XICRA pipeline

Release 1.2.4

Jose F Sanchez-Herrero

Dec 17, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION

XICRA is a pipeline that integrates multiple bioinformatic tools for the analysis of paired-end or single end reads from small RNA-seq data. It describes all major RNA biotypes present in the samples including miRNA and isomiRs, tRNA fragments (tRFs) and piwi associated RNAs (piRNAs). Results are generated for each sample and summarized for all samples in a single expression matrix.

The pipeline is written in Python with a modular architecture and based on open-source software and databases engines. The design of this bioinformatic tool allows miRNA analysis at different levels.

Multiple tasks are performed by several modules including:

- preparation of raw data
- quality analysis and trimming of the adapters
- merge reads (R1 & R2) that overlap
- map reads to reference genome and annotation
- quantification of RNA types
- miRNA and isomiR quantification (including the variant type)
- preparation of results for integrative visualization

The tool uses the miRTop database and its notation to quantify and report the miRNAs present in each sample. With the resulting matrix for each sample the analysis can be performed at the miRNA, isomiR, or variant type level.

The XICRA documentation includes:

- A *User's Guide* to get started.
- An example *Tutorial*.
- A *Quick Start Guide*.
- A list of *Frequently Asked Questions (FAQs)*
- Some *developer Guidelines* to contribute to the project.
- Additional *information* of the XICRA project.
- A list of *Glossary* terms.
- A list of *Bibliography*

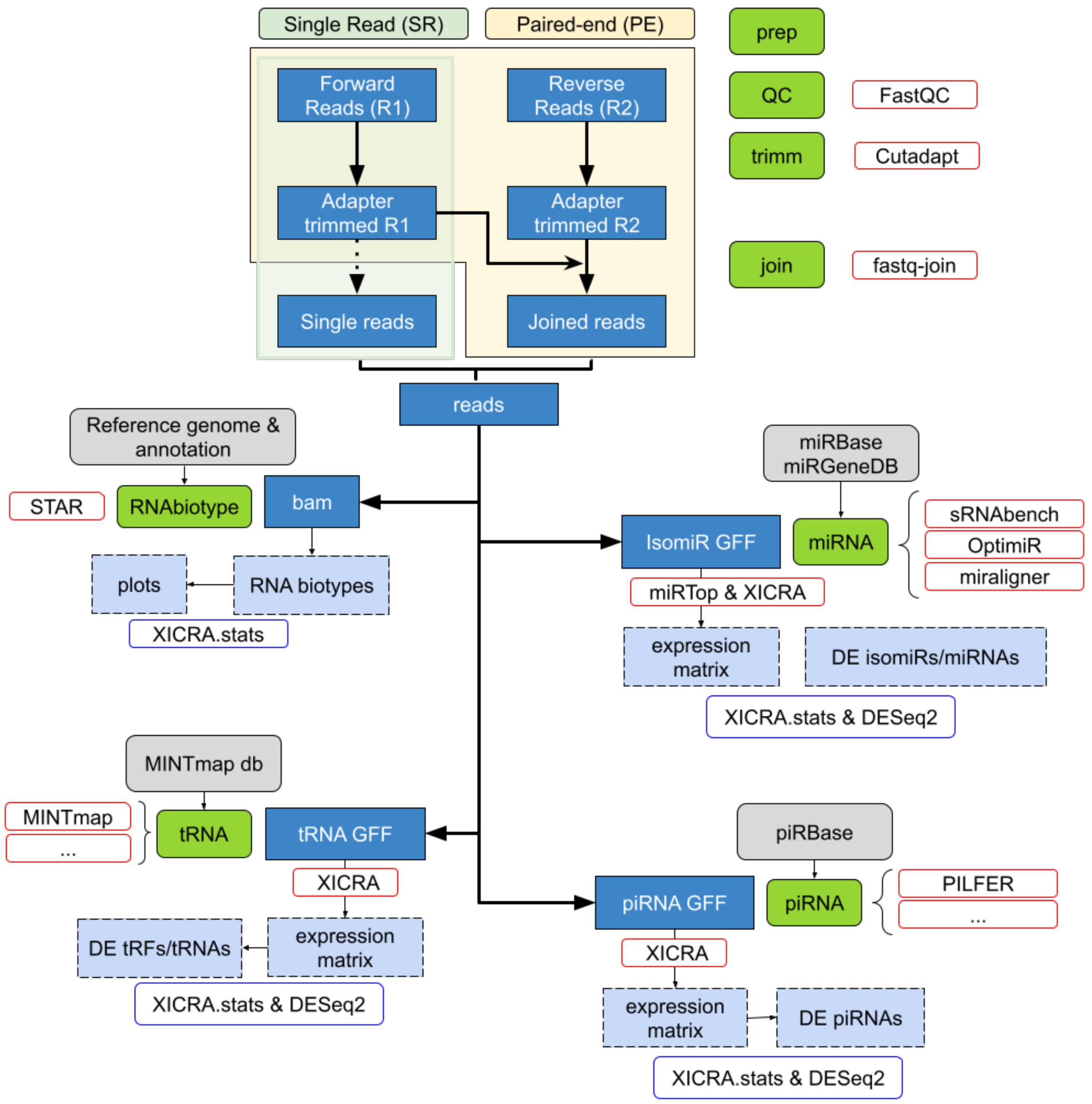
CHAPTER TWO

PIPELINE SCHEME

Here we show the scheme of the `XICRA` bioinformatic tool. It is divided in six main actions:

1. **Preparation of the input data**: preparation of the fastq files from a sequencing run.
2. **Quality analysis**: with quality check programs attending the input provided.
3. **Adapters trimming**: for each read the adapter sequences are filtered out.
4. **Read joining**: joins sequencing reads (paired-end).
5. **Mapping reads and feature counts**: generates a RNA biotype analysis, quantifying each RNA type present in the samples.
6. **miRNA analysis**: generates a miRNA analysis, with isomiR quantification and variant type information.

This information can be easily accessed using the accompanied R package XICRA.stats. Although the pipeline is designed to take paired-end reads, it also accepts single-end reads.


Single Read (SR)
Paired-end (PE)
prep
Forward Reads (R1)
Reverse Reads (R2)
QC
FastQC
trimm
Cutadapt
Adapter trimmed R1
Adapter trimmed R2
join
fastq-join
Single reads
Joined reads
reads
Reference genome & annotation
STAR
RNAbiotype
bam
plots
RNA biotypes
XICRA.stats
miRBase miRGeneDB
IsomiR GFF
miRNA
sRNAbench
OptimiR
miraligner
miRTop & XICRA
expression matrix
DE isomiRs/miRNAs
XICRA.stats & DESeq2
MINTmap db
MINTmap
...
tRNA
tRNA GFF
XICRA
DE tRFs/tRNAs
expression matrix
XICRA.stats & DESeq2
piRBase
piRNA GFF
piRNA
PILFER
...
XICRA
expression matrix
DE piRNAs
XICRA.stats & DESeq2

CHAPTER THREE

TABLE OF CONTENTS

3.1 Contents

3.1.1 Documentation

3.1.1.1 User's Guide

This User Guide provides information on XICRA usage and documentation. It is divided in several main chapters:

Installation

Python modules

There are several extra python module requirements that are needed and are summarized in the following table:

Module	Version
biopython	1.78
future	0.18.2
HCGB	0.3.2
mirtop	0.4.23
multiqc	1.10
networkx	2.5
pybedtools	0.7.10
termcolor	1.1.0
numpy	1.18.4
pandas	0.24.2

