| notebooks | seconds ago | |
| requirements | 20 minutes ago | |
| results | 20 minutes ago | |
| src | 20 minutes ago | |
| tmp | 20 minutes ago | |
| index.html | 20 minutes ago | 87 B |
| LICENSE | 20 minutes ago | 961 B |
| README.md | 20 minutes ago | 1.45 kB |

## Notebook generated fro the Chip-exo data analysis



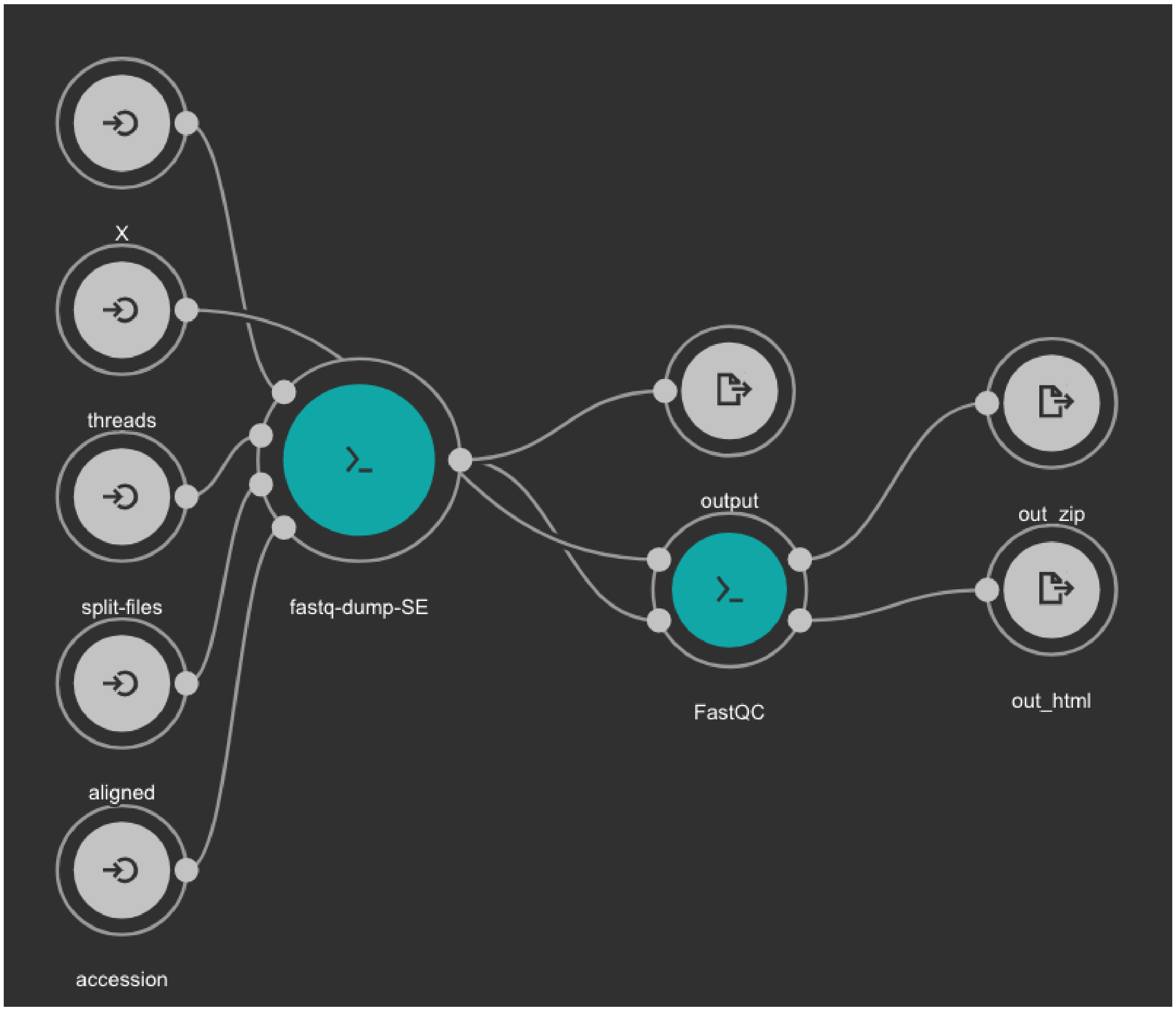
Select items to perform actions on them.

