# smallrnaseq

Oct 23, 2020

# Contents

Contents:

# CHAPTER 1

# Introduction

**smallrnaseq** is a Python package for processing of small RNA seq data. This is used to elucidate the small RNA contents of deep sequencing reads. For non Python users there is a command line interface that is quite simple to use. Typically a lot of disparate tools are integrated to create pipelines for this kind of analysis. This can be quite cumbersome. Our objective is to perform the analyses using Python packages as far as possible allowing almost all requirements to be installed using the pip tool. An aligner such as bowtie are still needed.

## 1.1 Command Line Interface

Installing the package provides the command *smallrnaseq* in your path. This allows users is a command line interface to the library without the need for any Python coding at all. It provides a set of pre-defined functions with parameters specified in a text configuration file. This is documented in detail in the Using smallrnaseq section.

## 1.2 Screencast

A screencast tutorial for using the command line interface: https://www.youtube.com/watch?v=m24cuLyTqg0

## 1.3 Citation

If you use this software in your work please cite the following article:

**Farrell, D. (2017). smallrnaseq : short non coding RNA-seq analysis with Python. Bioarxiv. https://doi.org/10.1101/110585**

## 1.4 FAQ

**Which version of python is supported?**

This package should work with python>=2.7 and >=3.6.

**Does this package work in windows?**

In theory yes, but this has not been tested. We strongly recommend using a linux OS. You can run a Linux operating system inside windows using virtualbox if you don't have linux on a separate computer. Just make sure you have enough memory (probably 8GB min).

**How about OSX?**

Yes it should run. It's recommended to install python with anaconda (see also bioconda).

**Can I use another aligner?**

As long as the aligner produces sam/bam file output it should work. It simply needs to be integrated into the package using a small amount of code. This can be done on request. Bowtie (v1) is recommended otherwise.

**How reliable is the novel miRNA prediction?**

It is hard to benchmark such an algorithm as there is no 'gold standard'. Our approach is broadly similar to the miRanalyzer one in that we use a feature classifier to score likely precursors then add some filters to remove unlikely candidates. The precursor score is only one factor. We do not use a more sophisticated model like miRDeep2. Our version is designed to be fast and easy to interpret the results. It is highly recommended to use another tool to compare the results to. If you think to much junk is being returned you can raise the score cutoff/read_cutoff or vice versa.

# CHAPTER 2

# Installation

## 2.1 Linux

On most linux operating systems installations of Python should include the pip tool. If not use your distributions package manager to install pip first. Then the simple call below should install all dependencies.

You should first run the following commands for installing the pre-requisites that might not be on your system already.

Ubuntu:

```
sudo apt install python-dev samtools liblzma-dev libbz2-dev zlib1g-dev liblzo2-dev␣
↪python-scipy
sudo pip install smallrnaseq
sudo apt install bowtie
```

Fedora:

```
sudo dnf install zlib-devel bzip2-devel xz-devel samtools swig redhat-rpm-config␣
↪python-devel
sudo pip install cython pysam
sudo pip install smallrnaseq
sudo dnf install bowtie
```

Then finally run:

```
pip install smallrnaseq
```

Or for the current github version you can use:

```
pip install -e git+https://github.com/dmnfarrell/smallrnaseq.git#egg=smallrnaseq
```

### 2.1.1 Snap package

You can install the application with a single command as a snap on Ubuntu and other supported linux systems. This has the advantage of convenient updates and not having to worry about any dependencies as everything is packaged in the snap. To install from the command line:

```
snap install smallrnaseq
```

You can also visit the store page for the app at https://snapcraft.io/smallrnaseq

### 2.1.2 For python 3 installs

You may need to use the command pip3 instead if python 2 is also on your system, like in Ubuntu. When installing packages with apt you likely need to specify python 3. e.g. python3-numpy instead of python-numpy.

### 2.1.3 For python 2.7 ONLY

You might also need the future package. Run *pip install future* to install.

## 2.2 Mac OSX

You will need to make sure you have Python. Anaconda is recommended as it provides most of the package requirements built in. Follow these steps in order:

Download and run the Mac OS X installer from https://www.continuum.io/downloads. The installer will automatically configure your system to use the Anaconda Python. Close the terminal and start a new one.

