
XICRA pipeline

Release 0.9.5

Jose F Sanchez-Herrero

Oct 27, 2021

CONTENTS

1	Introduction	1
2	Pipeline Scheme	3
3	Table of Contents	5
3.1	Contents	5
3.1.1	Documentation	5
3.1.1.1	User’s Guide	5
3.1.1.2	Developer Guidelines	18
3.1.1.3	Bibliography	19
3.1.2	Indices and tables	19

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 *<no title>*
- Some *developer Guidelines* to contribute to the project.
- Additional information of the XICRA project.
- A list of Glossary terms.
- A list of *Bibliography*

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.

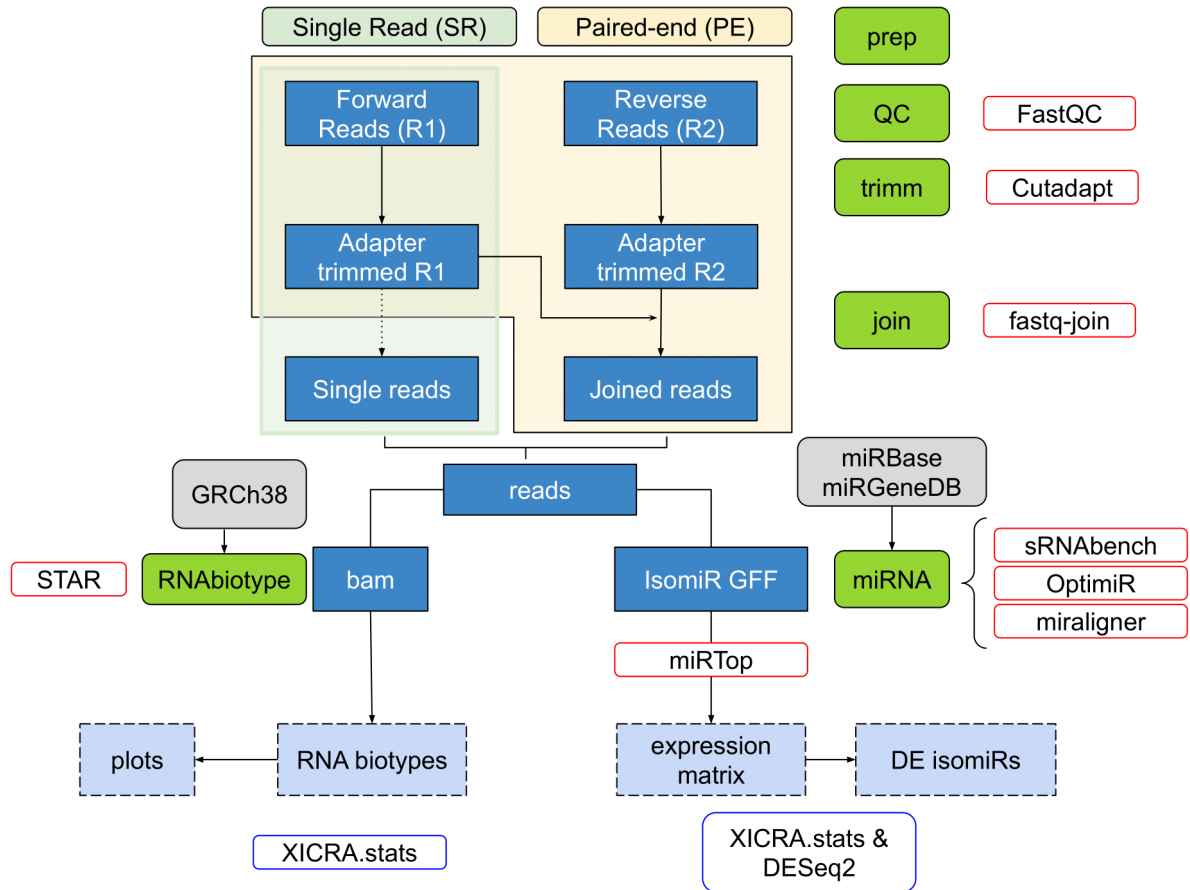


TABLE OF CONTENTS

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	sRNA-bench.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
feature-Counts	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 dependencies will be correctly 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

- *Installation*
 - *Requirements and dependencies*
 - * *System requirements*
 - * *XICRA requirements*
 - *Installing XICRA*
 - * *Installing from conda*
 - * *Installing from pip*
 - * *Installing from source*
 - *Get source code, for developers*
 - *Install XICRA from source*

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 appropriate choice according to your installing 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 circumstances (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 appropriate 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 functionality 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. Prepaion 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 analysis_tool`
 - `--threads X`: number of cores enabled to run the analysis.
 - `--software analysis_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 necessary 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 with any of the three available softwares `miraligner`, `optimir` and `sRNAbench` (they can be combined as seen in the example). Unfortunately, the downloading 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 appropriate 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 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.

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](#).

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 functionality 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 optional arguments

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.
- **--noTrim** – 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.

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 standardized 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_snps	sample-Name
iso-22-B175JXN0Q	AAACCGTTACCAT-TACTGAGTT	hsa-miR-451a	NA	0	0	0	0	69047
iso-23-B175JXN000	AAACCGTTACCAT-TACTGAGTTA	hsa-miR-451a	iso_add3p:0	0	0	1	0	169
iso-24-B175JXN0KF	AAACCGTTACCAT-TACTGAGTTAA	hsa-miR-451a	iso_add3p:0	0	0	2	0	1
iso-23-B175JXN005	AAACCGTTACCAT-TACTGAGTTC	hsa-miR-451a	iso_add3p:0	0	0	1	0	3
iso-23-B175JXN00P	AAACCGTTACCAT-TACTGAGTTG	hsa-miR-451a	iso_add3p:0	0	0	1	0	108
iso-23-B175JXN00Q	AAACCGTTACCAT-TACTGAGTTT	hsa-miR-451a	iso_3p:+1	0	1	0	0	35289
iso-24-B175JXN0KO	AAACCGTTACCAT-TACTGAGTTTA	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, differentiating 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 containing fastq reads. Provide a unique path or 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

3.1.1.2 Developer Guidelines

Guidelines for developing the XICRA project.

API Overview

Modules

Developer guidelines for XICRA modules.

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 Bibliography

3.1.2 Indices and tables

- genindex
- modindex
- search