| Name | Last Modified | File size |
|---|---|---|
| .. | seconds ago | |
| 00 - Project Report.ipynb | 20 minutes ago | 9.12 kB |
| 01 - Pre-processing QC.ipynb | 20 minutes ago | 3 kB |
| 02 - Samples trimming.ipynb | 20 minutes ago | 4.98 kB |
| 03 - Alignments.ipynb | 20 minutes ago | 5.5 kB |
| 04 - Peak Calling.ipynb | 20 minutes ago | 2.76 kB |
| 05 - MEME Motif.ipynb | 20 minutes ago | 4.53 kB |

### 2.3.4 CWL workflows

#### SRA download and QC workflow

This CWL workflow is designed to download FASTQ files from the NCBI SRA database using fastq-dump and then, execute fastqc generating a quality control report of the sample.



**Inputs**

- **accession**: SRA accession ID. Type: string. Required.
- **aligned**: Used it to download only aligned reads. Type: boolean. Optional.
- **split-files**: Dump each read into separate file. Files will receive suffix corresponding to read number. Type: boolean. Optional.
- **threads**: Number of threads. Type: int. Default: 1. Optional.
- **X**: Maximum spot id. Optional.

**Outputs**

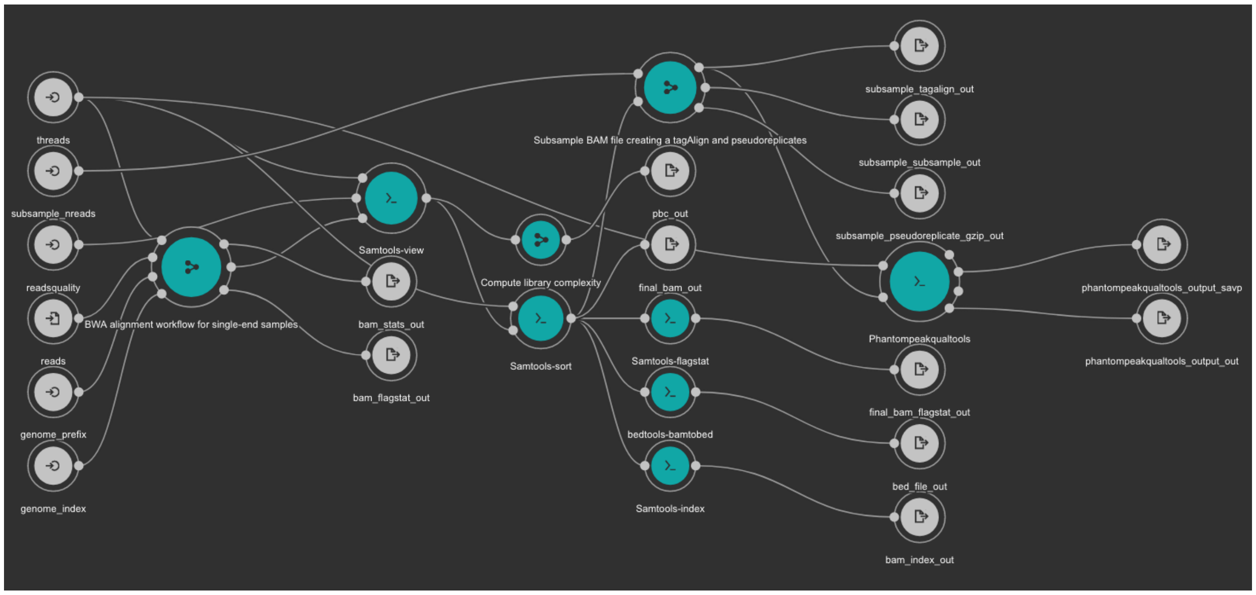
- **output**: Fastq files downloaded. Type: File[]
- **out_zip**: FastQC report ZIP file. Type: File[]
- **out_html**: FastQC report HTML. Type: File[]

### Samples Trimming

Our workflows uses Trimmomatic for read trimming. The Jupyter notebooks uses some basic Trimmomatic options that need to be modified depending on the FastQC quality control report generated for the sample.