These modules might have extra dependencies. Details of the list of all modules required are listed in `XICRA/config/python/python_requirements.txt`. And accessible here:

Python requirements details

Here is the content of file `XICRA/config/python/python_requirements.txt` containing all python modules and versions required by `XICRA`.

```
alabaster==0.7.12
appdirs==1.4.4
Babel==2.9.1
bottle==0.12.19
certifi==2021.5.30
charset-normalizer==2.0.5
commonmark==0.9.1
docutils==0.17.1
idna==3.2
imagesize==1.2.0
Jinja2==3.0.1
latexcodec==2.0.1
MarkupSafe==2.0.1
packaging==21.0
Pillow==8.3.2
pybtex==0.24.0
pybtex-docutils==1.0.1
Pygments==2.10.0
pyparsing==2.4.7
pytz==2021.1
PyYAML==5.4.1
recommonmark==0.7.1
requests==2.26.0
rinoh-typeface-dejavuserif==0.1.3
rinoh-typeface-texgyrecursor==0.1.1
rinoh-typeface-texgyreheros==0.1.1
rinoh-typeface-texgyrepagella==0.1.1
rinohtype==0.5.3
six==1.16.0
snowballstemmer==2.1.0
Sphinx==4.2.0
sphinxcontrib-applehelp==1.0.2
sphinxcontrib-bibtex==2.4.1
sphinxcontrib-devhelp==1.0.2
sphinxcontrib-htmlhelp==2.0.0
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==1.0.3
sphinxcontrib-serializinghtml==1.1.5
sphinxmark==0.2.1
urllib3==1.26.6
```

Although these dependencies will be fulfilled during the `XICRA` installation with `pip`, you might be interested in installing them yourself.

Using `pip` we can install them all at a glance.

```
pip install -r ./XICRA/config/python/python_requirements.txt
```

But again, following installation recommendations, we encourage you to create and install them within a virtual environment (See section: Python environment section for details).

You can test the presence of these `python` modules using the `XICRA config` module. Once you identified the missing dependencies and minimum versions required you can either install them and set them available within your `PYTHONPATH` or environment or you can execute the `XICRA config` with `install` option.

Software dependencies

Also, several software packages are also required. They are listed in `XICRA/config/software/dependencies.csv`, which is shown below:

software	version	export name	website
fastqc	0.11.4	fastqc	https://www.bioinformatics.babraham.ac.uk/projects/download.html#fastqc
sRNAbench	1.6	sRNAbench.jar	https://bioinfo2.ugr.es/srnatoolbox/standalone/
optimir		optimir	https://github.com/FlorianThibord/OptimiR
miraligner		miraligner.jar	https://github.com/lpantano/seqbuster
STAR	2.6.1	STAR	https://github.com/alexdobin/STAR
featureCounts	1.5.1	featureCounts	http://subread.sourceforge.net/
fastqjoin		fastq-join	https://expressionanalysis.github.io/ea-utils/
cutadapt	2.10	cutadapt	https://cutadapt.readthedocs.io/en/stable/
multiqc	1.8	multiqc	https://multiqc.info/
miRTop	0.4.23	mirtop	https://github.com/miRTop/mirtop

Most of the software are common software that any person doing bioinformatics should have, so you might have already available within your system. However, installing `XICRA` using `conda` all the dependecies will be correctky installed.

You can test for any missing software dependencies using the `XICRA config` module. Once you identified the missing dependencies and minimum versions required you can either install them and set them available within your `$PATH` or you can execute the `X config` with `install` option.

Contents

This is an installation guide for the `XICRA` pipeline.

XICRA is a pipeline composed of multiple lines of code that calls and automatizes the analysis from multiple bioinformatic tools. Several dependencies (python, perl, additional software and their dependencies and third-party software) are required for the XICRA analysis.

You first need to get XICRA (from different sources available): you can either install XICRA latest version using Conda and Python pip or if you want to contribute or see additional details of the project, it's recommended you *install the latest development version*.

Once you get the code, before running XICRA you must make sure you have all the dependencies fulfilled from section *Requirements and dependencies* using the `XICRA config` module.

Requirements and dependencies

System requirements

Take into account that you may need some system requirements to install XICRA already available in your system such as `python3`, `python3-dev` and `python3-venv` or `build-essential` libraries among others.

Check and install them by typing:

```
sudo apt install python3 python3-dev python-dev python3-venv python-pip
sudo apt install build-essential libssl-dev libffi-dev libxml2-dev libxslt1-dev zlib1g-
↪dev
```

XICRA will require `python` v3.6 and `java` (we tested in `openjdk` 14 2020-03-17).

XICRA requirements

We include details on the different modules required by XICRA to successfully run here:

Installing XICRA

If you want to run a stable version of XICRA, the installation can be done with `conda` or `pip` packages following instructions in *Installing from conda* and *Installing from pip*, correspondingly. We encourage you to install XICRA and all dependencies using the `conda` environment.

On the other hand if you are interested in contributing to XICRA development, running the latest source code, or just like to build everything yourself, you should follow the *Installing from source* instructions.

Additionally, there are a number of dependencies that might be necessary to install or check within your system in both cases. Choose the appopiate choice according to your intalling XICRA option selected.

Installing from conda

Unfortunately, a couple of executables are not available neither as a `conda` or `pip` packages. These packages are `miraligner` and `sRNAbench`. We have generated a shell script to retrieve `miraligner` and include it within your `conda` environment. However, `sRNAbench` is no longer available to be downloaded, thus, only the users with the software already installed will be able to run the XICRA analysis with `sRNAbench`. To create a new `conda` environment, install third party software, install XICRA and missing dependencies, do as follows:

```
## clone repo
git clone https://github.com/HCGB-IGTP/XICRA.git

## move to folder
cd XICRA

## create conda environemt
conda env create -n XICRA -f XICRA_pip/devel/conda/requirements.txt

## activate
conda activate XICRA

## install latest python code
pip install XICRA

## install missing software
sh XICRA_pip/XICRA/config/software/installer.sh
```

To check everything is fine, try executing the `config` module:

```
XICRA config
```

We additionally provide a supplementary R package for parsing and plotting some `XICRA` results, called XICRA.stats. See additional details here.

Install it in ``R` using:

```
## Install XICRA.stats version from GitHub:
## install.packages("devtools")
devtools::install_github("HCGB-IGTP/XICRA.stats")
```

Installing from pip

If you are not using a `conda` environment as you might have previously installed all dependencies, we encourage you to create a `python` environment containing all `python` modules required for `XICRA`.

```
## create enviroment
python3 -m venv XICRA_env

## activate it
source XICRA_env/bin/activate

## install XICRA and dependencies
pip install XICRA

## execute XICRA
XICRA -h
```

Follow additional pip installing instructions to learn about installing packages.

Also, as `XICRA` relies in multiple dependencies, external packages and third-party software, we encourage you to once you install `XICRA` using `pip`. Check for dependencies using the `XICRA config` module. See details in `XICRA config` module *section*.

Installing from source

Under some circumstancies (develop, bugs fixed, etc) you might be interested in obtaining the latest code version. Take into account, that you will need to install dependencies and fulfill requirements to have a working distribution.

Get source code, for developers

We have included files in the folder `devel/conda` for the build and configuration of the `conda` package. The `XICRA` project uses git as a version control system. To get the code, you can grab the latest version from the XICRA github website, and follow the *Install XICRA from source* instructions.

