sRNAtoolbox standalone and *Docker* manual

Last updated: 12/04/2022

Developed and maintained by:

http://bioinfo2.ugr.es/
Dept. of Genetics & Inst. of Biotechnology,
University of Granada, Spain

Questions and feedback: *sRNAbench*@gmail.com or hackenberg@ugr.es

1 What is sRNAtoolbox?

- ***sRNAtoolbox*** is a collection of several tools for RNA based research, including expression profiling from NGS data, differential expression, analysis of unmapped reads with blast and consensus target prediction and analysis.
- The key tool of *sRNAtoolbox* is ***sRNAbench*** (Barturen *et al.*, 2014) which is the successor program of **miRanalyzer** (Hackenberg *et al.*, 2009, 2011) a tool for expression profiling of small RNAs and prediction of novel microRNAs.
- *sRNAtoolbox* is implemented into a **webserver**, **a Docker**, and some of the programs can be furthermore downloaded as **standalone versions**.

1.1 Brief history

- In 2009 we published the first web-server for the analysis of miRNA-seq data in Nucleic Acids Research (Hackenberg et al., 2009).
- In 2011 an updated version was released and published in NAR again (Hackenberg et al., 2011).
- Shortly after the publication in 2011, novel features like the detection and classification of isomiRs were added.
- In 2013, miRanalyzer was completely redesigned, re-implemented and renamed to *sRNAbench* published (**PDF**) in 2014 (Barturen et al., 2014).
- In 2014 the differential expression module of *sRNAbench* was redesigned and named *sRNAde*. This program counts in its last version with 5 different DE programs generating furthermore a consensus differential expression.
- In 2014/15 *sRNAbench*, *sRNAde* and other tools were published together as ***sRNAtoolbox*** (Rueda et al., 2015).
- In 2017 we published a protocol for small RNA analysis in Methods Mol. Biol. (Gómez-Martín et al., 2017).
- In 2019 a ***sRNAtoolbox* update** (Aparicio-Puerta et al., 2019) was published adding features like batch mode, extended databases and improved differential expression.

1.2 Main features of the tools

sRNAbench:

sRNAbench was first published as the replacement tool of **miRanalyzer**, but later it was incorporated into the ***sRNAtoolbox*** collection of sRNA tools.

Most important features include:

- Expression profiling of microRNAs (*miRBase* (Kozomara et al., 2019), *MiRGeneDB* (Fromm et al., 2022), *miRCarta* (Backes et al., 2018) and *PmiREN* (Guo et al., 2020)) and other small RNAs (*RNAcentral* (The RNAcentral Consortium, 2019)). Libraries that should be profiled can be customized and defined easily by the user.

- Detection and classification of isomiRs allowing both redundant and non-redundant classification schemas.
- Prediction of novel microRNAs (animal and plant models)
- Built in pre-processing and quality control of fastq input data. All major protocols (*Illumina*, *NEBnext*, *NextFlex*, *Qiagen*) are supported.
- Analysis of samples with UMIs and spike-ins molecules.
- *sRNAbench* can detect automatically the library processing protocol and/or the species.
- A basic analysis of sequence variants is performed, i.e. the positions with variants are counted and provided in an output file.
- Genome mapped reads in *bedGraph* and *BigWig* format can be reported. Those can then be used visualization in programs like **IGV.**
- Extensive statistics: read length distributions of mapped, unmapped, assigned and unassigned reads or as a function of assigned RNA species (for example, read length distribution of all reads assigned to a miRNA), genome mapping statistics (mapping distribution in a multi-species assay), and visualizations of the mature miRNA alignments.
- Basic classification and analysis of tRNA derived fragments.

sRNAde:

- Detection of differentially expressed small RNAs based on 4 commonly used programs: *DESeq* (Anders and Huber, 2010), *DESeq2* (Love et al., 2014), *edgeR* (Robinson et al., 2010), and *NOISeq* (Tarazona et al., 2015) and a standard t-test.
- Detection of consensus differential expression.
- Detection of differences in isomiR profiles.
- By means of *sRNAde*, all individual *sRNAbench* result files can be summarized at a study level. Certain columns can extracted generating an expression matrix (feature matrix) output file (one column per sample).

sRNAblast:

- Unmapped or unassigned reads can be further analysed by means of *BLAST*. By default, the remote databases from NCBI are used (nr/nt by default).
- Output files are generated at read, species and kingdom levels.
- Often, a read can map to several species with the same quality. Therefore we perform a probabilistic single assignment to obtain the most likely species.

The results can either point towards contamination sources or biological meaningful information like the presence of unexpected viral or bacterial RNA molecules.

miRNAconsTarget:

- MiRNA Target prediction based on *Miranda* (John et al., 2004), *PITA* (Kertesz et al., 2007), *TargetSpy* (Sturm et al., 2010) and simple seed methods for animal

microRNAs. For plants Tapir (Bonnet et al., 2010) and psRobot (Wu et al., 2012) are used.

- Consensus targets at a miRNA/mRNA level (the target site positions need not to coincide between methods) and at a position level (the methods must call same miRNA/mRNA at the same position).

sRNAhelper:

We implemented a helper tool to aid the user in several different analysis steps.

- **sRNAhelper**: parse out sequences from RNAcentral at species or taxonomy level, extract subsets or manipulate fasta files with a given search pattern or remove duplicates from fasta files.
- **Populate**: download annotations from our database installing them directly
- **launcherLibs**: automate the execution for a larger number of samples (using library mode).
- Several tools that assist the user in the analysis and the generation of the local database.
- **Update**: Maintain your *sRNAtoolbox* binaries up to date.

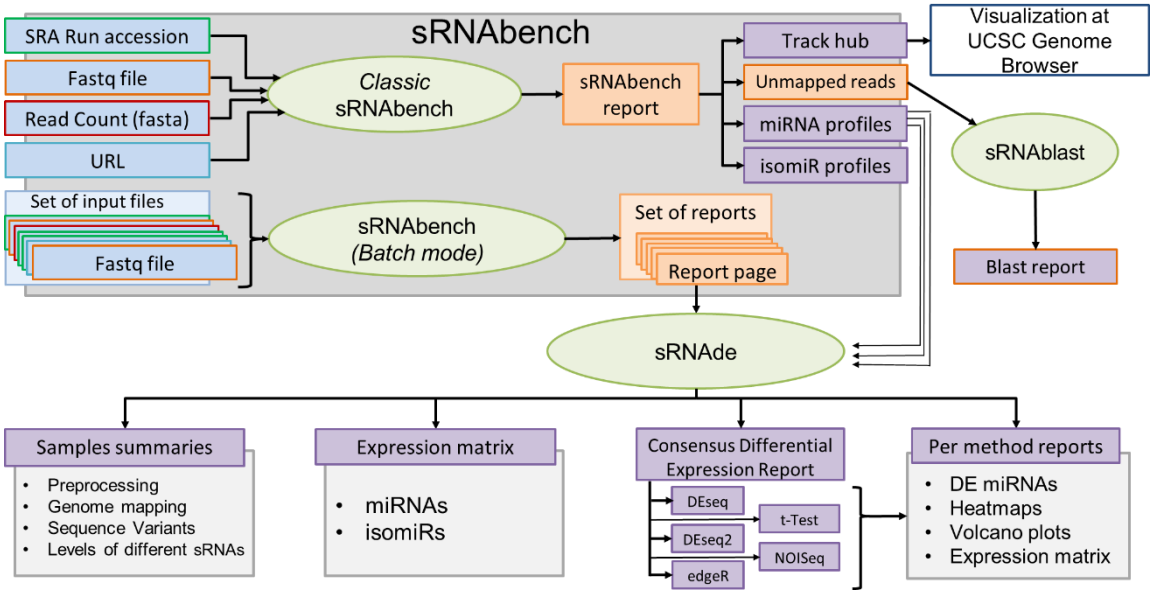


Figure 1: sRNAtoolbox graphical abstract (2019)

2 Getting started

There are 3 ways to use *sRNAtoolbox* (or parts of it): Webserver, Docker or standalone versions. We discourage the usage of standalone as several dependencies exist (Appendix 7.1).

2.1 Webserver

The easiest way is to use the webserver which can be accessed here: ***sRNAtoolbox* webserver**.

2.2 sRNAtoolbox Docker

The *sRNAtoolbox* docker provides the user with a number of preinstalled programs like all *sRNAtoolbox* tools, Vienna package, bowtie, samtools etc. needed for common small RNA data analysis.

First of all **Docker** must be installed. To install **Docker** in Ubuntu and MacOS please do the following:

```
1 – Install docker
sudo apt update
sudo apt install docker.io

2- Start docker as a service
      sudo systemctl start docker
      sudo systemctl enable docker
```

To install *Docker Desktop* in Windows please follow **these instructions**.

sRNAtoolbox docker is hosted in Dockerhub so the first step is to pull the image from it:

```
sudo docker pull ugrbioinfo/sRNAtoolbox:latest
```

After this, your *sRNAtoolbox* docker image is downloaded and now you can launch it:

```
sudo docker run --hostname sRNAtoolbox --name sRNAtoolbox --user srna --
workdir /home/srna -it ugrbioinfo/sRNAtoolbox:latest /bin/bash
```

Alternatively if you want to start the container with **shared folders**:

```
sudo docker run --hostname sRNAtoolbox --name sRNAtoolbox --user srna --
workdir /home/srna --mount
type=bind,source=MACHINE_FOLDER,destination=/shared/shared_folder -it
ugrbioinfo/sRNAtoolbox:latest /bin/bash
```

After the first use you can exit the container by typing “exit” and stop the container with the following:

```
sudo docker stop sRNAtoolbox
```

For further uses:

```
sudo docker start sRNAtoolbox
sudo docker exec -term=SCREEN -it sRNAtoolbox /bin/bash
```

2.3 Standalone versions

For some tools, standalone versions are available. Note that these depend on other programs like bowtie, samtools, blastn etc, and therefore the installation is not as straight forward as the usage of the Docker. It is explained in the Appendix 7.1.

2.4 Populate the database

The *Docker* includes a script called ***populate*** that downloads and installs user defined species. Over 300 species are available in our database (animals, plants, virus and different bacterial, fungi and virus collections). How to manually populate the database is described in Appendix 7.2.

To use it execute it in a terminal:

```
populate ‘pathTosRNAtoolboxDB (default: /opt/sRNAtoolboxDB)
```

populate has no mandatory parameters, by default the database is assumed to be in: */opt/sRNAtoolboxDB*

This script will open a dialog where you can select the species to install in the DB as it is shown in Figure 2.

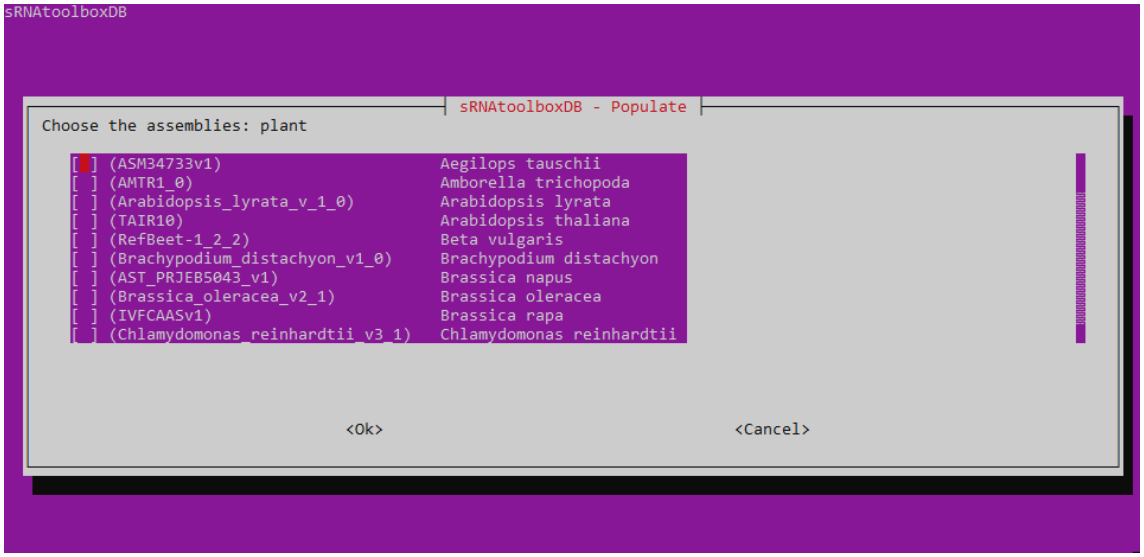


Figure 2: screenshot of populate. The species can be selected with ‘space’ – with enter you pass to next page

IMPORTANT: The species can be selected with ‘space’ and with enter you pass to next page

The result of the populate tool can be seen in the local *sRNAtoolbox*DB database (by default /opt/*sRNAtoolbox*DB) and in the ‘info’ file (by default /home/srna/info). Figure 3 shows a screenshot of this info file. It indicates the names of the reference files that are needed to use them with *sRNAbench*.

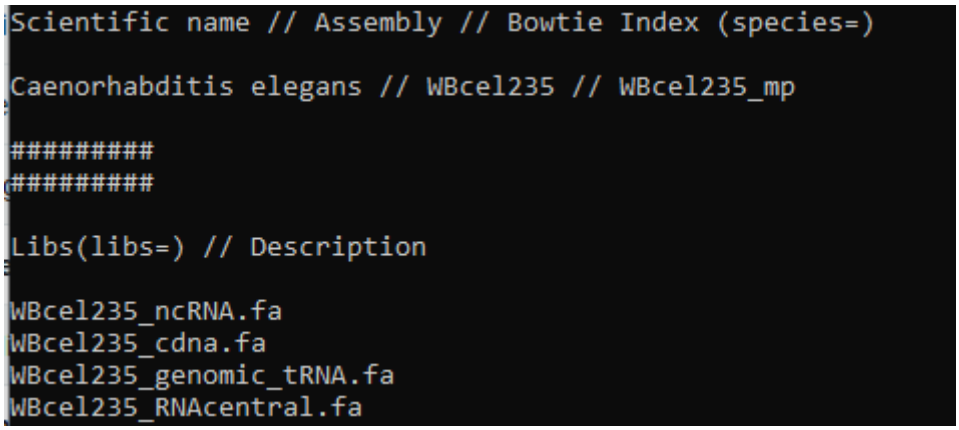


Figure 3: Screenshot of the 'info' file (in the root of the home directory of the user, by default /home/srna/info)

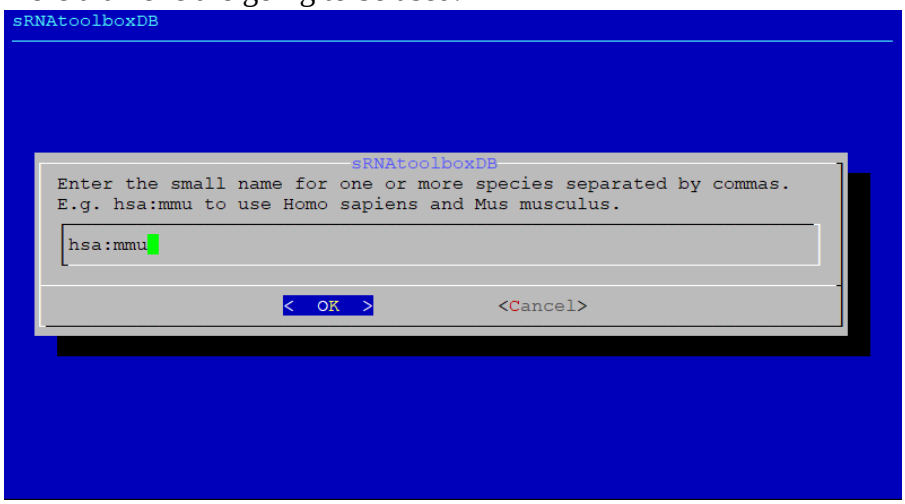
2.5 Launch *sRNAbench* with helper tool launcherLibs

The *Docker* includes a script called ***launcherLibs*** to prepare the commands/launch *sRNAbench* in library mode. This helper tool will generate a batch file that can either be manipulated or launched directly by the user.