You should then add the bioconda channels:

```
conda config --add channels conda-forge
conda config --add channels defaults
conda config --add channels r
conda config --add channels bioconda
```

Type this command to install the remaining requirements:

```
conda install pybedtools bx-python HTseq
```

smallrnaseq can then be installed using pip:

```
pip install smallrnaseq
```

### 2.2.1 bowtie on OSX

You can install bowtie with conda too but it may hang or give an error on the latest version of OSX (Sierra). Running `conda install bowtie=1.1.2` will install the older version which should work.

## 2.3 Windows

If you are a Windows user there are several options available:

1. Simply use linux running inside a virtualbox instance. see http://www.makeuseof.com/tag/how-to-use-virtualbox/
2. Install Windows Subsystem for Linux (WSL) and use the linux instructions above. This is confirmed to work on Windows 10.
3. Use conda and bioconda (see the OSX instructions below).

## 2.4 Vienna package

This is needed if you want to do novel miRNA prediction. It has to be installed separately on all systems. Go to https://www.tbi.univie.ac.at/RNA/#binary_packages and download the binary for your system. Note: this is not required if using the snap package.

## 2.5 Required dependencies

- numpy
- pandas
- matplotlib
- seaborn (requires scipy)
- HTSeq
- bx-python
- pyfaidx
- scikit-learn

## 2.6 Installing R for differential expression

R is **not** a requirement for this package but is currently needed to do differential expression analysis using the edgeR package. You will not need to use R directly at all. smallrnaseq handles pre-processing your count data according to the factors you want to compare.

Installation is via the package managers so on Ubuntu:

```
sudo apt install r-base
```

Go to https://cran.r-project.org/ and download the installers.

This is an extra package provided as part of the bioconductor project. You can install from the command line as follows:

```
## try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
biocLite("edgeR")
```

CHAPTER 3

# Using smallrnaseq

This page refers to using the Command Line Interface of smallrnaseq. For programmers using the API see code examples.

Installing the package provides the command *smallrnaseq* in your path. This allows users is a command line interface to the library without the need for any Python coding at all. It provides a set of pre-defined functions with parameters specified in a text configuration file. The primary input is one or more fastq files containing short read data from a small rna-seq experiment. Note: It is assumed they have already been adapter trimmed, if required.

## 3.1 Usage

Usage largely involves setting up the config file and having your input files prepared. Running the command *smallrnaseq -c default.conf* will create a new config file for you to work from if none exists. Just edit this with a text editor and then to execute:

```
smallrnaseq -c default.conf -r
```

Several other functions are available from the command line without the config file, i.e. to collapse or trim reads. Type *smallrnaseq -h* to get a list of options.

## 3.2 Configuration file settings

The advantage of configuration files is in avoiding long commands that have to be remembered or are prone to mistakes. Also the config files can be kept to recall what setting we used or to copy them for another set of files. The current options available in the file are shown below. The meaning of each option is explained explained below. If you are unsure or don't require an option value, leave it at the default. Options can be excluded from the file completely and defaults will be used but it's recommended to just leave unused options blank. You can also comment out lines with a '#' at the start if you want it to be ignored. The [base] heading should always be present an indicates a section of the file. The [aligner] section is for setting alignment parameters on a per library basis if you need to. The [novel] section is for a few novel mirna prediction settings. The [de] section is used for differential expression which can be ignored if you don't need that:

```
[base]
filenames =
path =
overwrite = 0
index_path = indexes
libraries =
ref_fasta =
features =
output = results
add_labels = 1
aligner = bowtie
mirna = 0
species = hsa
pad5 = 3
pad3 = 5
verbose = 1
cpus = 1

[aligner]
default_params = -v 1 --best
mirna_params = -v 1 -a --best --strata --norc

[novel]
score_cutoff = 0.8
read_cutoff = 50

[de]
sample_labels =
sep = ,
sample_col =
factors_col =
conditions =
logfc_cutoff =
```

