

**Title:**

**Methodologies for discovery and quantitative profiling of sRNAs in potato**

**Authors:**

Maja Križnik<sup>1</sup>, Maja Zagorščak<sup>1</sup>, Kristina Gruden<sup>1</sup>

**Affiliations:**

<sup>1</sup> Department of Biotechnology and Systems Biology, National Institute of Biology, Večna pot 111, 1000  
Ljubljana, Slovenia

**\* Corresponding author:** Maja Križnik, e-mail: [maja.kriznik@nib.si](mailto:maja.kriznik@nib.si)

## Running head:

Discovery and quantitative profiling of sRNAs

## Abstract

Small RNAs (sRNAs) are short non-coding RNAs involved in regulation of a wide range of biological processes in plants. Advances in high-throughput sequencing and development of new computational tools had facilitated the discovery of different classes of sRNAs, their quantification and elucidation of their functional role in gene expression regulation by target transcript predictions. The workflow presented here allows identification of different sRNA species: known and novel potato miRNAs, and their sequence variants (isomiRs), as well as identification of phased small interfering RNAs (phasiRNAs). Moreover, it includes steps for differential expression analysis to search for regulated sRNAs across different tested biological conditions. In addition, it describes two different methods for predicting sRNA targets, *in silico* prediction and degradome sequencing data analysis. All steps of the workflow are written in a clear and user-friendly way, thus they can be followed also by the users with minimal bioinformatics knowledge. We also included several in-house scripts together with valuable notes to facilitate data (pre)-processing steps and to reduce the analysis time.

**Key words:** sRNA, miRNA, siRNA, phasiRNA, degradome, potato, smallRNA identification, gene expression, small RNA target prediction, small RNA-seq data analysis

# 1. Introduction

Small RNAs (sRNAs) are defined as ~ 21-24 nt non-coding RNAs, that negatively regulate gene expression, predominantly at the post-transcriptional level. Since the discovery of their important role in regulation of multiple biological processes in eucaryotes, much effort has been put into adjustment of high-throughput sequencing technology for sRNA detection (i.e. library preparation methods) as well as into development of computational tools for their reliable identification and characterization in plants. Typical sRNA sequencing experiment begins with the preparation of a sRNA cDNA library from the RNA sample of interest, followed by the ‘massively parallel’ sequencing of millions of individual cDNA molecules from the library on a single run (*1*). First, sRNAs are size-selected, for example on a denaturing polyacrylamide gel. Then the adapters are sequentially ligated to the 3’ and 5’-end of sRNAs. The ligated products are converted to cDNA and PCR-amplified. Finally, libraries are purified (e.g. by gel purification) to isolate tagged library with a sRNA insert from adapter dimer contaminants and subjected to high-throughput sequencing (Fig. 1). Deeply sequenced sRNA libraries (e.g. more than 20 MIO reads) allow discovery of various types of sRNAs (including novel and lowly abundant sRNAs), as well as their read-count quantification (*2, 3*).

“[Fig. 1 near here]”

## 1.1. Computational tools for novel sRNA identification and categorization

Sequencing reads of a typical sRNA library represent a complex mixture of several types of molecules: microRNAs (miRNAs), small interfering RNAs (siRNAs) and a large number of ‘contaminating’ ribosomal (r)RNA, transfer (t)RNA, small nuclear (sn)RNA and small nucleolar (sno)RNA sequences. To classify obtained short RNA reads and to select sRNAs of interest in subsequent analyses different computational methods are usually employed. Known miRNA sequences can be readily identified by

matching searches to annotated miRNA databases such as miRBase database ((4); <http://www.mirbase.org>). A more difficult task is identification of *bona fide* novel miRNAs. With the development of sRNA sequencing techniques, many miRNA prediction tools have emerged for the discovery of novel miRNAs in plants. Typically, the first step of miRNA prediction tools is mapping of short RNA reads on selected plant genome following identification of genomic regions with the potential to form hairpin secondary structures with structural features similar to known hairpin miRNA precursors (i.e. pre-miRNAs) (5, 6). However, a typical genome contains thousands of regions with the predicted secondary structures that resemble miRNA precursors, of which only a low percentage is likely to represent actual miRNA producing loci (*MIR* loci). Usually, each miRNA prediction tool incorporates some additional annotation criteria to predict high-confidence miRNA precursor (pre-miRNA), mature miRNA (highest expressed read) and its corresponding complementary strand miRNA (previously termed as passenger or star strand miRNA\*). These criteria include:

- 1) limiting the miRNA precursor length to maximum of 300 nts
- 2) filtering based on folding free energy of predicted pre-miRNA precursor ( $< -0.2$  kcal/mol per nucleotide)
- 3) considering the strandedness of a candidate *MIR* locus
- 4) a discrete pattern of alignments to a single genomic strand, with a major species (the mature miRNA) separated by a short distance from a second, less abundant complementary miRNA (Fig. 2 left panel) (5, 7).

“[Fig. 2 near here]”

Following predictions, the identified miRNAs and pre-miRNA sequences and their corresponding *MIR* loci need to be properly annotated. Currently, mature miRNAs are named with the ‘miR’ prefix, followed by a numerical identifier that designates the miRNA family (e.g. miR156), whereas the names of plant

pre-miRNAs and *MIR* loci differ by the capitalization of the ‘miR’ prefix , and carry the family identifier (e.g. *MIR156*) (8). Assignment to known miRNA families is based on shared sequence similarity of the miRNA sequence, or the miRNA precursor sequence with annotated ones available in miRBase (6, 9). If there is no similarity detected, miRNAs can be assigned into a novel miRNA family. Furthermore, if two or more *MIR* loci are producing identical or similar miRNAs, they are assigned the same family number with sequential alphabetical suffixes (i.e., *MIR156a*, *MIR156b*) (6, 8) and corresponding miRNAs are named as miR156a, miR156b. miRNAs originating from the same hairpin precursor, previously annotated as mature miRNA (i.e. miR156) and miRNA\* (i.e. miR156\*), are currently named using -5p/-3p strand annotation, where the suffixes ‘-5p’ and ‘-3p’ are added (i.e. miR156-3p, miR156-5p), to designate from which arm of the precursor these two miRNA species arise (Fig. 3, Table 1) (10).

In addition to canonical miRNAs (i.e. miRNAs annotated in miRBase), miRNA variants of variable lengths can be also produced from the same hairpin precursor, which are termed as isomiRs (Fig. 3). The isomiRs are produced through individual or combined effects of:

- 1) Post-transcriptional enzymatic nucleotide additions, modifications, or removals that alter mature miRNA sequence during miRNA maturation (non-templated isomiRs);
- 2) imprecise or alternative cleavage of DCL1 during miRNA biogenesis resulting in different mature miRNA sequences with shifted start and/or end positions (templated isomiRs) (Fig. 3) (11, 12).

isomiRs can be distinguished by the canonical miRNA forms by adding numerical suffixes following their canonical miRNA identifiers (e.g., miR156a-5p.1 and miR156a-5p.2) (Fig. 3, Table 1) (10).

“[Fig. 3 near here]”

In contrast to miRNAs, endogenous siRNAs originate from long dsRNA molecules and the precursor is processed so that numerous siRNAs accumulate from both strands of the dsRNA (Fig. 2 right panel) (9).

Endogenous siRNAs are frequently 21 (phasiRNAs) or 24 nt long (heterochromatic siRNA; hc-siRNAs) (13–16), with phasiRNAs being the most prominent endogenous siRNA populations. In contrast to miRNA-encoding *MIR* loci, identification of phasiRNA and their corresponding *PHAS* loci requires a different computational prediction strategy. Based on sRNA sequences aligned to either genome or transcriptome reference, tools are searching for the pattern of phasing inside the mapped region (an example of phasing is given in Fig. 2 right panel). In general, these algorithms calculate the number of reads that are ‘in phase’ against those that are ‘out of phase’ in order to determine the probability that a particular locus is truly a *PHAS* locus (17, 18). Furthermore, one can also further classify phasiRNAs into ta-siRNA subgroup simply by comparing the mapping locations of the siRNA and its target transcript; if in trans (in the coding transcript), then siRNA can be further categorized in the ta-siRNA group (19). In contrast to miRNAs, nomenclature for endogenous siRNAs (i.e. phasiRNAs) has not yet been established, instead they are generally named either by

- 1) a unique sequence identifier given by the author, such as ((pha)siRNA1, (pha)siRNA2, etc.);
- 2) based on their *PHAS* loci of origin (e.g. TAS3-tasiRNA);
- 3) adapting the position-dependent coding system introduced by Allen et al. (2005, Fig. 4), where phasiRNAs originating from 5′ side of the miRNA cleavage site by 5′ direction phasing of the original transcript are sequentially named as 5′D1 (+) , 5′D2 (+), and those generated from 3′ side by 3′ direction phasing as 3′D1 (+) , 3′D2 (+), while phasiRNAs from the opposite strand are named as 5′D1 (-) , 5′D2 (-) and 3′D1 (-) , 3′D2 (-), respectively (Fig. 4) (20–22).

“[Fig. 4 near here]”

## 1.2. Finding sRNA targets in plants

Once discovered, the identification of the target transcripts for any given sRNA is a key step in determining its function. miRNAs and phasiRNAs direct post-transcriptional regulation of gene expression, by binding to the complementary site on their target mRNAs. The result of this binding is either immediate degradation of the transcript or repression of its translation, both dependent on the degree of complementary relation between the miRNA and its target sequence. In plants, both experimental and computational techniques are used to identify target mRNAs (23).

### 1.2.1. *In silico* target prediction

As plant miRNAs typically guide the cleavage of target mRNAs, bioinformatic predictions have been very successful in identifying sRNA targets (24, 25). Cleavage of mRNA requires a high degree of miRNA:target base pairing, especially at the 5' and central positions of the duplex relative to the miRNA. Positions 2–13 are relatively mismatch-free and contain somewhat fewer G:U base pairs. In general, a scoring scheme that requires perfect or near perfect match (one mismatch) within the 2-13 region has been widely employed in published computational methods for predicting plant sRNA targets (25, 26). Usually maximum three mismatches are allowed overall (27–30).

The secondary structure of mRNA is also very important for target prediction. The secondary structure around the target site will prevent sRNA binding to mRNA target (25, 31). Therefore, some target prediction methods, such as psRNATarget and imiRTP, employ additional tools (like RNAup, RNAduplex, etc.) for target-site accessibility evaluation, by calculating unpaired energy (UPE) required to 'open' secondary structure around a sRNA target site on the mRNA (Fig. 5) (25, 32, 33). Incorporating such target-site accessibility evaluation to sRNA target analysis was reported to significantly improve the prediction accuracy (32).

In plants, it has been observed that mismatches occurring around the center of miRNA/mRNA complementary region (positions 9-11) tend to disable the cleavage activity, however, the binding can still block the translation of the target (32). So far, psRNATarget is the only tool that is capable of distinguishing the sRNA mode of action, being either cleavage or translational inhibition (Fig. 5). Additionally, it was shown to be the fastest and among comparably reliable tools (24).

“[Fig. 5 near here]”

### 1.2.2. Degradome sequencing

The main limitation associated with all target prediction tools is a high false positive rate of predictions, since recognition between miRNA and target site is affected by other undiscovered factors as well (34). Additionally, target prediction tools can miss some *bona fide* targets due to non-conventional features of miRNA-mRNA interactions in plants (24). Generally, the target mRNA or protein level is negatively associated with its corresponding sRNA's expression level. Combining *in silico* target prediction with transcriptome experimental data has been shown to significantly reduce false positive predictions (35). However, one of the more effective approaches is to use degradome sequencing data (34).