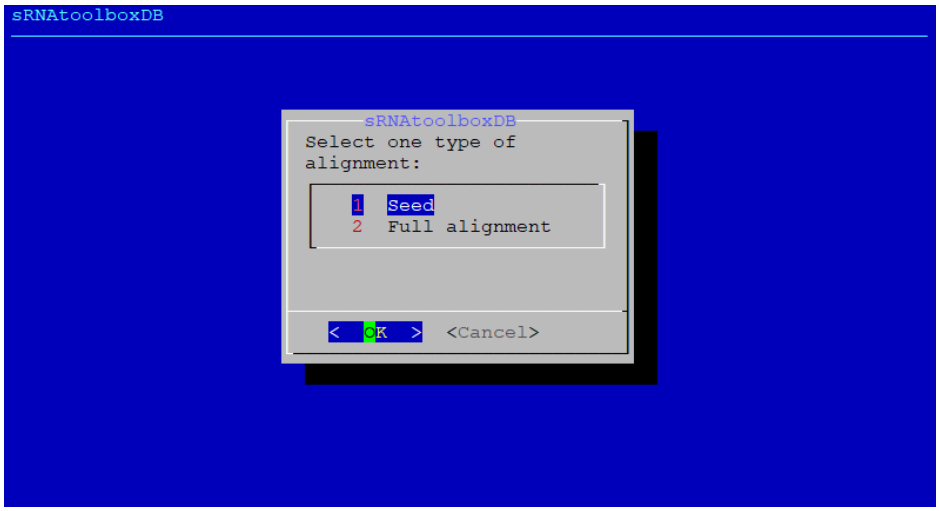
Similar to the ***populate*** tool is a dialog window where you can select some options. There are three possible scenarios depending on the inputFiles, but all of them have a set of options in common:

1- Species Selection

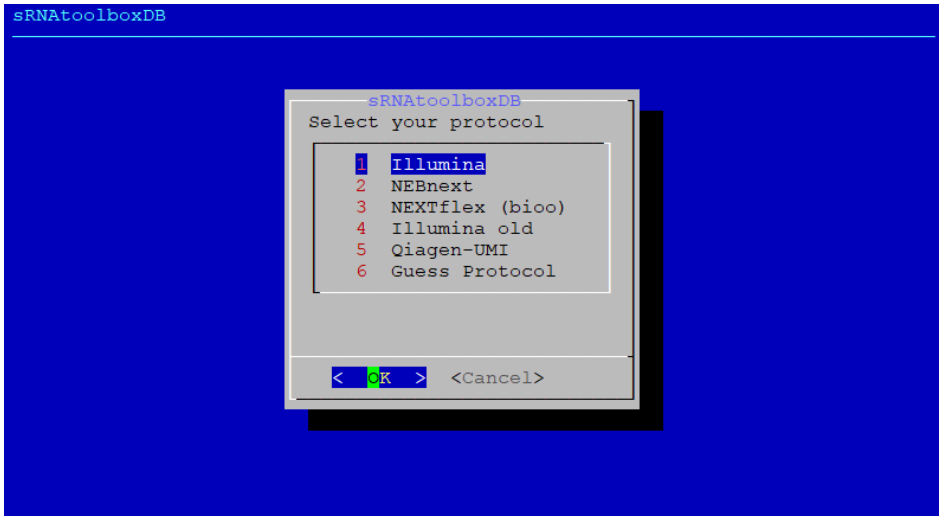
In this dialog you will have to provide small name of the species you want to use (eg. hsa for *Homo sapiens* or mmu for *Mus musculus*) separated by ":" if more than one are going to be used:



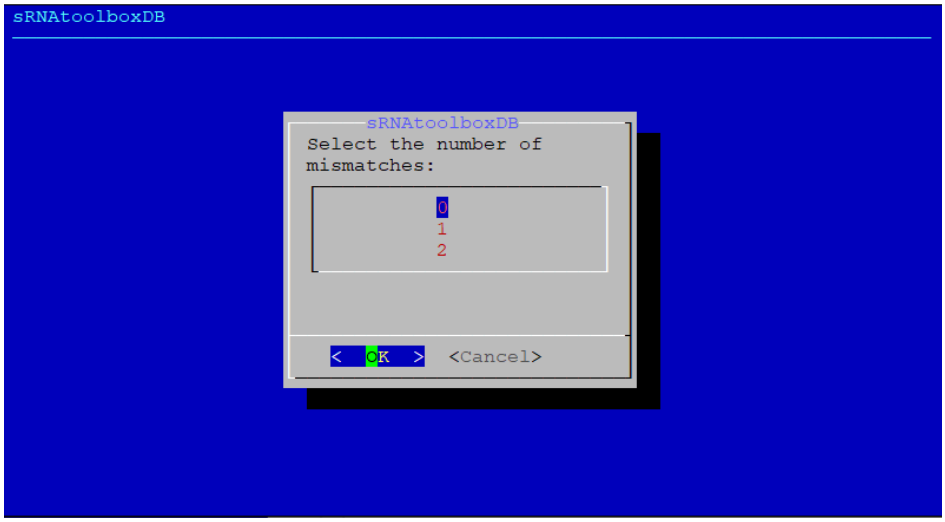
2- Type of alignment: You can choose between "seed" or "full alignment":



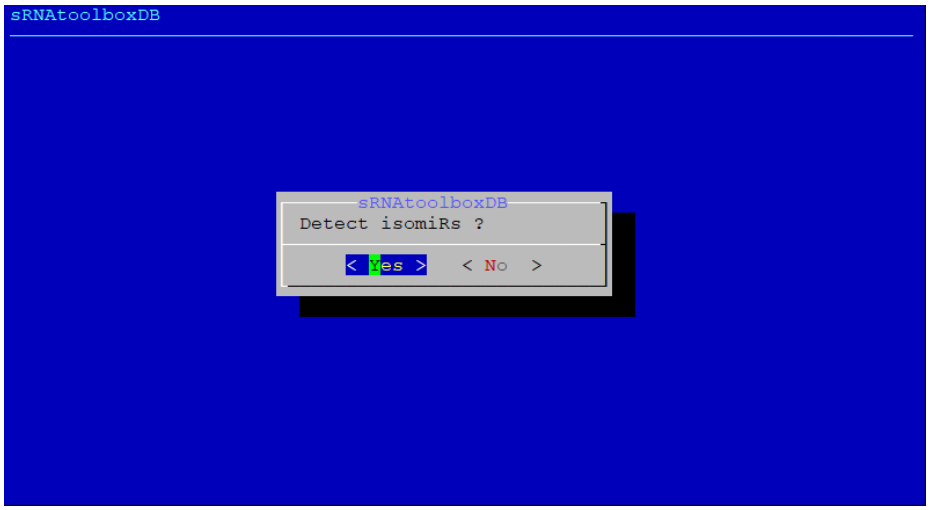
3- Protocol selection:



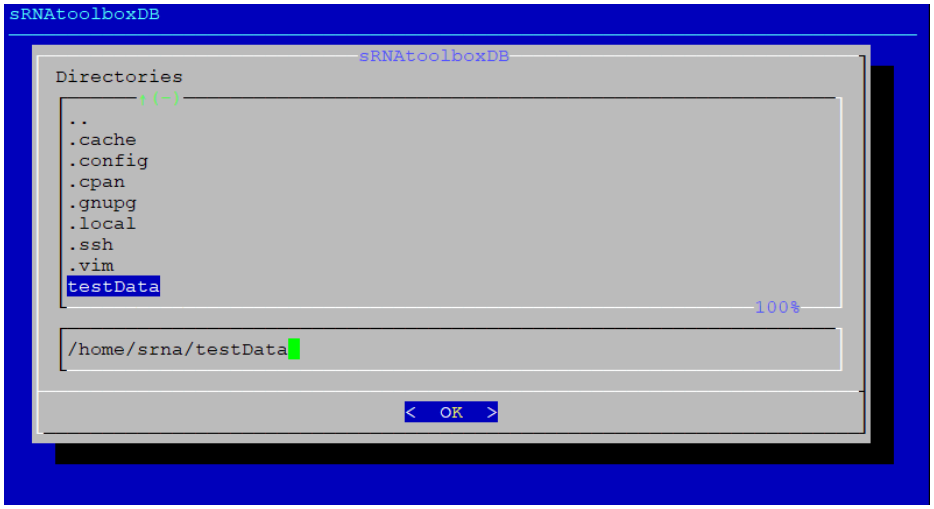
4- Number of mismatches:



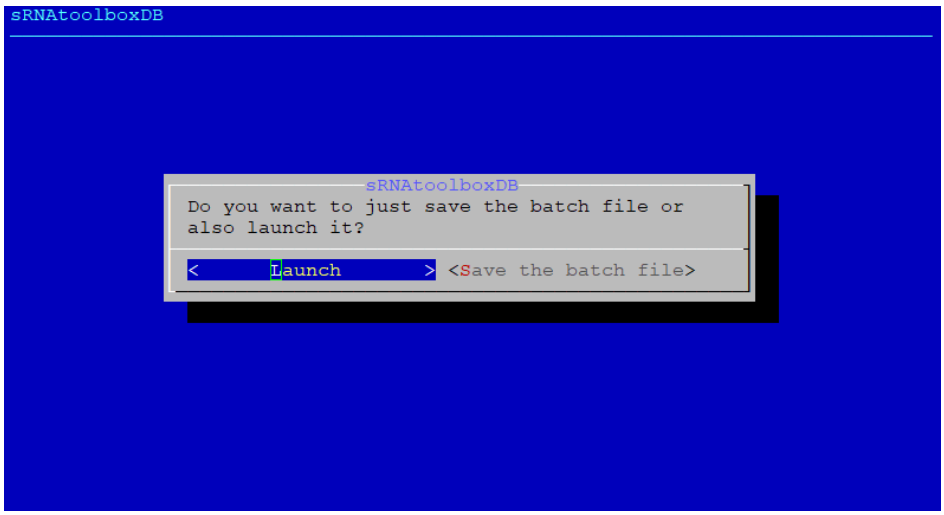
5- Detection of isomiRs:



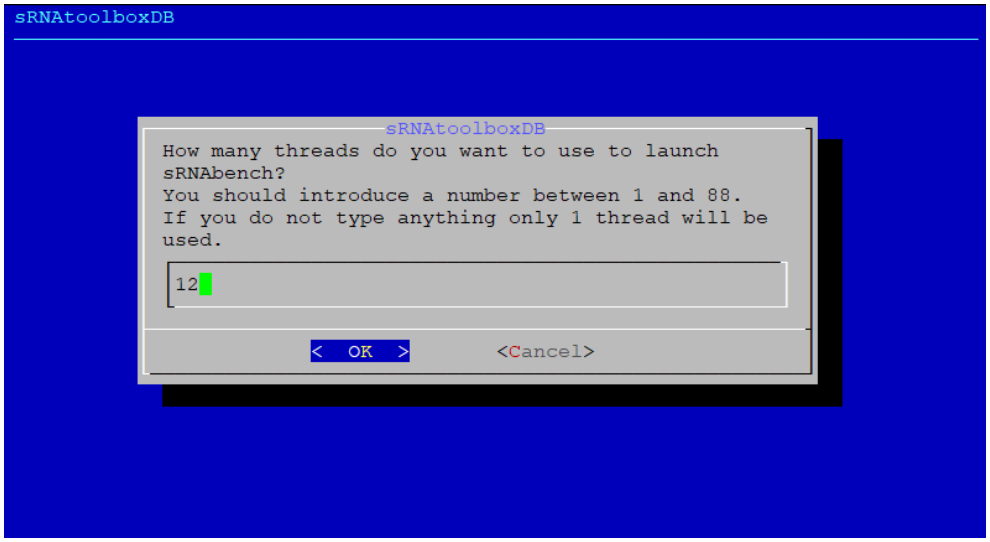
6- Input directory (only if not provided). Please notice that you can select with the spacebar the directory that it's highlighted (or enter into it):



7- Choose between *Launch* or just *Save the batch file*. If you choose Save the batch file it will be saved in the output directory.



8- If you choose *Launch* in the previous step, you can choose the number of threads to use and just launch *sRNAbench*.



As it was pointed out before, there are three possible scenarios that depends on the input and that are the following:

a. A *sampleSheet.tsv* is provided

In this scenario the ***launcherLibs*** script is launched as it follows:

```
launcherLibs --inputPath /home/srna/testData/ --outputPath
/home/srna/testResults --sampleSheetFile /home/srna/testData/sampleSheet.tsv
```

As the *sampleSheet* is provided, you specify in it the samples that are going to be analysed and the group to which they belong. You will find an example of a *sampleSheet.tsv* in */home/srna/testData/sampleSheet.tsv.*

In the *inputPath* parameter you should provide (but not necessarily) the input directory where the inputFiles. Analogously with the *outputPath*, that if it is not indicated you can choose with the utility.

At the end a *sampleSheet_DE.tsv* file will be generated in order to use it with the *sRNAde* tool.

b. All the samples are inside the same folder and a *sampleSheet.tsv* it is not provided

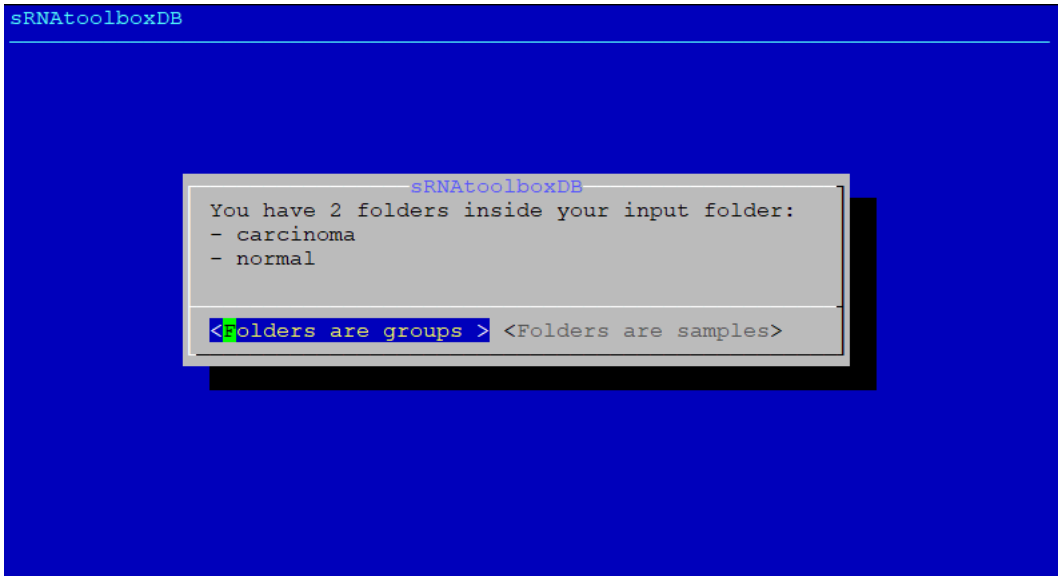
In this scenario ***launcherLibs*** script is launched as it follows:

```
launcherLibs
```

The tool will ask about all the items enumerated before (including input and output folders) and a *sampleSheet_DE.tsv* will be created at the end. You can edit this *sampleSheet_DE* in order to change the groups (that by default are *Group_1* in all samples) and use it with *sRNAde*.

c. The samples are inside different folders and a *sampleSheet.tsv* it is not provided

In this case two options are available, that each folder represents a different group or that each folder represents a sample. After selecting the input directory the utility will ask about this possibility with the following dialog:



You should choose between these two options.

a. Folders are groups

   If each folder represents a group (for further processing with *sRNAde*) when the following dialog is finished a *sampleSheet_DE.tsv* will be created and each sample will have assigned the group of the folder where it is.

b. The folders are different samples

   If each folder represents a sample all files inside a folder will be treated as the same sample and in the *sampleSheet_DE.tsv* all the samples will be assigned to the *Group_1*. You should edit that in order to use it with *sRNAde*.

2.6 Launch *sRNAde* with helper tool LaunchDE

The *Docker* includes a script called ***LaunchDE*** to launch *sRNAde*. To use it execute it in a terminal:

```
LaunchDE sampleSheet_DE.tsv sRNAbench_outputFolder/ sRNAde_outputFolder
```

The script has 3 parameters:

- *sampleSheet_DE.tsv*: A sample sheet as the one that you can obtain with launcherLibs. It should be a 3 column .tsv file.
- *sRNAbench_outputFolder*: The folder where the results of *sRNAbench* are located
- *sRNAde_outputFolder*: The output folder for *sRNAde* analysis.

3 sRNAbench

sRNAbench is a program for processing small-RNA data obtained from next generation sequencing platforms such as *Illumina* or *SOLiD*. Figure 3 shows the *sRNAbench* features and work flow.

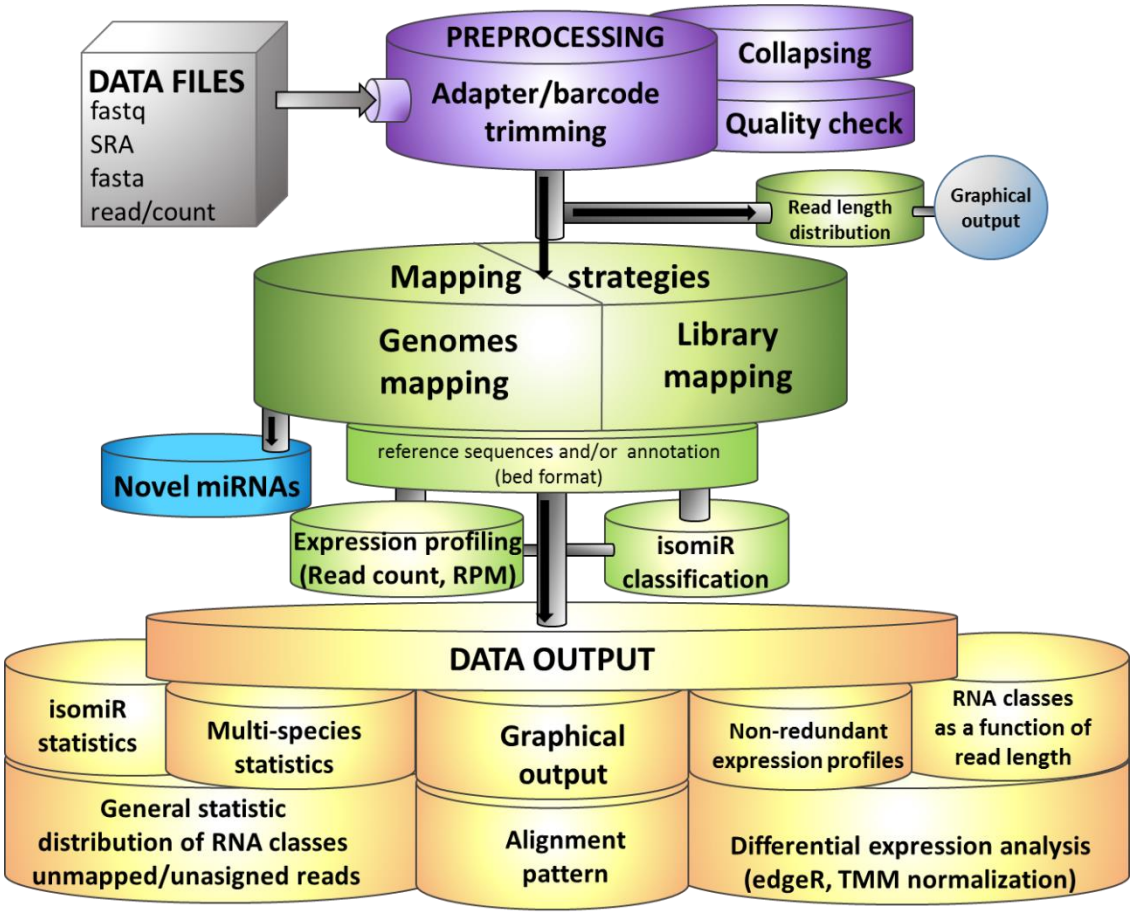


Figure 3: Schematic overview on sRNAbench features. On the bottom at the right, the differential expression analysis is mentioned which is performed by sRNAde: Differential expression.