Settings explained:

| name | example value | meaning |
|---|---|---|
| filenames | test.fastq | input fastq file(s) with reads, comma separated |
| path | testfiles | folder containing fastq files instead of listing files |
| index_path | indexes | location of bowtie or subread indexes |
| aligner | bowtie | which aligner to use, bowtie or subread |
| ref_fasta | hg19 | reference genome fasta file, optional |
| libraries | RFAM_human,mirbase-hsa | names of annotated library indexes to map to |
| features | Homo_sapiens.GRCh37.75.gtf | genome annotation file. ONLY needed for counting genomic features |
| output | smrna_results | output folder for temp files |
| add_labels | 1 | whether to add labels to replace the file names in the results (0 or 1) |
| mirna | 0 | run mirna counting workflow (0 or 1) |
| species | bta | mirbase species to use |
| pad5 | 3 | 3' flanking bases to add when generating mature mirbase sequences |
| pad3 | 5 | 5' flanking bases to add |
| verbose | 1 | print extra information |
| cpus | 1 | number of threads to use |
| sample_labels | samplefile.txt | csv file with sample labels |
| default_params | -v 1 –best | default alignment parameters |
| mirna_params | -v 1 -a –best –strata –norc | default miRNA alignment parameters |

## 3.3 Examples

### 3.3.1 Mapping to one or more libraries of known RNAs

This will simply consist of setting the *libraries* option to match the names of files you have created aligner indexes for. You can download RNA annotations in fasta format from http://rnacentral.org/. The index files should be placed in the *index_path* folder. To create an index using bowtie you supply a fasta file and the indexes are placed in a folder called 'indexes'. You can move them to another folder if needed:

```
smallrnaseq -b myrnas.fa
```

By default file names are replaced with short ids of the form s01, s02, etc. This also writes out a file called *sample_labels.csv* in the output folder. Set *add_labels = 0* if you don't want this behaviour.

### 3.3.2 Mapping to known miRNAs and finding novel miRs

Say we have a set of fastq files in the folder 'testfiles' that we want to count miRNAs in. We would set the options *mirna = 1* and *path = testfiles*. Note if just mapping to mirbase mature/precursor sequence you don't have to create an index file since it is generated automatically. If you are using any other libraries you should create the aligner index first. For novel miRNA discovery we need the reference genome and index for that.

### 3.3.3 Mapping to a reference genome with small RNA features

There are some advantages to using a full reference genome in that it allows us to see of the reads align outside the target transcripts and we might therefore exclude them. Also we can normalise the read counts by the sum of all mapped reads. This depends on what features you have in the gtf file. To count our reads using features in a reference genome, provide the *ref_genome* name which should correspond to the index name of the reference sequence. We also need to set *features*, a set of features stored in a gtf, gff or bed file. Remember that the coordinates in this file must correspond to the reference sequence. See Counting features for more information.

## 3.4 Outputs

The main outputs are csv files with the counts for each sample in a column, also produced is a 'long form' csv file with a separate row for every sample. These csv files can be opened in a spreadsheet. Temporary files such as collapsed read files are placed in a 'temp' folder inside the output folder.

When you run the mirna workflow on a set of samples, a file called mirbase_mature_counts.csv is created in the folder. The column names are mostly self explanatory. Each sample has a column for the raw counts and normalized counts. *mean_norm* is the normalized mean and *total_reads* is the sum of all raw read counts:

```
name,db,sample1,sample2,sample3,sample1 norm,sample2 norm,sample3 norm,total_reads,
↪mean_norm
bta-miR-486,mirbase-bta,1444,1070,5579,176722.56,47917.6,127569.57,239793,284398.062
bta-miR-122,mirbase-bta,693,10676,4,84812.14,478101.21,91.46,11409,149778.2825
bta-miR-423-5p,mirbase-bta,337,100,2008,41243.42,4478.28,45914.98,4270,62353.636
bta-miR-22-3p,mirbase-bta,315,1053,5655,38550.97,47156.29,129307.39,7462,45154.618
bta-miR-92a,mirbase-bta,332,891,2484,40631.5,39901.48,56799.21,5989,39094.122
bta-miR-192,mirbase-bta,659,656,1482,80651.08,29377.52,33887.45,2945,33098.118
bta-miR-21-5p,mirbase-bta,453,227,1311,55439.97,10165.7,29977.36,2054,27358.996
..
```

## 3.5 Differential expression

This workflow is done using a separate command and via some extra options not shown above for clarity. To execute this type of analysis you must have done gene counting (e.g. miRNAs) and have the results in a csv file. The analysis is then run using:

```
smallrnaseq -c default.conf -d
```

In the default file the additional options needed are in a separate [de] section. Most are blank by default. If you want to do differential expression analysis of the genes from your results the other main thing you need to provide is a **sample label file** that matches the filenames you have analysed to the conditions. You then choose which conditions you want to compare. The options are explained below.

| name | example value | meaning |
|---|---|---|
| count_file | mirna_counts.csv | csv file with gene counts |
| sample_labels | labels.txt | csv/text file with sample labels |
| sep | , | separator for sample labels text file |
| sample_col | file_name | column for the sample file names |
| factors_col | status | column for factor labels |
| conditions | control,infected | conditions to compare from factor column (each replicate will have the same condition) |
| logfc_cutoff | 1.5 | cutoff for log fold changes (lower are ignored) |

This analysis is run using the R edgeR package, so R must be installed. See Installing R.