Using the command-line, check you have a working distribution of `git` by typing `git --help` or install it by typing:

```
sudo apt update
sudo apt upgrade
sudo apt install git
```

Once you have `git` installed, to create a new conda environment, install third party software, install `XICRA` and missing dependencies for `XICRA` development, do as follows:

```
## clone repo
git clone https://github.com/HCGB-IGTP/XICRA.git

## move to folder XICRA_pip
cd XICRA/XICRA_pip

## create conda environemt
conda env create -n XICRA -f ./devel/conda/requirements.txt

## activate
conda activate XICRA

## install latest python code
pip install -r ./devel/pypi/requirements.txt
pip install -e .

## install missing software
sh ./XICRA/config/software/installer.sh
```

Install XICRA from source

Once you have `XICRA` source code available you only need to include the `XICRA` folder and main script within your path.

```
export PYTHONPATH=$PYTHONPATH":"$PWD"/XICRA"

export PATH=$PATH":"$PWD"/XICRA.py"
```

Take into account that before running `XICRA` you have to make sure you have all the dependencies fulfilled from section *Requirements and dependencies*. You can either install them yourself, use appropiate scripts for this purpose or use the `XICRA config` module to check, update and install all dependencies required.

Quick start

This is quick start guide for impatient people.

Installation

Build and activate a `conda` environment and get `XICRA` repository:

```
## clone repo
git clone https://github.com/HCGB-IGTP/XICRA.git

## create conda environemt
conda env create -n XICRA -f XICRA_pip/devel/conda/environment.yml

## activate the environment
conda activate XICRA

## install latest python code
pip install XICRA

## install missing software
sh XICRA_pip/XICRA/config/software/installer.sh
```

Sometimes, the activation of the environment is done with `source activate XICRA` instead of `conda activate XICRA`. Check everything is fine by executing the `config` module:

```
XICRA config
```

Execute XICRA

The functionallity of `XICRA` is divided in modules. Each of them in charge of different parts of the analysis. Here we show how to run each of the modules:

1. **Prepation of the data**: `prep`, `QC`, `trimm`, `join`.
   - `XICRA prep --input input_folder --output output_folder`: preparation of the data.
   - `XICRA QC --input input_folder`: quality analysis of the samples.
   - **`XICRA trimm --input input_folder --adapters_a XXXXX --adapters_A YYYYYYYY`: trimming the adapters.**
     - `--adapters_a XXXXX`: sequence of the 3' adapter of R1
     - `--adapters_A YYYYYYYY`: sequence of the 3' adapters of R2.
   - `XICRA join --input input_folder`: join paired end reads.
2. **Quantification of RNA types**: `biotype`
   - **`XICRA biotype --input input_folder --threads X --fasta file.fa --annotation file.gtf --limitRA`**
     - `--threads X`: number of cores enabled to run the analysis.
     - `--fasta file.fa`: fasta file with the reference genome to map reads.
     - `--annotation file.gtf`: reference genome annotation in GTF format.

  - `--limitRAM YYYY`: given RAM in bytes to run the analysis. **Note that:** this process consumes high RAM values.

3. **miRNA analysis**: `miRNA`

   - **`XICRA miRNA --input input_folder --threads X --software analyisis_tool`**
     - `--threads X`: number of cores enabled to run the analysis.
     - `--software analyisis_tool`: three options available, `miraligner`, `optimir` and `sRNAbench`.

Take into account that these are basic examples, each of the modules can be run with other different parameters. To see all the possible parameters of each module run `-h`. For example, for the `trimm` module:

```
XICRA trimm -h
```

Test example

Here we include a brief example on how to use `XICRA`.

First, we create a `conda` environment and install `XICRA` and its dependencies. See example details shown before. Then, we can test `XICRA` by using an example of 100 miRNA simulated and provided within the repository as an example of simulation.

```
## run XICRA example
ln -s ~/BMC_bioinformatics_paper/simulation/example/reads/

## prepare reads
XICRA prep --input reads/ --output_folder test_XICRA

## join reads
XICRA join --input test_XICRA --noTrim

## create miRNA analysis
XICRA miRNA --input test_XICRA --software miraligner optimir

## explore results
ls test_XICRA/report/
```

As a result, we will end up with a folder for each of the run analysis for every sample. Thus, in the `miRNA` folder, we will obtain the miRNA results for our samples, performed with two different softwares `miraligner` and `optimir`.

User data

In the presented example, nor `QC`, neither `trimm` steps were necessary (that is why `--noTrim` was added in the `join` command). However, with real data, running `QC` is highly recommended to check the quality of the samples (and filter outliers if necessary). Running the `trimm` command to eliminate the adapters will be necesary for NGS data.

On the other hand, the `biotype` was skipped as well. This step is only informative: it maps and annotates the reads and quantifies the RNA types present in each of the samples. **Note that:** this step is very time and RAM consuming.

The `miRNA` analysis can be performed whith any of the three available softwares `miraligner`, `optimir` and `sRNAbench` (they can be combined as seen in the example). Unfortunately, the donwloading of `sRNAbench` is no longer available, thus, only users with the software already installed will be able to run the miRNA analysis with it.

Finally, XICRA is also able to analyse single end reads, in this case --single_end should be added in the commands (and no join step would be necessary).

Deactivate environment

After finished the execution of any XICRA module or script, it is convenient to deactivate the environment. You can just close the terminal but it would be more appopiate to conveniently deactivate the environment first.

To do so, execute one of the following commands:

```
conda deactivate XICRA
```

```
source deactivate XICRA
```

Main modules

There are multiple modules available that perform several *steps* and generate multiple results.

XICRA modules require several command-line arguments and options to run. There are a number of shared *arguments* among all modules and some others specific of each module and specified within each one.

config

The XICRA pipeline provides this module config to help the user to check if the multiple dependencies and requirements are fulfilled.

We encourage XICRA users to run this module after the installation to check whether the multiple requirements and dependencies are correctly installed.

See also:

Additional information on XICRA configuration and requirements

- *Installation*
- *Requirements*
- config module (API)

How to run the config module

Once you have installed XICRA you should be able to run in the command line the pipeline.

If you type XICRA you should see a prompt with the different modules available. Following the pipeline name type the module of interest, in this case, config. As an example:

```
XICRA config -h
```

The different options and parameters for this module should appear in the command line prompt:

Module XICRA config

Parameters

- **--help** (*-h*,) – Show this help message and exit.

- **--debug** – Show additional message for debugging purposes.

Basically, this `XICRA config` module allows the user to check if the requirements are fulfilled.

```
XICRA config
```

prep

This module prepares fastq files from a sequencing run for later usage.

It initially checks the length of the name and advises the user to rename samples if exceeded a length (10 characters). Along `XICRA` there are a few string length limitations by different software that need to be sort out from the beginning of the process.

See additional details of this name format limitations under user-guide section: *format fastq files*

Once sample names are checked this module allows the user to copy files into the project folder initiate or only link them using a symbolic link to avoid duplicated raw data.

In addition, when multiples files have been generated for the same sample e.g different lanes this module can merge them. It concatenates these files according the common identifier and generates a unique file, one per paired-read if necessary, check *format fastq files*.

If using the the `--project` option, this module would create a single folder for each sample and a folder named as `raw` including the linked or copied fastq files. See additional details of project folder organization *here*.