3.1 Main features

- Two different ways can be used to profile the expression levels of small RNAs depending on whether a good genome sequence/annotation is available: i) mapping all short sequence reads first against the genome, obtaining the expression levels by means of an annotations in *fasta* or *BED* format at a second stage (**genome mode**) and ii) mapping against sequence libraries in fasta/Bowtie index format directly (**library mode** like it was done by *miRanalyzer*).
- An unlimited number of genomes can be used in the analysis at the same time without the need to pool all sequences into a single file/Bowtie index. This feature is especially important when analyzing the interaction between parasites and hosts, symbiosis or virus infected cells.
- Adapter trimming can be performed and *sRNAbench* accepts *fastq*, *fastq.gz*, read count and *fasta* input format.

- Most currently used library protocols are supported including UMIs and random adapters.
- Spike-in sequences can be provided.
- Extensive profiling of all microRNA sequence and length variants (isomiRs). Furthermore, NTAs (non-templated additions) can be detected for all sequence libraries and not only for microRNAs.
- Several summary and graphical summaries are available.
- The prediction of novel microRNAs was improved compared to miRanalyzer in the sense that it is much more specific now. The prediction is based on structural, sequence and biogenesis features.

Important novelties in 2.x:

UMIs and random adapters are supported

Spike-in sequences can be used

3.2 Quick start and working examples

The start-up database contains test data from this publication. The test data is obtained from primary effusion lymphoma cell line BC-1 (human) which additionally contains two viruses, human herpesvirus type 8 (HHV-8) and Epstein-Barr virus (EBV). This data set is aimed to show one of the principal strength of *sRNAbench*, the possibility to analyze multi-species assays.

In this section we will give a brief protocol for a basic analysis of small RNA sequencing data. For a more extensive protocol please see *sRNAtoolboxVM*: Small RNA Analysis in a Virtual Machine. In general, the commands will work directly if the user installed the database into ***/opt/sRNAtoolboxDB*. The full path to the java files should be specified always like this:**

```
java –jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar
```

Or (in Docker):

'*sRNAbench*' is a small, executable shell script that launches the java jar file.

```
sRNAbench
```

Following we will illustrate the use of *sRNAbench* by means of some examples:

3.2.1 Preprocessing

By means of the ***protocol*=<>** parameter (see **protocol=<I,NN,Ia,B,Q,S,guess>**), the use can specify the used library processing protocol or ***protocol=guess*** will make *sRNAbench* to guess the protocol.

An important step in the analysis of high-throughput small RNA sequencing data is the detection of the 3' adapter sequence. Many small RNAs (or RNA fragments) have lengths between 21/22 nt (mature microRNAs) and 33 nt (tRNA halves), and therefore

the adapter or part of it will be sequenced as well at nowadays typical read lengths of 36-100nt. As a consequence, the pre-processing consists normally of two basic steps: i) adapter trimming (detect and remove the adapter sequence) and ii) read collapsing (determine the unique reads and assign a read count to them, i.e. the number of times they have been sequenced).

Like mentioned above, a subsequence from the 5' end of the adapter sequence can be present in the read sequence. Its detection and removal is a highly parametrized process.

Briefly, *sRNAbench*:

- Aligns the first N nucleotides (**adapterMinLength** parameter, see below) from the 5' end of the adapter to the read allowing **adapterMM** mismatches and no gaps.
- By default the adapter is searched in the whole read, but by means of **adapterStart** the position where the search starts can be specified.

If the adapter is detected the read is trimmed at the first adapter position.

The following command will perform the pre-processing generating additionally length distribution files.

```
java -jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar input=SRR343332
output=/opt/sRNAtoolboxDB/out/SRR343332_pre adapterMinLength=6
adapter=TCGTATGCCG removeBarcode=5
```

Explanation of used Parameters

- **input:** if the input name is not an existing file, then *sRNAbench* will try to download from SRA (Short Read Archive). Otherwise, *fastq*, *fasta* and *read/count* format is accepted.
- **adapter:** the adapter sequence that will be trimmed of the 3' end of the reads
- **removeBarcode:** Three different barcodes have been added to the 5' end of the reads in this data set (SRR343332). The barcodes have a fixed length of 5 nt and they must be trimmed before trying to align the reads. This command removes a fixed number of nucleotides from the 5' end of each read.
- **adapterMinLength**: By default, *sRNAbench* forces the detection of at least 10nt of the adapter sequence. However, taking into account that the reads of SRR343332 have 36 nt length, out of which 5 nt correspond to the barcode will have at most 31 nt 'useful' information. Therefore we cannot use the default minimum adapter length as this would imply that we can only profile small RNAs equal or shorter than 31nt -10nt = 21nt. Therefore, we will set the minimum adapter length to 6nt allowing the profiling of small RNAs up to 25 nt. As the adapter length is quite short the allowed max. number of mismatches in adapter detection will be reduced to 0 (adapterMinLength=0)

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.
- **results.txt**: the summary of the run
- **log.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)

Files located in the ***stat*** folder:

- **readLengthFull.txt**: the length distribution of all raw reads after adapter trimming
- **readLengthAnalysis.txt**: the length distribution of all adapter trimmed reads in the analysis (i.e. after quality and length filtering)

3.2.2 microRNA profiling (Library mapping mode)

First, we will align the input data to miRBase libraries for the three species:

```
java -jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar
input=/opt/sRNAtoolboxDB/out/SRR343332_pre/reads_orig.fa
output=/opt/sRNAtoolboxDB/out/SRR343332_miR microRNA=hsa:ebv:kshv
```

Explanation of used Parameters

- **microRNA='short species name'**: In miRBase, the nomenclature makes reference to the species in the first 3-4 letters. For example, hsa-mir-21 refers to microRNA 21 in Homo sapiens while mmu-mir-21a would be the homologous microRNA in Mus musculus. This first part of the microRNA name is used to parse out the reference sequences of the species that should be analysed. **More than one species can be selected separating them by ':'.**
- **input=/opt/*sRNAtoolbox*DB/out/SRR343332_pre/reads_orig.fa**: the adapter trimmed reads from the last section are used as input.

Important output files

- **'mature_sense.grouped':** holds the expression values of the mature sequences. The format is explained here: Expression profiling: *.grouped Files
- **'mature_sense_SA.grouped'**: the single assignment expression file. Multiple mapping of reads is a well-known problem in HTS data analysis. MicroRNAs are frequently members of broader families that contain several genes with highly similar mature sequences. *sRNAbench* addresses this problem in two ways: i) a simple adjusted expression value is calculated dividing the read count of each read by the number of reference sequences or genome loci to which they map and ii) by determining a 'single assignment' expression value, i.e. each read

is only assigned to one sequence or loci, i.e. to the one with the highest expression value. Therefore, for each expression file, like 'mature_sense.grouped', one with 'single assignment' expression values are generated.

- 'hairpin_sense.grouped'; the expression of the precursor sequences including the mature sequences
- 'hairpin_sense_SA.grouped': the expression values of this file are based only on those reads that map to the precursor, **BUT NOT** to the mature sequences. This is because in the single assignment, the reads are first assigned to the mature sequences and then to the precursors. For most microRNAs, the number of reads uniquely mapped to the precursor but not to the mature sequence should be close to cero. Therefore, those precursors with high number of uniquely mapped reads could be incorrect microRNAs (false positive annotations).
- **reads.annotation**: the annotation of all assigned reads.
- **mappingStat.txt and mappingStat_sensePref.txt**: the mapping statistic to the different types of used annotations (here only microRNAs)
- ***hairpin*** **folder**: which holds two types of files: i) '*.align' files that show all reads mapped to the sequence (Figure 2c) and '*.countArray', which gives the percentage of read coverage based on unique reads (UR) and total read count (RC) and the RPM (read per million) expression per base.
- **microRNA_species.txt**: the distribution of microRNAs over the different species. Only written if more than one species for used for profiling.
- assignedReads and unAssignedReads files see here.
- **rnaComposition_readLength_sensePref.txt and rnaComposition_readLength.txt**: the distribution of detected types of RNAs as a function of length.

3.2.3 Genome mapping mode

The same analysis as carried out last section can also be performed by means of the genome mapping mode. Note that in the default database contains only human chromosome 22, and therefore only the microRNAs located on this chromosome can be profiled.

```
java -jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar
input=/opt/sRNAtoolboxDB/out/SRR343332_pre/reads_orig.fa
output=/opt/sRNAtoolboxDB/out/SRR343332_miRgenome microRNA=hsa:ebv:kshv
species=chr22:NC_007605:NC_009333
```

In comparison to the library mapping, there are only minor differences in the number of output files

Explanation of used parameters

- **species='name of the bowtie index'**: this parameter specifies the name of the bowtie index (of the genome assembly) that should be used to map the reads first against a genome. The bowtie indexes are located in the 'index' folder

within the *sRNAtoolbox* database. Several genome assemblies can be used simultaneously separating them by ':'

Important output files

- **genomeDistribution.txt**: the number of reads mapped to the different species. This file is also written if only one assembly is used as it contains also the frequency of redundant and non-redundant reads. By default, bowtie is launched with the –m parameter, i.e. only those reads that map at most N (-m N) times are reported (see
- seed=<int>: the length of the seed (-l parameter in Bowtie). (default: seed=19)
- **bowtieReportType=<String>**: in *#sRNAbench* parameters section). Those that map <= N times are called non-redundant and those that map > N times are called redundant.
- ***genomeDistribution*** **folder**: contains the reads assigned to the different assemblies in fasta format and the corresponding read length distribution.
- genomeMappedReads.fa: the reads mapped to any of the genome assemblies. Important: only non-redundant reads (by default those that are mapped at most 10 times to the assembly) are included (see
- seed=<int>: the length of the seed (-l parameter in Bowtie). (default: seed=19)
- **bowtieReportType=<String>**:).
- **genomeMappedReads.readLen**: the read length distribution of genome mapped reads.

3.2.4 Prediction of novel microRNAs

```
java -jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar
input=/opt/sRNAtoolboxDB/out/SRR343332_pre/reads_orig.fa
output=/opt/sRNAtoolboxDB/out/SRR343332_prediction microRNA=hsa:ebv:kshv
species=chr22:NC_007605:NC_009333 predict=true minReadLength=19
maxReadLength=25
```

Explanation of used parameters

The prediction of novel microRNAs is activated with predict=true. By default the models for animals are used. For plants, kingdom=plant needs to be set. The read lengths should be limited to reasonable microRNA sizes (including 3' length variants). Here with between 19 nt and 25 nt. See Prediction of novel microRNAs for all related parameters.

Important output files

- **novel.txt**: Summary of novel microRNAs
- **novel_mature.fa and novel_hairpin.fa**: mature and pre-microRNA sequences of novel microRNAs
- **folder** ***novel***: contains the alignments to the novel pre-microRNA sequences

3.2.5 Using other libraries

Other libraries can be analyzed with libs='library name'. If the library is given in *fasta* format, *sRNAbench* will first generate a bowtie index of this file. Otherwise, the library will be mapped to the genome in order to obtain the chromosome coordinates of the reference sequences.

Warning: In genome mode, only unspliced reference sequences should be given in *fasta* format. Spliced genes should be given as bowtie indexes (see Section 2.4.3 and Section 7.2).

```
java -jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar
input=/opt/sRNAtoolboxDB/out/SRR343332_pre/reads_orig.fa
output=/opt/sRNAtoolboxDB/out/SRR343332_libs microRNA=hsa:ebv:kshv libs=hg19-
tRNAs.fa plotLibs=true minRCplotLibs=100
```

Explanation of used parameters

- **libs**=<library name>: The name of the file that holds the reference sequences for this library. They can be given either in fasta format or as bowtie index. In genome mode, BED and GFF format is supported as well.
- **plotLibs**=<boolean>: if set to true, then the alignment files will be written out into a 'library name' folder. One file per reference sequence which has more than XXX mappings.
- **minRCplotLibs**=<int>: The alignment files only those sequences with a read count higher than 100 are plot

Note: ***plotLibs*=<*true*>** will also calculate the secondary structure of the reference sequence, BUT only if the sequence length is below a given threshold (**maxLenForSecStruc=<int>)**

Important output files

- **hg19-tRNAs_sense.grouped and hg19-tRNAs_antisense.grouped**: Reads mapped to the sense and antisense strands of the hg19-tRNA.fa library.
- **hg19-tRNAs_sense_SA.grouped and hg19-tRNAs_antisens_SA.grouped:** Single Assignment files of tRNA frequencies.
- ***hg19-tRNA* folder**: contains the alignments to the reference sequences (tRNA sequences in this example)

3.2.6 Detecting isomiRs

```
java -jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar
input=/opt/sRNAtoolboxDB/out/SRR343332_pre/reads_orig.fa
output=/opt/sRNAtoolboxDB/out/SRR343332_libs microRNA=hsa:ebv:kshv
isoMiR=true
```

Explanation of used parameters

IsomiRs can be detected adding isoMiR=true. See section 3.3.7 (Detection of isomiRs).

Important output files

- **mature.iso: isomiR information for each mature microRNA**. Explanation of output format
- **microRNAannotation: isomiR annotation at a read level.** Explanation of output format.
- **isomiR_summary.txt (in stat folder):** The number of reads found for all isomiR classes for each of the mature microRNAs. Explanation of the output format.
- **isomiR_NTA.txt and isomiR_otherVariants.txt (in 'stat' folder)**: isomiR summary of the sample. Explanation of the output format.
- **isomiR_lenNTA.txt (in 'stat' folder):** summary of non-templated additions as a function of 'addition length' (for example 1, 2, 3, etc. A's added). Explanation of the output format.

3.2.7 Visualizing alignments

Just like shown above for microRNAs, the alignment files and frequency counts along the annotation sequence can be generated also for other libraries.

```
java -jar /opt/sRNAtoolboxDB/exec/sRNAbench.jar
input=/opt/sRNAtoolboxDB/out/SRR343332_pre/reads_orig.fa
output=/opt/sRNAtoolboxDB/out/SRR343332_libs microRNA=hsa:ebv:kshv libs=hg19-
tRNAs.fa plotLibs=true
```

Explanation of used parameters

The alignment visualization files can be generated setting **plotLibs**=true. See this for further information.

Important output files

- For each library, and folder with the alignment files will be generated. In our example: ***hg19-tRNAs***

3.3 *sRNAbench* parameters

In general, the parameters have to be given in the following form: parameter=value. For example to specify the mandatory path to the local database: dbPath=/home/usr/...

3.3.1 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***)

- **microRNA=<species list>**: the species from which the microRNA annotations should be used for the analysis. An arbitrary number of species can be used given the short species names separated by ':'. For example microRNA=hsa:ebv will map the input reads simultaneously to human (hsa) and Epstein-Barr virus (ebv) microRNAs.
- **miRdb=<1,2,3,4,5>:** miRdb=1 (miRBase); miRdb=2 (MirGeneDB); miRdb=3 (PMiren); miRdb=4 (miRCarta); miRdb=5 (miRBase high confidence)
- **libs=<String>**: the name of the library file. Typically, those files would hold other types of small RNAs like tRNA, snoRNA, snRNA, piRNA, rRNA, yRNA, vaultRNA, etc. If only a name is given, than the program will search for the file in the default *sRNAbench* database folder ('libs'). If a full path is given the program will use this file which needs not to be within the *sRNAbench* database. The files can be given in *fasta* and *BED* format, or directly the Bowtie indexes (the basename of the index). **libs=<String> can be given several times on the command line!**
- **protocol=<I,NN,Ia,B,Q,S,guess>:** The protocol used for the library preparation.: protocol=I (Illumina); protocol=NN (NEBnext);protocol=Ia (old Illumina adapter TCGT...), protocol=B (NextFlex), protocol=Q (QIAseq/Qiagen with 12nt UMIs), protocol=S (SMARTer), protocol=guess (guess the protocol). Note that *sRNAbench* can handle adapter trimming, random adapters and UMIs. If your library preparation is not available, either try with guess, or you can custumize with the adapter= remove3pBases= removeBarcode= and umi= parameters (see 3.3.2)
- **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!**
- **species=<genome assembly list>**: the genome sequences that will be used. If this parameter is set, then the 'genome mode' will be used, 'library mode' otherwise. An arbitrary number of different genome sequences (bowtie indexes of these sequences) can be used separated by ':'. For example, hg19_5:NC_007605 will map the input reads simultaneously to hg19_5 (in this case, hg19_5 is the bowtie index basename of the human genome version hg19/NCBI37 path 5 genome sequence) and the Epstein-Barr virus genome sequence. Note that, the Bowtie indexes for the genome sequences must be located within the 'index' folder in the *sRNAbench* database having the basenames that are given on the command line.
- **solid=<Boolean>**: if set to true, SOLiD input data is expected. (default: solid=<false>)
- **tRNA=<tRNA library>:** A tRNA specific analysis will be carried out including the detection and classification of tRNA fragments.
- **homolog=<species list>:** A string of short species names is accepted in the same format as explained above (microRNA=). For example, homolog=mmu:rno would map the reads (after the profiling of known microRNAs) to the hairpin sequences of mouse and rat in order to detect putative novel microRNAs based

on homology. homolog=all will use all species except those given with microRNA=.