### BWA based alignment and quality control workflow

This workflow use BWA as base aligner. It also use SAMtools and bedtools for file conversion and statistics report. Finally, Phantompeakqualtools is used to generate quality control report for the processed samples.



**Inputs**

- **genome_index**: Aligner indexes directory. Type: Directory. Required. Variable ALIGNER_INDEX in the Jupyter Notebooks.
- **genome_prefix**: Prefix of the aligner indexes. Generally, it is the name of the genome FASTA file. It can be used as os.path.basename(GENOME_FASTA) in the Jupyter Notebooks. Type: string. Required.
- **readsquality**: Minimum MAPQ value to use reads. We recommend for ChIP_exo dataa value of: 30. Type: int. Required.
- **threads**: Number of threads. Type: int. Default: 1. Optional.
- **subsample_nreads**: Number of reads to be used for the subsample. We recomend for ChIP_exo dataa value of: 500000. Type: int. Required.

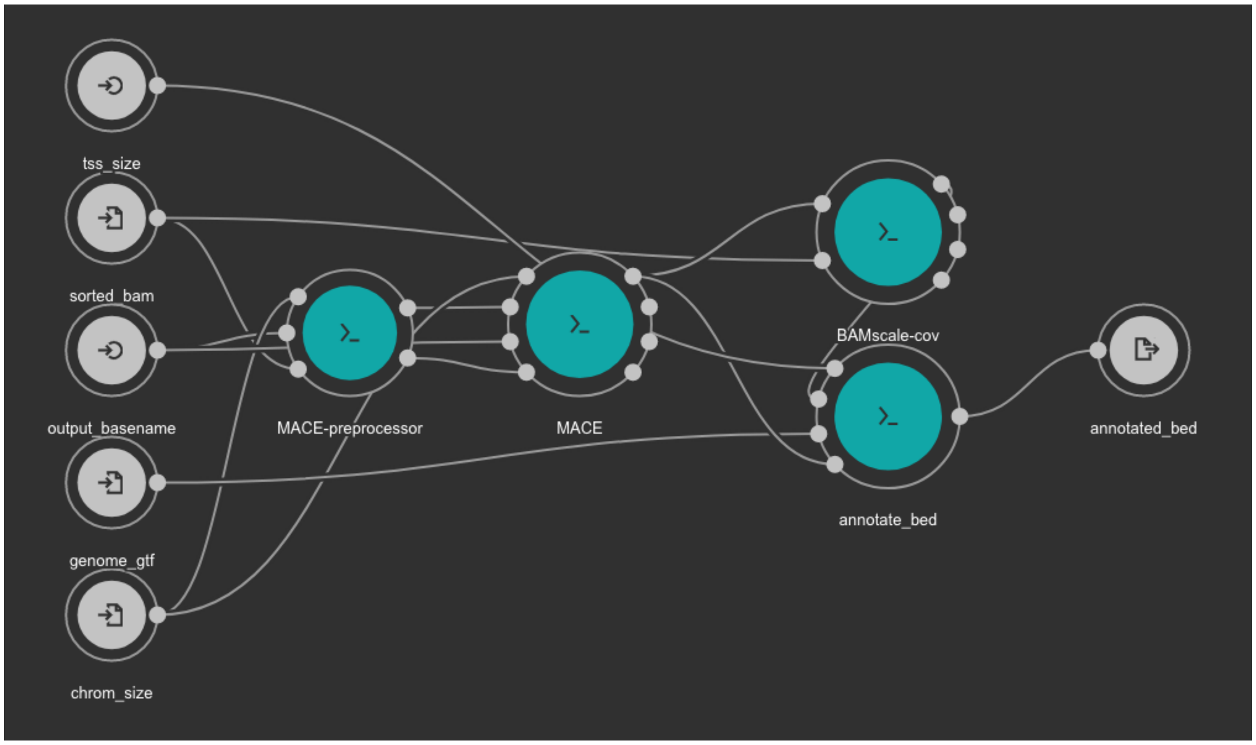
- **reads**: FastQ file to be processed. Type: File. Required.