To experimentally verify the sRNA-induced cleavage of target mRNAs, Addo-Quaye et al. (2008) and German et al. (2009) introduced the parallel analysis of RNA ends (PARE), also known as degradome sequencing (36, 37). Experimental studies have shown that sRNA-guided cleavage of mRNA targets occurs exactly between the 10<sup>th</sup> and 11<sup>th</sup> nucleotide of complementarity relative to the sRNA 5' end. The resulting upstream fragment of the cleaved target rapidly degrades, while the downstream fragment is stable *in vivo* (38). The approach takes advantage of the free 5'-monophosphate and poly(A) tail present on the downstream 3' fragment after cleavage. Typically, degradome libraries are constructed by ligation



of custom 5'-RNA adapter containing a 3' MmeI restriction endonuclease site to a polyA-enriched fragments, followed by reverse transcription and second-strand synthesis (Fig. 6). cDNA is then digested with the MmeI restriction enzyme, which cleaves off 20 nucleotides 3' from the recognition site, leaving only 20 nucleotides of the target transcript linked to the adapter. Next, a 3' dsDNA adapter with degenerate nucleotides in the overhang region (to match the overhangs generated by MmeI digestion) is ligated to the MmeI digestion products and the resulting fragment is PCR-amplified, purified (e.g. by PAGE) and subjected to high-throughput sequencing (Fig. 6) (25, 37).

After sequencing, reads are aligned back to reference transcripts and used as evidence for sRNA mediated cleavage. Some degradome processing tools, such as CleaveLand, PAREsnip and SeqTar, have been developed to assist in identifying sRNA targets by combining the information of detected cleavage site on mRNA together with sRNA-mRNA complementarity and providing support for sRNA-target cleavage interaction (25, 39–41). The limitation of degradome sequencing is that it can detect only sRNA-mRNA target interactions that result in mRNA cleavage. It can also miss sRNA-target pairs when targets are lowly expressed or their 3' ends are highly unstable (36).

“[Fig. 6 near here]”

## 2. Materials

1. Preferential operating system: Ubuntu based Linux distribution with Graphical User Interface (GUI).
2. Software requirements for pre-processing of sRNA sequencing data (P) and for sRNA data analysis (A):
  - a) cutadapt (<https://cutadapt.readthedocs.io/en/stable/>) (P)
  - b) Java Platform, Standard Edition (SE), Runtime Environment (<https://www.oracle.com>) (P)
  - c) FastQC (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc>) (P)
  - d) FASTX-Toolkit ([http://hammonlab.cshl.edu/fastx\\_toolkit/commandline.html](http://hammonlab.cshl.edu/fastx_toolkit/commandline.html)) (P)
  - e) The small RNA Workbench (version 3 or later) (<http://srna-workbench.cmp.uea.ac.uk/>) (P)
  - f) Bowtie (<https://sourceforge.net/projects/bowtie-bio/>) (P, A)
  - g) SAMtools v1.6 or later (<http://samtools.sourceforge.net/>) (P, A)
  - h) Bedtools (<https://bedtools.readthedocs.io/en/latest/index.html>) (P, A)
  - i) ShortStack v3.8.5 (<https://github.com/MikeAxtell/ShortStack>) (A)
  - j) ViennaRNA package (<https://www.tbi.univie.ac.at/RNA/>) (A)
  - k) Bowtie2 (<https://sourceforge.net/projects/bowtie-bio/files/>) (A)
  - l) GenomeTools (<http://genometools.org/>) (A)
  - m) Uitas (<http://www.smallrnagroup.uni-mainz.de/software.html>) (A)
  - n) CleaveLand4 v.4.5 (<https://github.com/MikeAxtell/CleaveLand4/>) (A)
  - o) R (<https://www.r-project.org/>) (A)
  - p) Rstudio (optional) (<https://rstudio.com/products/rstudio/>) (A)

## 3. Methods

The bioinformatic workflow consists of two main parts (Fig. 7), pre-processing and sRNA data analysis.

The sRNA sequencing data are first pre-processed in order to remove adapter sequences, low-quality and

low complexity reads, and contaminating reads such as rRNAs, tRNAs, snRNAs and snoRNAs. Cleaned sRNA reads are then separately subjected to miRNA or phasiRNA analysis. Identified sRNAs are then quantified and subjected to differential expression analysis. Finally, to reveal the functions of sRNAs of interest, target prediction can be performed *in silico* and/or by employing the degradome sequencing data analysis.

“[Fig. 7 near here]”

### 3.1.Pre-processing of sRNA sequencing data

#### 3.1.1. Adapter removal (trimming) and short reads filtering

1. Place the sRNA sequencing file(s) into selected working directory. We recommend to create a new subdirectory (e.g. Pre-processing) within your working directory to keep files organized. To create the subdirectory run the following command within your working directory:

```
$ mkdir Pre-processing
```

Position yourself to Pre-processing subdirectory:

```
$ cd Pre-processing
```

2. Remove adapter sequences and filter the sRNA reads using cutadapt:

```
$ cutadapt -a TGGAATTCTCGGGTGCCAAGG --format fastq \  
--discard-untrimmed -m 18 -M 26 \  
-o sRNA_sample_trimmed.fastq \  
sRNA_sample.fastq
```

This command removes the 3'-adapter 'TGGAATTCTCGGGTGCCAAGG' of Illumina's small RNA TruSeq kit, from the 3' end of each sRNA read in the input sRNA sequencing file `sRNA_sample.fastq`. Reads without adapters are discarded with the argument `--discard-untrimmed`. sRNA reads shorter than 18 nt and longer than 26 nt (arguments `-m`, `-M`) are filtered out. The trimmed reads are stored in subdirectory `Pre-processing` in the output file `sRNA_sample_trimmed.fastq`. Sometimes 3' -adapters in the raw reads have been already removed by the sequencing service facilities. In this case, trimming of the sRNA reads by `cutadapt` should be run without the `-a` (adapter) argument.

### 3.1.2. Quality control and filtering of low-quality sRNA reads

Sequencing errors in sRNA sequencing data introduced during sequencing may impact the interpretation of the downstream analysis. Usually, the quality of sRNA reads obtained by Illumina sequencing is high (Phred quality score  $\geq 30$ ) due to their short sizes, generally not requiring additional filtering. However, the quality control of sRNA reads before proceeding with further analyses is considered good practice.

1. Check sRNA read quality by the FastQC tool. The following command will generate the reports for the trimmed sRNA sequencing sample, `sRNA_sample_trimmed.fastq`.

Run:

```
$ fastqc sRNA_sample_trimmed.fastq
```

2. Examine per base sequence quality in the FastQC generated reports. Open `sRNA_sample_trimmed.html` file or inspect the `fastqc_data.txt` file in generated

sRNA\_sample\_trimmed.fastqc.zip archive. Note, that you have to first decompress the zip archive using the command:

```
$ unzip sRNA_sample_trimmed.fastqc.zip
```

3. Filter out the reads from sRNA\_sample\_trimmed.fastq using FASTQ Quality Filter of the FASTX-Toolkit with lower Phred score.

```
$ fastq_quality_filter -q 20 -p 100 -v -i sRNA_sample_trimmed.fastq -o  
sRNA_sample_trimmed_qfiltered.fastq
```

This command will take sRNA\_sample\_trimmed.fastq and check for reads with the Phred quality score > 20 (argument -q) across the whole sequence (argument -p), and output the high quality reads into the file sRNA\_sample\_trimmed\_qfiltered.fastq.

4. Additionally, examine adapter content and sequence length distribution in the FastQC generated reports to check for the efficiency of adapter removal. After adapter removal, two peaks, one at 21 nt and one at 24 nt should be visible. Observed peaks at higher lengths may indicate presence of bases from adapter sequences in the sRNA sequences.

5. Convert the trimmed fastq file sRNA\_sample\_trimmed.fastq (or sRNA\_sample\_trimmed\_qfiltered.fastq, if you performed quality filtering in step 3) into fasta format using fastq\_to\_fasta of FASTX-Toolkit:

```
$ fastq_to_fasta -v -i sRNA_sample_trimmed.fastq -o  
sRNA_sample_trimmed.fasta
```

or:

```
$ fastq_to_fasta -v -i sRNA_sample_trimmed_qfiltered.fastq -o  
sRNA_sample_trimmed_qfiltered.fasta
```

6. Remove low complexity sequences (sequences containing less than three distinct nucleotides) and invalid reads (sequences containing undefined nucleotides, e.g. Ns) by using Filter Tool of the UEA small RNA Workbench. Launch the `srna-workbench` startup program from the command line using the following command:

```
$ java -jar /pathTo/sRNAWorkbenchStartup.jar
```

where `/pathTo/` denotes the full or relative path to the directory in which `srna-workbench` is installed.

7. If you are running the sRNA Workbench in GUI mode, open the Filter tool, import sRNA sample in fasta format from the output of step 5 and select: ‘filter low complexity’, ‘filter invalid sequences’, check the option ‘output redundant file’, define the output location of the resulting file (e.g. `/Path_to_working_directory/Pre-processing/`) and click ‘run’. Upon completion of the analysis the resulting file with a suffix ‘`_filter_R.fasta`’ (e.g. `sRNA_sample_trimmed_filter_R.fasta`) will appear at your designated directory. For an alternative see <http://srna-workbench.cmp.uea.ac.uk/>.

### **3.1.3. Removal of rRNA, tRNA, snRNA and snoRNA sequences**

This step includes removal of rRNA, tRNA, snRNA and snoRNA sequences from the sRNA sequencing data to avoid erroneous annotation of miRNAs or siRNAs. Due to a large number of sRNA reads to be scanned for similarity with rRNA, tRNA, snRNA and snoRNA sequences, this step can be also performed

after the miRNA and siRNA prediction pipelines. In this case, mapping will be performed on the limited set of sRNA (i.e. only those predicted as miRNAs or siRNAs), which will reduce the analysis time.

1. Download potato rRNAs, tRNAs, snRNAs, snoRNAs sequences from RNACentral database (<https://rnacentral.org/search?q=potato>) for each type separately.

2. Create a new `ncRNAs` subdirectory within the Pre-processing:

```
$ mkdir ncRNAs  
$ cd ncRNAs
```

3. Place all four downloaded fasta files into the `ncRNAs` subdirectory (make sure that there are no other `.fasta` files within this directory) and concatenate the downloaded fasta files into a single fasta file named `ncRNAs.fasta` using `cat` command within the `ncRNAs` subdirectory as follows:

```
$ cat *.fasta > ncRNAs.fasta
```

This combined fasta file `ncRNAs.fasta` will serve as a reference for mapping of sRNA reads from the pre-processed sRNA file `sRNA_sample_trimmed_filter_R.fasta`. You can use any mapping tool, here we will present mapping steps using Bowtie. Prior to mapping, make sure that sequences in `ncRNAs.fasta` and `sRNA_sample_trimmed_filter_R.fasta` files contain thymines (Ts) instead of uracils (Us), as mapping tools are adapted to run with genomes as references (*see Note 1*).

4. Generate a Bowtie index named `ncRNA_index` from `ncRNAs.fasta` within the `ncRNAs` subdirectory using `bowtie-build`:

```
$ bowtie-build ncRNAs.fasta ncRNA_index
```

5. Run Bowtie:

```
$ bowtie -a -v 0 -p 10 ncRNA_index \  
-f ../sRNA_sample_trimmed_filter_R.fasta -S > ncRNA_alignments.sam
```

The argument `-a` instructs Bowtie to report all valid alignments. The `-v 0` argument allows no mismatch in the alignment. The `-p 10` argument uses 10 threads to do the alignment. Bowtie outputs SAM file (argument `-S`) `ncRNA_alignments.sam` containing both mapped and unmapped reads. Run `$ bowtie --help` to get details on all bowtie parameters.

The following four steps (6-9), enable conversion among different formats (`.sam`, `.bam`, `.fastq`) to obtain the final fasta file (`sRNAs_unmapped.fasta`), which contains sRNA sequences not mapped to rRNA, tRNA, snRNA and snoRNA sequences, and can be used for sRNA data analysis.

6. Position (`cd ..`) to Pre-processing subdirectory and export unmapped reads from the `ncRNA_alignments.sam` file using `samtools view` (SAMtools):

```
$ samtools view -Sh -f4 ./ncRNAs/ncRNA_alignments.sam >  
sRNAs_unmapped.sam
```



Run `$ samtools view --help` to get details on parameters.

7. Convert the SAM file into BAM using `samtools view` (SAMtools):

```
$ samtools view -b -S sRNAs_unmapped.sam > sRNAs_unmapped.bam
```

8. Convert the BAM file into FASTQ using `bamToFastq` (bedtools):

```
$ bamToFastq -i sRNAs_unmapped.bam -fq sRNAs_unmapped.fq
```

Run `$ bamToFastq -h` to get details on parameters.