- **mature=<String>**: the name of the library that holds the mature microRNAs (for example mature.fa from miRBase) (default: mature=mature.fa)
- **hairpin=<String>**: the name of the microRNA precursor sequences (for example hairpin.fa from miRBase) (default: hairpin=hairpin.fa)
- **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 *sRNAbench*. 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=#)

3.3.2 Preprocessing: Adapter trimming

- **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>)
- **guessAdapter=<boolean>:** the program tries to guess the adapter. **Important**: This parameter overrules the adapter= parameter! Briefly, *sRNAbench* will align the first 250000 reads to the genome using the bowtie seed functionality (the adapters will not count for the mismatches). Out of all aligned reads, the adapter sequence is defined as the most frequent 10-mer starting at the first mismatch. (default=<false>)
- **recursiveAdapterTrimming=<boolean>:** The adapter is recursively detected and trimmed. If at least ***adapterMinLength*** bases of the adapter cannot be found within the read sequence, then the adapter is recursively detected only at the 3' end of the read. 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>)

3.3.3 Preprocessing: Barcodes and random adapters

- **remove3pBases=<int>**: removes <int> nucleotides after adapter trimming from the 3' part of the read. This might be useful for many 'random adapter' like protocols that aim to avoid ligase bias. (Default: remove3pBases=0)
- **removeBarcode=<int>:** eliminates the first <int> bases from the 5' end of the read (default: removeBarcode=0)

3.3.4 UMIs (Unique Molecular Identifier)

- umi=[umi code]: This parameters allows to take into account different UMI (Unique Molecular Identifier) designs.
    - ***Fragment - Adapter - UMI design (like Qiagen):* umi=3pA<INT>** (example for Qiagen standard would be umi=3pA12)
    - ***Fragment-UMI-Adapter design:* umi=3p<INT>**

Differences between umi=3pA<INT> and umi=3p<INT>.

Preprocessing steps for umi=3pA<INT>:

- Detect the adapter given with the parameter ***adapter=*** using the adapter trimming parameters (see 3.3.2)
- Eliminate the whole adapter sequnce given by ***adapter=***
- extract the UMI sequence (length <INT>) that starts directly 3' after the adapter sequence generating a new read that consist only of FRAGMENT-UMI
- Group the reads generating unique read sequences & Read Count
- Eliminate the UMI from the reads to generate the input reads (each FRAGMENT-UMI) sequence is only represented once.

Preprocessing steps for umi=3p<INT>:

- Trimm off the adapter from the reads adapter trimming parameters (see 3.3.2). The resulting read is FRAGMENT-UMI
- Group the reads generating unique read sequences & Read Count
- Eliminate the UMI (the length is given with <INT>) from the reads to generate the input reads (each FRAGMENT-UMI) sequence is only represented once.

3.3.5 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)

3.3.6 Preprocessing: Quality control

qualityType=[min,mean]: Activates the filtering by quality. At most 'maxQfailure' nucleotides of a read can have Phred Scores below a minimum. (minQ=20 by default). By default this filter is not applied)

maxQfailure=<int>: the number of nucleotides that can have Phred Scores below the threshold (minQ=20).

minQ=<int>: the minimum PhredScore (default 20).

phred=<int>: the Phred Score codification (by default phred=33).

3.3.7 Mapping parameters

The following parameters are passed to the Bowtie aligner. For more information, please see the Bowtie manual page

- **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)
- **bowtieReportType=<String>**: This parameter regulates the bowtie reporting behaviour and **must be given within single quotation marks on the command line** . The default combination 'bowtieReportType=–a –m' indicates that only reads that map at most **m** times to the genome are reported. Note that all type of parameters can be passed to bowtie, for example '-k' would be another possibility (see the bowtie manual for more details)
- **bowtieReportCount=<int>**: this parameter specifies the corresponding number to the 'bowtieReportType' parameter. For example:
  - 'bowtieReportType=-a -m' && bowtieReportCount=20 would pass to bowtie the following parameters '-a -m 20'
  - 'bowtieReportType=-a -k' bowtieReportCount=5 would pass '-a -k 5'
  - **If all mappings should be reported, 'bowtieReportType=-a ' 'bowtieReportCount= ' needs to be specified on the command line (with a space after the =)**
- **chunkmbs=<int>**: "*The number of megabytes of memory a given thread is given to store path descriptors in –best mode*" (from Bowtie manual): Default: chumkmbs=<256>

- **bowtieAdd=<String>:** A string that will be added to the bowtie command line. In this way, any other parameters can be passed to Bowtie. Default: ‘bowtieAdd=--best --strata’
- **microRNAmappingOrientation=<String>**: This is a bowtie parameter: --nofw (does not map to the forward strand) or --norc (does not map to the reverse complementary strand). **It is only applied in the microRNA profiling**. This parameter is applied to both, library and genome mode. By default this parameter is not given on the command line and the mapping is performed against both strands.
- **libsmappingOrientation=<String>**: This is a bowtie parameter: --nofw (does not map to the forward strand) or --norc (does not map to the reverse complementary strand). **It is only applied to the libraries given by libs=** . This parameter is applied to both, library and genome mode. By default this parameter is not given on the command line and the mapping is performed against both strands
- **tRNAmappingOrientation=<String>:** This is a bowtie parameter: --nofw (does not map to the forward strand) or --norc (does not map to the reverse complementary strand). **It is only applied to the tRNA library given by tRNA=** . This parameter is applied to both, library and genome mode. By default this parameter is not given on the command line and the mapping is performed against both strands

3.3.8 Profiling parameters

There are several parameters that influence the profiling of known elements.

- **winUpMir=<int>:** the upstream flanking for the detection of microRNAs (default=<3>)
- **winDownMir=<int>:** the downstream flanking for the detection of microRNAs. For example, if the (default=<5>)

Explanation of winUpMiR and winDownMiR: For example, hsa-miR122-3p maps to the '+' strand of chr18 with start position 56118356 and end position 56118377. This region is than extended adding the downstream and upstream flanking: From (56118356-winUpMiR) to (56118377+winDownMiR). All reads that lie within this region are assigned to the reference element (hsa-miR122-3p in this case).

- **winUpLibs=<int>:** Same as winUpMiR but applied to libs reference sequences. Default winUpLibs=<0>
- **winDownLibs=<int>:** Same as winDownMiR but applied to libs reference sequences. Default winDownLibs=<0>
- **hierarchical=<boolean>:** Apply a hierarchical classification. If hierarchical=<true>, then the reads mapped to libs=<sequence library> are removed from the input after each library so they cannot map again (like in miRanalyzer). If hierarchical=<false>, than reads can map to different libraries. Default: hierarchical=<true>
- **base=<0,1>:** for bed format input. base=1 means that the coordinates are 1-based, base=0 means that the coordinates are 0-based (default: base=<0>)

- **matureMM=<int>:** Number of allowed mismatches between the genome and the known mature microRNA (for the detection of the genome coordinates). Default: matureMM=<0>
- **hairpinMM=<int>:** Number of allowed mismatches between the genome and the known pre-microRNA (for the detection of the genome coordinates) Default: hairpinMM=<0>

3.3.9 Detection of isomiRs

- **isoMiR=<boolean>**: Classifies and quantifies the isomiR distribution at the microRNA and sample level. Default: isoMiR=<false>
- **fullIsoStat=<boolean>:** Writes out a full isoMiR stat (as a function of the different species and 3p and 5p). Default: fullIsoStat=<false>
- **isoLibs=<boolean>**: Detect non-templated additions for the reference sequences given by the libs= paramter. Default: isoLibs=<false>
- **isomiRseed=<int>:** This parameter marks the number of 5' nucleotides which are not used to detect NTAs (non-templated additions). Default: isomiRseed=<18>
- **minRCiso=<integer>:** minimum RC for isomiR profiling (if RC is smaller, this RNA is not considered). Default: minRCiso=10
- **nonRedundantisoMiRclass=<true,false>:** true --> Perform a non-redundant isomiR classification (each read belongs to only one isomiR class); false --> each read can belong to different classes (like length variant and sequence variant). Default: nonRedundantisoMiRclass=true
- **minVarFreq=[0,1]**: The frequency of a sequence variant is calculated as the number of reads with the sequence variant divided by the total number of reads covering the position. The parameter sets the minimum frequency (number between 0 and 1) needed to report the sequence variant in the output. Default: minVarFreq=0.1.

3.3.10 Output options

- **graphics=<boolean>**: if true, *sRNAbench* generates graphic files using R/ggplot2. Default: graphics=<false>
- **plotMiR=<boolean>**: plot out the microRNA alignments to the hairpin folder. Default: plotMiR=<false>
- **plotLibs=<boolean>**: plot out the alignments for the libraries (those given with libs=). Default: plotLibs=<false>
- **minRCplotLibs=<integer>**: The minimum read-count in order to write out the libs alignment file. Default: minRCplotLibs=<200>
- **minRCplotMiR=<integer>**: The minimum read-count in order to write out the microRNA alignment file. Default: minRCplotMiR=<20>
- **maxLenForSecStruc=<int>**: The secondary structure is calculated if the length of the reference sequence is below this threshold. Only applies if

plotLibs=<true> and the read-count of the RNA >= minRCplotLibs. **Default: maxLenForSecStruc=<200>**

- **tRNA=<String>**: If a library from the genomic tRNA database is used, this tag will summarize the tRNA mappings by codons. The value must be the same library name set with the libs= tag.

3.3.11 Produce BedGraph output

- **bedGraph=<boolean>**: writes out a bedGraph file with the genome mappings. (only in genome mode). Default: bedGraph=<false>
- **bedGraphMode=<String>: Will generate several files:** a bed file with the regions that have continuous read coverage; a bedGraph file with both strands, a bedGraph file for the forward strand, a bedGraph file with the reverse mappings and a file with the chromosome sequence length (to ease the conversion of bedGraph to bigWig). It can take to parameters: FA (the read count is fully assigned to each position) or MA (multiple assignment adjusted), i.e. if a read has read count 10 and maps to two positions, to each are assigned 10/2 = 5. Default: bedGraphMode=FA
- **bedGraphIntervals=<String>:** This parameter allows to define several length intervals. Each lenght interval will produce one bed graph file. The string must have this format: Inteval1_start-Interval1_end:Interval2_start-Interval2_end . For example: bedGraphIntervals=19-23:24-24:28-33 would generate 3 different bed graph files, i) for reads with lengths between 19 nt and 23 nt, ii) for reads with read length 24 nt, iii) for read with lengths between 28 and 33nt. Default: parameter not set

3.3.12 Prediction of novel microRNAs

- **predict=<boolean>**: predict=true The prediction is turned on. Default: predict=false
- **kingdom=<animal,plant>:** the input data is from animal or plant in order to use the appropriate prediction parameters. Default: kingdom=<animal>
- **maxDistNovel=<int>:** The maximal distance between the end of the putative 5p-arm microRNA and the start of the putative 3p arm microRNA. Default: not set (the default values for animal (60) and plant (180) are used).
- **novelName=<String>:** The short name used for the novel microRNAs. For example, hsa (human), mmu (mouse), rno (rat), etc. Default: novelName=<new>
- **novelHomolog=<String>:** the species that should be used to assign a name to a novel microRNA that do have a homologous in the microRNA database (determined by the seed sequence). For example: novelHomolog=has:ptr:mmu:rno By default, all animal or plant species from miRBase are used depending on whether kingdom is ‘plant’ or ‘animal’.

3.3.13 Make Genome Distribution statistics

- **writeGenomeDist=<boolean>**: writes out a mapping statistic as a function of chromosome.

- **splitToSpecies=<true,false>:** true --> i) write out the reads that map to a given index (genome assembly) in fasta format, ii) generate the read length distribution. Default: splitToSpecies=true
- **chromosomeLevel=<true,false>:** true --> make the mapping statistics at a chromosome level (and not at a genome level). IMPORTANT: in order to generate the genome level statistics, the sequence ids of the chromosome sequences must be 'manipulated' like: >chr1:hsa (chromosome 1 of homo sapiens- hsa). This makes *sRNAbench* to use the 'hsa' tag for the statistics. If this tag does not exist, chromosome level is used. Default: chromosomeLevel=false
- **mainSpecies=<String>:** If there is a genome assembly to which multiple mapping reads (those that map with the same quality to more than one assembly) should be assigned preferentially. Default: not used
- **genomeDistunique=<true,false>:** Only reads that map uniquely to one species will be considered. Multiple mapping reads will appear in the statistics with 'mixed'. Default: genomeDistunique=false
- **chrMappingByLength=<true,false>:** Make chromosome statistics as a function of read length. Default: chrMappingByLength=false
- **chromosomes=<String>:** A string that specifies the chromosomes that should be analysed. The chromosomes are separted by ':'. Default: not used (all chromosomes in index)

3.3.14 Program names

- **RNAfold=<String>:** The name of the RNAfold program. For example, if both Vienna 2.0 and Vienna 1.8.5 or before are installed on the computer. Default: RNAfold=<RNAfold>

3.4 *sRNAbench* feature and implementation

3.4.1 Analysis steps

Very briefly, *sRNAbench* can be used in two different modes: Genome mode and Library mode (see below). Both modes share a common pre-processing step which consists of i) adapter trimming, ii) quality control, iii) collapsing of all identical reads into one unique entry assigning a read count (the number of times a given read was obtained in the experiment). Note that for both ways, the analysis can be carried out in a hierarchical (by default) or non-hierarchical or redundant way. Hierarchical means that all reads that map to a given library are removed from the analysis and can therefore not map again. In this mode, each read can map only to one annotation group. Both modes share also the mapping order: 1) MicroRNAs (microRNA=), 2) putative homologous (homolog=), 3) other libraries (libs=). Note that an unlimited number of libs= can be used. They will be used in the order as they appear on the command line.

3.4.2 Genome mode

If a genome sequence is given at the command line (species=) than all reads are mapped first to the genome. In a second step, the genome coordinates of the reference small RNA annotations (microRNAs and those given by libs=) are determined. For *fasta* annotations, those are mapped to the genome and the chromosomal coordinates are

retrieved in BED format. If the annotation input was given in BED format, the corresponding coordinates are adopted directly.

3.4.3 Library mode

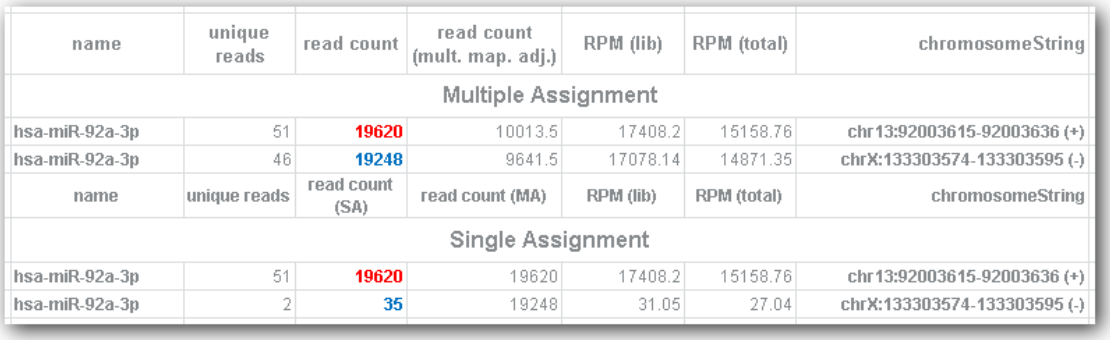
If no genome sequence is available, *sRNAbench* proceeds nearly identical to miRanalyzer. Instead of mapping to the genome sequence, the reads are successively mapped first to a microRNA annotation and after this to the libraries given as libs=.

3.4.4 Ambiguous mapping treatment

A general problem in all high-throughput sequencing experiments is how to deal with ambiguously mapping reads. Ambiguous mapping can arise if i) several alternative transcripts of one gene are given in the reference library or ii) the read maps with the same quality to different genome loci. The first point is of particular importance in mRNA-seq experiments in order to infer the correct expression values of the different isoforms of a gene. For small RNA sequencing, the problem of different transcripts from the same locus virtually does not exist, however several microRNAs have more than one gene in the genome. In order to address this problem, *sRNAbench* generates two different output files: i) a multiple assignment file – each read counts for all loci to which it maps and ii) a single assignment file – each read is assigned only to one locus. The single assignment expression files are generated starting from the multiple assignments (for example mature_sense.grouped):

- The multiple assignment file is ordered by the read count in a descending way.
- The most frequent RNA maintains its expression value and all reads that map to it are removed.
- From the second most frequent RNA to the last, in each step the remaining reads that have been assigned previously to this RNA are summed and removed afterwards. In this way, each read is assigned only once – out of the RNAs to which it maps with the same quality, it is assigned to the most frequent one (based on the read count).