How to run the prep module

Executing the following:

```
XICRA prep -h
```

The different options and parameters for this module should appear in the command line prompt:

Module XICRA prep help

Parameters **--help** (*-h*) – Show this help message and exit.

Module XICRA prep Input/Output

Parameters

- **--input** – Folder containing the files with reads. Files could be .fastq/.fq/ or fastq.gz/.fq.gz. See --help_format for additional details. REQUIRED.
- **--output_folder** – Output folder. Name for the project folder.
- **--single_end** – Single end files. Default mode is paired-end. Default OFF.
- **--batch** – Provide this option if input is a file containing multiple paths instead a path.
- **--in_sample** – File containing a list of samples to include (one per line) from input folder(s). Default OFF.
- **--ex_sample** – File containing a list of samples to exclude (one per line) from input folder(s). Default OFF.
- **--detached** – Isolated mode. No project folder initiated for further steps. Default OFF.
- **--include_lane** – Include the lane tag (*L00X*) in the sample identification. See --help_format for additional details. Default OFF.

- **`--include_all`** – Include all file name characters in the sample identification. See --help_format for additional details. Default OFF.

`Module XICRA prep options`

Parameters

- **`--threads`** – Number of CPUs to use. Default: 2.
- **`--copy_reads`** – Instead of generating symbolic links, copy files into output folder. Default OFF.
- **`--merge_Reads`** – Merge files corresponding to the same sample. Used in combination with --include_lane and --include_all will produce different results. Please check, --help_format or https://xicra.readthedocs.io/en/latest/user_guide/info/info_index.html
- **`--rename`** – File containing original name and final name for each sample separated by comma. No need to provide a name for each pair if paired-end files. If provided with option '--merge_Reads', the merged files would be renamed accordingly.

`Module XICRA prep additional information`

Parameters

- **`--help_format`** – Show additional help on name format for files.
- **`--help_project`** – Show additional help on the project scheme.
- **`--debug`** – Show additional message for debugging purposes.

- For further information of the module functionallity, check this *page*.

Output of prep for each sample

After the `prep` module excution, a data folder will be generated. Inside it, for each sample, a new folder will be created (called as the sample). These sample folders will contain links to all the raw files that correspond to each sample (if --copy_reads has been selected, instead of links the copied files).

QC

This module calls fastqc, a quality check program, to analyze the quality of each sample.

By default, creates a final MultiQC report with all the samples. It is useful to check if there are outliers among the input samples. It can be disabled using the option --skip_report.

How to run the QC module

Executing the following:

```
XICRA QC -h
```

The different options and parameters for this module should appear in the command line prompt:

`Module XICRA QC help`

Parameters **`--help`** (*-h*) – Show this help message and exit.

`Module XICRA QC Input/Output`

Parameters

- `--input` – Folder containing the files with reads. Files could be .fastq/.fq/ or fastq.gz/.fq.gz. See --help_format for additional details. REQUIRED.
- `--output_folder` – Output folder. Name for the project folder.
- `--batch` – Provide this option if input is a file containing multiple paths instead a path.
- `--in_sample` – File containing a list of samples to include (one per line) from input folder(s). Default OFF.
- `--ex_sample` – File containing a list of samples to exclude (one per line) from input folder(s). Default OFF.
- `--detached` – Isolated mode. No project folder initiated for further steps. Default OFF.
- `--include_lane` – Include the lane tag (*L00X*) in the sample identification. See --help_format for additional details. Default OFF.
- `--include_all` – Include all file name characters in the sample identification. See --help_format for additional details. Default OFF.

`Module XICRA QC options`

Parameters

- `--skip_report` – Do not report statistics using MultiQC report module. Default OFF.
- `--threads` – Number of CPUs to use. Default: 2.

`Module XICRA QC additional information`

Parameters

- `--help_format` – Show additional help on name format for files.
- `--help_project` – Show additional help on the project scheme.
- `--help_multiqc` – Show additional help on the multiQC module.
- `--debug` – Show additional message for debugging purposes.

- For further information of the module functionallity, check this *page*.

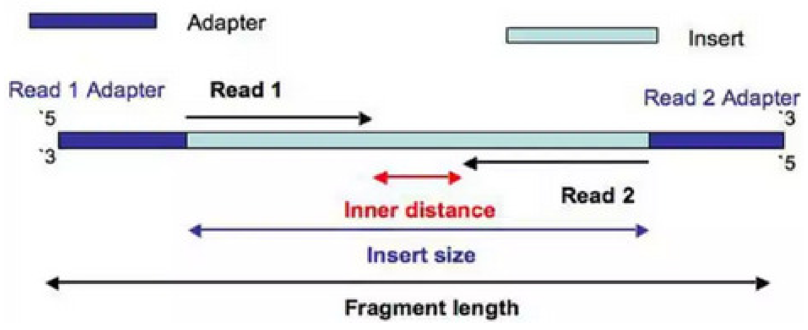
Output of QC

After the `QC` module excution,in the data folder for each sample, a new directory will be created. It will be called ‘fastq’ and will contain the the quality analysis.

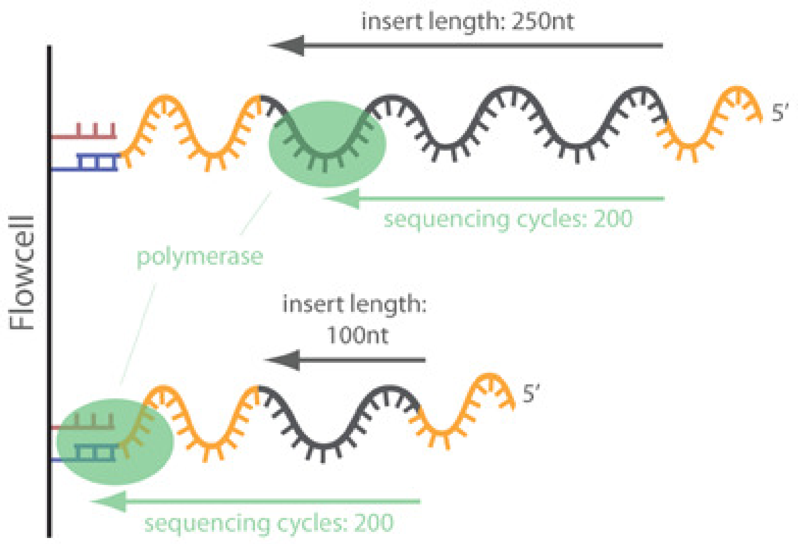
In addition, if --skip_report was not selected, a folder called ‘report’ will also be generated. In this folder, the different reports that may be created after the execution of other modules (as `trimm`) will be stored. These reports will always show information of all the samples. In this case, a subfolder called ‘fastqc’ will be built with the MultiQC report in order to visualize the quality analysis of all samples.

trimm

This module trimms sequencing adapters that could be present in next generation sequencing files. Adapters have to be ligated to every single DNA molecule during library preparation. For Illumina short read sequencing, the corresponding protocols involve (in most cases) a fragmentation step, followed by the ligation of certain oligonucleotides to the 5' and 3' ends. These 5' and 3' adapter sequences have important functions in Illumina sequencing, since they hold barcoding sequences, forward/reverse primers (for paired-end sequencing) and the important binding sequences for immobilizing the fragments to the flowcell and allowing bridge-amplification.