**Outputs**

- **bam_flagstat_out**: SAMtools flagstats for unfiltered BAM file. Type: File.
- **bam_stats_out**: SAMtools stats for unfiltered BAM file. Type: File.
- **final_bam_flagstat_out**: SAMtools flagstats for filtered BAM file. Type: File.
- **bed_file_out**:: Aligned reads in BED format. Type: File.
- **final_bam_out**: Final BAM file filtered and sorted. Type: File.
- **bam_index_out**: BAM index file. Type: File.
- **pbc_out**: Library complexity report. Type: File.
- **phantompeakqualtools_output_out**: Phantompeakqualtools main output. Type: File.
- **phantompeakqualtools_output_savp**: Phantompeakqualtools SAVP output. Type: File.
- **subsample_pseudoreplicate_gzip_out**: Subsample pseudoreplicates tagAlign gzipped. Type: File[].
- **subsample_tagalign_out**: Subsample tagAlign gzipped. Type: File[].
- **subsample_subsample_out**: Subsample shuffled tagAlign gzipped. Type: File[].

### Peak caller workflow using MACE

This workflow uses MACE as peak caller tool. The annotation is created from the GTF file using a *in-house* python script available here. BAMscale is used for the quantification of the resulting peaks.



**Inputs**

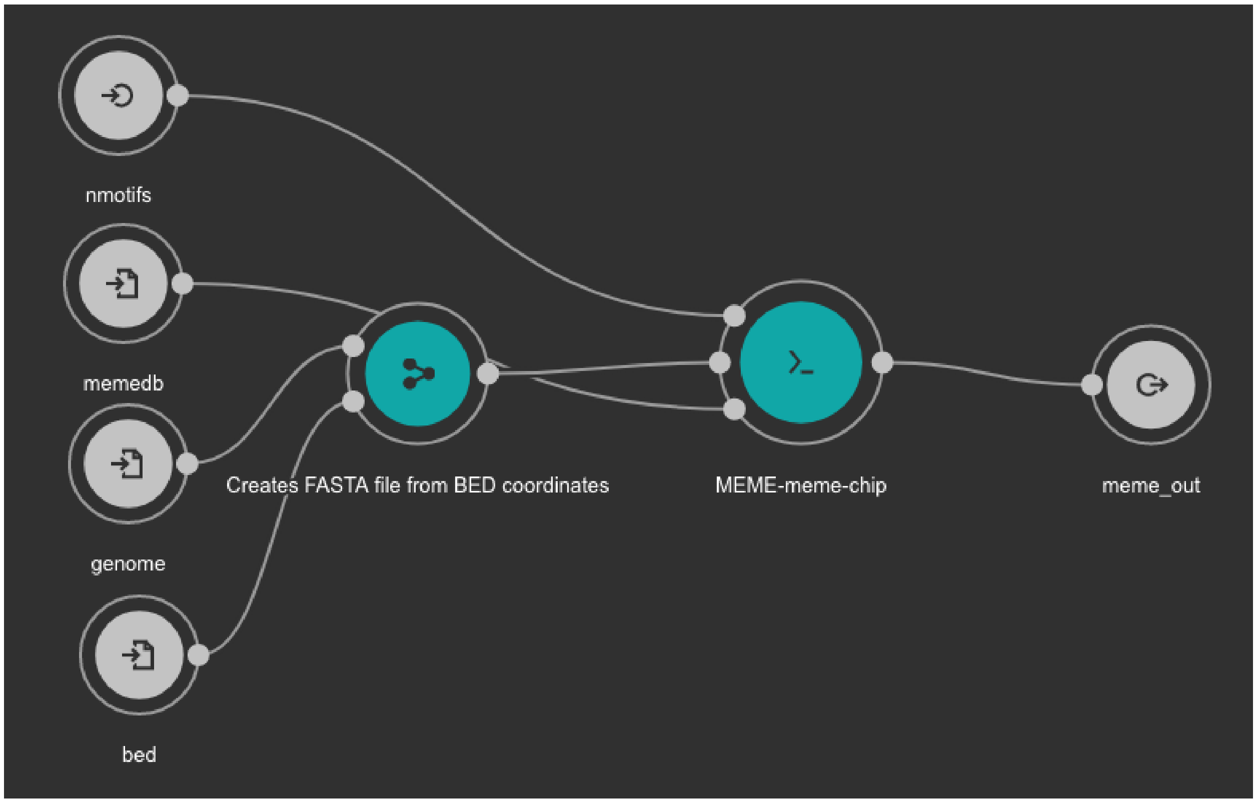
- **chrom_size**: Chromosome size file. Tab or space separated text file with 2 columns: first column is chromosome name, second column is size of the chromosome. Type: File. Required. Variable GENOME_CHROMSIZES in the Jupyter Notebooks.
- **output_basename**: Prefix for the output file. Type: string. Required.
- **genome_gtf**: Genome GTF file. Variable GENOME_GTF in the Jupyter Notebooks. Type: File. Required.
- **tss_size**: Number of bp to use for TSS definition. We recommend use 1000. Type: int. Required.