Figure 3 illustrates the result of this procedure. The mature microRNA hsa-miR-92a-3p can be obtained from two genes, located on chromosomes 13 and X. In this example, the read counts are 19620 and 19248 respectively, while the corresponding single assignment read counts are 19620 and 35. This means that all ambiguously mapping reads are assigned to the locus on chromosome 13 leaving a single assignment read count of 35 for the mature microRNA located on the X chromosome. Note that these 35 reads do map to this locus in an unambiguous way, i.e. they map with higher quality to the X chromosome than to chromosome 13. In this way, the number of uniquely mapped reads can be calculated. The locus on the X chromosome had a total read count of 19248, out of which 35 mapped exclusively to this locus. Therefore we can infer that 19248-35=19213 reads mapped to both loci. This allows us finally to conclude that 19620-19213 = 407 reads map exclusively to the locus on chromosome 13. Note that these numbers could mean different things: i) the microRNA is transcribed from chromosome 13 and the 35 exclusively mapping reads from the X chromosome are obtained due to sequencing errors or ii) the microRNA is transcribed from both loci, but at much higher levels from chromosome 13. In either case, the single assignment file might serve in some cases as starting point for further investigation. In this concrete example it shows that the locus on chromosome 13 is very likely the important one.



name	unique reads	read count	read count (mult. map. adj.)	RPM (lib)	RPM (total)	chromosomeString
Multiple Assignment						
hsa-miR-92a-3p	51	19620	10013.5	17408.2	15158.76	chr13:92003615-92003636 (+)
hsa-miR-92a-3p	46	19248	9641.5	17078.14	14871.35	chrX:133303574-133303595 (-)
name	unique reads	read count (SA)	read count (MA)	RPM (lib)	RPM (total)	chromosomeString
Single Assignment						
hsa-miR-92a-3p	51	19620	19620	17408.2	15158.76	chr13:92003615-92003636 (+)
hsa-miR-92a-3p	2	35	19248	31.05	27.04	chrX:133303574-133303595 (-)

Figure 7: Comparison between multiple and single read assignment.

The generation of the non-redundant files is performed in a strict order. **First, the 'sense' files have preference over the 'antisense' files**. This implies that a read that maps to both, the sense and the antisense strand of a given library will be always assigned to the sense strand. Therefore, the single assignment antisense expression files count only 'true' antisense reads.

In the same line, the single assignment files of mature microRNAs have preference over the pre-microRNA files ('hairpin'). This implies that the single assignment pre-microRNA expression files only contain those reads that mapped to the precursor sequence but not to the corresponding mature microRNAs. Those reads can correspond to unannotated mature sequences (mostly for those microRNAs that only have one arm annotated), loop sequences, degradation products or atypical cleavage products.

The methods used to generate single assignment files depend on the order in the multiple assignment file. This order might change between replicates and conditions. Therefore those files should not been used for differential expression analysis.

3.4.5 IsomiR/isoRNA detection and classification

Most mature microRNAs do not only exist in their canonical form in the cell, but a high number of different sequence variants can be reproducibly detected. 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).

Non-templated additions frequently do not match the genome or the pre-microRNA sequence and would therefore cause mismatches in the alignment. In order to not discard those cases, the Bowtie seed alignment option is used. This option scores only the first L nucleotides (L=20 by default) and therefore does not take into account the mismatches at the 3' end of the read caused by the post-transcriptionally added nucleotides.

In order to detect isomiRs, *sRNAbench*, i) maps the reads to the genome or pre-microRNA sequences using the Bowtie seed option, ii) determines the coordinates of the mature microRNAs , iii) clusters all reads that map within a window of the canonical mature microRNA sequence (see 'microRNA expression profiling'), iv) applies a hierarchical classification schema which is described below.

Frequently, a single read can have more than one modification compared to the canonical microRNA sequence. For example, it can be both, 5' trimmed and adenylated.

This fact opens two possibilities: 1) work with a redundant classification (allowing that a read can belong to more than one isomiR class) or 2) apply a hierarchical classification schema. *sRNAbench* works exclusively with a non-redundant classification schema which is summarized in Figure 4.

Briefly, all reads assigned to a given mature microRNA are checked if they belong to one of the following classes:

1. The read is identical to the canonical sequence (usually the miRBase entry).
2. The read has non-templated additions (added A, T(U), C or G), i.e. nucleotides at the 3' end that do not match to the reference (template). By default, *sRNAbench* starts at position 18 detecting the longest run of A's, G's, C's or T's that do not match to the template.
3. The read starts and ends at the same position as the canonical sequence in the pre-microRNA, but shows sequence variation (most likely due to sequencing errors, but RNA editing events and SNPs might exist as well).
4. The read starts or ends at the same position as the canonical version. For this case we can distinguish 4 groups:
    a) 3' trimmed read: the read starts at the same position as the canonical sequence (same 5' end) but it is shorter than the canonical sequence.
    b) 3' extended read: the read starts at the same position as the canonical sequence (same 5' end) but it is longer than the canonical sequence.
    c) 5' trimmed read: the read ends at the same position as the canonical sequence (same 3' end) but it is shorter than the canonical sequence.
    d) 5' extended read: the read ends at the same position as the canonical sequence (same 3' end) but it is longer than the canonical sequence.
5. The read does not coincide neither in 5' nor in 3' with the canonical sequence (multiple length variant).

Note that this classification gives preference to non-templated additions compared to other variants. This is because some NTAs might be biologically meaningful, i.e. at least some microRNAs are stabilized by monoadenylation.

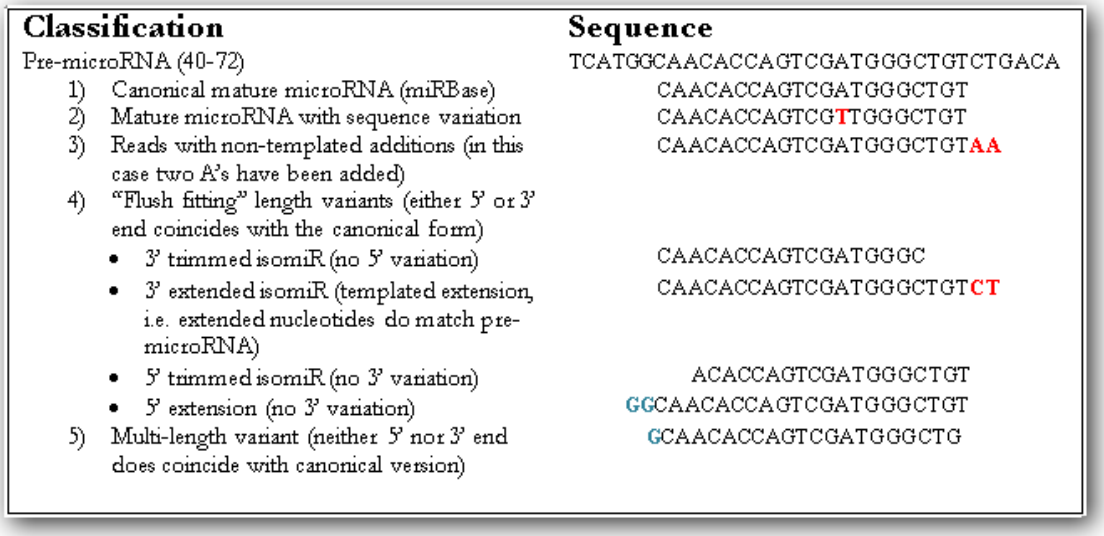


Figure 8: The hierarchical isomiR classification applied by sRNAbench.

sRNAbench does not only detect the canonical (miRBase) microRNA sequence but also all isomiRs (sequence variants) Frequently, a single sequencing read can show different post-transcriptional modifications. For example, it can be both, 3' trimmed and adenylated. In order to keep the classification schema simple, we applied a hierarchical classification: For each read we test in this order if it belongs to one of the following five classes:

1. The read is identical with the canonical sequence (miRBase entry); (label: exact)
2. The read starts and ends at the same position as the canonical sequence in the pre-microRNA, but shows sequence variation (most likely due to sequencing errors, but RNA editing events and SNP might exist as well).
3. The read has non-templated additions (added A, T(U), C or G), i.e. nucleotides at the 3' end that do not match with the reference (template)
4. The read starts **OR** ends at the same position as the canonical version. For this case we can distinguish 4 groups:
    a) 3' trimmed read: the read starts at the same position as the canonical sequence (same 5' end) but it is shorter than the canonical sequence
    b) 3' extended read: the read starts at the same position as the canonical sequence (same 5' end) but it is longer than the canonical sequence
    c) 5' trimmed read: the read ends at the same position as the canonical sequence (same 3' end) but it is shorter than the canonical sequence
    d) 5' extended read: the read ends at the same position as the canonical sequence (same 3' end) but it is longer than the canonical sequence
5. The read does not coincide neither in 5' nor in 3' with the canonical sequence (multiple length variant)

3.4.6 Prediction of novel microRNAs

The prediction of novel microRNAs is based on a modified method described before in the *sRNAbench* paper and the detection of novel plant miRNAs (Barley microRNAs). **Note that all thresholds that are mentioned below are described and motivated in the *sRNAbench* paper, although the concrete number could have varied slightly.** The prediction can be divided in four steps: Clustering, candidate generation, candidate evaluation and post-processing of accepted candidates.

Cluster Reads:

- The reads are mapped first to the genome sequence
- Reads that map to nearly identical positions in the genome are clustered into 'read clusters' in the following way:
    1. the reads are sorted by read count (read frequency)
    2. the most frequent read is assigned to the first cluster (the coordinates of the read cluster are determined by the coordinates of the most frequent read)
    3. for all other reads
        - check if the read lies within a window defined by ClusterStart – 3 nt and ClusterEnd + 5 nt on the same strand (flankings were added in order to assign all isomiRs to the same read cluster)
        - if the read belongs to an existing cluster, the associated read information (sequence and the read count) is added to the cluster
        - if the read does not belong to an existing cluster, a new cluster gets opened.

Generate Candidates:

After clustering all reads, read cluster pairs with distances of less than specified either by the parameter 'maxDistNovel' or 180 nt for plants or 60 nt for animals are built. Furthermore, the read clusters must not overlap having at least a distance of 3nt. The hairpin sequence is defined by the following coordinates:

- Start: the start coordinate of the 5' read cluster - 11 bp
- End: end coordinate of the 3' read cluster + 11 bp.

For most *bona fide* miRNAs there should be two read clusters corresponding to the two arms processed from the pre-microRNA sequence but virtually no other reads overlapping the read clusters (stacks that correspond to the mature sequences).

Sometimes RNA secondary structure programs like RNAfold (used by *sRNAbench*) will not calculate the correct structure if no adequate constraints are provided. Therefore, if the two read clusters overlap only partially in the calculated secondary structure (low number of bindings), we calculate the duplex formed by the two read clusters by means of RNAcofold. Therefore, candidates are generated in two ways:

- Extract the genomic sequence spanned by the two read clusters and calculate the secondary structure by means of RNAfold
- Calculate the duplex between the two read clusters using RNAcofold

Evaluate the candidates:

A candidate is predicted as miRNA if:

- Dominant read cluster is located on the stem (does not fold back onto its-self)
- 'Theoretical' star sequence has reads (2 nt overlap allowing a deviation of 1 nt)
- The number of bindings in the duplex >= 14
- The 5' homogeneity (fraction of reads that start at the same position as the dominant read – 1 if perfectly homogeneous) must be above a minimum threshold of 0.18 in animals and 0.1 in plants
- 5 out of the following 6 conditions must be met:
    - 5' homogeneity >= 0.52 (animals) or 0.5 (plants)
    - Duplex bindings >= 17
    - At most 3 clusters in the hairpin region (loop sequences, muRNA can sometimes built additional clusters)
    - 'In cluster ratio' (the fraction of reads that belong to either guide or passenger strand) >= 0.94 (animal) o >= 0.85 (plants)
    - 'Dominant fraction': the read count fraction of the dominant read >= 0.25 (animal) or 0.40 (plants)
    - The read clusters that form the duplex are the two most expressed ones in hairpin region (met always if only two clusters exist)

3.4.7 Sequence variants

sRNAbench implements the detection of putative sequence variants based on the mismatches reported by bowtie1. For each position in the hairpin sequence, the frequencies of the different mismatches are reported. A putative sequence variant is reported if it is above the threshold specified by **minVarFreq**, a number between 0 and 1. In order to minimize the impact of sequencing errors, strict quality control should be used (see section 3.3.6)

3.5 Output files

3.5.1 Summary and log files

results.txt:

The results of the different steps, i.e. preprocessing, mapping, and prediction of novel microRNAs are summarized.

logFile.txt:

Different analysis steps are logged, but additionally possible warnings and errors are written to this file.

3.5.2 *Fasta* files

Several *fasta* output files will be generated:

reads_orig.fa:

Reads after the preprocessing, i.e. after adapter trimming, length and quality filtering and collapsing. The sequence name encodes the 'read count'. The format is >ID#'read count'.

>689339#**8**

AGATTATGAGATATGAGGGCA

Means that the read AGATTATGAGATATGAGGGCA has been obtained **8** times in this sample.

reads.fa:

Reads that have not been mapped (either to the genome or any of the used libraries).

assignedReads.fa (in stat folder)

All reads that have been assigned to any of the annotations. In genome mode this means 'genome mapped' & 'assigned', while in library mode mapped and assigned is the same.

unAssignedReads.fa (in stat folder)

Reads that could not be assigned to an annotation. In genome mode this means 'genome mapped' but not assigned, and in library mode it means 'unmapped'.

genomeMappedReads.fa (in stat folder)

Reads mapped to the genome (not written in library mode)

3.5.3 Expression profiling: *.grouped Files

Those files hold the expression profiling of a given annotation:

1. name: the name of the element
2. unique reads: the number of unique reads mapped to this element
3. read count: the total number of reads mapped to this element
4. read count (mult. map. adj.): each read is divided by the number of times that it mapped (to different genome locations (genome mode) or sequences in the library (sequence library mode))

5. RPM (lib): The Read Per Million normalized by the total number of reads mapped to the library
6. RPM (total): The Reads Per Million normalized by the total number of genome mapped reads (genome mode) or the total number of reads in the analysis (sequence library mode)
7. coordinate string: The coordinate string depends whether the genome mode or library mode was used: for genome mode the format is: "chromosome,chromosome start,chromosome end,strand" and refers to genome position of the annotation. Note that this coordinates are generated by *sRNAbench* if the annotation input was fasta format, and it is taken from the annotation file if this was in BED format. For 'library mode', relative coordinates are given, i.e. for microRNAs the coordinates of the 5p/3p arms are given.
8. RPM_adj (lib): The adjusted read count normalized by the total number of reads mapped to the library
9. RPM_adj (total): The

3.5.4 Expression profiling: Single Assignment files: *_SA.grouped

These files are obtained from the *.grouped files and the read.annotation file (see Ambiguous mapping treatment).

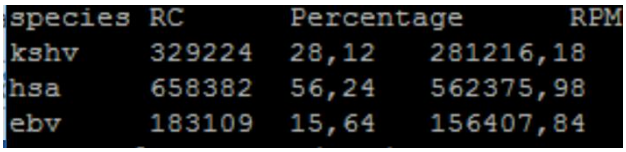
WARNING: this file should not be used for differential expression analysis. Please see Differential expression based on *.grouped files how to use single assignment in differential expression.

1. name: the name of the element
2. unique reads: the number of unique reads mapped to this element
3. read count(SA): the single assignment read count, i.e. the total number of reads assigned to this element
4. read count(SA): the total number of reads mapped to this element (this read count ignores whether some reads are mapped as well to other elements.
5. RPM (lib): The Read Per Million normalized by the total number of reads mapped to the library
6. RPM (total): The Reads Per Million normalized by the total number of genome mapped reads (genome mode) or the total number of reads in the analysis (sequence library mode)
7. coordinate string: The coordinate string depends whether the genome mode or library mode was used: for genome mode the format is: "chromosome,chromosome start,chromosome end,strand" and refers to genome position of the annotation. Note that this coordinates are generated by *sRNAbench* if the annotation input was fasta format, and it is taken from the annotation file if this was in BED format. For 'library mode', relative coordinates are given, i.e. for microRNAs the coordinates of the 5p/3p arms are given.

3.5.5 microRNA_species.txt (stat folder)

The distribution of microRNAs over the different species. This file is only written if more than one species was used for profiling.