In Illumina sequencing, adapter sequences will only occur at the 3' end of the read and only if the read length is longer than the insert size:



However, in the case of miRNAs, there will be adapter contamination in the 3' reads but also in the 5', due to their short nature. As the adapters are synthetic they need to be removed. Note that, the 3' adapter of the R2 will be the same as the 5' of R1, and the 5' of R2 will be the same as the 3' of R1.

The adapter sequences must be introduced by the user to be trimmed, using the parameters --adapters_a and --adapters_A, for the adapters attached to the 3' of read 1 and read 2 correspondingly, see this *section*. The process is done by the software Cutadapt.

How to run the trimm module

Executing the following:

```
XICRA trimm -h
```

The different options and parameters for this module should appear in the command line prompt:

`Module XICRA trimm help`

Parameters **`--help`** (*-h*) – Show this help message and exit.

`Module XICRA trimm Input/Output`

Parameters

- **`--input`** – Folder containing a project or reads, according to the mode selected. Files could be .fastq/.fq/ or fastq.gz/.fq.gz. See –help_format for additional details. REQUIRED.
- **`--output_folder`** – Output folder.
- **`--single_end`** – Single end files. Default mode is paired-end. Default OFF.
- **`--batch`** – Provide this option if input is a file containing multiple paths instead a path.
- **`--in_sample`** – File containing a list of samples to include (one per line) from input folder(s). Default OFF.
- **`--ex_sample`** – File containing a list of samples to exclude (one per line) from input folder(s). Default OFF.
- **`--detached`** – Isolated mode. –input is a folder containing fastq reads. Provide a unique path o several using –batch option.
- **`--include_lane`** – Include the lane tag (*L00X*) in the sample identification. See --help_format for additional details. Default OFF.
- **`--include_all`** – IInclude all file name characters in the sample identification. See --help_format for additional details. Default OFF.

`Module XICRA trimm options`

Parameters

- **`--adapters_a`** – Sequence of an adapter ligated to the 3' end (of read 1). See –help_trimm_adapters for further information.
- **`--adapters_A`** – Sequence of an adapter ligated to the 3' read in pair (of read 2). See --help_trimm_adapters for further information.
- **`--extra`** – provide extra options for cutadapt trimming process. See --help_trimm_adapters for further information.
- **`--skip_report`** – Do not report statistics using MultiQC report module. [Default OFF]. See details in –help_multiqc
- **`--threads`** – Number of CPUs to use. Default: 2.

`Module XICRA trimm additional information`

Parameters

- **`--help_format`** – Show additional help on name format for files.
- **`--help_project`** – Show additional help on the project scheme.

    - **`--help_trimm_adapters`** – Show additional help of the trimm module.
    - **`--help_multiqc`** – Show additional help on the multiQC module.
    - **`--debug`** – Show additional message for debugging purposes.

- For further information of the module functionallity, check this *page*.

Output of trimm for each sample

After the `trimm` module excution, for each sample, a new fastq file is generated in the in the "trimm" folder of the sample with the same name as the raw one + "_trimm".

It generates by default a MultiQC report in the folder "report/trimm", this report may be useful to detect outliers or if there are similar amounts of trimmed sequences for all samples.

biotype

This module generates a RNA biotype analysis. The aim of this computation is to check if there are samples with a very different configuration, outliers that show different proportions of uniquely mapped, multimapped or no mapped reads or with different quantities of miRNA, misc_RNA,… than the rest. If this happens, it could be due to possible differences in sample manipulation, extraction, library preparation, sequencing, etc. Those samples should be excluded.

The module is divided in two steps:

1. Mapping the reads: performed with STAR software.
2. Feature counts: perforfed with featureCounts.

The output of this module is 'descriptive', thus, we won't need the output to continue with the analysis, it is an informative step to know the RNA types that are present in each sample.

Note that: the mapping process performed with STAR software requires high RAM values. We are working on the implementation of alternatives to be able to execute the `biotype` module in computers with less RAM capacity.

How to run the biotype module

Executing the following:

```
XICRA biotype -h
```

The different options and parameters for this module should appear in the command line prompt:

`Module XICRA biotype help`

Parameters **`--help`** (*-h*) – Show this help message and exit.

`Module XICRA QC Input/Output`

Parameters

- **`--input`** – Folder containing the files with reads. Files could be .fastq/.fq/ or fastq.gz/.fq.gz. See --help_format for additional details. REQUIRED.
- **`--output_folder`** – Output folder. Name for the project folder.
- **`--single_end`** – Single end files. Default OFF. Default mode is paired-end.
- **`--batch`** – Provide this option if input is a file containing multiple paths instead a path.

- `--in_sample` – File containing a list of samples to include (one per line) from input folder(s). Default OFF.
- `--ex_sample` – File containing a list of samples to exclude (one per line) from input folder(s). Default OFF.
- `--detached` – Isolated mode. No project folder initiated for further steps. Default OFF.
- `--include_lane` – Include the lane tag (*L00X*) in the sample identification. See --help_format for additional details. Default OFF.
- `--include_all` – Include all file name characters in the sample identification. See --help_format for additional details. Default OFF.

`Module XICRA biotype options`

Parameters

- `--threads` – Number of CPUs to use. Default: 2.
- `--annotation` – Reference genome annotation in GTF format.
- `--limitRAM` – limitRAM parameter for STAR mapping. Default 20 Gbytes.
- `--noTrim` – Use non-trimmed reads [or not containing '_trim' in the name].
- `--skip_report` – Do not report statistics using MultiQC report module. Default OFF. See details in --help_multiqc

`Module XICRA biotype parameters`

Parameters

- `--no_multiMapping` – Set NO to counting multimapping in the feature count.By default, multimapping reads are allowed. Default: False
- `STRANDED` (*--stranded*) – Select if reads are stranded [1], reverse stranded [2] or non-stranded [0], Default: 0.

`Module XICRA biotype reference genome`

Parameters

- `--fasta` – Reference genome to map reads.
- `--genomeDir` – STAR genomeDir for reference genome.

`Module XICRA biotype additional information`

Parameters

- `--help_format` – Show additional help on name format for files.
- `--help_project` – Show additional help on the project scheme.
- `--help_RNAbiotype` – Show additional help on the RNAbiotype paired-end reads process.
- `--debug` – Show additional message for debugging purposes.

- For further information of the module functionallity, check this *page*.

Output of biotype

Inside the data folder of each sample, a 'map' directory will be generated containing a report from the mapping of MultiQC. After that, a final report will also be created in the 'report' folder with the featureCounts information of all samples.

miRNA

The `XICRA` pipeline provides this module, `miRNA`, to generate the miRNA analysis. MicroRNAs (miRNAs), a class of small non-coding RNAs, have an average length of 21–23 nucleotides (nt) and modulate gene expression post-transcriptionally. Most miRNA expression studies based on next generation sequencing (NGS) data, summarize all the reads mapping to a specific miRNA locus or miRNA sequence with or without mismatches and assign it to a single miRNA entity, this is, a unique miRBase reference database entry (miRBase database is a searchable database of published miRNA sequences and annotation).

However, this type of analysis neglects the fact that **not all reads are identical to the reference sequence in miRBase, which is called the canonical sequence**. Small RNA sequencing NGS methodology has revealed that miRNAs can frequently appear in the form of multiple sequence variants or isoforms (termed isomiRs). Each isomiR can modulate gene expression post-transcriptionally.

With the `miRNA` module we capture all the available information at different levels, the analysis can be done at miRNA or isomiR level.