### 3.5.1 Example

The sample label file below shows how we use the above options. Our filenames are in the Run_s column. We want to compare the conditions 3 and 6 months samples in the age_s (the 'factor') column. mirna_mature_counts.csv is the file where counts from our mirna analysis are stored. This should have a column for each sample. Note that you could use counts output from any program as long as they are csv and have the right column names, see the output file formats section. It is assumed you are using replicates. Note that column names are case sensitive:

```
Run_s age_s   isolate_s
SRR3457948    3 months        animal1
SRR3457949    6 months        animal1
SRR3457950    6 months        animal4
SRR3457951    15 months       animal4
SRR3457952    3 months        animal5
SRR3457953    6 months        animal5
SRR3457954    15 months       animal5
SRR3457955    3 months        animal6
SRR3457956    6 months        animal6
...
```

So the config file will look like this:

```
[de]
sample_labels = SraRunTable.txt
sep = tab #tab delimiter
count_file = mirna_mature_counts.csv
sample_col = Run_s
factors_col = age_s
conditions = 3 months,6 months
logfc_cutoff = 1.5
```

## 3.6 Adapter trimming

Since there are numerous programs to perform this task it is left to the user to perform trimming of the reads prior to input.

## 3.7 Aligners

Traditional sequence alignment tools like BLAST are not well suited for next generation sequencing where one needs to align millions of short sequences very quickly. This has given rise to the development of a new class of short read aligners of which there are now dozens available. For small RNA-seq data alignment present certain specific challenges but standard aligners used for normal RNA-seq are usually adequate with the right parameters. The aligner and parameters will have an effect on the results, so trying more than one might be a good idea.

Currently smallrnaseq integrates bowtie (version 1) and subread. Though others can be added on request. These are free and easy to install on linux and OSX systems. On Ubuntu the following command installs both:

```
apt install bowtie subread
```

### 3.7.1 Links

- http://subread.sourceforge.net/

- http://bowtie-bio.sourceforge.net/index.shtml

## 3.8 Alignment settings

The parameters used for the alignment/mapping procedure can be important in the final counts produced, irrespective of the aligner used.

This can be a complex topic in itself and general users will be confused by the many options. The command line tool for smallrnaseq takes the simple approach of providing a default alignment parameter for general mapping to libraries, another for mapping miRNAs and one for reference genomes. All can be changed in the config file if needed. You can also set custom parameters per library in the aligner section.

### 3.8.1 Bowtie

For general mapping `-v 1 --best` is used. `-v 1` reports read mappings with up to one mismatch, options `--best` orders the mappings from best to worse alignments.

In miRDeep2 when mapping to the mature miRNAs (miRBase sequences) for mature quantification the following parameters are used:

```
-v 1 -a --best --strata --norc
```

Here `-a` means report all valid alignments, options `--best --strata` orders the mappings from best to worse alignments according to the strata definition of bowtie. `--norc` means do not map reads to the reverse complement of the sequences.

For reference genome mapping miRDeep2 uses these parameters:

```
-n 0 -e 80 -l 18 -a -m 5 --best --strata
```

### 3.8.2 Subread

`-m 2 -M 1` is the default for general alignment to libraries. If you use subread you can check the parameters by typing `subread-align --help` at the command line, or refer to the website.