- **Species:** the short name of the species
- **RC:** the read count assigned to the given species
- **Percentage:** percentage of assigned reads
- **RPM:** read per million expression value



species	RC	Percentage	RPM
kshv	329224	28,12	281216,18
hsa	658382	56,24	562375,98
ebv	183109	15,64	156407,84

Figure 9: microRNA_species.txt file

3.5.6 isomiR output: mature.iso files

This file holds the isomiR composition of each mature microRNA. The file has the following columns:

1. name: the name of the mature microRNA
2. pre-microRNA: the name of the precursor sequence
3. RC: the read count of the mature sequence (canonical sequence and all isomiRs)
4. UR: unique reads mapped to the mature sequence (canonical sequence and all isomiRs)
5. RPM (lib): The Read Per Million normalized by the total number of reads mapped to the library
6. RPM (total): The Reads Per Million normalized by the total number of genome mapped reads (genome mode) or the total number of reads in the analysis (sequence library mode)
7. arm: the arm - either 3p or 5p
8. isoString: this string holds all values for all detected isomiRs. The values for the different isomiR types are separated by '|'. For decoding the string please see next paragraph.
    - ***nta#BASE_RC***: all non-templated nucleotide additions. For example: nta#A_9568, means that 9568 reads in the sample do present one or more 3' terminal As which are not present in the reference sequence (either genome or miRBase hairpin sequence)
    - ***nta#BASE#ADDITIONS_RC***: non-templated nucleotide additions of a given length. For example: nta#T#1_125 , means that 125 reads present mono-uridylation at its 3' end (1 U added which is not present in the reference sequence) .
    - ***lv5p_RC***: 5' length variants .For example, lv5p_862 means that 862 reads do show any type of 3' length variation.

- ***lv3p_RC***: 3' length variants. lv3p_222835 means that 222835 reads do show any type of 3' length variation.
- ***mlv_RC***: Multiple length variants

3.5.7 canonical.txt

This file holds the expression values of the canonical microRNA sequences (as defined by the annotation). Only the exact sequences are profiled. The file is only written if **isoMiR=true** is set. The format is the same as in the grouped files.

3.5.8 isomiR annotation

The file *microRNAannotation.txt* assigns to each read mapped to a known microRNA an isomiR related label.

- read: the read sequence
- name: the name of the mature microRNA
- preMicro: the name of the pre-microRNA
- isoClass: the assigned isomiR class
- NucVar: the observed nucleotide variation (reference > sample)
- read count: the read count
- RPM library normalized
- RPM total input (or total genome mapped) normalized

3.5.9 Per mature microRNA isomiR summary (isomiR_summary.txt)

- name: name of the mature microRNA
- UR: number of unique reads
- RC: read count
- RPM(total): Reads Per Million normalized to all preprocessed input (library mapping)/genome mapped (genome mode) reads
- RPM(lib): Reads Per Million normalized to all reads mapped to a known microRNA
- Canonical_RC: Read count of the canonical sequence
- NTA(A): number of reads with a non-templated A addition
- NTA(U): number of reads with a non-templated U addition
- NTA(C): number of reads with a non-templated C addition
- NTA(G): number of reads with a non-templated G addition
- lv3pE: number of reads with 3' length extension (longer than the canonical sequence)
- lv3pT: number of reads with 3' length trimming (shorter than the canonical sequence)

- lv5pE: number of reads with 5' length extension (longer than the canonical sequence)
- lv5pT: number of reads with 5' length extension (shorter than the canonical sequence)
- mv: number of reads classified as multiple length variants

3.5.10 Sample isomiR summary

Two files: “isomiR_NTA.txt” and “isomiR_otherVariants.txt” summarize the isomiR statistics for NTA (non-templated additions) and several length variants.

- name: the name of the isomiR type:
    - A (adenine addition), C (cytosine addition), T (U/T addition), G (G addition),
    - lv5pT: 5' trimmed
    - lv5pE: 5' extended
    - lv5p: 5' length variant (lv5pT + lv5pE)
    - lv3pT: 3' trimmed
    - lv3pE: 3' extended
    - lv3p: 3' length variant (lv3pT + lv3pE)
    - mv: multiple length variants
- totalRC: the number of reads mapped to microRNAs
- NTA_count: the number of reads that belong to a given isomiR type
- wMean: the weighted mean (NTA_count/totalRC)
- mean: the mean isomiR ratio of the sample. For each microRNA, an isomiR ratio is calculated like: (number of reads belonging to a certain isomiR type) / (total number of reads mapped to the microRNA).
- stdDev: standard deviation of the mean.

3.5.11 Sample summary as a function of NTA length

‘isomiR_lenNTA’ (located in the ***stat*** folder) lists the non-templated additions as a function of the number of added nucleotides.

- length: the number of added (non-templated) nucleotides
- RC_A: number of reads with a non-templated A addition of a given length
- weighted_mean_A: the weighted mean of A-NTA’s (NTA_count/totalRC)
- mean_A: the mean isomiR ratio of the sample. For each microRNA, an isomiR ratio is calculated like: (number of reads belonging to a certain isomiR type) / (total number of reads mapped to the microRNA). In this case it is the number of reads that have a certain number of A’s added.
- stdDev_A: the standard deviation of mean_A

Other columns are for T (U), C and G additions and the meaning is as explained above for A additions.

3.5.12 The “reads.annotation” file

This file represents a summary at the read level, i.e. each read is listed individually together with all the annotation to which it mapped. It contains the following columns:

1. read sequence
2. the read count
3. RPM: The Reads Per Million normalized by the total number of genome mapped reads (genome mode) or the total number of reads in the analysis (sequence library mode)
4. Classification group. It consists of the name of annotation file (for libs=) or hairpin/mature (microRNAs from miRBase) plus the mapping orientation (sense or anti-sense). The format is “AnnotationGroup#Orientation”. Note that the name can be “mixed” if the read maps to several different annotations.
5. Mapped Annotations: This column contains all annotations to which the read maps. The format for the mapping to one element is: “AnnotationGroup#AnnotationName#Orientation#CoordinateString”. As a read can map to several different annotations, those are separated by '$'. For example, “hv_030312_v2_18#MLOC_55934.3#**antisense**#s**$**hv_030312_v2_18#MLOC_55933.2#**sense**#s” means that i) the corresponding read mapped to twice (two annotation strings separated by one dollar sign), ii) the read maps one gene in sense and another one in antisense direction, iii) “hv_030312_v2_18 is the name of the library (AnnotationGroup) given at the common line like this libs=, iv) The gene names (AnnotationName) are MLOC_55933.2 and MLOC_55934.3

3.5.13 Read Length distribution

Several files are written summarizing the read length distribution:

- **readLengthFull.txt**: length distribution without setting any thresholds like minimum length or minimum read count.
- **readLengthAnalysis.txt**: distribution of the reads that are used for the analysis.
- **assignedReads.readLen**: the read length distribution of all assigned reads, i.e. those reads that could be assigned to any of the used annotations.
- **unAssignedReads.readLen**: in genome mode: the reads that have been mapped to the genome but could not be assigned to an annotation; in library mode: the reads that could not be mapped (assigned) to any of the annotations
- **genomeMappedReads.readLen**: the read length distribution of genome mapped reads (this file does not exist in library mode)
- ***readLength*** **folder**: the length distribution of reads mapped to the different annotations, i.e. **mature.readLen** holds the read length distribution of reads assigned to a mature microRNA (guide or passenger strand)

All files have the following columns:

1. Read Length: the length of the sequenced RNA fragment/molecule
2. UR: the number of unique reads of a given length
3. Percentage_UR: the percentage of unique reads of a given length
4. RC: the read count that corresponds to a given length (sum of all reads with a given length)
5. Percentage_RC: The percentage of the total read count picked up by a given length
6. RPM: The Reads Per Million expression value as a function of read length

Note that there might be slight differences between both files due to the quality criterion which is not applied in the 'readLenghtFull.txt'

3.5.14 Distribution format

Several files do have the same format, as the mappingStat.txt, *mappingStat_sensePref.txt* or *genomeDistribution.txt.*

- name: The name of the library and the mapping orientation (sense, anti-sense) coded into a string like this: 'library#orientation'
- UR: number of unique reads assigned to the 'category'
- URperc: the percentage of unique reads assigned to the category
- RC: total read count assigned
- RCperc: the percentage of reads assigned to the category
- RPMassigned: The read per million normalized by the total number of **assigned reads**, i.e. those that could be mapped (library mode) or mapped/assigned (genome mode) to any of the RNA elements in the libraries. This column should sum 1,000,000
- RPManalysis: The reads per million normalized by the number of input reads (library mode) or number of genome mapped reads (genome mode)

3.5.15 RNA distribution summary

A summary of the mapping process can be found in the mappingStat.txt and mappingStat_sensePref.txt files. Frequently, a read maps to several annotations, both in sense and antisense direction. In the 'sensePref' file, preference is given to the sense orientation. This means if a read maps both, in sense and antisense orientation, only the sense mappings are considered for the statistics. The format is explained here.

3.5.16 The genome distribution: genomeDistribution.txt

The small read distribution over the different species. The mapping to the genome is generally done specifying a maximum number of allowed chromosome positions to which a read can map. If this number is exceeded, than the read is labels as 'high-redundant (HR)'. Those reads are not used for the profiling but they are used for the genomeDistribution.txt file. For example in Figure 7 it can be seen that 1959 reads are highly redundant (orange), i.e. they map to more than 10 position in the chromosome 22 (10 is the default value which can be changed with the bowtieReportCount parameter).

Furthermore, it can be seen that 23% of all reads do map to the human herpesvirus type 8 (NC_009333) – marked in green.

The format is explained here.

name	UR	URperc	RC	RCperc	RPMassigned	RPManalysis
NC_009333-chr22	25	0.037473394	121	0.008568786	212.5839135	85.68786108
NC_007605	3432	5.144347513	184735	13.08227026	324559.4154	130822.7026
chr22-NC_007605	19	0.028479779	22	0.001557961	38.65162064	15.5796111
NC_009333	3568	5.348202776	329706	23.34859663	579257.7835	233485.9663
chr22	6941	10.40411308	52642	3.727917672	92486.30064	37279.17672
chr22_HR	834	1.25011242	1959	0.138729355	3441.751129	1387.293552
NC_009333-NC_007605	1	0.001498936	1	7.08E-05	1.756891847	0.708164141
NC_009333-chr22-NC_007605	1	0.001498936	1	7.08E-05	1.756891847	0.7081
unmapped	51893	77.78427317	842915	59.6922177	0	596922.177

Figure 10: the genomeDistribution.txt read with Excel

3.6 RNA distribution as a function of length

rnaComposition_readLength_sensePref.txt and *rnaComposition_readLength.txt* give the distribution of RNA type frequencies as a function of read length.

The format gives in the first column the read length and in all other columns the % that picks up a certain RNA type of this read length. All rows need to sum 100%. This file allows to answer questions like for example: which percentage of 33nt long reads are mapped to tRNAs?

3.7 Alignment files - processing pattern

The visualization of the alignments to the pre-microRNA sequences can be found in the ***hairpin*** folder (those to other libraries, within a folder with the library name). All reads that map to the sequence are shown and the Drosha/Dicer (DLC-1) processing pattern can be studied. The alignment files have extension 'align' (see Figure 8).

```
GCGACTGTAAACATCCTCGACTGGAAGCTGTGAAGCCACAGATGGGCTTTCAGTCGGATGTTTGCAGCTGC  7,791(11,310.4/46,881.5) adj:7,639.6
(((.(((((((((((..((((((((((((((....)))).....)))))))))))))))))))))).)))
     TGTAAACATCCTCGACTGGAAGCT                                          1,433(2,080.3/8,622.9)  adj:1,433   +   .
     TGTAAACATCCTCGACTGGA                                              1,317(1,911.9/7,924.9)  adj:1,317   +   .
     TGTAAACCTCCTCGACTGGAAGCT                                          554(804.3/3,333.6)  adj:554 +   .
     TGTAAACCTCCTCGACTGGA                                              550(798.5/3,309.6)  adj:550 +   .
     TGTAAACATCCTCGACTGGAAGC                                           478(693.9/2,876.3)  adj:478 +   .
                                    CTTTCAGTCGGATGTTTGCAGC             398(577.8/2,394.9)  adj:398 +   .
     TGTAAACATCCTCGACTGGAA                                             318(461.7/1,913.5)  adj:318 +   .
     TGTAAACATCCTCGACTGGAAGA                                           208(302/1,251.6)  adj:208   +   .
     TGTAAACCTCCTCGACTGGAAGC                                           187(271.5/1,125.3)  adj:187 +   .
     TGTAAACATCCTCGACTGGAAG                                            159(230.8/956.8)  adj:159   +   .
     TGTAACCATCCTCGACTGGAAGCT                                          155(225/932.7)  adj:155 +   .
     TGTAAACCTCCTCGACTGGAA                                             134(194.5/806.3)  adj:134   +   .
     TGTAACCATCCTCGACTGGA                                              104(151/625.8)  adj:104 +   .
     TGTAAACATCCTAGACTGGA                                              78(113.2/469.4)  adj:39 +   .
     TGTAAACCTCCTCGACTGGAAG                                            74(107.4/445.3)  adj:74 +   .
     TGTAAACATCCTAGACTGGAAGCT                                          72(104.5/433.3)  adj:36 +   .
                                     TTTCAGTCGGATGTTTGCAGC             71(103.1/427.2)  adj:71 +   .
     TGTAAACATACTCGACTGGAAGCT                                          66(95.8/397.1)  adj:66   +   .
     TGTAAACCTCCTCGACTGGAAGA                                           66(95.8/397.1)  adj:66   +   .
     TGTAAACATCCTCGACTGGAAGT                                           61(88.6/367.1)  adj:61   +   .
     TGTAAACATACTCGACTGGA                                              54(78.4/324.9)  adj:54   +   .
     TGTAAACATCCTCGACTGGAAA                                            49(71.1/294.9)  adj:49   +   .
      GTAAACATCCTCGACTGGAAGCT                                          48(69.7/288.8)  adj:48   +   .
                                    CTTTCCGTCGGATGTTTGCAGC             46(66.8/276.8)  adj:46   +   .
```

Figure 11: Visualization of the read alignments to the human microRNA has-mir-30a.

For each ‘align’ file, a corresponding ‘countArray’ file is written

position	uniqueCount	percUnique	totalCount	percTotalCount	rpm_lib	countStartEnd	rpm_startEndPosition
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3.7.1 Novel microRNAs

If the predict=true parameter was set, *sRNAbench* will predict novel microRNAs. The results are written into the ***novel*** folder and into 3 files: ***novel.txt***, ***mature_novel.fa*** (*fasta* file with mature sequences) and ***hairpin_novel.txt*** (*fasta* file with pre-microRNA sequences).

The ***novel.txt*** file has the following format:

name: the name of the novel microRNA, which can be either the name of a known microRNA family (in this case the microRNA is known, but novel in the species) or a name like xxx-novel-number.

seqName: the chromosome/scaffold/contig name

start: start coordinate of the novel pre-microRNA in the chromosome/scaffold/contig

end: end coordinate of the novel pre-microRNA

strand: the strand

total RC: the total read count of the pre-microRNA

5pSeq: the mature microRNA sequence of the 5' arm

5pRC: the read count of the mature 5p (only the “canonical” sequence, no isomiRs are counted)

3pSeq: the mature microRNA sequence of the 3' arm

3pRC: the read count of the mature 3p (only the “canonical” sequence, no isomiRs are counted)

duplexType: in the current version (May 2017) this can be only ***duplex*** (Dicer/Drosha pattern between the two most expressed read clusters of the pre-microRNAs). Likely in the future other types will be added (non-canonical biogenesis pathways)

isHairpin: can be true (the pre-microRNA has a strict hairpin structure) or false (the pre-microRNA does not have a strict hairpin structure)

hasHomolog: the novel microRNA does have putative homologs in other species

hairpinSeq: the sequence of the hairpin sequence (pre-microRNA + flankings)

duplexQuality: exact=strict 2nt overhangs, lax= at least one of the extremes has either 1 nt or 3 nt overhang

detectionType: RNAfold: the duplex can be found in the secondary structure predicted by RNAfold; RNAcofold: the duplex cannot be found in secondary structure but RNAcofold predicts a Drosha/Dicer duplex.

mature5pFluctuation: the fraction of reads that start at the same position as the mature sequence (the canonical microRNA sequence)

InClusterRatioSense: the fraction of reads that are members of the two most expressed clusters or are a putative loop sequence.

Dominant2AllRatioMature: the ratio between the dominant read (the canonical sequence) and the sum of all reads of a cluster.

matureBindings: the number of bindings in the duplex (between 5p/3p arms)

top2ClusterAreGuideAndStar: the two most expressed clusters correspond to the guide and passenger sequence.

noCluster: the number of read clusters in the hairpin sequence (should be normally 2; one corresponding to guide and one to passenger strand)

outOf6Fulfilled: the number of flexible conditions (see the 6 previous columns) that are fulfilled.

3.7.2 Putative sequence variants: microRNA_seqVariants.txt

- Name: the name of the hairpin sequence
- Type: the sequence variant (like T>A: T in reference and A in read)
- RC: the total read count of the putative sequence variant
- noReads: the number of different unique reads that show the sequence variant
- RCpos : the total read count at the given position
- Perc: the fraction: RC/RCpos
- Position: the position in the hairpin sequence

3.8 Tips and tricks

3.8.1 Overwrite the mapping parameters

It is possible to “overwrite” the global mapping parameters given at the command line (noMM=, seed=, alignType= and bowtieReportType= bowtieReportCount=) for some analysis steps. This is mainly useful for the 'Library mode'. The parameters can be changed by adding a parameter string to the library name (or to the homologous microRNAs): For example:

snoRNA.fa#1#20#n#-m#20

would cause *sRNAbench* to align the reads to the library named snoRNA.fa with noMM=1, seed=20, alignType=n and bowtieReportType=-m bowtieReportCount=20.

Therefor in the bowtie parameters this will appear –n 1 –l 20 –m 20

homolog=mmu:rno:mmu:ptr#2#20#v#-a

will make *sRNAbench* to align the reads to the putative homologous microRNAs from human (hsa), mouse (mmu), rat (rno) and chimp (ptr) with parameters: noMM=2 alignType=v –a (all –best –strata alignments will be reported).