Its functionalyty is divided in three steps:

1. miRNA analysis
2. Standarization of the results
3. Expression count matrix generation with miRTop (Command line tool to annotate with a standard naming miRNAs e isomiRs).

The analysis can be performed with three different softwares:

- Miraligner: maps small RNA data to miRBase repository.
- Optimir: algorithm for integrating available genome-wide genotype data into miRNA sequence alignment analysis.
- sRNAbench: application for processing small-RNA data obtained from NGS platforms. Unfortunately, the downloading of this tool is no longer available, thus, only users with `sRNAbench` already installed will be able to run the `XICRA` analysis with it.

According to our tests, published in this article, `miraligner` is the software with the best performance for the miRNA analysis.

How to run the miRNA module

Executing the following:

```
XICRA miRNA -h
```

The different options and parameters for this module should appear in the command line prompt:

`Module XICRA miRNA help`

Parameters **`--help`** (*-h*) – Show this help message and exit.

`Module XICRA miRNA Input/Output`

Parameters

- `--input` – Folder containing a project or reads, according to the mode selected. Files could be .fastq/.fq/ or fastq.gz/.fq.gz. See –help_format for additional details. REQUIRED.
- `--output_folder` – Output folder.
- `--single_end` – Single end files. Default mode is paired-end. Default OFF.
- `--batch` – Provide this option if input is a file containing multiple paths instead a path.
- `--in_sample` – File containing a list of samples to include (one per line) from input folder(s). Default OFF.
- `--ex_sample` – File containing a list of samples to exclude (one per line) from input folder(s). Default OFF.
- `--detached` – Isolated mode. –input is a folder containing fastq reads. Provide a unique path o several using –batch option.
- `--include_lane` – Include the lane tag (*L00X*) in the sample name. See –help_format for additional details. Default OFF.
- `--include_all` – Include all characters as tag name before read pair, if any. See –help_format for additional details. Default OFF.
- `--noTrimm` – Use non-trimmed reads (or not containing '_trim' in the name).

`Module XICRA miRNA options`

Parameters

- `--threads` – Number of CPUs to use. Default: 2.
- `--species` – Species tag ID. Default: hsa (Homo sapiens).
- `--database` – Path to store miRNA annotation files downloaded: miRBase, miRCarta, etc.
- `--miRNA_gff` – miRBase hsa GFF file containing miRNA information.
- `--hairpinFasta` – miRNA hairpin fasta file.
- `--matureFasta` – miRNA mature fasta file.
- `--miRBase_str` – miRBase str information.

`Module XICRA miRNA software`

Parameters `--software` – Software to analyze miRNAs, sRNAbench, optimir, miraligner. Provide several input if desired separated by a space. REQUIRED.

`Module XICRA miRNA additional information`

Parameters

- `--help_format` – Show additional help on name format for files.
- `--help_project` – Show additional help on the project scheme.
- `--help_miRNA` – Show additional help on the miRNA paired-end reads process.
- `--debug` – Show additional message for debugging purposes.

- For further information of the module functionallity, check this *page*.

Output of miRNA for each sample

As the rest of the modules, the `miRNA` module will generate a folder in each of the sample directories called "miRNA". Inside this folder another two will be created for each of the softwares selected. For example, if we have executed `--software optimir miraligner` we will obtain four output folders:

- data/sampleName/miRNA/optimiR
- data/sampleName/miRNA/miraligner
- data/sampleName/miRNA/optimiR_miRTop
- data/sampleName/miRNA/miraligner_miRTop

The first two folders will store the outputs of the corresponding softwares in their particular format.

The folders ended in "_miRTop" will contain the results in the miRTop standarized format.

Finally, the expression count matrix will be stored in .tsv format. Following the previous example, these files would be located in:

- data/sampleName/miRNA/optimiR_miRTop/counts/mirtop.tsv
- data/sampleName/miRNA/miraligner_miRTop/counts/mirtop.tsv

Expression count matrix for each sample

As a result for each sample (and software used) we will end up with a table like this, mirtop.tsv, called the expression count matrix. Here we can see an example of this matrix:

UID	Read	miRNA	Variant	iso_5p	iso_3p	iso_add3p	iso_snp	sampleName
iso-22-B175JXN0Q	AAACCGTTACCATTACTGAGTT	hsa-miR-451a	NA	0	0	0	0	69047
iso-23-B175JXN00O	AAACCGTTACCATTACTGAGTTA	hsa-miR-451a	iso_add3p:1	0	0	1	0	169
iso-24-B175JXN0KF	AAACCGTTACCATTACTGAGTTAA	hsa-miR-451a	iso_add3p:2	0	0	2	0	1
iso-23-B175JXN005	AAACCGTTACCATTACTGAGTTC	hsa-miR-451a	iso_add3p:1	0	0	1	0	3
iso-23-B175JXN00P	AAACCGTTACCATTACTGAGTTG	hsa-miR-451a	iso_add3p:1	0	0	1	0	108
iso-23-B175JXN00Q	AAACCGTTACCATTACTGAGTTT	hsa-miR-451a	iso_3p:+1	0	1	0	0	35289
iso-24-B175JXN0KO	AAACCGTTACCATTACTGAGTTTA	hsa-miR-451a	iso_3p:+2	0	2	0	0	675

- UID: unique identifier (UID) for each sequence defined by miRTop.

- Read: DNA sequence.
- miRNA: miRNA precursor, identifier defined by miRBase for each miRNA canonical sequence.
- Variant: variant type of each isomiR, 'NA' for the canonical sequence (checkout the miRTop variant nomenclature).
- The following four columns indicate the amount of base pairs added or substracted, compared to the canonical sequence.
- SampleName: raw read count expression for this sample.

Output of miRNA, comparing samples

On the other hand, as other modules, `miRNA` also builds an output to compare samples. In the folder report/miRNA, three different files will be created for each software executed. For example, if we have run `--software miraligner`, we will obtain the following files:

- **report/miRNA/miRNA_expression-miraligner_dup.csv**: Matrix with the number of reads of each UID of each sample that are duplicated. Normally, they occur when some bases are added at the beginning and the end, so it cannot be differentiated if they are 3p:+1;5p:+2 or 3p:+2;5p:+1. In those cases, they will both have the same UID. They are removed (they typically have very few counts).
- **report/miRNA/miRNA_expression-miraligner.csv**: Final matrix (without the duplicated UIDs). Number of counts of each UID of each sample, to be further analyzed with `R`.
- **report/miRNA/miRNA_expression-miraligner_seq.csv**: table with the DNA sequence corresponding to each UID.

The analysis of the matrix stored in miRNA_expression-miraligner.csv can be done at the isomiR level, differenciating by UID, variant type or miRNA (just considering the miRNA identifier). It can be done with the package XICRA.stats.

Command-line shared arguments

Here we include a brief description of the shared command-line arguments for some of `XICRA` modules.

Mode:

—project Project mode. Requires as `--input` a folder containing an initialized `BacterialTyper` project [Default].

--detached Isolated mode. `--input` is a folder containining fastq reads. Provide a unique path o several using `--batch` option

Input/Output:

—input string Folder containing a project or reads, according to the mode selected. Files could be `.fastq/.fq` or `fastq.gz/.fq.gz`. See `--help_format` for additional details.

--single_end Single end files [Default OFF]. Default mode is paired-end.

—batch Provide this option if input is a file containing multiple paths instead a path.