9. Convert the FASTQ file into FASTA file using `fastq_to_fasta` (FASTX-Toolkit):

```
$ fastq_to_fasta -v -i sRNAs_unmapped.fq -o sRNAs_unmapped.fasta
```

More details on the SAMtools options and parameters can be found at

[www.htslib.org/doc/samtools.html](http://www.htslib.org/doc/samtools.html), while details of the FASTX-Toolkit are available at

[http://hannonlab.cshl.edu/fastx\\_toolkit/](http://hannonlab.cshl.edu/fastx_toolkit/).

## **3.2. sRNA data analysis**

### **3.2.1. Known miRNAs identification**

To identify known potato miRNAs, the sRNA sequences from fasta file

`sRNAs_unmapped.fasta` need to be aligned to known (annotated) potato miRNAs.

1. Download the annotated potato miRNA sequences in fasta format from the miRBase database (<http://www.mirbase.org>). Click 'Browse' and select 'Solanum tuberosum', then choose 'mature sequence' as sequence type, click 'select all' and 'fetch sequences'. Save the file as

`stu_mature.fasta`.

2. Download our in-house `FASTA_unique_sequences_all_IDs.sh` and `FASTA_unique_sequences_all_IDs.py` scripts to prepare non-redundant fasta file of unique annotated miRNA sequences (*see Note 2*) using the following commands:

```
$ apt update # to install Git with Apt
```

```
$ apt install git
```

```
$ cd ..
```

```
$ git clone https://github.com/NIB-SI/Potato_sRNA_analysis.git
```

Note that using this command `Potato_sRNA_analysis` directory, containing six in-house scripts within `scripts` subdirectory, and some input/output illustrative files, will be downloaded into your current working directory. You can check the current directory using `$ pwd` command.

Prior to running the scripts, install all prerequisites according to the commands listed in the `README.md` file within the `scripts` subdirectory.

3. Position into `scripts` subdirectory and run commands:

```
$ cd Potato_sRNA_analysis/scripts/
```

```
$ chmod 755 *
```

This will set executable permission for all scripts.

```
$ ls -al
```

This will list all files (and directories) within the current directory.

4. Run the `FASTA_unique_sequences_all_IDs.sh` script as follows:

```
$ bash FASTA_unique_sequences_all_IDs.sh ../../Pre-  
processing/stu_mature.fasta ../output/ 1
```

where `../output` stands for relative path to your results directory and 1 for number of threads used.

Upon completion of the analysis the resulting file `unique_stu_mature.fasta` will appear in the output directory.

5. Run the script `sRNA_counts.pl` (already downloaded in section 3.2.1., step 2) within the `scripts` subdirectory with the following command:

```
$ ./sRNA_counts.pl -seqtab ../output/unique_stu_mature.fasta \  
-output ../output/known_miRNAs_counts.txt \  
../../Pre-processing/sRNAs_unmapped.fasta
```

For more information run `$ ./sRNA_counts.pl -help`. Note if your current working directory is not `scripts/`, you will need to adapt relative paths.

This command will align sRNAs from the pre-processed reads file `sRNAs_unmapped.fasta` to annotated potato miRNA sequences given in the file `unique_stu_mature.fasta` (argument `-seqtab`) and count the number of their occurrences. Results are written in a tab delimited file (argument `-output`) `known_miRNAs_counts.txt`. If working on multiple

sRNA sequencing files *see* **Note 3**. Make sure that sequences in both files

`sRNAs_unmapped.fasta` and `unique_stu_mature.fasta` both contain either Ts or Us (*see* **Note 1**) and that there are no duplicated headers in `unique_stu_mature.fasta`.

6. To obtain a FASTA file of known potato miRNAs present in the pre-processed reads file

`sRNAs_unmapped.fasta`, from `known_miRNAs_counts.txt` file filter out all the miRNAs having zero counts in the second column and generate subsetted fasta containing only known miRNAs detected in pre-processed reads file, using following commands within the `scripts` subdirectory:

```
$ awk -F "\t" '{
    count=0;
    for (i=2;i<=NF;i++){
        if ($i==0) ++count;
    }
    if (count<(NF-1)) print
}' \
../output/known_miRNAs_counts.txt | cut -d ' ' -f1 | sed '1d' > \
../output/IDs.txt
```

```
$ awk -F "\t" '{
    count=0;
    for (i=2;i<=NF;i++){
        if ($i==0) ++count;
    }
    if (count<(NF-1)) print
}' \
```

```

../output/known_miRNAs_counts.txt | cut -f1 | sed '1d' > \
../output/longIDs.txt

```

Two awk commands will generate two IDs lists, i.e. list of short and extended IDs, for which sums of counts are  $> 0$ . Short and extended IDs are concatenated (by column) to create translation table for short ID – extended ID replacement.

```

$ paste -d "\t" ../output/IDs.txt ../output/longIDs.txt > \
../output/alias.txt
$ rm ../output/longIDs.txt

```

To create a subsetted fasta, using short IDs, run:

```

$ xargs samtools faidx \
../output/unique_stu_mature.fasta < ../output/IDs.txt > \
../output/known_miRNAs.fasta
$ rm ../output/IDs.txt

```

To return extended IDs using alias translation table run:

```

$ awk 'NR==FNR{a[$1]=$2;next}
NF==2{$2=a[$2]; print ">" $2;next}
1' FS='\t' ../output/alias.txt FS='>' \
../output/known_miRNAs.fasta | sponge ../output/known_miRNAs.fasta
$ rm ../output/alias.txt

```

Upon completion of the analysis the resulting file `known_miRNAs.fasta` will appear in the `output` directory.

If you have multiple sRNA sequencing files and you would like to perform differential expression analysis only on known miRNAs move to section 3.2.6.

### 3.2.2. Novel miRNAs identification

To identify novel unannotated potato miRNAs and their loci of origin (i.e. *MIR* loci), sRNA reads from pre-processed sRNA sequencing file `sRNAs_unmapped.fasta` are subjected to ShortStack pipeline (7). The pipeline requires the reference genome file in fasta format. Make sure that ShortStack and all its dependencies have been installed and added to your PATH (*see Note 4*).

1. Create a new subdirectory, `Novel_miRNA_analysis`, which will help you to keep files organized. Position (`cd ..`) to your working directory and create the subdirectory using the following command:

```
$ mkdir Novel_miRNA_analysis
```

2. Position to `Novel_miRNA_analysis` subdirectory and copy the pre-processed sRNA sequencing file `sRNAs_unmapped.fasta` file into `Novel_miRNA_analysis` subdirectory:

```
$ cd Novel_miRNA_analysis
```

```
$ cp ../Pre-processing/sRNAs_unmapped.fasta
```

3. Download the latest potato genome sequences (e.g. PGSC\_DM\_v4.03) from The SOL Genomics Network [ftp://ftp.solgenomics.net](http://ftp.solgenomics.net) using `wget` command:

```
$ wget  
ftp://ftp.solgenomics.net/genomes/Solanum_tuberosum/assembly/PGSC_DM_v4.  
03/PGSC_DM_v4.03_pseudomolecules.fasta.zip
```

4. Unzip/extract the zip archive:

```
$ unzip PGSC_DM_v4.03_pseudomolecules.fasta.zip
```

5. Rename the file to `genome.fasta`

```
$ mv PGSC_DM_v4.03_pseudomolecules.fasta genome.fasta
```

6. Run ShortStack within `Novel_miRNA_analysis` subdirectory as follows:

```
$ ShortStack --readfile sRNAs_unmapped.fasta --genomefile genome.fasta -  
-mincov 5 --bowtie_cores 10 --sort_mem 10G
```

When working on multiple files consider using the command given in **Note 5**. To optimize the ShortStack analysis pipeline running time, one can increase the number of available processor threads used for mapping step by adding `--bowtie_cores [integer]` argument (default value is 1). An additional option that will decrease the analysis time is to increase the maximum memory usage during BAM file sorting (`--sort_mem`; default is 768M; use K/M/G suffixes to specify kilobytes, megabytes, and gigabytes). In the given example, 10 threads and 10 gigabytes of working memory will be used for computation. Run `$ ShortStack --help` for additional information on available parameters.

7. Import the ShortStack's output file `Results.txt` into a spreadsheet application (e.g. LibreOffice Calc, OpenOffice Calc, etc.) and extract miRNAs by selecting all that have a 'Y' (yes) in the 13<sup>th</sup> column (MIRNA). These represent miRNAs that have passed all ShortStack's tests and for which also complementary strand miRNAs have been detected.
8. Using the filter option within any spreadsheet application, one can optionally exclude low expressed miRNAs from the `Results.txt` file by filtering out miRNAs having less than 30 raw reads per sample (10<sup>th</sup> column; MajorRNAReads).

### 3.2.3. Assigning the predicted miRNAs to known or novel *MIR* genes and families

The output file `Results.txt` from ShortStack may also contain the predictions of already annotated potato *MIR* loci/pre-miRNAs and miRNAs. In order to separate them from potential novel *MIR* candidates, all genome locations of *MIR* loci/pre-miRNAs predicted by ShortStack need to be compared with already annotated *MIR* loci/pre-miRNAs locations.

1. Download the *MIR* loci/pre-miRNA genome location information of potato in the gff3 format from <ftp://mirbase.org/> latest version genomes subdirectory. Within `Novel_miRNA_analysis` subdirectory run the following command to download the gff3 file:

```
$ wget ftp://mirbase.org/pub/mirbase/CURRENT/genomes/stu.gff3
```



If you are working with the another genome version of potato (e.g. PGSC\_DM\_v4.03) that was used for gff3 file preparation deposited in miRBase (i.e. SolTub3.0), you should manually prepare the corresponding gff3 file by mapping annotated potato pre-miRNA sequences deposited in the miRbase database to your potato reference genome (follow the next steps 2-8).

2. Within Novel\_miRNA\_analysis subdirectory generate a Bowtie2 index named as genome\_index from the potato reference genome (file: genome.fasta) using bowtie2-build:

```
$ bowtie2-build genome.fasta genome-index
```

3. Download annotated potato pre-miRNA sequences from miRBase database (<http://www.mirbase.org>). Click 'Browse' and select 'Solanum tuberosum', then choose 'stem-loop sequence' as sequence type, click 'select all' and 'fetch sequences'. Save the file into the Novel\_miRNA\_analysis subdirectory as stu\_pre-miRNAs.fasta.

4. Align annotated potato pre-miRNAs to potato genome using Bowtie2:

```
$ bowtie2 -a -x genome-index -f stu_pre-miRNAs.fasta -S aligned_pre-miRNAs.sam
```

Run `$ bowtie2 --help` to get details on all bowtie2 parameters.

Results of this command are stored in the SAM file format. The four steps (5-8) that follow, enable conversions among different formats (.sam, .bam, .bed) to finally obtain gff3 file containing the location of pre-miRNAs in potato genome.

5. Convert sam file into BAM using `samtools view` command (SAMtools):

```
$ samtools view -bS aligned_pre-miRNAs.sam > aligned_pre-miRNAs.bam
```

6. Sort BAM file using `samtools sort` command (SAMtools):

```
$ samtools sort aligned_pre-miRNAs.bam > sorted_aligned_pre-miRNAs.bam
```

7. Convert sorted BAM into bed file using `bamToBed` command (bedtools):

```
$ bamToBed -i sorted_aligned_pre-miRNAs.bam > sorted_aligned_pre-  
miRNAs.bed
```

8. Convert bed file into gff3 using `gt-bed_to_gff3` command (GenomeTools):

```
$ gt bed_to_gff3 sorted_aligned_pre-miRNAs.bed > sorted_aligned_pre-  
miRNAs.gff3
```

To search for putative overlaps between annotated and predicted *MIR* loci/pre-miRNAs we also need the locus information of *MIR* loci/pre-miRNAs predicted by ShortStack, as ShortStack's locus information (first column of the `Results.txt` table) sometimes does not match the exact location of the predicted pre-

miRNA in the genome. ShortStack start and end predicted positions of the locus usually cover narrower intervals compared to ones of the actual pre-miRNAs. Therefore, we suggest to perform separate mapping of the pre-miRNA sequences predicted by ShortStack to the genome to obtain proper coordinates to prepare the table `locations_predicted_pre-miRNAs.txt` (see example at Table 2) linking identified *MIR* loci/pre-miRNAs with miRNAs (see steps 9-13).