Note that, i) for alignType=v, the seed= parameter is ignored, and ii) the global parameters are only temporarily overwritten. For the next lib= library, the global parameters will be applied unless they are overwritten again.

3.8.2 Using bowtie indexes as libraries in genome mode

This possibility makes sense only if the library contains spliced sequences. The previously genome mapped reads are mapped again to the provided bowtie index. The mapping parameters to the bowtie index can be overwritten. Let's analyze the following example:

```
java -jar sRNAbench.jar input=SRR069835_part.fastq species=chr22 microRNA=dme
dbPath=”path to database” adapter=CTGTAGGCAC noMM=0 seed=18 alignType=n
libs=refSeq#0#20#v#20
```

First, all reads are mapped to the genome with global parameters (1 mismatch, seed length 18, seed alignment mode). The genome mapped reads (that do not map to microRNAs) are then mapped to the bowtie index refSeq with noMM=0 and alignType=v. In this way, for all bowtie indexed libraries, stricter parameters can be chosen.

3.8.3 Multi-species analysis

A new feature in *sRNAbench* is the possibility to analyze several species at the same time. This is especially interesting when infected cells or host/parasite interactions are analyzed. However, if the different genomes in the analysis share “chromosome names” (like chr1 in human and mouse genome), the genome sequences needs to be manipulated first. For example in the web-server, we added to each sequence name the short species name. In this way, chr1:hsa (human) could be distinguished from chr1:mmu mouse. A simple command can add this information to the genome sequences prior to built the bowtie indexes and the seqOBJ zip file.

```
cat hg19.fa | awk '{ print $1”hsa” }' > hg19_mp.fa
```

3.8.4 Construction of shared libraries

Sometimes, many different RNA species are annotated in a single file. If the classification is known, *sRNAbench* can use them for the summary files. Briefly, the ID of the sequences should be separated by the class name by a ':'. For example '>NR_046235:ribosomal_RNA'. We implemented several helper tools that can generate this prepared annotation files from primary Ensembl and NCBI annotation files (see also the *sRNAbench* helper tools)

3.8.5 Prediction of novel microRNAs

In general, we recommend to predict the novel microRNAs in a separate run and with strict parameter settings: noMM=0 alignType=v minRC=2

3.8.6 profile tRNAs

sRNAbench has some additional features when used with tRNA libraries from the Genome tRNA database (see also the *sRNAbench* helper tools) . Before using it with *sRNAbench*, the description field should be removed from the fasta file. In Linux, this can be done easily by means of this command: **cat eukaryotic-trnas.fa | awk '{ print $1 }' > eukaryotic-trnas_woDesc.fa** being *eukaryotic-trans.fa* the input file.

When analyzing this tRNAs, another parameter can be set at the command line: tRNA=”name of the tRNA library”. An additional file will be generated at an anti-codon level.

4 *sRNAde*: Differential expression

The *sRNAde* program can be used in two different ways.

i. It can be applied to a user generated expression matrix

ii. It takes a number of individual *sRNAbench* runs as input. For the runs the relation between the samples, i.e. defining the experimental groups must be provided.

Independently of the input type, two main results are generated. First, heatmaps are calculated using Hierarchical Clustering (implementation: hclust for R, parameters method=”complete”) which allows the user to visualize the clustering of the samples and to detect outlier samples. Second, the differential expression is assessed using 3 frequently used programs: edgeR, DESeq and NOISeq. Additionally, the program also generates a consensus differential expression file which gives the user the possibility to increase the stringency using only those microRNAs/sRNA that have been detected as differentially expressed by more than one method.

If *sRNAbench* output is used as input, additional analysis types are available. For example, the module generates a summary of sequencing, adapter removal, genome mapping statistics, isomiR differential expression, RNA distribution summaries, and many more.

The differential expression can be launched by means of the next command:

```
java -jar sRNAde.jar input=<path to input folders> output=<name of the output folder> grpString=<the names of the input samples>
```

4.1 Mandatory parameters: *sRNAbench* input

- **input=<String>**: the path to the *sRNAbench* output folders of the individual *sRNAbench* runs. Please note that all sample output folders must be within the same directory. For example, if the default output was used: input=$*sRNAtoolbox*DB/out
- **output=<String>**: the path of the output folder.
- **grpString=<String>**: the group string must contain the names of the different *sRNAbench* output folders in the following way: f1_1:f2_1#f1_2:f2_2 being f1_1 the first folder (sample) of the first group (controls in a case/control study) f2_1 the the second folder of the first group, f1_2 the first folder of the second group (cases) etc

4.2 Mandatory parameters: expression matrix input

- **input=<String>:** the path to the expression matrix (for the format, please see below in this section).
- **output=<String>:** the name of the output folder. The output folder will be placed in the directory given with input=the path of the output folder
- **matrixDesc=<String>**: description of the matrix samples, i.e. assign an experimental group (a label like 'cancer', 'control' etc.) to each sample. The string for an experimental setup with 4 control samples and 4 cancer samples would be: control,control,control,control,cancer,cancer,cancer,cancer

Expression matrix format:

The expression matrix should be in standard format, names on the first column and expression values in all other columns (see following example).

name	SRR837842|control	SRR837839|control	SRR837836|control	SRR837833|control	SRR837843|cancer	SRR837840|cancer	SRR837837|cancer	SRR837834|cancer
hsa-miR-671-5p	41	130	28	87	71	104	102	33
hsa-miR-361-3p	237	69	3	262	203	63	70	161
hsa-miR-484	326	197	25	622	50	138	72	339
hsa-miR-30b-3p	301	92	17	84	118	98	42	114

Figure 12: format of the expression matrix (from http://bioinfo5.ugr.es/static/WebManual_sRNAtoolbox.pdf)

Important: If with input= a file is given, then automatically the differential expression is launched treating this input data as expression matrix. This analysis includes both, the differential expression and a cluster analysis. Some more parameters exist for this analysis: General parameters for differential expression and heatmaps.

4.3 Additional parameters for sample descriptions

- **sampleDesc=<String>: The user can provide a name for each sample (those names will appear in the output files and graphics. For example sampleDesc=healthy01:healthy02:cancer01:cancer02. The order of the samples is the same as defined by grpString but note that sampleDesc should not contain '#' .**
- **grpDesc=<String>: If not set, the groups will be named 'grp'. The user can give names to the groups like healthy#cancer by means of a string that contains as many group separators '#' as defined with grpString=**

4.4 Differential expression based on *.grouped files

- **diffExpr=<boolean>:** Perform differential expression analysis. Default: diffExpr=<false>. **Currently, at least two samples must exist per condition otherwise only the RPM (Read Per Million) normalized expression matrix will be generated.**
- **diffExprFiles=<String>:** The files that should be used for differential expression analysis. Note that this file needs to contain a column with the read count (needed by the programs like edgeR or DEseq and that several files can be used separating the file paths by '|'.
- **minRCexpr=<Integer>**: The minimum read count that ALL samples of at least one group (condition) must have so that the entity (microRNA/gene etc) is included into the read count expression matrix (input for edgeR, DEseq and NOIseq). Note that the read count is expected to be in the third column (like in the 'grouped' files from *sRNAbench*). Default: minRCexpr=1.
- **minRPMexpr=<double>**: The minimum read per million (RPM) expression value that ALL samples of at least one group (condition) must have so that the entity (microRNA/gene etc) is included into the expression matrix (the RPM expression matrix is calculated from the read count matrix. (Default: minRPMexpr=1)
- **DEmode=[a,b]**: If a entity (microRNA or gene name etc) is encountered twice in the data: a) will only use the expression value of the most frequent while b) will sum up the expression values using the column of adjusted expression values (column 4 in the 'grouped' files from *sRNAbench*) . Default DEmode=b
- genomeFiles=<true,false>: true --> use the chromosome string (from the grouped files) for 'hashing', i.e. calculate differential expression for each loci at which a microRNA/gene is located. This parameter makes only sense for *sRNAbench* output files when the genome mode was used. (Default genomeFiles=false)
- **makeSingleAssignDE=<Boolean:** true The differential expression is calculated using the 'single assignment' mode, i.e. a multiple mapping read is only assigned once - to the most expressed loci. The most expressed loci is determined over all used samples. Note, if single assignment is used than the name of the annotation must be selected (annotName= ; see next parameter). Default: makeSingleAssignDE=false. See Ambiguous mapping treatment .
- **annotName=<String>**: The name of the library for which the single assignment differential expression should be calculated, i.e. the names of the library in the

reads.annotation file. For example, for microRNAs annotName=mature or for tRNA fragments annotName=tRNA (or annotName=hg19-tRNAs (if libs=hg19-tRNAs was given). Important: right now only ‘sense’ mapping reads are considered.

IMPORTANT: the user should not use the Single Assignment files for the differential expression analysis. (those with _SA in its name). For single assignment differential expression the **makeSingleAssignDE=true s**hould be used.

i. the diffExprFiles are used to generate a read count expression matrix

ii. the read count matrix is ordered by read count

iii. in decreasing read count order, all reads are assigned to one element and then eliminated to avoid multiple assignment.

4.5 Make summary files

- stat=<true,false>: A study summary will be caclulated for a given file. This analysis will: i) summarize the information of one column of a given file over all samples, ii) generate an 'expression matrix' like output format. The first column holds the names of microRNAs/genes etc. and all other column represent the values in the different samples. iii) calculate per-group (mean and std. deviation) and between group statistics (t-test).
- statFiles=<String>: The files that should be used for stat=true analysis. Note that several files can be used separating the file paths by '|'
- colData=<int>: The column that should be used for stat=true. Note that his number is 0-based, i.e. the first column will be colData=0.
- minRCdata=<double>: The minimum value that all samples of at least one group must fulfil.
- folderData=<String>: If the file that should be summarized is located within a subfolder of the *sRNAbench* output folder, the name of this folder needs to be given here. If no folder is given, the file given by (statFiles=) should be in the root of the *sRNAbench* folders.

4.6 IsomiR analysis

There are different ways to analyze the isomiR generation among different conditions. The comparison can be performed at a microRNA level (iso=true) or at the sample level (isoSummary=true). isoSummary=true will summarize the files isomiR_NTA.txt and isomiR_otherVariants.txt and does need not any additional parameters.

- **iso=<true,false>:** true isomiR analysis, i.e. comparison of isomiR ratios between groups. This analysis will: i) calculate the isomiR ratios (either (isomiR type RC/ total RC)) or (isomiR type RC/ canonical RC) for all microRNAs and samples. ii) calculate the isomiR ratios which are significantly different between two groups.
- **isoFile=<String>:** The file that should be used for isomiR analysis. Default isoFile=mature.iso

- **minRCiso=<int>:** The minimum read count of the microRNA so that the isomiR ratios are calculated. Default: minRCiso=10
- **isoCanonical=<true,false>:** true --> calculate additionally the isomiR ratios as (isomiR type RC / canonical RC). Default: isoCanonical=false

4.7 isomiR analysis at a read level

- **readIso=<true,false>:** true This analysis will: i) generate a expression matrix using the column specified by colReadIso; ii) calculate the differential expression (if colReadIso=5)
- **colReadIso=<5,6,7>:** The column that should be used (from the microRNAannotation.txt file). Default: colReadIso=5 (read count); Others: 6 (RPMlib), 7 (RPMtotal)
- **minExprReadIso=<int>:** The minimum value found in the column (defined by colReadIso=) that each sample in at least one group (condition) must accomplish. Default: minExprReadIso=1
- **detectIsoMiRs=<String>:** A string that encodes the names of all isomiR types that should be analyzed. Note that the names of the different isomiR types must be separated by '|' Default: "nta#A|nta#A#1|nta#T|nta#T#1|nta#C|nta#G|lv5p|lv3p"

4.8 Make expression matrix from fasta files

This function can be used to make expression matrixes out of every fasta file produced by *sRNAbench*.

- mappedReadsDE=<true,false>. True This analysis will: i) generate an expression matrix using the read counts from the fasta files (like >ID#read_count).ii) calculate the differential expression using edgeR, DEseq and NOIseq. Default: mappedReadsDE=false
- minRCreadLevel=<int>: The minimum read count that each sample in at least one group (condition) must accomplish. Default: minRCreadLevel=10
- fastaFiles=<String>: The fasta file(s) that should be used. Several can be defined separating by '|'.
- folderData=<String>: Defines the subfolder within the *sRNAbench* output folder were the file is located. By default, no subfolder is defined and the fasta file is expected within the root of the output folder.

4.9 Analyse annotated reads

Analyse the annotated reads at a read level.

- readLevel=<true,false>: true This analysis will: i) generate an expression matrix using the annotated read counts from reads.annotation file; ii) calculate the differential expression using edgeR, DEseq and NOIseq.

- minRCreadLevel=<int>: The minimum read count that each sample in at least one group (condition) must accomplish. Default: minRCreadLevel=10
- readLevelExprCol=<1,2>"," The column that should be used to generate the expression matrix: 1 --> RC; 2 --> RPM (total). Default: readLevelExprCol=2
- annotCol=<3,4>: The annotation level that should be used: 3 = groups#orientation; 4 = group#name#orientation. Default: annotCol=4

4.10 General parameters for differential expression and heatmaps

Note that this parameters will influence both, and *sRNAbench* differential expression analysis and user input expression matrix analysis.

- **fdr=<double>**: false discovery rate for DEseq and edgeR. Default: fdr=0.05
- **noiseq=<double>**:corresponding noiseq parameter. Default: noiseq=0.8
- **hmPerc=<double> = 1**: the percentile of expression that should be used for the heatmap. Default: hmPerc=1
- **hmTop=<int>**: if hmPerc = 1 then takee the (hmTop) top expressed genes/microRNAs to generate the heatmap. Default: hmTop=50
- **percentil =<double>**: the percentil applied to expression matrix. Default **percentil =-1** (not applied)
- **top=<int>**:the top X entries from the expression matrix that should be used for the DE analysis. Default: top = -1

4.11 Differential Expression output

4.11.1 "diffExpr=true" output

A number of output files will be generated. The base name of the files are:

"**base name of corresponding grouped file**"**_minRCexpr_'DEmode'**

For example, if mature_sense.grouped (default file) was used, the names will be ***mature_sense_1_RCadj***

As by default minRCexpr=1 and DEmode=b (the adjusted Read count is used).

The default output for a DE analysis with two groups (control and cancer) will produce the following output files:

- **mature_sense_1_RCadj_control_vs_cancer.xlsx**: The output generated by edgeR, DEseq and NOI-seq.
- **mature_sense_1_RCadj.mat**: the adjusted read count expression matrix which is used as input for edgeR, DEseq and NOI-seq
- **mature_sense_1_RCadj_totalRPM.mat:** The normalized read count matrix (normalized with the total number of reads)

- **mature_sense_1_RCadj_libraryRPM.mat:** The normalized read count matrix (normalized with the number of reads mapped to the corresponding library – microRNAs in this case)
- **mature_sense_1_RCadj_TMM_normalized.tsv:** the TMM normalized expression matrix (generated by edgeR)
- **mature_sense_1_RCadj_heatmap_perc0.87.png:** the cluster analysis and heatmap taking the top 50 microRNAs (percentile 87 here)
- **mature_sense_1_RCadj_heatmap_perc0.87_median_normalized.png:** the cluster analysis and heatmap taking the top 50 microRNAs applying median normalization (rest the median value)

4.11.2 "iso=true" output

The isomiR ratio can be defined in two different ways:

- Default: isomiR type read number / total read count of microRNA (canonical read count plus all isomiRs)
- Activated by **isoCanonical=true**: isomiR type read number / canonical (miRBase) read count

IsomiR patterns are analyzed for an *.iso file (isoFile=<String>) . The first step in the analysis of the isomiR patterns consists in the generation of an isomiR ratio matrix. The files have 'mat' extension and the name of the file indicates the analyzed isomiR type. For example, **iso_nta#T.mat** indicates that this file holds the isomiR ratios for non-templated additions of U/T. **iso_nta#T_canonical.mat** is the corresponding matrix for the isomiR ratio based on the canonical read count.

**.ttest and *.sig files*

.ttest* files do hold all comparisons, while **.sig* files do hold only the statistically significant isomiR ratio differences (after FDR correction). For each isomiR ratio matrix, all possible comparisons are calculated (for two groups –> 1 comparison, for three groups –> 3 comparisons, for four groups –> 6 comparisons, etc.). The nomenclature is like before, adding the group numbers. For example, **iso_nta#T_1_2.ttest holds the ttest outcome for the comparison between the first and the second group. The files have the following columns:

- microRNA: name of the mature microRNA
- mean_1: the mean isomiR ratio in first group
- var_1: the standard deviation of the isomiR ratios in first group
- mean_2: the mean isomiR ratio in the second group
- var_2: the standard deviation of the isomiR ratios in second group
- p: the exact p-value (t-test)
- FDR: the FDR corrected p-value

4.11.3 "seqStat=true" output files

The file *sequencingStat.txt* summarizes the sequencing statistics:

- sample: The name of the sample
- raw reads: number of raw input reads
- adapter cleaned: number of adapter cleaned reads
- reads in analysis: number of reads in analysis (applying length thresholds to adapter cleaned reads)
- unique reads in analysis: number of unique reads in analysis (applying length thresholds to adapter cleaned reads)
- genome mapped reads: number of reads mapped to the genome
- unique reads mapped to genome: number of unique reads mapped to the genome

5 sRNAblast (Docker)

sRNAblast can be used to further analyse unmapped reads. The program uses blastn to align input reads against a BLAST database (***nr*** database by default).

sRNAblast uses the *sRNAbench* input layer and therefore it accepts the same input files.

```
java -jar sRNAblast input='reads_file' output='output_directory' maxReads=100
```

5.1 sRNAblast parameters

Parameter	Description
input=file	The path to the input file (supported formats: fastq, read/count, fasta). This is the only mandatory parameter.
output=<folder>	The name of the oputput folder. Default: output=/opt/*sRNAtoolbox*DB/out
maxReads=<int>	The number of top expressed reads that we will be blasted. Default: maxReads=500
blastDB=<String>	The name of the blast database. The parameter can take this values. Default: blastDB=nr
minIdent=<int>	The minimum identity for a blast result. Default: minIdent=90 (90% sequence similarity)
maxEvalue=<int>	The maximum E-value. Default: maxEvalue=2
word_size=<int>	The word size used for blast search. Default word_size=9 http://www.genebee.msu.su/blast/blast_faqs.html

5.2 sRNAblast output files

Figure 13 shows a typical sRNAblast output directory.