**Outputs**

- **annotated_bed**: Annotated detected peaks in BED format. Type: File

### MEME Motif detection workflow

Motif detection is executed using the MEME suite.



**Inputs**

- **bed**: BED file with detected peaks. Type: File. Required.
- **memedb**: MEME database for use by Tomtom and CentriMo. Type: File. Required.

- **genome**: Genome FASTA file. Variable GENOME_FASTA in the Jupyter Notebooks. Type: File. Required.
- **nmotifs**: Maximum number of motifs to find. We recommend use 10. Type: int. Required.

**Outputs**

- **meme_out**: MEME output directory. Type: Directory

### MEME databases

MEME workflow depends on the MEME databases. Go to the MEME Suite Download page: http://meme-suite.org/doc/download.html

Download the latest version for the Motif Databases and GOMo Databases.

The downloaded files should be uncompressed in a directory **data/meme**. The final directory should be:

```
localhost:~> cd data
localhost:~> mkdir meme
localhost:~> cd meme
localhost:~> wget http://meme-suite.org/meme-software/Databases/motifs/motif_
↪databases.12.18.tgz
localhost:~> wget http://meme-suite.org/meme-software/Databases/gomo/gomo_databases.3.
↪2.tgz
localhost:~> tar xzf motif_databases.12.18.tgz
localhost:~> tar xzf gomo_databases.3.2.tgz
localhost:~> rm gomo_databases.3.2.tgz motif_databases.12.18.tgz
localhost:~> tree -d
.
├── gomo_databases
└── motif_databases
    ├── ARABD
    ├── CIS-BP
    ├── CISBP-RNA
    ├── ECOLI
    ├── EUKARYOTE
    ├── FLY
    ├── HUMAN
    ├── JASPAR
    ├── MALARIA
    ├── MIRBASE
    ├── MOUSE
    ├── PROKARYOTE
    ├── PROTEIN
    ├── RNA
    ├── TFBSshape
    ├── WORM
    └── YEAST

19 directories
```

See also an example in our test project: https://ftp.ncbi.nlm.nih.gov/pub/pm4ngs/examples/chipexo-single/data/

### 2.3.5 Test Project

A test project is available (read-only) at https://ftp.ncbi.nlm.nih.gov/pub/pm4ngs/examples/chipexo-single/

### 2.3.6 Extra requirements

#### Creating BWA indexes

This workflow uses BWA for sequence alignment. The BWA index creation is not included in the workflow, that's why we are including an small section here to describe how the BWA indexes can be created.

The **genome.fa** file should be copied to the genome directory.

```
localhost:~> conda activate /home/veraalva/chipexo-single/bin/bioconda
localhost:~> conda activate --stack /home/veraalva/chipexo-single/bin/jupyter
localhost:~> cd chipexo-single/data
localhost:~> mkdir genome
localhost:~> cd genome
localhost:~> mkdir BWA
localhost:~> cd BWA
localhost:~> cwl-runner --no-container ../../../bin/cwl-ngs-workflows-cbb/tools/BWA/
↪bwa-index.cwl --sequences genome.fa
localhost:~> cd ..
localhost:~> tree
.
├── BWA
│   ├── genome.fa
│   ├── genome.fa.amb
│   ├── genome.fa.ann
│   ├── genome.fa.bwt
│   ├── genome.fa.pac
│   └── genome.fa.sa
└── genome.fa

1 directory, 7 files
```

Here all files inside the directory **BWA** are created by the workflow.