9. Combine all ShortStack's predicted pre-miRNA sequences (as a default ShortStack output, each pre-miRNA is given in separate file, see example on the Fig. 8) into a single fasta file `pre-miRNAs.fasta` using in-house `combine_sequences.sh` script (already downloaded in section 3.2.1., step 2).

10. Run the `combine_sequences.sh` script:

```
$ bash ../Potato_sRNA_analysis/scripts/combine_sequences.sh \  
/pathTo_ShortStack_output/MIRNAs/ \  
../Potato_sRNA_analysis/output/
```

The script will also write all ShortStack's predicted mature miRNAs and their complementary miRNAs (also given in separate files) into two separate files named `mature_miRNAs.fasta` and `star_miRNAs.fasta` (keep both files as you will need them later on). All generated output files will be stored at `/Potato_sRNA_analysis/output/`. Denote: if you wish to run the analysis for some other data set and do not wish to merge it with existing output, change the output directory (here `../Potato_sRNA_analysis/output/`), or alternatively specify different names for output files (i.e. `pre-miRNAs.fasta`, `mature_miRNAs.fasta` and `star_miRNAs.fasta`) within the script.

“[Fig. 8 near here]”

11. To obtain coordinates of the ShortStack’s predicted pre-miRNAs in the genome, follow the same steps as used for mapping of annotated potato pre-miRNAs to the genome (see section 3.2.3., steps 4-8).
12. Import the resulting gff3 file from the previous step into any spreadsheet application and generate `locations_predicted_pre-miRNAs.txt` table (see example at Table 2) containing five columns of interest (Chromosome, Start\_position, End\_position, miRNA, Orientation).
13. Before proceeding to the next step, check that chromosome names defined in both files, `locations_predicted_pre-miRNAs.txt` and `sorted_aligned_pre-miRNAs.gff3`, are consistent (i.e. in the same form, e.g. chr00, chr01, chr12).
14. Run the `MIR_loci_overlaps.R` script (already downloaded in section 3.2.1., step 2) using the following command:

```
$ R -e  
"shiny::runApp('../Potato_sRNA_analysis/scripts/MIR_loci_overlaps.R')"
```

“[Fig. 9 near here]”

15. Import the `locations_predicted_pre-miRNAs.txt` (for miRNA file) and `sorted_aligned_pre-miRNAs.gff3` (for gff file) by selection of files using Choose miRNA Files and Choose gff file option in the App (Fig. 9).

16. Select sequentially (in the exact order) columns of interest (mouse click into column), containing: chromosomes, start position, end position and orientation and click the button `Generate!` to create the intersection table.
17. Upon the completion of the analysis, the intersection table will automatically appear and can be downloaded in multiple file formats. The table reports all pre-miRNAs (*MIR* loci) which overlap with location of already annotated pre-miRNAs/*MIR* loci, and thus correspond to known *MIR* loci.
18. Compare the mature miRNA (highest expressed read) and their complementary miRNA sequences predicted by ShortStack (see the files `mature_miRNAs.fasta` and `star_miRNAs.fasta` at `Potato_sRNA_analysis/output` location, if output not defined otherwise in the step 10) of overlapped pre-miRNAs (*MIR* loci) considered as already annotated ones (from the previous step), with canonical potato miRNAs from miRBase (given in file: `unique_stu_mature.fasta`) produced from the same pre-miRNA (*MIR* loci). If the miRNA sequences are not identical, obtained mature miRNA and/or complementary miRNAs predicted by ShortStack can be considered as isomiRs. For their nomenclature see example at Table 1.
19. From `pre-miRNAs.fasta` file, select (copy) all novel pre-miRNA sequences using filtering option in spreadsheet software, i.e. ones showing no genome overlap with annotated pre-miRNAs and save the file as `novel_pre-miRNAs.fasta` into `Novel_miRNA_analysis` subdirectory.
20. Combine extracted pre-miRNA sequences from the previous step with sequences of all annotated pre-miRNAs from miRBase (`stu_pre-miRNAs.fasta`, already downloaded in section 3.2.3., step 3) into a single fasta file within `Novel_miRNA_analysis` subdirectory:

```
$ cat novel_pre-miRNAs.fasta stu_pre-miRNAs.fasta > combined_pre-  
miRNAs.fasta
```

The resulting file `combined_pre-miRNAs.fasta` will be used for classification of pre-miRNAs (and their encoding miRNAs) into known or novel miRNA families.

21. Go to the CD-HIT-EST clustering program available at [http://weizhonglab.ucsd.edu/cdhit\\_suite/cgi-bin/index.cgi?cmd=cd-hit-est](http://weizhonglab.ucsd.edu/cdhit_suite/cgi-bin/index.cgi?cmd=cd-hit-est) (or install it locally according to instructions at <https://github.com/weizhongli/cdhit>) (42), import `combined_pre-miRNAs.fasta` file from the previous step and run the program using default parameters and an identity threshold of 0.8.
22. Examine the resulting CD-HIT\_EST Sorted cluster file. Group the sequences showing similarities with annotated pre-miRNAs into corresponding known miRNA families (Fig. 10 A), and sequences that do not show similarity into novel miRNA families (Fig. 10 B).

“[Fig. 10 near here]”

23. You then similarly cluster miRNA sequences using CD-HIT-EST clustering program or compare their sequences with annotated miRNAs and pre-miRNAs using BLASTN within miRBase (<http://www.mirbase.org/search.shtml>) to make sure that classification into families was properly performed.
24. Apply proper nomenclature to non-annotated miRNAs and to newly identified *MIR* loci/pre-miRNAs according to miRNA annotation guidelines (see Table 1) (5, 6). Non-overlapped pre-miRNAs (from the step 19) grouped into the same known miRNA family and encoded at different

genomic loci, are usually producing identical or very similar miRNAs, hence they are regarded as newly identified paralogous *MIR* loci. Paralogous *MIR* loci/pre-miRNAs are assigned the same (family) number with sequential alphabetical suffixes (e.g. *MIR156a*, *MIR156b*, etc.), and corresponding canonical miRNAs are named as, e.g. miR156a, miR156b, respectively (5, 6). For miRNAs produced from the same pre-miRNA precursor (i.e. from one specific *MIR* loci), position on hairpin (5' arm or the 3' arm) need to be additionally defined to finalize their nomenclature (e.g. miR156a-3p, miR156a-5p) (5, 6).

25. To define the position of ShortStack's predicted miRNAs on their pre-miRNAs (5' arm or the 3' arm) run in-house `group_miRNA_sequences.Rmd` script (already downloaded in section 3.2.1., step 2). Prior to running the script ensure that you move (or copy) your ShortStack's result subdirectory `MIRNAs` within the `/Potato_sRNA_analysis/input/` (or redefine input/output directories by changing the values of `inputFolder` and `outputFolder` variables at the beginning of the script, i.e. relative paths defined as `inputFolder = "../input/MIRNAs"` and `outputFolder = "../output/"`).

Run the script as follows:

```
$ mv /pathTo_ShortStack_output/MIRNAs/ ../Potato_sRNA_analysis/input/
$ Rscript -e
"rmarkdown::render('../Potato_sRNA_analysis/scripts/group_miRNA_sequences.Rmd') "
```

The script `group_miRNA_sequences.Rmd` will examine all ShortStack's predicted pre-miRNAs (given in `MIRNAs` subdirectory) and extract and group the miRNAs and their complementary sequences (Fig. 8) based on their position on the hairpin (5' arm or the 3' arm).

In the `Potato_sRNA_analysis/output` subdirectory (if you did not redefine `outputFolder`) examine resulting file `miRNA_5p-3p_grouping.txt`. miRNA sequences given in the 2<sup>nd</sup> column of the file originate from 5' arm of the precursor, and thus get '-5p' suffix in their name (e.g. miR156a-5p), whereas the miRNA sequences from the 3<sup>rd</sup> column originate from the 3' arm and get '-3p' suffix (e.g. miR156a-3p). If, for example, two miRNAs from the same family, but from the different *MIR* loci, are identical in sequence, we recommend using an unique miRNA identifier which involves the information of all possible *MIR* loci giving rise to specific identical miRNA (see example at Table 1) (28), since we cannot be certain from which of possible *MIR* loci (e.g. given in miRBase and/or predicted by the miRNA prediction tools) the majority of this specific miRNA originate. In addition, having the unique identifier for miRNA sequences will simplify the counting of miRNA abundance and differential expression analysis.

26. Prepare a FASTA file with all novel miRNA sequences (e.g. `novel_miRNAs.fasta`) having unique headers to be used for counting and differential expression analysis. For this, one can use any text editor.

Example:

```
>miR403a-3p
```

```
CTAGATTACGACAAACTC
```

'miR403' in the header of the FASTA file corresponds to miRNA family name obtained from CD-HIT-EST clustering program (step 22), 'a' denotes *MIR* loci obtained from `MIR_loci_overlaps.R` script output (step 17) and '-3p' suffix denotes 3' arm of precursor obtained from `group_miRNA_sequences.Rmd` script output (step 25).



Save the file as `novel_miRNAs.fasta` into `Novel_miRNA_analysis` subdirectory.

If you would like to count the abundance and perform differential expression analysis only on novel miRNAs, see section 3.2.1, step 5 and section 3.2.6.

#### 3.2.4. Identification of miRNA variants (i.e. isomiRs)

1. If you followed novel miRNA prediction steps by ShortStack and run our in-house script `group_miRNA_sequences.Rmd` (previous section) you can find the isomiRs in the `isomiRs.txt` file at `/Potato_sRNA_analysis/output/` location. Note that isomiRs generated from the same pre-miRNA share the same cluster name (default naming by ShortStack) and differ by numeric suffix preceded by a dot (e.g. `Cluster_1.1`, `Cluster_1.2`, etc.).
2. Replace the ‘Cluster’ names of isomiRs with proper isomiR names according to miRNA annotation guidelines (see Table 1, *see* **Note 6**). This can be done using any spreadsheet application.
3. If you did not follow the novel miRNA prediction steps by ShortStack and you did not run `group_miRNA_sequences.Rmd` script, or you want to identify additional miRNA variants of known and novel miRNAs from sRNA sequencing data, download the `isomiRID.py` script and the configuration file `Config.txt`, from the <https://github.com/lfelipedeoliveira/isomiRID> (43). Note, one can also use `git clone` command to download both files at once. Run the following commands within your working directory:

```
$ cd ..
```

```
$ git clone https://github.com/lfelipedeoliveira/isomiRID.git
$ cd isomiRID
$ chmod 755 *
```

4. Within isomiRID directory, open the Config.txt file in text editor and define the full paths of :

- Pre-processed sRNA sequencing sample(s)

```
lib: /Path_to_working_directory/Pre-
processing/sRNAs_unmapped.fasta sRNA_sample1 fa
```

Besides path to the sample (e.g. /Path\_to\_working\_directory/Pre-processing/sRNAs\_unmapped.fasta), you need to define sample name that will be used in the final table (e.g. sRNA\_sample1), and the format of the sample (fa for .fasta and fq for .fastq). In case of multiple samples, write information one per line.

- known and novel pre-miRNA sequences (merged in one fasta file)

```
main_ref:
/Path_to_working_directory/Novel_miRNA_analysis/combined_pre-
miRNAs.fasta yes
```

- reference genome

```
filter_ref:
/Path_to_working_directory/Novel_miRNA_analysis/genome.fasta
yes
```

- known and novel miRNA sequences (merged in one fasta file here named as known\_novel\_miRNAs.fasta)

known\_miRNAs:

/Path\_to\_working\_directory/isomiRID/known\_novel\_miRNAs.fasta

To prepare known\_novel\_miRNAs.fasta use the following commands within

isomiRID subdirectory:

```
$ cat ../Potato_sRNA_analysis/output/known_miRNAs.fasta >
```

```
known_novel_miRNAs.fasta
```

```
$ cat ../Novel_miRNA_analysis/novel_miRNAs.fasta >>
```

```
known_novel_miRNAs.fasta
```

- **Bowtie**

bowtie\_path: /usr/bin/bowtie (if you followed exact rules written at software package homepage, otherwise use command `whereis bowtie` to obtain a proper path)

- **Bowtie-build:**

bowtie-build\_path: /usr/bin/bowtie-build (if you followed exact rules written at software package homepage, otherwise use command `whereis bowtie-build` to obtain a proper path)

Check that all input fasta files listed in `Config.txt` contain Ts instead of Us, otherwise consider using the command give in **Note 1**.