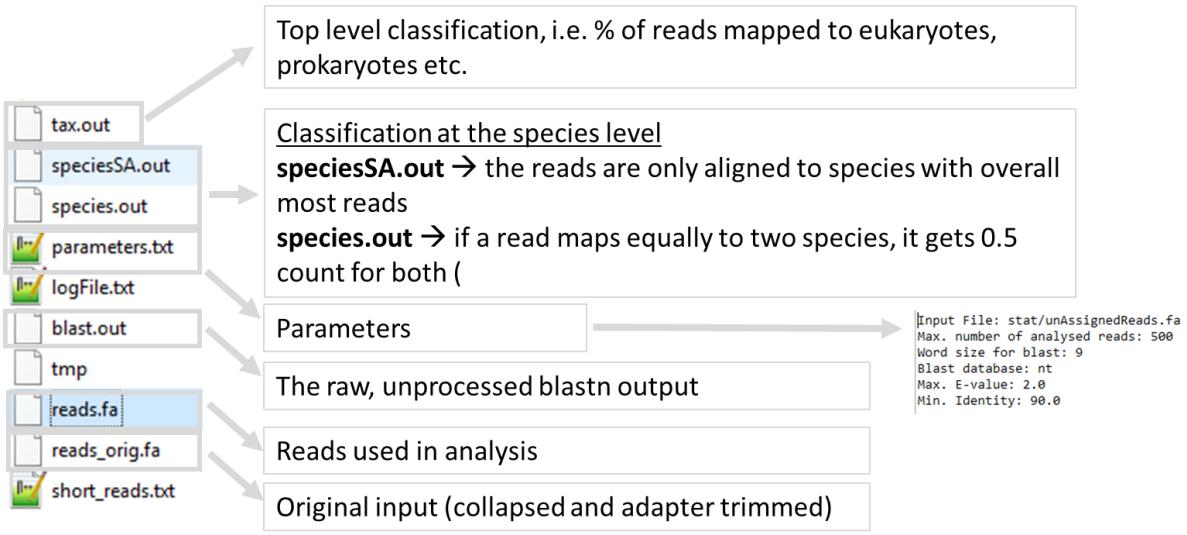


Figure 13: Content of typical sRNAblast output directory

Note: By default, the NCBI remote database is used. That means that sRNAblast needs nearly no local resources, BUT is rather slow.

A local blast can be used by adding remote=local to the command line. For this:

A local database needs to be created (how to create local DB and available databases):

The environmental variable BLASTDB needs to be set assigning the value of the directory with the blast database files. Type 'echo $BLASTDB' in the terminal, if no path is given back, add this to your .bashrc

BLASTDB=/home/user/blastDB

5.2.1 Format of tax.out, speciesSA.out and species.out

Column	description
species	Name of the species or taxonomic level
RC	The total number of mapped reads (might be not an integer if multiple mapping reads exist)
%RC	The percentage of reads mapped to this category

6 miRNAconstargets

This program allows to calculate consensus miRNA target predictions, both in animals and plantas.

In animals, 4 different methods are currently implemented: Miranda , TargetSpy, PITA and a simple seed method (a target gets predicted if a seed match is detected in a transcript)

6.1 Launch miRNAconstargets

For animals:

```
java -jar miRNAconsTargets.jar
```

or in the Docker:

```
miRNAconsTargets
```

For plants:

```
miRNAconsTargets_plants.py
```

or in the Docker:

```
miRNAconsTargets_plants
```

6.2 miRNAconstargets parameters

Both implementations (plant and animals) have exactly the same parameters. The parameters are positional and must be given in this order: 1) microRNA file (fasta), 2) target file (i.e. 3'UTR sequences in fasta format), 3) output directory, 4) number of threads, 5) program string and 6) program parameters.

The program string needs to specify the programs separted by ':'. The available programs are: TargetSpy (TS), miranda (MIRANDA), pita (PITA) and SEED (simple seed method). For example MIRANDA:PITA would use Miranda and Pita while PITA:SEED:TS would use Pita, seed method and TargetSpy.

The program parameters need to be given in the same order as in 'program_string'. For example: TS:SEED '-s:1-8' would change the default parameters of TargetSpy to 'detect only target sites with seeds' (-s) and the definition of the seed region to: seed is from position 1 to position 8 (default is from 2 to 8)

7 Apendix

7.1 Standalone versions

Currently *sRNAbench* and *sRNAde* can be downloaded as standalone versions **(download standalone versions**).

7.1.1 Dependencies

sRNAbench and *sRNAde* are implemented in JAVA and need apart from a JRE the following third party software that needs to be installed **and placed in the PATH** first.

- Vienna RNA package for RNA Secondary Structure Prediction and Comparison Vienna package 2. ***sRNAbench* will only work with Vienna 2.0 or higher!**
- Bowtie - An ultra-fast memory-efficient short read aligner (Bowtie). ***sRNAbench* will only work with Bowtie1 but not Bowtie2**.
- For the differential expression program *sRNAde*: edgeR package, DEseq / DEseq2 NOISeq (R packages) and The Apache Commons Mathematics Library
- 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.

7.1.2 Get started with the standalone

sRNAbench and *sRNAde* rely on a local database where most of the library files, genome sequences, Rscripts and Bowtie indexes need to be stored. The database can have any arbitrary name and the easiest way to generate it is by means of the "start-up" DB following the next steps:

1. Download the "start-up" database: *sRNAtoolbox*DB. Please note that this does not generate a full database, i.e. it includes only microRNAs for *Human, human herpesvirus type 8 (HHV-8) and Epstein-Barr virus (EBV).* To populate the database, please see below.
2. Extract it to the directory of your choice: *tar xvzf sRNAtoolboxDB.tgz* . **We recommend to use /opt/*sRNAtoolbox*DB** (All programs take this directory as default value)
3. Download the most recent version and replace the *sRNAbench*.jar and *sRNAde*.jar files form the database: *sRNAbench*.jar && *sRNAde*.jar

After extracting the database, you should see the following folders (:

- **libs:** The default location for all sequence libraries except the genome sequences
- **index:** This folder contains the Bowtie indexes of the genome sequences
- **seqOBJ:** This folder contains the prepared genome sequences in order to access them rapidly (the files can be generated by means of the makeSeqObj program).
- **exec**: the programs, i.e. jar and other executable files
- **out**: The default ***output*** folder. This directory is always used if the output= parameter is omitted

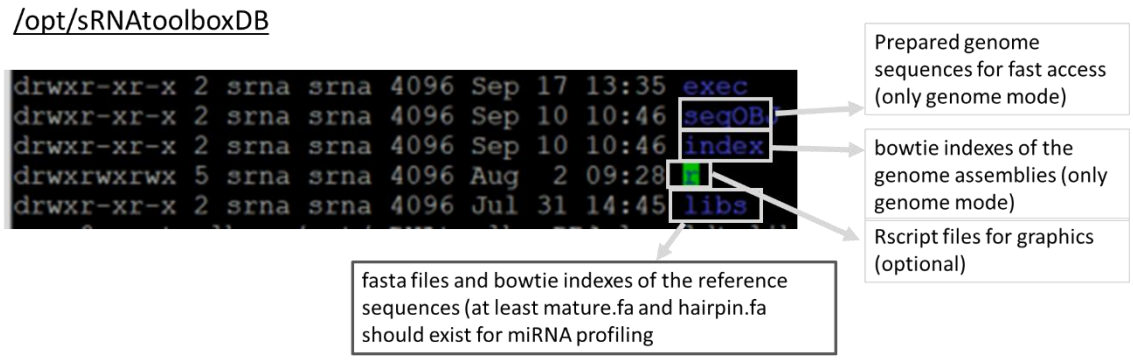


Figure 3: typical folders in sRNAtoolboxDB

7.2 Manually populate *sRNAtoolbox*DB

There are two ways to populate the database: manually or by means of the populate program that downloads annotations directly from our annotation database.

> **Note: populate is preinstalled in the Docker. Download only if you are using the standalone programs.**

7.2.1 Genome sequences

In order to add a genome sequence to the database, two steps are needed:

1. Apply 'bowtie-build' to the genome sequence(s) and place the obtained 6 bowtie index files into the ***index*** folder of the *sRNAtoolbox* database
2. Apply makeSeqObj to the genome sequence(s) and place the obtained zip file into the ***seqOBJ*** folder. For example, **java -jar makeSeqObj.jar hg19.fa .** Please note that this might take a while (hours) in case of big genomes. Furthermore, it is quite memory demanding, so probably the heap space needs to be increased by means of -Xmx on the command line. Finally, the bowtie index base names must be the same then the one of the genome assembly 'zip' file.

7.2.2 microRNAs

Known microRNA sequences are best obtained from miRBase or mirGeneDB. Download mature and hairpin sequences and extract them into the ***libs*** folder of the *sRNAbench* database. In linux, move to the ***sRNAtoolboxDB/libs*** folder and type:

wget -nd ftp://mirbase.org/pub/mirbase/CURRENT/mature.fa.gz

wget -nd ftp://mirbase.org/pub/mirbase/CURRENT/hairpin.fa.gz

> **Note: microRNA annotations need to be given for 'mature' and 'hairpin' sequences (that contain the pre-microRNA). Currently, only *fasta* input is supported. Furthermore, the miRBase nomenclature starting with the short species name 'hsa', 'mmu', 'ath', etc. or miRGeneDB starting with upper case letters like 'Hsa', 'Mmu' is expected**

7.2.3 Other small RNA species

Other small RNA annotations can be given either in ***fasta***, *BED* or *GTF* format or as **Bowtie indexes**. *BED* and *GTF* format will be only valid if a genome sequence is specified with species= (see below *sRNAbench* parameters). Note that the annotations in *BED* format **need to be** from the same assembly as the used genome sequence. If no extension is given, *sRNAbench* will assume the existence of a Bowtie index in the ***libs*** folder aligning directly against it. Otherwise, for a library with 'fa' extension either the coordinates are obtained by mapping against the genome sequence (if species= is set) or a Bowtie index is generated first by *sRNAbench*.

7.2.4 *sRNAbench* helper tools

We developed some helper tools in order to facilitate the usage of Ensembl and NCBI annotations. The web-version of the helper tools can be accessed here. For the standalone versions see here: sRNAhelpers

8 FAQs

Q: When profiling in genome mode, I cannot find some microRNA which are present in library mode

A: In genome mode, *sRNAbench* maps the microRNA precursor sequences (hairpins) first to the genome. If those cannot be mapped, the corresponding microRNAs will no get profiled.

9 References

Anders, S., Huber, W., 2010. Differential expression analysis for sequence count data. Genome Biol. 11, R106. https://doi.org/10.1186/gb-2010-11-10-r106

Aparicio-Puerta, E., Lebrón, R., Rueda, A., Gómez-Martín, C., Giannoukakos, S., Jaspez, D., Medina, J.M., Zubkovic, A., Jurak, I., Fromm, B., Marchal, J.A., Oliver, J., Hackenberg, M., 2019. sRNAbench and sRNAtoolbox 2019: intuitive fast small RNA profiling and differential expression. Nucleic Acids Res. 47, W530–W535. https://doi.org/10.1093/nar/gkz415

Backes, C., Fehlmann, T., Kern, F., Kehl, T., Lenhof, H.-P., Meese, E., Keller, A., 2018. miRCarta: a central repository for collecting miRNA candidates. Nucleic Acids Res. 46, D160–D167. https://doi.org/10.1093/nar/gkx851

Barturen, G., Rueda, A., Hamberg, M., Alganza, A., Lebron, R., Kotsyfakis, M., Shi, B.-J., Koppers-Lalic, D., Hackenberg, M., 2014. sRNAbench: profiling of small RNAs and its sequence variants in single or multi-species high-throughput experiments. Methods Gener. Seq. 1. https://doi.org/10.2478/mngs-2014-0001

Bonnet, E., He, Y., Billiau, K., Van de Peer, Y., 2010. TAPIR, a web server for the prediction of plant microRNA targets, including target mimics. Bioinformatics 26, 1566–1568. https://doi.org/10.1093/bioinformatics/btq233

Fromm, B., Høye, E., Domanska, D., Zhong, X., Aparicio-Puerta, E., Ovchinnikov, V., Umu, S.U., Chabot, P.J., Kang, W., Aslanzadeh, M., Tarbier, M., Mármol-Sánchez, E., Urgese, G., Johansen, M., Hovig, E., Hackenberg, M., Friedländer, M.R., Peterson, K.J., 2022. MirGeneDB 2.1: toward a complete sampling of all

major animal phyla. Nucleic Acids Res. 50, D204–D210. https://doi.org/10.1093/nar/gkab1101

Gómez-Martín, C., Lebrón, R., Rueda, A., Oliver, J.L., Hackenberg, M., 2017. sRNAtoolboxVM: Small RNA Analysis in a Virtual Machine, in: Dalmay, T. (Ed.), MicroRNA Detection and Target Identification: Methods and Protocols, Methods in Molecular Biology. Springer, New York, NY, pp. 149–174. https://doi.org/10.1007/978-1-4939-6866-4_12

Guo, Z., Kuang, Z., Wang, Y., Zhao, Y., Tao, Y., Cheng, C., Yang, J., Lu, X., Hao, C., Wang, T., Cao, X., Wei, J., Li, L., Yang, X., 2020. PmiREN: a comprehensive encyclopedia of plant miRNAs. Nucleic Acids Res. 48, D1114–D1121. https://doi.org/10.1093/nar/gkz894

Hackenberg, M., Rodríguez-Ezpeleta, N., Aransay, A.M., 2011. miRanalyzer: an update on the detection and analysis of microRNAs in high-throughput sequencing experiments. Nucleic Acids Res. 39, W132–W138. https://doi.org/10.1093/nar/gkr247

Hackenberg, M., Sturm, M., Langenberger, D., Falcón-Pérez, J.M., Aransay, A.M., 2009. miRanalyzer: a microRNA detection and analysis tool for next-generation sequencing experiments. Nucleic Acids Res. 37, W68–W76. https://doi.org/10.1093/nar/gkp347

John, B., Enright, A.J., Aravin, A., Tuschl, T., Sander, C., Marks, D.S., 2004. Human MicroRNA Targets. PLOS Biol. 2, e363. https://doi.org/10.1371/journal.pbio.0020363

Kertesz, M., Iovino, N., Unnerstall, U., Gaul, U., Segal, E., 2007. The role of site accessibility in microRNA target recognition. Nat. Genet. 39, 1278–1284. https://doi.org/10.1038/ng2135

Kozomara, A., Birgaoanu, M., Griffiths-Jones, S., 2019. miRBase: from microRNA sequences to function. Nucleic Acids Res. 47, D155–D162. https://doi.org/10.1093/nar/gky1141

Love, M.I., Huber, W., Anders, S., 2014. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. 15, 550. https://doi.org/10.1186/s13059-014-0550-8

The RNAcentral Consortium, 2019. RNAcentral: a hub of information for non-coding RNA sequences. Nucleic Acids Res. 47, D221–D229. https://doi.org/10.1093/nar/gky1034

Robinson, M.D., McCarthy, D.J., Smyth, G.K., 2010. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26, 139–140. https://doi.org/10.1093/bioinformatics/btp616

Rueda, A., Barturen, G., Lebrón, R., Gómez-Martín, C., Alganza, Á., Oliver, J.L., Hackenberg, M., 2015. sRNAtoolbox: an integrated collection of small RNA research tools. Nucleic Acids Res. 43, W467–W473. https://doi.org/10.1093/nar/gkv555

Sturm, M., Hackenberg, M., Langenberger, D., Frishman, D., 2010. TargetSpy: a supervised machine learning approach for microRNA target prediction. BMC Bioinformatics 11, 292. https://doi.org/10.1186/1471-2105-11-292

Tarazona, S., Furió-Tarí, P., Turrà, D., Pietro, A.D., Nueda, M.J., Ferrer, A., Conesa, A., 2015. Data quality aware analysis of differential expression in RNA-seq with NOISeq R/Bioc package. Nucleic Acids Res. 43, e140. https://doi.org/10.1093/nar/gkv711

Wu, H.-J., Ma, Y.-K., Chen, T., Wang, M., Wang, X.-J., 2012. PsRobot: a web-based plant small RNA meta-analysis toolbox. Nucleic Acids Res. 40, W22–W28. https://doi.org/10.1093/nar/gks554