#### Creating BED files from GTF

For generating a BED file from a GTF.

The **genes.gtf** file should be copied to the genome directory.

```
localhost:~> conda activate /home/veraalva/chipexo-single/bin/bioconda
localhost:~> conda activate --stack /home/veraalva/chipexo-single/bin/jupyter
localhost:~> cd chipexo-single/data
localhost:~> mkdir genome
localhost:~> cd genome
localhost:~> cwl-runner --no-container ../../bin/cwl-ngs-workflows-cbb/workflows/UCSC/
↪gtftobed.cwl --gtf genes.gtf
localhost:~> tree
.
├── genes.bed
├── genes.genePred
├── genes.gtf
```

(continues on next page)

(continued from previous page)

```
└── genome.fa

0 directory, 4 files
```

Here the files **genes.bed** and **genes.genePred** are created from the workflow.

# CHAPTER THREE

# EXTRA LINKS

## 3.1 Project Templates Installation

### 3.1.1 Project Templates with Python virtual environment

Install Cookiecutter and other basic Python packages using the **requirements.txt** file.

```
localhost:~> virtualenv venv-templates
localhost:~> source venv-templates/bin/activate
localhost:~> pip install -r https://raw.githubusercontent.com/ncbi/pm4ngs/master/
↪requirements.txt
```

### 3.1.2 Project Templates with Conda/BioConda

Conda should be already installed and configured.

```
localhost:~> wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.
↪sh
localhost:~> sh Miniconda3-latest-MacOSX-x86_64.sh
```

A Conda virtual environment will be created with name **templates** using these instructions:

```
localhost:~> wget https://raw.githubusercontent.com/ncbi/pm4ngs/master/conda-
↪requirements.yaml
localhost:~> conda env create -f conda-requirements.yaml
```

If Conda is installed with prefix **/gfs/conda** you should see the available environments like in this block:

```
localhost:~> conda env list
# conda environments:
#
base                     /gfs/conda
templates             *  /gfs/conda/envs/templates

localhost:~>
```

To activate the templates env

```
localhost:~> conda activate templates
localhost:~>
```

### 3.1.3 Using the Template Project

This project template uses the workflow defined in the project cwl-ngs-workflows-cbb. Depending on the execution environment selected: **docker**, **conda** or **programs in the path** the project template will check the availability of the Bioinformatic tools required by the workflow.

## 3.2 Project Description YAML file

Cookiecutter accept a YAML file as a config file for the project template creation. This YAML file is created from the parameters:

```
{
    "author_name": "Roberto Vera Alvarez",
    "email": "veraalva@ncbi.nlm.nih.gov",
    "project_name": "my_ngs_project",
    "dataset_name": "my_dataset_name",
    "is_data_in_SRA": "y" or "n",
    "ngs_data_type": ["RNA-Seq", "ChIP-Seq", "ChIP-exo"],
    "sequencing_technology": ["single-end", "paired-end"],
    "create_demo": "y" or "n",
    "number_spots": "5000000",
    "organism": "human",
    "genome_dir": "/gfs/data/genomes/igenomes/Homo_sapiens/UCSC/hg19",
    "genome_name": "hg19",
    "aligner_index_dir": "{{ cookiecutter.genome_dir }}/ALIGNER",
    "genome_fasta": "{{ cookiecutter.genome_dir }}/genome.fa",
    "genome_gtf": "{{ cookiecutter.genome_dir }}/genome.gtf",
    "genome_gff": "{{ cookiecutter.genome_dir }}/genome.gff",
    "genome_gff3": "{{ cookiecutter.genome_dir }}/genome.gff3",
    "genome_bed": "{{ cookiecutter.genome_dir }}/genome.bed",
    "genome_chromsizes": "{{ cookiecutter.genome_dir }}/genome.sizes",
    "genome_mappable_size": "hg19",
    "genome_blacklist": "{{ cookiecutter.genome_dir }}/hg19-blacklist.bed",
    "fold_change": "2.0",
    "fdr": "0.05",
    "use_docker": "y" or "n",
    "pull_images": "y" or "n",
    "use_conda": "y" or "n",
    "cwl_runner": "cwl-runner",
    "cwl_workflow_repo": "https://github.com/ncbi/cwl-ngs-workflows-cbb",
    "create_virtualenv": "y" or "n",
    "use_gnu_parallel": "y" or "n",
    "max_number_threads": "16"
}
```