## 5. Define additional parameters in the `Config.txt` file:

- To search only for templated isomiRs (Fig. 3) set a double "no" separate by a space:

M3: no no

M5: no no

If you want to identify also non-templated isomiRs see **Note 7**.

- Set the size range (min and max) of reads that should be considered within this analysis.

RangeSize: 18 26

-Set the cutoff of read abundance (amount of reads that must be present in sRNA sequencing sample)

cutoff: 30

If you are analyzing two or more files, the pipeline will report only isomiRs whose sum of the reads in all samples exceeds 30 (e.g. (sample1+sample 2+sample 3) > 30 raw reads).

6. Run `isomiRID.py` script within the `isomiRID` directory. Make sure to keep the `Config.txt` with the original name and in the same directory as `isomiRID.py` script file.

```
$ python2 isomiRID.py
```

Script `isomiRID.py` will create `./Results` subdirectory.

7. Position to the `Results/r0/Cutoff` subdirectory to obtain the `r0-Cutoff30.txt` file (see example file within `Potato_sRNA_analysis/output/` directory under the name `TestFile_r0-Cutoff.txt`). It contains the list of predicted isomiRs in the 1<sup>st</sup> column (`OriginalSeq`) and link to the corresponding precursor(s) name in the 4<sup>th</sup> column (`preMIRref`).
8. Compare the sequences of predicted isomiRs with canonical known and novel miRNAs identified in previous sections. One can use Vertical LOOKUP function built in spreadsheet application of choice for this step.
9. To make further analysis less complex, extract only non-matched (i.e. novel) sequences.

10. Apply the nomenclature to detected isomiRs from the previous step (see Table 1 for help). One can use spreadsheet application for this step.

Example of non-matched (i.e. novel isomiR sequence):

TCGATAAACCTCTGCATCCAGC from `TestFile_r0-Cutoff.txt` file should be renamed to `stu-miR162a,b.1-5p`

11. Prepare a single fasta file with isomiR sequences (e.g. `isomiRs.fasta`) to be used for counting and differential expression analysis and save the file. Copy names and sequences from the spreadsheet and format them in fasta format (see example below).

Example:

```
>miR162a,b.1-5p
TCGATAAACCTCTGCATCCAGC
```

If you would like to count the abundance and perform differential expression analysis only on isomiRs, see section 3.2.1., step 5 and section 3.2.6.

### 3.2.5. Identification of *PHAS* loci and phasiRNAs

To identify *PHAS* loci and phasiRNAs, map sRNA reads to reference genome or transcriptome sequences (latter only to detect protein-coding *PHAS* loci) using ShortStack. Note, if you have already run the ShortStack pipeline for novel miRNA prediction (section 3.2.2), you can filter out the *PHAS* loci from `Results.txt` file based on the PhaseScore value (Phasing score for a phase size of 21 or 24 nts). Higher phasing scores indicate stronger phasing signature, with phase scores ranging from very near 0 (worst) up (7). You can define the cut-off for the PhaseScore depending on your biological question, for very strict identification of *PHAS* loci PhaseScore > 30 is recommended (7).

1. First create a new subdirectory (e.g. `phasiRNA_analysis`) within your working directory which will help you to keep files organized. Position (`cd ..`) to working directory and create the subdirectory using the following command:

```
$ mkdir phasiRNA_analysis
```

2. Move the pre-processed sRNA sequencing file `sRNAs_unmapped.fasta` and potato genome sequences (file: `genome.fasta`) into `phasiRNA_analysis` subdirectory:

```
$ mv ../Pre-processing/sRNAs_unmapped.fasta .
```

```
$ mv ../Novel_miRNA_analysis/genome.fasta .
```

3. Within `phasiRNA_analysis` directory run the ShortStack. To detect all non-coding and protein coding *PHAS* loci use potato genome as genomefile:

```
$ ShortStack --readfile sRNAs_unmapped.fasta \  
--genomefile genome.fasta \  
--mincov 5 --bowtie_cores 10 \  
--sort_mem 10G --nohp
```

4. To detect only protein-coding *PHAS* loci use potato transcriptome as genomefile. Download the potato coding sequences (CDS) from [github.com/NIB-SI/\\_p\\_stRT\\_Merged-PhurejaDM-geneModels](https://github.com/NIB-SI/_p_stRT_Merged-PhurejaDM-geneModels) directory (**44**), save the file as `transcriptome.fasta` into `phasiRNA_analysis` subdirectory:

```
$ wget https://github.com/NIB-SI/_p_stRT/raw/master/Merged-PhurejaDM-  
geneModels/StPGSC4.04n_seq_3_PGSC-cds-rep_ITAG-cds.fasta.zip
```

```
$ unzip StPGSC4.04n_seq_3_PGSC-cds-rep_ITAG-cds.fasta.zip
$ mv StPGSC4.04n_seq_3_PGSC-cds-rep_ITAG-cds.fasta transcriptome.fasta
```

## 5. Run the ShortStack:

```
$ ShortStack --readfile sRNAs_unmapped.fasta \
--genomefile transcriptome.fasta \
--mincov 5 --bowtie_cores 10 \
--sort_mem 10G --nohp
```

Usage of `--nohp` argument will skip the miRNA prediction step, for the meaning of the other parameters see section 3.2.2., step 6. Note that ShortStack's output (`Results.txt`) allows only examination of sRNA coverage at predicted *PHAS* loci in an analyzed sample or comparison of the coverage between the multiple samples, however, it does not report sequences of phasiRNAs. For this purpose, an additional tool should be used (i.e. *unitas*) (*17*).

- Unitas phasiRNA prediction pipeline takes only read alignment files in SAM format as input. Therefore, you should first convert the alignment BAM file produced by ShortStack (within ShortStack output directory) into a SAM format file using `samtools view` (SAMtools) command as follows:

```
$ samtools view -h ./ShortStack_output_directory/aligned_sRNAs.bam >
aligned_sRNAs.sam
```

- Set executable permissions for the *unitas* (*17*) perl script :

```
$ chmod 755 * /pathTo/units/*pl
```

where `/pathTo/` denotes the full or relative path to the directory from which `units` is available.

8. Then run the `units` as follows:

```
$ perl /pathTo/units/units_1.5.3.pl -skip_mapping -input  
aligned_sRNAs.sam -phasi 21 -phasi24 -species solanum_tuberosum
```

The script looks for 21 and 24 nt phasing patterns (argument `-phasi`) within the `aligned_sRNAs.sam` file. Run `$ perl /pathTo/units/units_1.5.3.pl --help` and `$ perl /pathTo/units/units_1.5.3.pl -show_options` for more options.

9. Examine the resulting files within `units` directory. Predicted `phasiRNA` sequences are reported in the resulting file `units.phasiRNA.sorted.fas`. They are sorted according to their *PHAS* loci of origin. Note, that in this file `phasiRNA` sequences may occur multiple times when assigned to more than one *PHAS* loci. A resulting file, named `units.phasiRNA.align`, contains the information of predicted *PHAS* loci (genome location, sRNA coverage) as well as visualization of aligned `phasiRNAs`.

10. Move `phasiRNAs` from `units.phasiRNA.sorted.fas` into a new fasta file (e.g. `phasiRNAs.fasta`) and save the file into `phasiRNA_analysis` subdirectory. This file will be used for obtaining `phasiRNAs` counts and differential expression analysis (section 3.2.6).



```
$ cp /pathTo/units/UNITAS_dd-mm-  
yy_filename_#X/units.phasiRNA.sorted.fas ./phasiRNAs.fasta
```

Note that UNITAS\_dd-mm-yy\_filename\_#X represents output subdirectory created by unitas, which is named according to the current date, input file name and analysis run (e.g.: UNITAS\_01-01-2017\_input.fasta\_#1).

### 3.2.6 Differential expression analysis

1. Combine all identified known and novel potato miRNAs, isomiRs, and phasiRNAs from previous sections in one fasta file using the append command as follows within your working directory:

```
$ cd ..  
$ mkdir DEA  
$ cd DEA  
$ cat ../Potato_sRNA_analysis/output/unique_stu_mature.fasta >  
potato_sRNAs.fasta  
  
$ cat ../Novel_miRNA_analysis/novel_miRNAs.fasta >> potato_sRNAs.fasta  
  
$ cat ../isomiRID/isomiRs.fasta >> potato_sRNAs.fasta  
  
$ cat ../phasiRNA_analysis/phasiRNAs.fasta >> potato_sRNAs.fasta
```

2. Align sRNA reads from pre-processed sRNA sequencing sample(s) to all identified sRNAs in the potato\_sRNAs.fasta file and obtain their occurrences by executing sRNA\_counts.pl script (see section 3.2.1., step 5). Output of the sRNA\_counts.pl script (here noted as

`count_table.txt`, see **Note 3**) is the count table that is used as input file to perform differential expression analysis in R using limma-voom method (45).

3. Call R by typing `$ R` in terminal. Type `> quit()` to exit R at any moment.

4. Install the limma package (46):

```
> if (!requireNamespace("BiocManager", quietly = TRUE))  
  install.packages("BiocManager")  
> BiocManager::install("limma")  
> library(limma)
```

5. Install the edgeR package (47):

```
> if (!requireNamespace("BiocManager", quietly = TRUE))  
  install.packages("BiocManager")  
> BiocManager::install("edgeR")  
> library(edgeR)
```

6. Import count table (`count_table.txt` file from step 2, see example file within

`Potato_sRNA_analysis/output/` directory under the name

`TestFile_count_table.txt`) into object `x` using:

```
> x <-  
read.table("/path_to_the_file/count_table.txt", header=TRUE, row.names="sR  
NAs", sep="\t", stringsAsFactors=FALSE)
```

7. Define the groups:

```
> group <- factor(c(1,1,1,2,2,2))
```

In this example, two different groups (treated, control), each in three biological replicates, are defined. Note that defined grouping should always follow a sample sequence in the count table.

8. Convert `x` object into a `DGEList`-object using the `DGEList` function, and specify the library sizes of the samples.

```
> y <- DGEList(counts=x, group=group,  
lib.size=c(xxxxxxx, xxxxxxx, xxxxxxx, xxxxxxx, xxxxxxx, xxxxxxx) )
```

Make sure to replace `xxxxx` with library size numbers.

9. Convert raw counts from `DGEList` to CPM (counts per million).

```
> cpm <- cpm(y)
```

10. Remove sRNAs that are low expressed.

```
> keep.exprs <- rowSums(cpm>2) >=3
```

In this example, only sRNAs having a CPM value strictly higher than 2, in at least three samples, are kept.

