miRNAgFree manual

Last updated 15/7/2017

1 Introduction

miRNAgFree is a piece of software that allows for mature microRNA prediction using sRNAseq/miRNAseq/sncRNAseq without needing a genome or miRNA sequence libraries. This tool is therefore ideal for non-model species with genomes yet to be sequenced or for those lacking an appropriate quality.

This program can use input files accepted by sRNAbench[1] (also developed by our group) which includes fasta, fastq and read counts. Fundamentally, the workflow consists of 6 steps: preprocessing of the input (adapter trimming, quality control), read filtering (although optional, many unwanted redundant conserved sequences can be removed providing an eukariotyc ribosomal RNA library for example), clustering of the reads, calculation of duplexes using RNAcofold[2], selection of microRNA-like duplexes and output layer (which includes duplex visualization).

Our approach outperforms previous similar attempts because microRNA biogenesis features were taken into account. We benchmarked miRNAgFree using a set of species with high quality genomes (including *h.sapiens, mus musculus, C.elegans, D.Rerio)* using publicly available datasets. When measuring the specificity on the guide strand of the microRNA we obtained over 90% accuracy for the most expressed quartile of duplexes.

2 Getting started

2.1 Standalone version

Currently miRNAgFree can be downloaded as a standalone program:

2.1.3 Dependencies

miRNAgFree is implemented in JAVA (so it needs a JRE) and has the following third party software as dependencies (need to be installed **and placed in the PATH** first):

- Vienna RNA package for RNA Secondary Structure Prediction and Comparison Vienna package 2. **miRNAgFree will only work with Vienna 2.0 or higher!**

  The instructions below are for installing the ViennaRNA package from source. However, pre-compiled binaries for various Linux distributions, as well as for Windows users are available from Download section of the main ViennaRNA homepage.

Linux Quick-install

```
sudo apt-add-repository ppa:j-4/vienna-rna
sudo apt-get update
sudo apt-get install vienna-rnatar -zxvf ViennaRNA-2.4.0.tar.gz
```

Rest of OS

Check out the software Download Section: https://www.tbi.univie.ac.at/RNA/index.html#download

- (Optional-for *libsfilter* only) Bowtie - An ultra-fast memory-efficient short read aligner (Bowtie). **miRNAgfree will only work with Bowtie1 but not Bowtie2**.

Linux Quick-install

```
sudo apt-get update
sudo apt-get install bowtie
```

Rest of OS
You may download either Bowtie sources or binaries for your platform from the Download section of the Sourceforge project site. Binaries are currently available for 64-bit Intel architectures running Linux, Windows, and Mac OS X. Check out the software Download Section: https://sourceforge.net/projects/bowtie-bio/files/bowtie/

- (Optional-for SRA data download only)SRA tool kit: only if the user wants to use data from SRA as input files as those need to be converted to fastq first.

Check out the Downloads section for SRA-tools: https://github.com/ncbi/sra-tools/wiki/Downloads

Check out the installation guidelines: https://github.com/ncbi/sra-tools/wiki/HowTo:-Binary-Installation

3 Main features, implementation and modes

- Adapter trimming can be performed and miRNAgFree accepts *fastq*, *fastq.gz*, read count and *fasta* input format
- A preprocessing filtering step can be included. This is useful in a number of scenarios: for example to remove unwanted ribosomal sequences or if there are some already described microRNAs that you are not interested in. Reads mapping to provided libraries will not be considered for downstream analysis.
- Biogenesis features were taken into account so it's very specific
- Several parameters can be adjusted to allow for more sensitivity and that can be tuned depending on the needs of your experiment/hypothesis (check the **Parameters section**).

3.1 Implementation steps

The general workflow consists in i) preprocessing of the input reads, ii) read filtering (optional if the user provides a filter library like ribosomal RNA), iii) clustering of the reads, iv) calculation of duplexes with RNAcofold , v) detection of microRNA-like duplexes and vi) output layer including visualization the detected miRNA duplexes.

3.1.1 Clustering method

The mature microRNAs are normally represented at the read level by the canonical sequence (i.e. the one in miRBase) and its isomiR sequences. Therefore, in order not to predict a microRNA several times due to duplexes formed by its isomiR sequences, we first cluster together all reads. Briefly the algorithm performs the following steps on a sorted read list (descending order)

(1) Open a cluster with the most abundant read as dominant read

(2) Take the most abundant read as reference and align all other reads against it. The reads can align with a 3 nt overhang at the 5' end allowing by default 1 mismatch (suppl. figure 1a). The last nucleotides are ignored as those might be NTAs (non-templated additions)

(3) Remove all assigned reads so they are not considered for other clusters

(4) Repeat steps 1-3 until no reads are left.