**Parameters**

- **author_name**: Project author name
- **email**: Author's email
- **project_name**: Name of the project with no space nor especial characters. This will be used as project folder's name.

- **dataset_name**: Dataset to process name with no space nor especial characters. This will be used as folder name to group the data. This folder will be created under the **data/{{dataset_name}}** and **results/{{dataset_name}}**.
- **is_data_in_SRA**: If the data is in the SRA set this to **y**. A CWL workflow to download the data from the SRA database to the folder **data/{{dataset_name}}** and execute FastQC on it will be included in the **01 - Pre-processing QC.ipynb** notebook.

  If this option is set to **n**, the fastq files should be copied to the folder **data/{{dataset_name}}/**
- **ngs_data_type: Select one of the available technologies to process:**
  1. RNA-Seq
  2. ChIP-Seq
  3. ChIP-exo
- **sequencing_technology: Select one of the available sequencing technologies in your data:**
  1. single-end
  2. paired-end

  Mixed datasets with single and paired-end samples should be processed independently.
- **create_demo**: If the data is downloaded from the SRA and this option is set to **y**, then only the number of spots specified in the next variable will be downloaded. Useful to test the workflow.
- **number_spots**: Number of sport to download from the SRA database. It is ignored is the **create_demo** is set to **n**.
- **organism**: Organism to process, e.g. human. This is used to link the selected genes to the NCBI gene database.
- **genome_dir**: Absolute path to the directory with the genome annotation to be used by the workflow.
- **genome_name**: Genome name , e.g. hg38 or mm10.
- **aligner_index_dir**: Absolute path to the directory with the aligner indexes.
- **genome_fasta**: Absolute path to the directory to the genome fasta.
- **genome_gtf**: Absolute path to the directory with the genome GTF.
- **genome_gff**: Absolute path to the directory with the genome GFF.
- **genome_gff3**: Absolute path to the directory with the genome GFF3.
- **genome_bed**: Absolute path to the directory with the genome BED. All these files are note required to exist. It depends on the workflow executed.
- **genome_chromsizes**: Genome chromosome sizes file like hg19.chrom.sizes.
- **genome_mappable_size**: Genome mappable size used by MACS. For human can be hg38 or in case of other genomes it is a number.
- **genome_blacklist**: Genome blacklist file.
- **fold_change**: A real number used as fold change value, e.g. 2.0.
- **fdr**: Adjusted P-Value to be used, e.g. 0.05.
- **use_docker**: Set this to **y** if you will be using Docker.
- **pull_images**: Set this to **y** if you want pull the required docker images during the project structure creation.

- **use_conda**: Set this to **y** if you want to use Conda. The environments required by the **ngs_data_type** to process will be installed during the project structure creation.
- **cwl_runner**: Absulute path to the cwl-runner.
- **cwl_workflow_repo**: Always use: https://github.com/ncbi/cwl-ngs-workflows-cbb. This repo will be cloned in the **bin** folder.
- **create_virtualenv**: Set this to **y** if not using Docker nor Conda for creating a Python virtual environment in a folder **venv**.
- **use_gnu_parallel**: Use GNU Parallel for parallel execution of the jobs.
- **max_number_threads**: Number of threads available in the host

# CHAPTER FOUR

# REFERENCE

1. Vera Alvarez R, Pongor LS, Mariño-Ramírez L and Landsman D. Containerized open-source framework for NGS data analysis and management [version 1; not peer reviewed]. F1000Research 2019, 8(ISCB Comm J):1229 (poster) (doi: 10.7490/f1000research.1117155.1)

# CHAPTER FIVE

# PUBLIC DOMAIN NOTICE