--in_sample string File containing a list of samples to include (one per line) from input folder(s) [Default OFF].

—ex_sample string File containing a list of samples to exclude (one per line) from input folder(s) [Default OFF].

Options:

—threads int Number of CPUs to use [Default: 2]

Additional information:

—debug Show additional message for debugging purposes.

Details of the XICRA project folder

TODO

Tutorial

Software Information

This is a general guide for the software that we employed along the `XICRA` pipeline.

Contents

FastQC

FastQC [1] is a quality control tool for high throughput sequence data. It 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 analysis 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

Read further information about FastQC: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Cutadapt

STAR

Feature counts

mirTop

Miraligner

MINTmap

sRNAbench

Optimir

Input File Format Information

Contents

Sample name details

`XICRA` accepts a wide range of file names. The format for fastq files can be:

- name.fastq.gz
- name_1.fastq.gz, with '1' or '2' to specify the read
- name_R2.fastq.gz, 'R1' or 'R2' to specify the read
- name_L001_R1.fastq.gz, with the lane information as 'L00X' or '00X' after the name
- name_L001_R1_001.fastq.gz, with '00X' at the end of the file name. This naming is used when the fastq of a sample had been cut in different files.
- name_L001_XYZ_R1_001.fastq.gz, there can be extra info for each file, XYZ.

The input file names should be structured considering the following aspects:

Length limitation

There is a limitation for the sample ID ('name') of 25 characters.

`XICRA` provides an option to rename samples if necessary in the module `prep`, option **--rename**.

Extensions

The suported extensions are:

- name_L00x_R2.fastq
- name_L00x_R2.fq
- name_L00x_R2.fastq.gz
- name_L00x_R2.fq.gz

Single-end files

It is possible to provide NGS single-end files although some steps of the process could not be accomplished using single-end files.

- name.fastq.gz
- name.fastq
- name.fq

Use option **--single-end in the different XICRA modules.**

Paired-end files

Paired-end files are full supported. The format for these files are:

- name_1.fastq.gz or name_R1.fastq.gz
- name_2.fastq.gz or name_R2.fastq.gz

No parameter is needed in to specify this kind of files.

Lane information

Files might contain lane information (*L00x* and/or *00x*). `XICRA` supports these names as long as follow these examples:

- name_L00x_R1.fastq.gz, name_L00x_R2.fastq.gz
- name_L00x_1.fastq.gz, name_L00x_2.fastq.gz

Name extensions

It can also be the case that the reads of a sample are divided in different files. In those cases, the files should contain a name final extension:

- name1_L001_R1_001.fastq.gz, name1_L001_R2_001.fastq.gz
- name1_L001_R1_002.fastq.gz, name1_L001_R2_002.fastq.gz
- name1_L002_R1_001.fastq.gz, name1_L002_R2_001.fastq.gz
- name1_L002_R1_002.fastq.gz, name1_L002_R2_002.fastq.gz

Extra information

In some cases, files might contain other extra information. In the following example, XYZ is the extra information:

- name1_L001_XYZ_R1_001.fastq.gz, name1_L001_XYZ_R2_001.fastq.gz
- name1_L001_XYZ_R1_002.fastq.gz, name1_L001_XYZ_R2_002.fastq.gz

Sample identification

`XICRA` will store the names of all the input files. After that, it will identify the samples. It can be the case that more than one file belong to the same sample. In order to pass this information to `XICRA`, a combination of the following parameters may be needed depending on the characteristics of the input file names:

Option: --include_lane

If you want to include lane tags (*L00X*, *00X*) into each each sample name (differentiate samples considering the lane): Use option **--include_lane within each module** and the lane tag will also be used to identify samples.

However, if you want to consider as a single sample the different lanes, you need to merge the corresponding fastq files: Use option **--merge_Reads** within module `prep`.

As an example, considering the input files:

- name1_L001_R1.fastq.gz, name1_L001_R2.fastq.gz
- name1_L002_R1.fastq.gz, name1_L002_R2.fastq.gz
- name1_L003_R1.fastq.gz, name1_L003_R2.fastq.gz
- name1_L004_R1.fastq.gz, name1_L004_R2.fastq.gz

  1. By adding the option `--include_lane` in **all modules**, `XICRA` will identify four samples:
     - Sample 1: name1_L001_R1, name1_L001_R2
     - Sample 2: name1_L002_R1, name1_L002_R2
     - Sample 3: name1_L003_R1, name1_L003_R2
     - Sample 4: name1_L004_R1, name1_L004_R2

     Remember to use option **--include_lane within each module**.
  2. By adding the options `--include_lane --merge_Reads` **within module prep**, `XICRA` will only identify one sample, merging all the corresponding files:
     - Sample 1: sample1_R1, sample1_R2

Option: --include_all

In some cases, files might contain other extra information and it is necessary to use all the information of the file name to identify samples:

If that is the case use **--include_all in al modules** .

If you want to merge fastq files that only differ in the final extension (_001, _002, ...):

Use options **--merge_Reads --include_all within module prep** and only **--include_all in the rest of the modules**.

As an example, considering the input files:

- name1_L001_XYZ_R1_001.fastq.gz, name1_L001_XYZ_R2_001.fastq.gz
- name1_L001_XYZ_R1_002.fastq.gz, name1_L001_XYZ_R2_002.fastq.gz
- name1_L002_XYZ_R1_001.fastq.gz, name1_L002_XYZ_R2_001.fastq.gz
- name1_L002_XYZ_R1_002.fastq.gz, name1_L002_XYZ_R2_002.fastq.gz

  1. By adding the option `--include_all` in **all modules**, `XICRA` will identify four samples:
     - Sample 1: name1_L001_XYZ_R1_001, name1_L001_XYZ_R2_001
     - Sample 2: name1_L001_XYZ_R1_002, name1_L001_XYZ_R2_002
     - Sample 3: name1_L002_XYZ_R1_001, name1_L002_XYZ_R2_001
     - Sample 4: name1_L002_XYZ_R1_002, name1_L002_XYZ_R2_002

     Remember to use option **--include_all within each module**.

2. By adding the options `--include_all --merge_Reads` **within module prep**, `XICRA` will identify two samples:

   - Sample 1: name1_L001_XYZ_R1, name1_L001_XYZ_R2
   - Sample 2: name1_L002_XYZ_R1, name1_L002_XYZ_R2

   Remember to use option **--include_all within each module**.

Frequently Asked Questions (FAQs)

Contents

This is a collection of FAQs for `XICRA` tutorial and interpretation of results.

See also:

Please read further information on each section following the links provided or the main *User Guide for XICRA*.

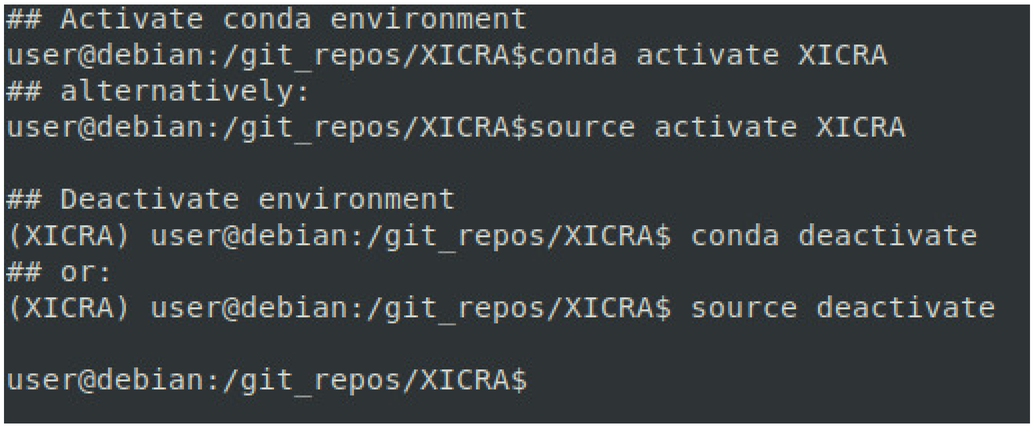
Installation

- What is required for the installation?