### 3.8.3 Configuration file

In the aligner section set your parameters. In the example below bos_taurus is the name of the reference genome. We have also used custom settings for mirna and another library of tRNAs.

```
[aligner]
mirna_params = -n 1 -l 20
bos_taurus = -v 1 -k 50
bosTau8-tRNAs = -v 0 --best
```

### 3.8.4 Code example

If using the package in your python code, aligner parameters are set via the aligners module. This is done before calling mapping routines such as `map_rnas`.

for example:

```
from smallrnaseq import aligners
aligners.BOWTIE_PARAMS = '-v 0 --best'
aligners.SUBREAD_PARAMS = '-m 0 -M 1'
```

## 3.9 References

- Shi, J., Dong, M., Li, L., Liu, L., Luz-Madrigal, A., Tsonis, P. A.,

  … Liang, C. (2015). mirPRo–a novel standalone program for differential expression and variation analysis of miRNAs. Scientific Reports, 5, 14617. http://doi.org/10.1038/srep14617

- Friedländer, M. R., Mackowiak, S. D., Li, N., Chen, W., & Rajewsky,

  N. (2012). miRDeep2 accurately identifies known and hundreds of novel microRNA genes in seven animal clades. Nucleic Acids Research, 40(1), 37–52. http://doi.org/10.1093/nar/gkr688

CHAPTER 4

# Code Examples

## 4.1 Counting microRNAs

Requires you provide at least one fastq file and have a short read aligner installed. You should also specify the species being mapped to:

```
import smallrnaseq as smrna
res = smrna.map_mirbase(files=['test_1.fastq','test_2.fastq'], overwrite=True,␣
↪aligner='bowtie',
                    species='hsa', pad5=3, pad3=5)
```

## 4.2 Counting isomiRs

This method is used to count isomiRs using results from previously mapped reads. So a sam file is required:

```
smrna.count_isomirs(samfile, truecounts, species='bta')
```

## 4.3 Mapping to the genome

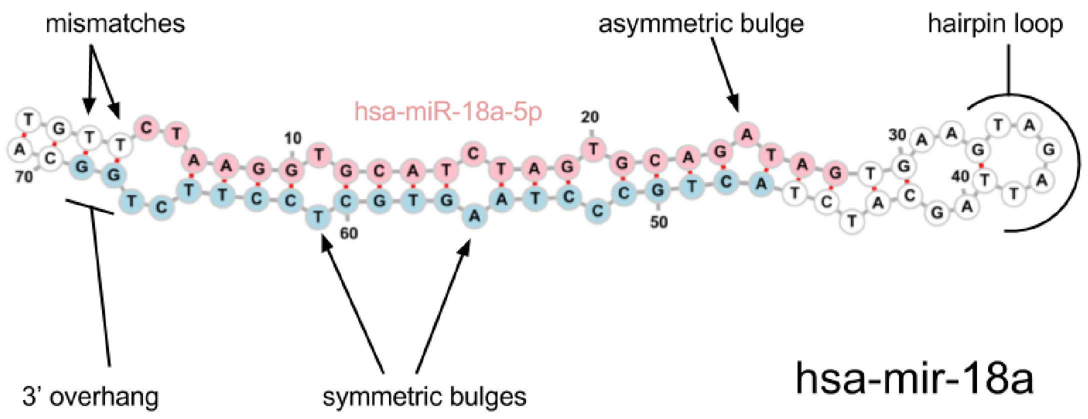
This requires a reference genome and a gtf file with miRNA features:

```
featcounts = srseq.map_genome_features(['test_1.fastq'], 'bos_taurus', gtffile,
                                    outpath='ncrna_map', aligner='subread',␣
↪merge=True)
```

## 4.4 Novel miRNA prediction

The built-in method for novel prediction should be considered a somewhat ‘quick and dirty’ method at present but is relatively fast and convenient to use. The basic idea is to take clusters of reads that could be mature sequence and find suitable precursors. Structural features of each precursor are then scored using a classifier. The best candidate is selected is there is at least one. We have followed a similar approach to the miRanalyzer method.

The following features are currently used in our algorithm, most are the same as those used in sRNAbench (miRanalyzer). The diagram below may help to clarify some of the terminology used.



To predict miRNAs you need to have run mapping on genome. Then use the sam file and read counts to get the true reads and input this into the method find_mirnas with a reference genome fasta file. The reference fasta must match the bowtie index you used for alignment:

```
from smallrnaseq import novel
import pandas as pd
#single file prediction
readcounts = pd.read_csv('countsfile.csv')
samfile = 'mysamfile.sam'
reads = utils.get_aligned_reads(samfile, readcounts)
new = novel.find_mirnas(reads, ref_fasta)
```