3.1.2 Detection of miRNA duplexes

The detection of novel microRNAs consists of two steps: i) selection of clusters that resemble those formed by real guide microRNAs and their isomiRs and ii) the assignment of the most likely passenger sequence.

(1) Sort clusters by total expression value in descending order.

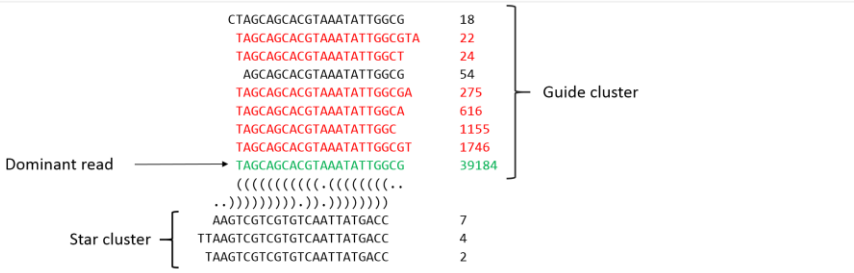
(2) Pick the most expressed cluster and remove it from the list and evaluate its microRNA potential based on several criteria like the 5' homogeneity or the ratio of the most expressed read to the read count sum of the cluster. The same thresholds as in *Barturen et al., 2014* were used. The user can choose between lax and strict settings for both animals and plants.

(3) For all putative cluster pairs, calculate all duplexes for the (M) most expressed reads in cluster (i) vs the (N) most expressed clusters of cluster (j) and remove all that do not have perfect 2 nucleotide 3' Drosha/Dicer overhangs (strict) or at least 1-3 nt overhang (lax mode)

(4) Sort the duplexes by energy ratio (mean free energy divided by the sequence length) and assign the energetically most favorable read and its cluster to the guide cluster.

3.2 Strict and lax mode

If one considers an example cluster-duplex (like the one presented in the figure below) there are a number of thresholds that must be surpassed for a given set of features. These thresholds were calculated as explained in *Barturen et al., 2014*.



Feature	Feature description	Threshold
5'fluctuation	Fraction of reads starting at the same position as the dominant read (reads shown in red).	Lax threshold = 0.106 Strict threshold= 0.508
Dominant to all ratio	Ratio between most frequent read (shown in green) and the rest.	Lax threshold = 0.395 Strict threshold = 0.614
Number of unique reads in cluster	Minimum number of unique reads in the dominant cluster (example case 8)	Threshold = 4
Number of reads for the dominant read	Minimum number of reads required for the dominant read (example case 39184)	Threshold = 10
Number of bindings	Minimum accepted number of bindings in the duplex	This parameter is chosen by user (default=14)

3.3 Processing mode

Under construction

3.4 Kingdom: animal and plant

Under construction

4 Quick start

4.1 Simple case: Using default parameters

```
java –jar miRNAgFree.jar input=example_file.fastq
output=example_output_folder adapter= TCGTATGCCG
```

Explanation of used Parameters

- **input=example_file.fastq** : a fastq, fasta and *read/count* format resulting from your sequencing experiment is accepted.
- **adapter=TCGTATGCCG**: the adapter sequence that will be trimmed of the 3' end of the reads
- **output=example_output_folder** : the directory where the output will be stored

Important output files

- **reads.fa and reads_orig.fa:** after the process finishes, reads.fa contains all unmapped reads, while reads_orig.fa contains the initial set of adapter trimmed and cleaned reads.
- ***logFile .txt***: a log file that protocols all steps – it contains all errors and warnings that might have occurred.
- ***short_reads.txt***: the reads filtered out due to minReadLength parameter (default 15nt)
- ***microRNAs.txt***: a file containing different measures of expression (including RPM and Read Counts) for each microRNA prediction. They are presented in descending order.
- ***mature.fa:*** a multifasta file containing both strand sequences of the predicted microRNA.

4.2 A bit more advanced: setting mode to lax and adding a filtering library

By default, miRNAgFree uses a set of parameters quite strict, which yields a very specific set of predictions. However, this can be easily reverted using a more lax mode, which will be far more sensitive sacrificing some of its specificity.

The user can also add a library of sequences to filter unwanted RNA material (ribosomal RNA, already described microRNAs, etc). Reads mapping to this library will not be considered for downstream analysis.

```
java –jar miRNAgFree.jar input=example_file.fastq
output=example_output_folder adapter= TCGTATGCCG
libsFilter=ribosomal.fa mode=lax
```

Explanation of used Parameters

- **input=example_file.fastq** : a fastq, fasta and *read/count* format resulting from your sequencing experiment is accepted.
- **adapter=TCGTATGCCG**: the adapter sequence that will be trimmed of the 3' end of the reads
- **output=example_output_folder** : the directory where the output will be stored
- **libsFilter=ribosomal.fa :** library used to filter unwanted sequences previous to the prediction. Please note that no library is included, you should provide your own.
- **mode=lax** miRNAgFree can use a set of lax or strict thresholds for a set of features, it uses *strict* by default (check the Features section).