In order to correctly install `XICRA`, it is necessary to have python3 and python development and virtual environment libraries installed. See additional details in section *System requirements*.

- The common errors during the installation are:
- How do I know the conda environment is activated?

Once you execute the activation of the environment, either using the script `conda activate XICRA` or by executing `source activate XICRA`, you should see a tag in the command-line, `(XICRA)`, prompt as shown in the following image.



```
## Activate conda environment
user@debian:/git_repos/XICRA$conda activate XICRA
## alternatively:
user@debian:/git_repos/XICRA$source activate XICRA

## Deactivate environment
(XICRA) user@debian:/git_repos/XICRA$ conda deactivate
## or:
(XICRA) user@debian:/git_repos/XICRA$ source deactivate

user@debian:/git_repos/XICRA$
```

Once you deactivate the environment this tag should disappear.

Read additional details in Conda official documentation website.

3.1.1.2 Developer Guidelines

Guidelines for developing the `XICRA` project.

API Overview

Modules

Developer guidelines for `XICRA` modules.

prep

This module prepares fastq files from a sequencing run for later usage.

For each sample, a folder called as the sample will be generated conatining the links to the raw data (or copied raw data files) that belong to the sample.

Workflow

TODO: build image

Functions

QC

This module calls fastqc, a quality check program, to analyze the quality of each sample.

For each sample, a folder called "fastqc" is generated with the quality analysis, reported in an html file.

In addition, by default, a multiQC report is generated for all the samples.

Workflow

TODO: build image

Functions

Other modules

The `QC` module also uses the `multiQC_report` module:

trimm

This module cuts the introduced adapter sequences of the input reads.

For each sample, a folder called "trimm" is generated with the trimmed reads, called as the original ones +"_trimm".

In addition, a multiQC report is generated.

Workflow

TODO: build image

Functions

Other modules

The `trimm` module also uses the `multiQC_report` module:

biotype

This module works in two steps. First, it maps the unjoined reads to a given reference genome. Secondly, it generates a summary of the RNA types found in each sample.

Mapping

The mapping step is performed with STAR software and the code is in a separate script, `mapReads.py`. To introduce the reference genome the are two options:

1. Introduce the path to the fasta sequence of the human genome, option --fasta, and indicate the path to a reference genome annotation in GTF format, --annotation.
2. Introduce the STAR indexed genome file, option --genomeDir. This directory, called 'STAR_index' is created after the execution of STAR, thus, if it is already generated it can be reused, check the STAR documentation.

The --limitRAM parameter is also important. It indicates the maximum RAM (in bytes) that the computation will use to prevent the computer collapse. Note that, the STAR software requires high values of RAM in order to do the mapping. Thus, a 'regular' laptop may not be able to perform this computation. We are currently working in the implementation of alternatives to this software that can be used with any computer.

As a result, the mapping step generates a 'map' folder with a BAM file and a report from MultiQC for each sample.

Feature counts

The second step performs the RNA biotype analysis using the featureCounts software, code located in the script `RNAbiotype.py`. After the computation for each sample, a final MultiQC report is generated to compare the composition of each sample and, eventually, detect outliyers.

Workflow

TODO: build image

Functions

Other functions

This module calls other scripts:

miRNA

This module analyses the joined or single reads, that have been previously trimmed of different samples. To do so, it uses three possible softwares, which can be run at the same time if desired.

For each sample, an expression matrix using the miRTop nomenclature is generated, containing the information of the counts at miRNA or isomiR level; also describing the variant type of each isomiR.

In addition, an expression matrix comparing all the samples is created.

Workflow

TODO: build image

Functions

Other functions

This module calls the function:

Here we include and API documentation for `XICRA` in order to provide third parties to use the functionality of `XICRA` application.

This pipeline is composed of multiple modules and scripts that are separated in a main script:

- `XICRA.py`

This main script integrates and connects all available modules and analysis arranged in several directories:

- `XICRA/modules`
- `XICRA/scripts`
- `XICRA/data`
- `XICRA/other_tools`
- `XICRA/config`

Contents

XX

3.1.1.3 Additional information

Contents

Project Status

The project is available on XICRA github. You can report issues or contribute to the project by forking the project and creating pull requests. See additional details on XICRA *developer guidelines* and how to work with Git.

Background

MicroRNAs (miRNAs), a class of small non-coding RNAs (ncRNAs), have an average length of 21–23 nucleotides (nt). They have been widely studied as endogenous regulatory molecules that modulate gene expression post-transcriptionally by inducing target mRNA silencing and decay. Additional roles beyond negative modulation of mRNA function have also been proposed.

Most miRNA expression studies based on next generation sequencing (NGS) performed to this date have summarized all the reads mapping to a specific miRNA locus or miRNA sequence with or without mismatches and assign it to a single miRNA entity (a miRBase reference database entry). However, this type of analysis neglects the fact that not all reads are identical to the mature reference sequence in miRBase. Small RNA sequencing NGS methodology has revealed that miRNAs can frequently appear in the form of multiple sequence variants or isoforms (termed isomiRs).

`XICRA` is a pipeline to analyze small RNA sequencing (small RNA-seq) data, which allows detecting and quantifying isomiR level variation in miRNA.

License

This is brief description of the license terms for `XICRA`.

Credits

Give credit to who deserves

Citation

Please cite the `XICRA` project, code or documentation using the following links:

Github statistics

These are Github statistics generated for each release of the code.

```
GitHub stats for 20xx/xx/xx - 20xx/xx/xx (tag: vX.X.X)
```

What's new?

See below information on differences among each release of the code generated.

3.1.1.4 Glossary

miRNA Class of small non-coding RNAs (ncRNAs), have an average length of 21–23 nucleotides. They are endogenous regulatory molecules that modulate gene expression post-transcriptionally by inducing target mRNA silencing and decay. Additional roles beyond negative modulation of mRNA function have also been proposed.

isomiR miRNAs can frequently appear in the form of multiple sequence variants or isoforms (termed isomiRs).

Reads An inferred sequence of base pairs (or base pair probabilities) corresponding to all or part of a single DNA fragment.

mirBase A searchable database of published miRNA sequences and annotation.

mirTop Command line tool to annotate with a standard naming miRNAs e isomiRs.

Pypi Python package index (Pypi, https://pypi.org/) is a repository of software for the Python programming language.

reStructuredText (ReST) reStructuredText format (http://docutils.sourceforge.net/rst.html). An easy-to-read, what-you-see-is-what-you-get plaintext markup syntax and parser system.

Sphinx Python module for documentation. See details in http://www.sphinx-doc.org/en/master/

3.1.1.5 Bibliography

3.1.2 Indices and tables

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] FastQC A Quality Control tool for High Throughput Sequence Data. URL: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (visited on 2020-01-24).

INDEX