## 4.5 Differential Expression

Assuming we have all the raw files, they need to be adapter trimmed. Optionally you can remove other ncrnas before counting your target rnas class, though that may not be advisable. The following code maps all the files to bovine mature miRNAs and counts the mapped genes, then saves the results to a csv file which has the counts in one column per sample. You can skip this if you already have the counts file:

```
import pandas as pd
import smallrnaseq as smrna
from smallrnaseq import base, utils, de

path = 'pathtodata'
base.BOWTIE_INDEXES = 'bowtie_index'
refs = ['mirbase-bta'] #name of bowtie index

files = glob.glob(path+'/*.fastq')
```

(continues on next page)

(continued from previous page)

```
outpath = 'ncrna_map'
#map to selected annotation files
counts = smrna.map_rnas(files, refs, outpath, overwrite=True)
R = smrna.pivot_count_data(counts, idxcols=['name','db'])
R.to_csv('mirna_counts.csv',index=False)
```

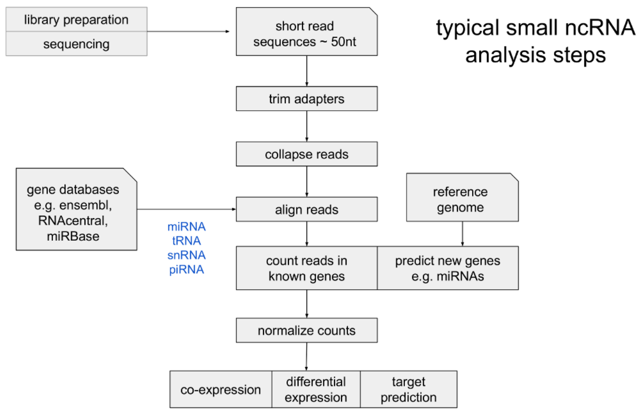
# CHAPTER 5

# Methodology

smallrnaseq is a Python package that implements some of the standard approaches for quantification and analysis of sncRNAs. This is usually implemented as part of a 'pipeline' that goes from raw fastq files to final counts of specific genes (e.g. coding transcripts or RNA species).

For command line usage see the 'using smallrnaseq' section.

Python programmers may find the various modules useful in creating their own more flexible workflows. A functional approach is used with a flat hierarchy of modules with limited use of classes.



## 5.1 Counting known miRNAs

This program uses the known miRNA sequences from miRBase for counting of species specific sequences. The current release used is version 22 (March 2018).

## 5.2 Counting isomiRs

Much miRNA expression profiling uses the read counts of all isoforms (isomiRs) of the canonical mature sequence summed together. While this may be appropriate for many cases it is also useful to consider the isoforms separately. This is true for several reasons:

1. you only wish to count the exact canonical sequence.
2. if the canonical mature sequence in miRBase is not even common in the tissue or samples you are studying and you wish to know this information.
3. your samples contain two isoforms that differ such that one will not be detected in (for example) a PCR assay, giving an inconsistent result.
4. one or more isoforms can better distinguish between sets of samples (i.e. control and disease) than the entire sum of counts.

### 5.2.1 Classifying isomiRs

Sequence variants include 5' and 3' trimming and extension, non-templated additions (enzymatically addition of a nucleotide to the 3' end, i.e. adenylation, uridylation). A single read can have more than one modification so sRNAbench provided a useful non-redundant hierarchical method of classifying isomiRs. See reference below. A similar scheme has been used by Loher et al. The sRNAbench scheme is illustrated below:

### 5.2.2 isomiR counting

We have implemented the sRNAbench naming scheme. isomiR data is output by default when the `map_mirbase` method is called.

If required, you can explicitly call the routine to count isomirs, assuming you have already aligned to the mirbase sequences and have a sam file. You should also provide the original read counts loaded from the csv file which is always created when the reads are collapsed.

```
smrna.count_isomirs(samfile, truecounts, species='bta')
```

### 5.2.3 References

- Barturen, G., Rueda, A., … Hackenberg, M. (2014). sRNAbench: profiling of small RNAs and its sequence variants in single or multi-species high-throughput experiments. Methods in Next Generation Sequencing, 1(1), 21–31.
- Telonis, A.G. et al., 2015. Beyond the one-locus-one-miRNA paradigm: microRNA isoforms enable deeper insights into breast cancer heterogeneity. Nucleic Acids Research, 43(19), pp.9158–9175.
- Loher,P., Londin,E.R. and Rigoutsos,I. (2014) IsomiR Expression Profiles in Human Lymphoblastoid Cell Lines Exhibit Population and Gender Dependencies. Oncotarget, 5, 8790–8802.