11. Normalize filtered counts.

```
> y1 <- y[keep.exprs, , keep.lib.sizes=TRUE]
> y1 <- calcNormFactors(y1)
```

12. Create design matrix and define contrasts.

```
> design = cbind(treated = c(1,1,1,0,0,0), control = c(0,0,0,1,1,1))
> contrastMatrix = makeContrasts("treated-control", levels=design)
```

13. Transform data using voom **(48)**.

```
> v <- voom(y1, design, plot=TRUE)
```

14. Fit linear models

```
> fit <- lmFit(v, design)
> fit2 = contrasts.fit(fit, contrastMatrix)
> fit2 <- eBayes(fit2)
```

15. Extract results (treated-control comparison) into the table.

```
> Final_results <- topTable(fit2, coef=1, number=1000000)
> write.table(treated_control, file= "Final_results.txt", quote = FALSE,
sep="\t")
```

### 3.2.7 sRNA target prediction

We will use two pipelines for sRNA target prediction. First, we will describe *in silico* target prediction by psRNATarget:

1. Go to the 'psRNATarget: A Plant Small RNA Target Analysis Server' available at <http://plantgrn.noble.org/psRNATarget/> website and select 'Submit small RNAs and targets' tab.
2. Upload sRNA file containing potato sRNA sequences (`potato_sRNAs.fasta`).
3. Upload the potato transcript sequences (`transcriptome.fasta`) already downloaded in section 3.2.4., step 4 (44).
4. Choose scoring schema *Shema V2* (latest release) and modify the following parameters :
  - a) reduce the parameter *# of top targets* from 200 to 50 (# denotes number), as we do not expect that specific sRNAs would have such large number of targets.
  - b) Reduce the maximum *expectation* from 5 to 3 to capture targets with good complementary matching.
5. Click the button Upload & Submit
6. Upon completion of the analysis download the resulting file. The file contains the predicted targets for sRNAs and sRNAs mode of action (cleavage or translational inhibition) (see example on the Fig. 5).

sRNA targets can also be determined through experimental data, e.g. degradome analysis. For degradome analysis download Cleaveland pipeline from [sites.psu.edu/axtell/software/](http://sites.psu.edu/axtell/software/). Make sure that Cleaveland and all its dependencies have been installed and added to your PATH (**Note 8**).

1. Download the raw degradome sequencing data, either from sequence service or from public database. If needed, trim the adapters and examine the quality of the reads by following the steps described in section 2.1. Create new directory `Target_prediction` and save degradome FASTA file in it:

```
$ cd ..  
$ mkdir Target_prediction  
$ cd Target_prediction
```

2. Run the `CleaveLand.pl` script:

```
$ perl /pathTo/CleaveLand.pl -e degradome.fasta \  
-u ../DEA/potato_sRNAs.fasta \  
-n ../phasiRNA_analysis/transcriptome.fasta \  
-o ./output_directory/ \  
-c 3 -t > degradome_analysis_results.txt
```

Three input files are required for the analysis:

- a) degradome sequencing data (path to the file is set by the argument `-e`
- b) list of sRNAs (path to the file is set by the argument `-u`
- c) transcript sequences (path to the file is set by the argument `-n`).

Maximum three categories are selected for reporting (from 0 to 3) by setting the `-c` argument to 3, whereas category 4 is excluded. Results are written in tabular format by adding the `-t` argument.

The results list the identified cut transcripts and the sRNA that is responsible for the degradation as well as the level of confidence given by the category (0 to 3).

## 4. Notes

1. To replace all uracils (Us) in fasta file (e.g. `ncRNAs.fasta`) with thymines (Ts), first install the `moreutils` and then run `sed` commands as follows:

```
$ apt-get install moreutils  
$ sed '/^[^>]/ y/uU/tT/' /Path_to_working_directory/Pre-  
processing/ncRNAs/ncRNAs.fasta | sponge /Path_to_working_directory/Pre-  
processing/ncRNAs/ncRNAs.fasta
```

2. The file `stu_mature.fasta` may contain the identical miRNA sequence which occur multiple times under different names (e.g. the identical sequence `TTGACAGAAGATAGAGAGCAC` correspond to `stu-miR156a-5p`, `stu-miR156b-5p`, `stu-miR156-c-5p` and `miR156-d-5p` annotated in `miRBase`).

3. To analyze multiple sRNA sequencing files with the script `sRNA_counts.pl` run this command:

```
$ ./sRNA_counts.pl -segtab ../input/unique_stu_mature.fasta -output  
../output/known_miRNAs_counts.txt ../input/file*
```

Of note, `file*` will take all the fasta files that are present in same directory as the script. The resulting count table will contain the list of sRNAs in rows and their counts of the analyzed sRNA sequencing samples in columns.

4. The following commands can be used to add ShortStack and its dependency programs to your PATH. Note that this would be temporary, meaning that it will work only for that particular bash session (will be lost upon closing the terminal). Make sure that you replace 'Path' with your own data destination.

```
$ export PATH=$PATH:/Path/ShortStack-3.8.5/
$ export PATH=$PATH:/Path/ViennaRNA-2.4.2/
$ export PATH=$PATH:/Path/bowtie-1.2.2-linux-x86_64/
$ export PATH=$PATH:/Path/samtools-1.6/
```

5. To run multiple files with ShortStack specify the fasta file names within the argument `--readfile`. Separate the file names or paths (if the files are not within the same directory) with commas.

```
$ ShortStack --readfile sRNA1.fasta,sRNA2.fasta,sRNA3.fasta \
--genomefile genome.fasta --mincov 5 --bowtie_cores 10 --sort_mem 10G
```

6. In some cases, isomiRs produced from specific *MIR* loci are identical to the canonical miRNAs produced from another *MIR* loci (e.g. paralogous loci). As we cannot be fully certain from which loci this particular miRNAs sequence originate, one can assign to miRNA an identifier which represents all possible origins. However, much easier is to keep only canonical miRNA identifier and check later on (i.e. upon differential expression analysis) whether miRNA can be produced as isomiR from another *MIR* loci.
7. If you want to identify non-templated 5' or 3' end isomiRs (Fig. 3) set within `Config.txt` file:



```
M3: yes 3
```

```
M5: yes 3
```

Since the non-templated isomiRs do not match with the pre-miRNA sequence, using this command the three nucleotides from 5' and 3' ends of sRNA are first removed before subsequent sRNA mapping on the pre-miRNA reference file.

8. The following commands can be used to add Cleaveland (`Cleaveland.pl` and `GSTAr.pl` scripts) and its dependency programs to your PATH. Note that this would be temporary, meaning that it will work only for your particular bash session (will be lost upon closing the terminal). Make sure that you replace `Path` with your own data destination.

```
export PATH=$PATH:/Path/CleaveLand/GSTAr_v1-0.pl
export PATH=$PATH:/Path/Cleaveland/CleaveLand.pl
export PATH=$PATH:/Path/ViennaRNA-2.4.2/
export PATH=$PATH:/Path/bowtie-1.2.2-linux-x86_64/
export PATH=$PATH:/Path/samtools-1.6/
export PATH=$PATH:/Path/R/
```

You also need to install perl module `Math::CDF`:

```
$ apt-get install build-essential
$ perl -MCPAN -e 'install Math::CDF '
```

## Acknowledgements

We thank Henrik Krnec for providing the script `sRNA_counts.pl`. The work was financed by the Slovenian Research Agency (research core funding No. P4-0165 and projects J4-7636 and J4-1777).

## 5. References

1. Pritchard CC, Cheng HH, and Tewari M (2012) MicroRNA profiling: Approaches and considerations. *Nat Rev Genet* 13:358–369
2. Bilichak A, Golubov A, and Kovalchuk I (2017) Small RNA Library Preparation and Illumina Sequencing in Plants, In: Kovalchuk, I. (ed.) *Plant Epigenetics: Methods and Protocols*, pp. 189–196 Springer US, Boston, MA
3. Shore S, Henderson JM, Lebedev A, et al (2016) Small RNA library preparation method for next-generation sequencing using chemical modifications to prevent adapter dimer formation. *PLoS One* 11:1–26
4. Kozomara A, Birgaoanu M, and Griffiths-Jones S (2019) MiRBase: From microRNA sequences to function. *Nucleic Acids Res* 47:155–162
5. Axtell MJ and Meyers BC (2018) Revisiting Criteria for Plant MicroRNA Annotation in the Era of

6. Meyers BC, Axtell MJ, Bartel B, et al (2008) Criteria for annotation of plant MicroRNAs. *Plant Cell* 20:3186–3190
7. Axtell MJ (2013) ShortStack : Comprehensive annotation and quantification of small RNA genes. *19:740–751*
8. Budak H, Bulut R, Kantar M, et al (2016) MicroRNA nomenclature and the need for a revised naming prescription. *Brief Funct Genomics* 15:65–71
9. Ambros V, Bartel B, Bartel DP, et al (2003) A uniform system for microRNA annotation. *9:277–279*
10. Kozomara A and Griffiths-Jones S (2014) miRBase : annotating high confidence microRNAs using deep sequencing data. *42:68–73*
11. Budak H and Akpinar BA (2015) Plant miRNAs: biogenesis, organization and origins. *Funct Integr Genomics* 15:523–531
12. Debat HJ and Ducasse D a. (2014) Plant microRNAs: Recent Advances and Future Challenges. *Plant Mol Biol Report* 32:1257–1269
13. Lu C, Lu C, Kulkarni K, et al (2006) MicroRNAs and other small RNAs enriched in the Arabidopsis RNA-dependent RNA polymerase-2 mutant. *Genome Res* 16:1276–1288
14. Chen H-M, Chen L-T, Patel K, et al (2010) 22-Nucleotide RNAs trigger secondary siRNA biogenesis in plants. *Proc Natl Acad Sci U S A* 107:15269–15274
15. Cuperus JT, Carbonell A, Fahlgren N, et al (2010) Unique functionality of 22-nt miRNAs in triggering RDR6-dependent siRNA biogenesis from target transcripts in Arabidopsis. *Nat Struct Mol Biol* 17:997–1003

16. Kasschau KD, Fahlgren N, Chapman EJ, et al (2007) Genome-wide profiling and analysis of Arabidopsis siRNAs. *PLoS Biol* 5:0479–0493
17. Gebert D, Hewel C, and Rosenkranz D (2017) unitas : the universal tool for annotation of small RNAs. 18:1–14
18. Chen H-M, Li Y-H, and Wu S-H (2007) Bioinformatic prediction and experimental validation of a microRNA-directed tandem trans-acting siRNA cascade in Arabidopsis. 104:3318–3323
19. Morgado L and Johannes F (2017) Computational tools for plant small RNA detection and categorization. *Brief Bioinform* 20:1–12
20. Allen E, Xie Z, Gustafson AM, et al (2005) microRNA-Directed Phasing during Trans-Acting siRNA Biogenesis in Plants. *Cell* 121:207–221
21. Lin Y, Lin L, Lai R, et al (2015) MicroRNA390-Directed TAS3 Cleavage Leads to the Production of tasiRNA-ARF3/4 During Somatic Embryogenesis in *Dimocarpus longan* Lour. *Front Plant Sci* 6:1–15
22. Allen E and Howell MD (2010) miRNAs in the biogenesis of trans-acting siRNAs in higher plants. *Semin Cell Dev Biol* 21:798–804
23. Djami-Tchatchou AT, Sanan-Mishra N, Ntushelo K, et al (2017) Functional Roles of microRNAs in Agronomically Important Plants—Potential as Targets for Crop Improvement and Protection. *Front Plant Sci* 8:1–24
24. Srivastava PK, Moturu TR, Pandey P, et al (2014) A comparison of performance of plant miRNA target prediction tools and the characterization of features for genome-wide target prediction. *BMC Genomics* 15:1–15
25. Ding J, Zhou S, and Guan J (2012) Finding MicroRNA Targets in Plants: Current Status and Perspectives. *Genomics, Proteomics Bioinforma* 10:264–275

26. Fahlgren N and Carrington JC (2010) miRNA Target Prediction in Plants, In: Meyers, B.C. and Green, P.J. (eds.) Plant MicroRNAs, Methods in Molecular Biology, pp. 51–57 Humana Press
27. Xiao B, Yang X, Ye C-Y, et al (2014) A diverse set of miRNAs responsive to begomovirus-associated betasatellite in *Nicotiana benthamiana*. BMC Plant Biol 14:1–9
28. Križnik M, Petek M, Dobnik D, et al (2017) Salicylic Acid Perturbs sRNA-Gibberellin Regulatory Network in Immune Response of Potato to Potato virus Y Infection. 8:1–14
29. Moyo L, Ramesh S V., Kappagantu M, et al (2017) The effects of potato virus Y-derived virus small interfering RNAs of three biologically distinct strains on potato (*Solanum tuberosum*) transcriptome. Virol J 14:1–17
30. Li S, Le B, Ma X, et al (2016) Biogenesis of phased siRNAs on membrane-bound polysomes in *Arabidopsis*. Elife 5:1–24
31. Kertesz M, Iovino N, Unnerstall U, et al (2007) The role of site accessibility in microRNA target recognition. Nat Genet 39:1278–1284
32. Dai X and Zhao PX (2011) PsRNATarget: A plant small RNA target analysis server. Nucleic Acids Res 39:1–5
33. Mückstein U, Tafer H, Hackermüller J, et al (2006) Thermodynamics of RNA-RNA binding. 22:1177–1182
34. Dai X, Zhuang Z, and Zhao PX (2011) Computational analysis of miRNA targets in plants: Current status and challenges. Brief Bioinform 12:115–121
35. Huang JC, Morris QD, and Frey BJ (2007) Bayesian Inference of MicroRNA Targets from Sequence and Expression Data. J Comput Biol 14:550–563
36. Addo-Quaye C, Eshoo TW, Bartel DP, et al (2008) Endogenous siRNA and miRNA Targets Identified by Sequencing of the *Arabidopsis* Degradome. Curr Biol 18:758–762