Commented [b1]: Change for example.fa?

Commented [EA2]: Modes/features section where we explain the different modes

5 Parameters

5.1 Basic parameters

- **input=<String>**: the path to the input file (*fastq*, *read/count*, *fasta*). This is the only mandatory parameter. *read/count* format is simply like this

- **dbPath=<path to folder>**: the full path to the sRNAtoolbox database (default: dbPath=*/opt/sRNAtoolbox*)
- **output=<Folder>**: The output folder. (Default: output=*dbPath/out*)
- **libsFilter=<String>**: the name of the libraries that should be use to filter out certain reads prior to the expression profiling of microRNAs and other libraries. The reads mapping to those libraries are filtered out. The libraries should be given in fasta format or as bowtie indexes. **libsFilter=<String> can be given several times on the command line!**
- **solid=<Boolean>**: if set to true, SOLiD input data is expected. (default: solid=<false>)
- **p=<int>:** The number of threads that will be assigned. This is applied both to the bowtie alignment but also to the parallelized parts of miRNAgFree. Default: p=<4>
- **sep=<String>:** Only applies to fasta input format! This parameter allows to give the separator by which the 'ID' and the 'Read Count' are separated. For example: >1-45798 (ID=1, Read Count = 45798) would need sep=- (Default: sep=#)
- **mode=<strict,lax>** mode=strict: Both, for the thresholds and the patterns the high confidence parameters are used.
- **procMode=<strict,lax>** Processing (Drosha,Dicer) mode, strict: a strict 2nt overhang is requiered; lax = a 1 nt tolerance is allowed
- **nrNonDomGuide=<int>** the number of non-dominant reads that should be checked for hybrids in the dominant cluster guide). Default: nrNonDomGuide=-1 (only dominant read of the most expressed cluster considered). This is parameter M in the microRNA detection description (section 3.1.2).
- **nrNonDomStar=<int>** the number of non-dominant reads that should be checked for hybrids of the star cluster (passanger). Default: nrNonDomStar=-1 (only the dominant read of the passenger cluster is considered). This is parameter N in the microRNA detection description (section 3.1.2).
- **kingdom=<String>** Indicate if the data is from animal (kingdom=animal) or plant (kingdom=plant). This paramter only affects the prediction of novel microRNAs. Default: kingdom=animal

Commented [EA3]: Quitar de la ayuda?

Commented [EA4]: Añadir kingdom to the Modes section

5.2 Preprocessing trimming parameters

- **adapter=<String>:** the adapter sequence. If this parameter is NOT given on the command line, then the input is assumed to be adapter trimmed already (if guessAdapter=<false>)!
- **recursiveAdapterTrimming=<boolean>:** recursive search for the adapter at the 3' end. This function might be indicated for read length 36 if sRNA populations of length between 27 and 34 should be analyzed (default=<false>)
- **holdNonAdapter=<boolean>:** include also those reads into the data analysis for which the adapter sequence was not found (default: holdNonAdapter=<false>)
- **adapterStart=<int>:** the base in the read where the adapter search should be started in 0-based coordinates (default: adapterStart=0)
- **adapterMinLength=<int>:** the minimum length of the adapter that needs to be detected (default: adapterMinLength=10)
- **adapterMM=<int>:** the maximum number of mismatches allowed between the adapter sequence and the read (default: adapterMM=1)
- **writeNonAdapter=<boolean>:** write out the reads for which the adapter was not found (default: writeNonAdapter=<false>)

5.3 Preprocessing: Length and count thresholds

- **maxReadLength=<int>:** the maximum length of a input read (filters out all reads that are longer than <int>) (by default this filter is not applied)
- **minReadLength=<int>:** the minimum read length for a input read (filters out shorter reads) (default: minReadLength=15)
- **minRC=<int>:** the minimum read count of a read. Filter out reads with less read count than <int> (default: minRC=1)

5.4 Mapping parameters

These parameters will be used for the *libsfiltering* option.

- **noMM=<int>**: the number of mismatches. (default: noMM=0)
- **alignType=[n,v]:** the alignment type; can be either 'n' (-n parameter in Bowtie, i.e. alignType=n) or 'v' (-v parameter in bowtie, i.e. alignType=v). Note that when setting 'v', the seed parameter will have no effect. Briefly, 'n' will perform a seed alignment (only the first nucleotides are used for the alignment, i.e. mismatches outside the seed region do not count). For example, to detect isomiRs, 'n' must be used. On the other side, 'v' aligns the whole read, i.e. all mismatches do count. (default: alignType=n)
- **seed=<int>**: the length of the seed (-l parameter in Bowtie). (default: seed=19)