## 5.3 Counting genomic features

Counting features means **counting the overlap of your reads with the locations of gene annotations** (the features) in a reference genome. A short read aligner is the first step. Note that if you are aligning to a human-sized reference genome, creating the index from the fasta file can take some time. Also this process can use a lot of memory. See the section on Aligners for more information.

After read alignment to a reference genome the next step in transcriptomics is usually to count the features that each read intersects. This done using the sam or bam file generated by the aligner (e.g. bowtie) along with a gtf or gff file that stores the annotated genes - the features. These files are available for many spedcies from Ensembl. These will include annotations for non coding RNAs.

### 5.3.1 Counting features

```
import smallrnaseq as smrna
import pandas as pd
readcounts = pd.read_csv('counts.csv')
fcounts = smrna.count_features(samfile, gtffile, truecounts=readcounts)
```

This returns a pandas dataframe of the form:

```
                   name   reads
259         _no_feature  290705
50            _unmapped  145281
645  ENSBTAT00000060484   15162
287  ENSBTAT00000042309    6284
538  ENSBTAT00000051764    5575
...
```

This will merge the gtf file fields with the read counts, which can also be done explicitly using:

```
res = base.merge_features(fcounts, gtffile)
```

### 5.3.2 Links

- HTSeq feature counting
- Ensembl ftp downloads
- Ensembl genome annotation

## 5.4 Novel miRNA prediction

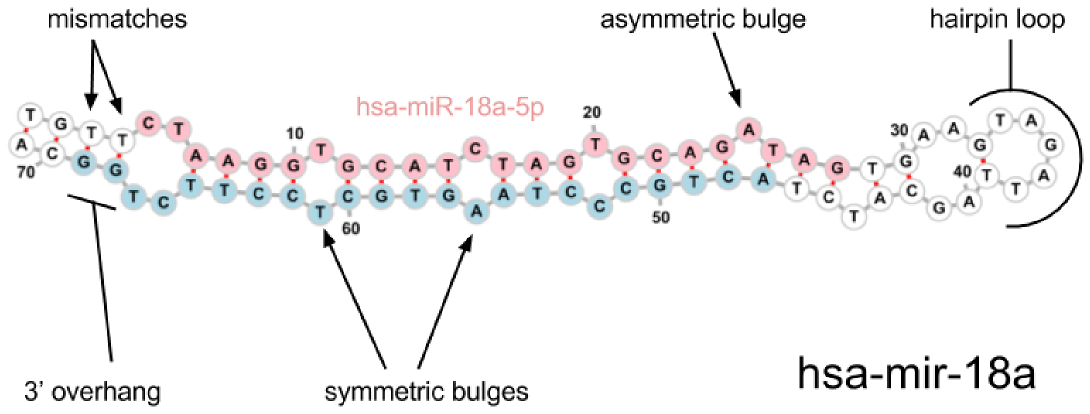
**Novel miRNAs** in a species are generally those for which the mature sequence is not present in miRBase. This may be because the species has not been well studied or insufficient evidence is available to consider a sequence a real miRNA. There are problems with false positives in general for miRNA discovery from deep sequencing and this should be kept in mind when running an analysis of your reads. It is advised to use at least two algorithms for such an analysis. Most of the higher abundance miRNAs have been identified in the commonly studied species. Detection/prediction of new mirs will therefore involve looking at the low abundance (or tissue specific) forms which will need further evidence such as conservation, experimental verification and perhaps identification of function.

There are by now multiple algorithms available for predicting novel miRNAs from small RNA sequencing data. The most popular is probably miRDeep2. This algorithm is accessible via the mirdeep2 module assuming you have installed mirdeep2 as well. However we include a native algorithm in smallrnaseq for novel prediction, this is described below.

### 5.4.1 Method

The built-in method for novel prediction should be considered a somewhat 'quick and dirty' method at present but is relatively fast and convenient to use. The basic idea is to take clusters of reads that could be mature sequence and find suitable precursors. Structural features of each precursor are then scored using a classifier. The best candidate is selected is there is at least one. We have followed a similar approach to the miRanalyzer method.