37. German M a, Luo S, Schroth G, et al (2009) Construction of Parallel Analysis of RNA Ends (PARE) libraries for the study of cleaved miRNA targets and the RNA degradome. *Nat Protoc* 4:356–362
38. Llave C (2002) Endogenous and Silencing-Associated Small RNAs in Plants. *Plant Cell Online* 14:1605–1619
39. Addo-Quaye C, Miller W, and Axtell MJ (2009) CleaveLand: A pipeline for using degradome data to find cleaved small RNA targets. *25:130–131*
40. Folkes L, Moxon S, Woolfenden HC, et al (2012) PAREsnip: A tool for rapid genome-wide discovery of small RNA/target interactions evidenced through degradome sequencing. *Nucleic Acids Res* 40:1–10
41. Zheng Y, Li YF, Sunkar R, et al (2012) SeqTar: An effective method for identifying microRNA guided cleavage sites from degradome of polyadenylated transcripts in plants. *Nucleic Acids Res* 40:1–18
42. Huang Y, Niu B, Gao Y, et al (2010) CD-HIT Suite: A web server for clustering and comparing biological sequences. *26:680–682*
43. Felipe L, Oliveira V De, Christoff AP, et al (2013) isomiRID : a framework to identify microRNA isoforms. *29:2521–2523*
44. Petek M, Zagorščak M, Ramšak Ž, et al (2020) Cultivar-specific transcriptome and pan-transcriptome reconstruction of tetraploid potato. *Sci Data* 7:1–15
45. Law CW, Alhamdoosh M, Su S, et al (2018) RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR. *5:1–29*
46. Ritchie ME, Phipson B, Wu D, et al (2015) Limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res* 43:1–13

47. Robinson MD, McCarthy DJ, and Smyth GK (2009) edgeR: A Bioconductor package for differential expression analysis of digital gene expression data. 26:139–140
48. Law CW, Chen Y, Smyth GK, et al (2014) voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. Genome Biol 15:1–17

## Figure Captions

Fig. 1: Schematic illustration of sRNA library preparation for Illumina platform. Isolated sRNAs are ligated to the 3'-preadenylated adapter and the 5'-adapter. The ligated products are converted to cDNA and PCR-amplified. Finally, libraries are purified to isolate tagged library with a sRNA insert and subjected to high-throughput sequencing.

Fig. 2: Difference in the prediction scheme for miRNAs and phasiRNAs. miRNA identification (left panel) is primarily based on the prediction of the classical hairpin structure of a pre-miRNA yielding one highly abundant mature miRNA (depicted in red), and less abundant complementary miRNA (depicted in blue). phasiRNA identification (right panel) relies on the detection of a phased pattern of mapped short reads (depicted in green).

Fig. 3: Overview of small RNA nomenclature. Schematic illustration of miR156 family members originating from two paralogous *MIR* loci is given. Two paralogous *MIR* loci, *MIR156a* and *MIR156b* are encoding two hairpin precursors, of which each precursor is processed by DICER (DCL) enzymes (predominantly DCL1) giving rise to two canonical miRNAs, one generated from the 3' arm and the second one from the 5' arm of the precursor. Canonical miRNAs from *MIR156a* and *MIR156b* precursors can differ by few nucleotides or share identical sequence. In this given example, 5' arm canonical miRNAs, miR156a-5p and miR156b-5p, have an identical sequence UUGACAGAAGAUAGAGAGCAC (<http://www.mirbase.org/>). The 3' arm canonical miRNAs have not been annotated yet. In addition to canonical miRNAs, their sequence variants (termed as isomiRs) can originate from the same precursor. These variants are either templated with shifted start and end positions produced by alternative DCL1 cleavage or non-templated, emerging through nucleotide additions, removals or modifications by other enzymes. Pol II - RNA polymerase II.

Fig. 4: Schematic illustration of phasiRNA biosynthesis and their naming. The biogenesis of phasiRNAs is initially triggered by miRNAs that direct the cleavage of a single-stranded protein-coding or non-coding



transcript, resulting in 5' or 3' cleavage fragments, of which one is degraded while the other is used as a template for synthesis of complementary strands forming dsRNA which is further processed in phased intervals into phasiRNAs. The example shows phasiRNA produced from 5' cleavage fragment by phasing in 5' direction. For phasiRNAs, nomenclature is not established yet. The Fig shows the nomenclature introduced by Allen et al. (2005) in which the processing direction is noted by a 5' or 3' prefix; phasiRNA are sequentially named as D1, D2, D3 etc., of which phasiRNA, the nearest to the miRNA cleavage site gets the identifier D1, the orientation is indicated by adding the suffix (+) for the positive (original transcript) strand, or (–) for the synthesized negative strand.

Fig. 5: Example of *in silico* prediction of sRNA-mRNA target pairs obtained by psRNATarget tool.

Fig. 6: Schematic illustration of library preparation for degradome sequencing. Procedure takes advantage of the free 5'-monophosphate (5'-P) remaining on the 3' fragment after sRNA-mediated cleavage, to which a 5'-RNA adapter (depicted in blue) that includes a MmeI recognition site is ligated. Product is reverse-transcribed, slightly PCR amplified and cleaved with MmeI. Next, a 3'-adapter with degenerate nucleotides in the overhang region (depicted in purple) is ligated to the MmeI digestion products and the resulting material is amplified by PCR, purified and subjected to high-throughput sequencing.

Fig. 7: The computational workflow for analyzing sRNA sequencing data.

Fig. 8: Example of a predicted *MIR* loci by ShortStack tool. The location on the genome and the sequence of a pre-miRNA is given at the top. Below the sequence base-paired secondary structure is predicted in dot-bracket notation followed by derived more abundant miRNA and its complementary miRNA (miRNA-star) sequence. The additional sequences mapped to the same pre-miRNA represent the miRNA variants (isomiRs). The numbers on the right represent lengths and abundance of reads. l – length, a – number of reads for mapping sequence.

Fig. 9: Shiny app (used) to obtain overlapping genome locations between *MIR* loci predicted by ShortStack and annotated potato *MIR* loci.

Fig. 10: Examples of the CD-HIT-EST clustering result. A) pre-miRNA1 clustered with known miRNA family miR9471 identified in tomato (sly-Solanum lycopersicum). According to miRBase, two paralogous loci *MIR9471a* and *MIR9471b* have been so far identified only in tomato, but none in potato. Therefore the name of this pre-miRNA (and *MIR* locus) in potato would be *MIR9471a*. B) pre-miRNA2 did not cluster with any annotated miRNA family, thus it is grouped into novel miRNA family. After the submission to the miRBase registry, it will get the also the new family name. \*- representative sequence from the cluster, + - positive strand was used for clustering, %- the identity between the sequence and the representative.

**Table Captions**

Table 1: Plant miRNA nomenclature.

Table 2: Example for the preparation of the file locations\_predicted\_pre-miRNAs.txt.

## Tables

Table 1

Naming convention	Explanation	Example
miR	'miR' prefix designates mature miRNA	miR156
<i>MIR</i>	Capitalized 'MIR' prefix used to designate <i>MIR</i> loci/pre-miRNA	<i>MIR156</i>
miR <u>XXX</u> , <i>MIR</i> <u>XXX</u>	Number after 'miR/ <i>MIR</i> ' prefixes designate the miRNA family	miR156 family
a, b, c, etc.	Characters after miRNA family identifier denotes paralogous <i>MIR</i> loci and their corresponding miRNAs which sequences are identical or very similar	For <i>MIR</i> loci: <i>MIR156a</i> , <i>MIR156b</i> For miRNAs: miR156a, miR156b
-3p or -5p	Suffix to differentiate among miRNA originating from the 3' or 5' end of the same pre-miRNA, respectively	miR156a-3p, miR156a-5p, miR156b-3p, miR156b-5p
.1, .2, .3, .4	Suffix to differentiate among different isomiRs originating from the same pre-miRNA	miR156a-3p.1, miR156a-5p.1, miR156a-3p.2, miR156a-5p.2
miRXXX <u>a,b,c-5p</u>	Identical sequences originating from two or more <i>MIR</i> loci are given unique miRNA name/identifier which contains information of all possible <i>MIR</i> loci giving rise to identical miRNA	miR156a,b,c-5p miR156d-j-5p

Table 2

Chromosome	Start_position	End_position	miRNA	Orientation
chr00	17410704	17410824	miRNA1	+
chr01	731598	731720	miRNA2	+

5'-P **sRNA** 3'-OH







↓ 3'-adapter ligation

5'-P  3'

↓ 5'-adapter ligation

5'   3' 

↓ reverse transcription

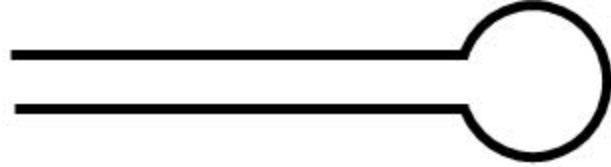
P5   **cDNA**   **index**  P7 

↓ PCR amplification

↓ purification

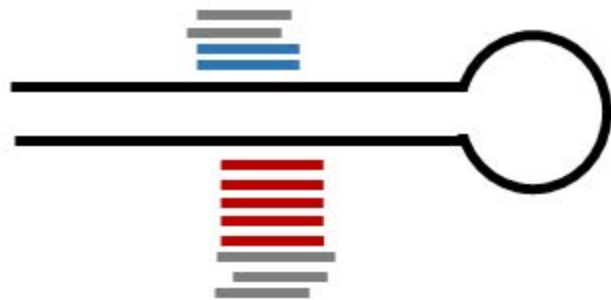
high-throughput sequencing



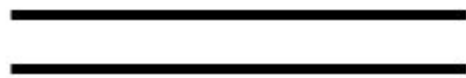
Single stranded precursor which folds into hairpin-like structure.



miRNAs



Processing of precursor yields one highly expressed miRNA (in red), and complementary miRNA (miRNA\*, in blue) and few miRNA variants (in grey).