The following features are currently used in our algorithm, most are the same as those used in sRNAbench (miRanalyzer). The diagram below may help to clarify some of the terminology used.



| Feature | Description |
|---|---|
| Length | The length of the longest hairpin structure |
| Stem length | The length of the longest hairpin structure stem |
| Mfe | The mean free energy of the hairpin |
| Loop length | The number of bases in the loop of the hairpin |
| Loop GC | The GC-content of the loop |
| GC | The GC-content of the small hairpin |
| Asymmetric bulges | The number of asymmetric bulges and mismatches in the stem |
| Symmetric bulges | The number of symmetric bulges and mismatches in the stem |
| Bulges | The number of bulges in the stem |
| Longest bulge | The number of non-pairing nucleotides of the longest bulge |
| Hairpin mismatches | The number of single mismatches in the hairpin |
| Mature mismatches | The number of single mismatches in the mature microRNA region of the hairpin |
| Triplet-SVM features | All features that were proposed by Xue et al. |

#### Algorithm:

The basic steps for novel precursor detection are as follows.

1. if multiple samples, read counts for all samples are put together using each sam file and total counts for each unique read (after collapsing the original fastq files)
2. the reads are sorted by read count
3. reads are clustered using cluster trees
4. read clusters are themselves clustered to detect pairs within 100 nt (for animals). these are considered to be possible mature/star arms and checked for hairpin structure

5. single clusters are checked for precursors by creating multiple possible precursors at the 5’ and 3’ ends and evaluating the features
6. a precursor is discarded if:

- it has no hairpin
- its read cluster overlaps with the hairpin loop
- it has less than 19 bindings in the stem
- it has less than 11 bindings to the region occupied by the read cluster

7. a random forest classifier/regressor trained on known positives and negatives is used to score the precursor features
8. the precursor with the lowest energy and highest score is used as the most likely candidate
9. for single clusters this region is considered the mature arm and the star sequence is estimated
10. the list of novel miRNAs with precursor, mature, star, reads is output

### 5.4.2 Command line usage

Using the command line tool it is simply a matter of setting `novel = 1` in the config file. Reads will first be mapped to mirbase to remove known miRNAs and also any libraries you included in the indexes option. The novel prediction step from the command line tool will produce a file called `novel_mirna.csv` in the output folder.

### 5.4.3 Code examples

```python
from smallrnaseq import novel
import pandas as pd
#single file prediction
readcounts = pd.read_csv('countsfile.csv')
samfile = 'mysamfile.sam'
reads = utils.get_aligned_reads(samfile, readcounts)
new = novel.find_mirnas(reads, ref_fasta)
```

### 5.4.4 References

- Kang, W. & FriedlÃ¤nder, M.R., 2015. Computational Prediction of miRNA Genes from Small RNA Sequencing Data. Frontiers in Bioengineering and Biotechnology, 3, p.7.
- Hackenberg, M. et al., 2009. miRanalyzer: a microRNA detection and analysis tool for next-generation sequencing experiments. Nucleic Acids Research, 37(Web Server), pp.W68–W76.
- Friedländer, M.R. et al., 2012. miRDeep2 accurately identifies known and hundreds of novel microRNA genes in seven animal clades. Nucleic acids research, 40(1), pp.37–52.
- Shi, J. et al., 2015. mirPRo–a novel standalone program for differential expression and variation analysis of miRNAs. Scientific Reports, 5, p.14617.
- Xue, C. et al., 2005. Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. BMC bioinformatics, 6, p.310.
- Lopes, I.D.O.N. et al., 2014. The discriminant power of RNA features for pre-miRNA recognition. BMC bioinformatics, 15(1), p.124.

# CHAPTER 6

# smallrnaseq

## 6.1 smallrnaseq package

### 6.1.1 Submodules

### 6.1.2 smallrnaseq.aligners module

### 6.1.3 smallrnaseq.analysis module

### 6.1.4 smallrnaseq.app module

### 6.1.5 smallrnaseq.base module

### 6.1.6 smallrnaseq.config module

### 6.1.7 smallrnaseq.de module

### 6.1.8 smallrnaseq.ensembl module

### 6.1.9 smallrnaseq.mirdeep2 module

### 6.1.10 smallrnaseq.novel module

### 6.1.11 smallrnaseq.plotting module

### 6.1.12 smallrnaseq.srnabench module

### 6.1.13 smallrnaseq.tests module

### 6.1.14 smallrnaseq.trf module

### 6.1.15 smallrnaseq.utils module

### 6.1.16 Module contents

# CHAPTER 7

# Indices and tables

- genindex
- modindex
- search