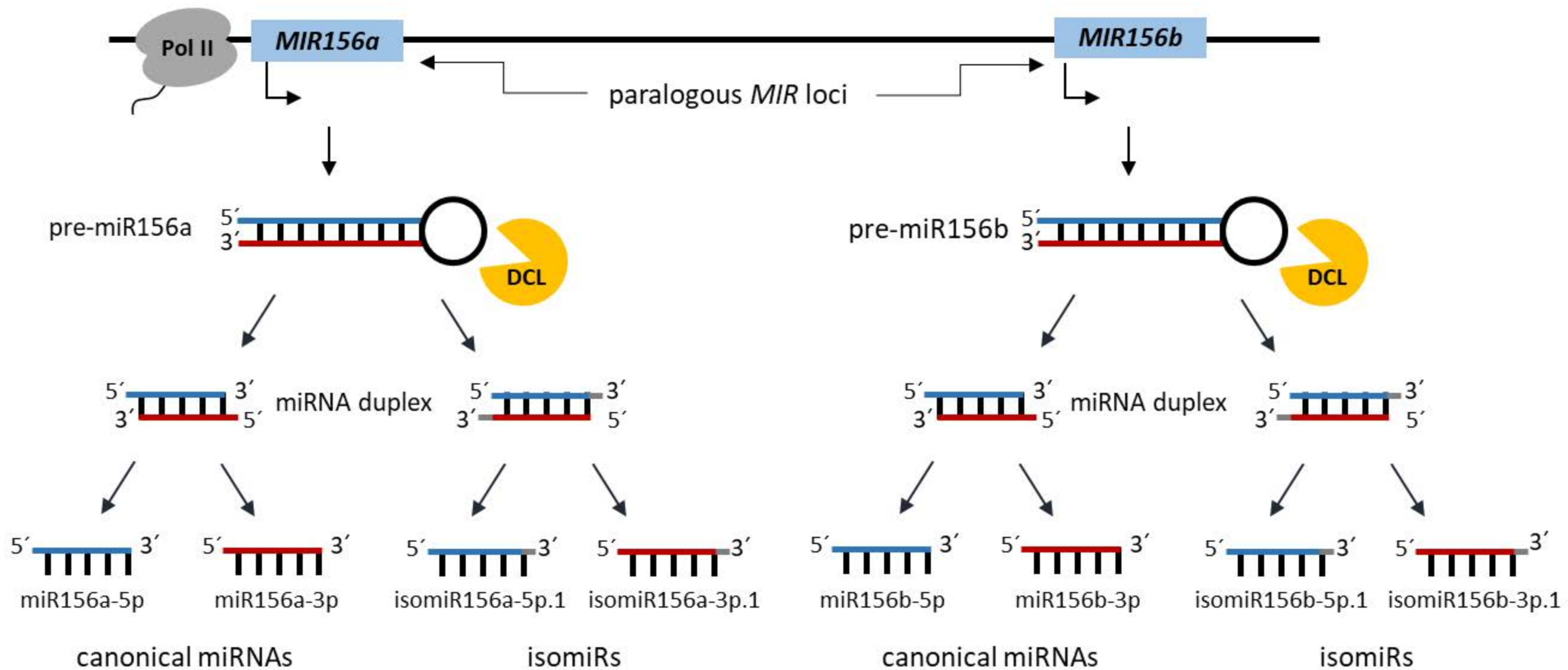
Double-stranded RNA precursor.

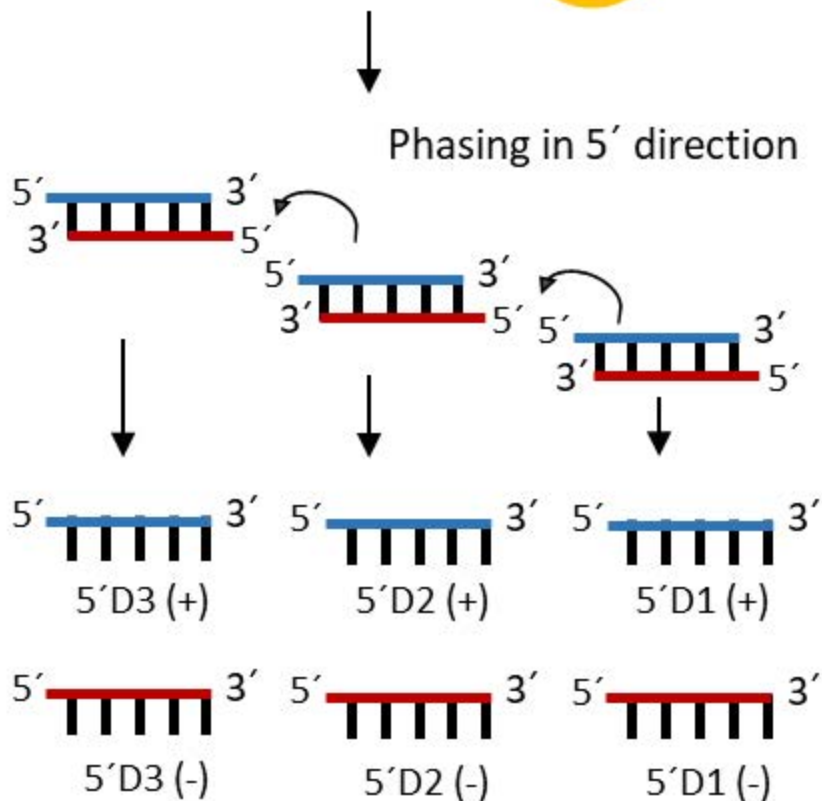
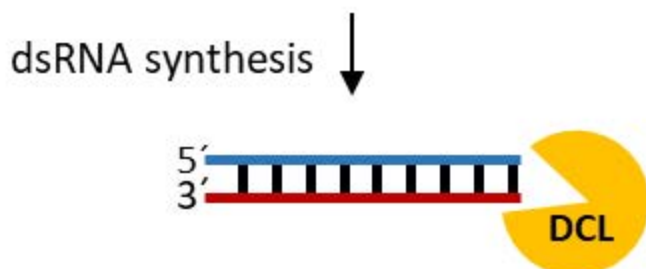
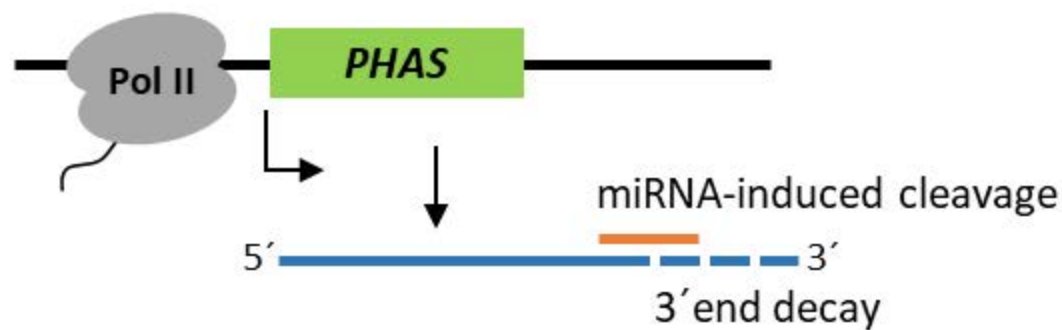


phasiRNAs



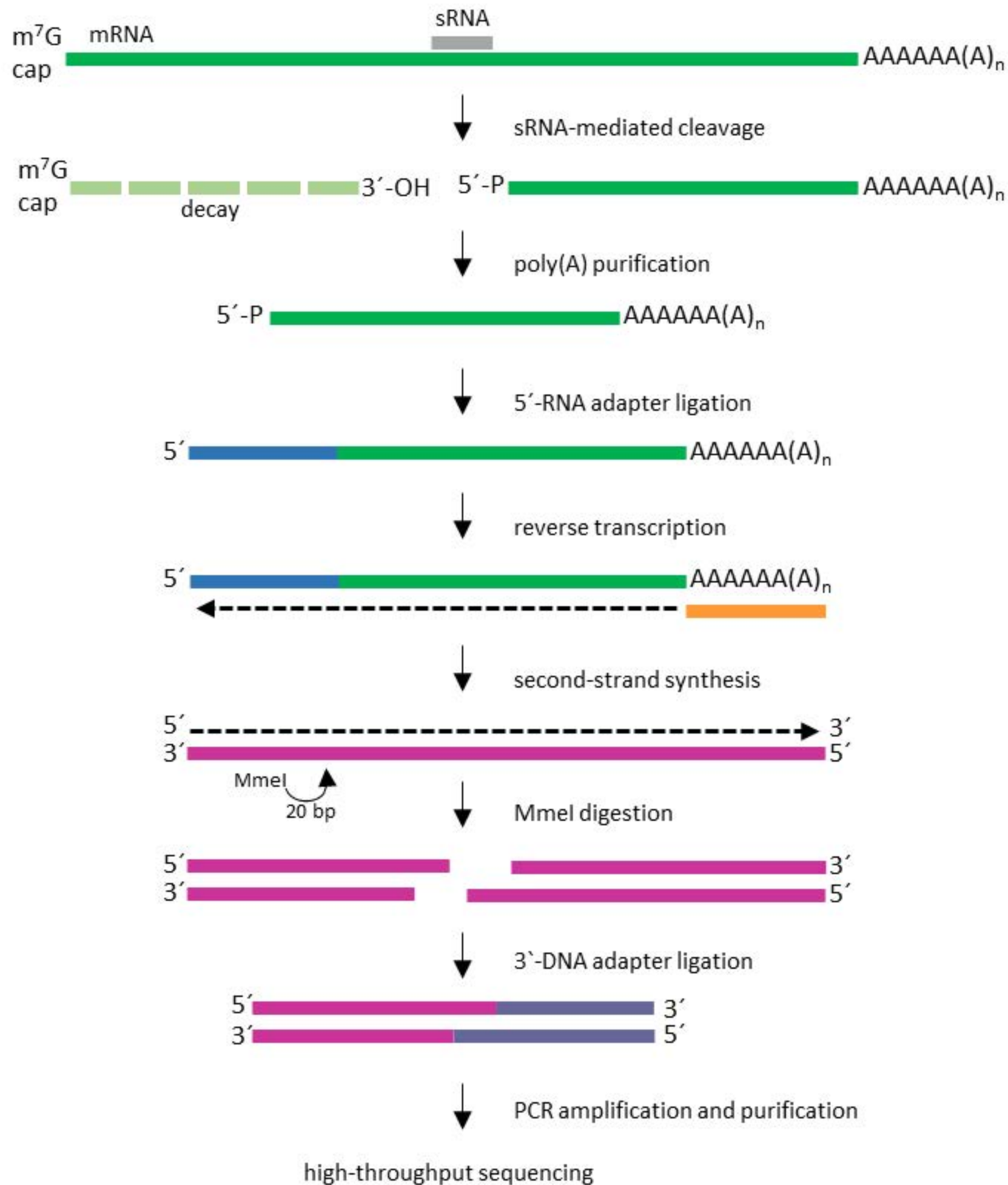
Precursor is sliced sequentially in regular intervals yielding a large number of phasiRNAs.





miRNA Acc.	Target Acc.	Expect	UPE	Alignment	Target Description	Inhibition	Multiplicity
miRNA	Sotub07g027430	1.0	9.227	<div> <div>miRNA</div> <div>21</div> <div>UCGUACACGGGACGAAGAGGU</div> <div>1</div> <div> <div>::::</div> <div>::::::::::::::::::::</div> </div> <div>Target</div> <div>668</div> <div>AGCAAGUGCCCUGCUUCUCCA</div> <div>688</div> </div>	NAC domain protein	Cleavage	1





## Pre-processing

Adapter removal  
& short reads filtering  
*cutadapt*

Quality check  
*FastQC*

Low-quality  
reads filtering  
*FASTX-Toolkit*

Low-complexity  
& invalid reads  
filtering  
*Filter Tool*

Contaminating reads  
filtering  
*Bowtie*

RNAcentral

## sRNA data analysis

### miRNA analysis

Known miRNAs  
identification  
*sRNA\_counts*

miRBase

Novel miRNAs  
identification  
*ShortStack*

genome

isomiRs  
identification  
*ShortStack, isomiRID*

genome

Known and novel  
MIR loci  
*MIR\_loci\_overlaps*

genome

### phasiRNA analysis

phasiRNA  
identification  
*unitas*

genome

transcriptome

PHAS loci  
identification  
*ShortStack, unitas*

genome

transcriptome

### Differential expression analysis

sRNA  
quantification  
*sRNA\_counts*

statistical analysis  
*limma-voom*

### Target prediction

*In silico* target  
prediction  
*psRNATarget*

transcriptome

Degradome-Seq  
analysis  
*Cleaveland*

transcriptome

MIR loci location: chr08:41408630-41408754 Strand: +

GUCACAUCUCCGUAGUCCUGUCGCAGAUGACUUUCGCCCUCUCAUUCAGCCCCCUUUUCUCUAAAUCAAUUAUUUGAUUUCAAUGUGGUAGGACGAGAGUCAUCUGUGACAGGAUAAUGCAAGAUCAAGUUUU

[illegible]

.....ACGAGAGUCAUCUGUGACAGG..... miRNA 1-21 a=667

.....UGUCGAGAUACUUUCGCC..... miRNA-star l=21 a=9

.....CGAGAGUCAUCUGUGACAGG..... 1=20 a=2

.....CGAGAGUCAUCUGUGACAGGA..... l=21 a=37

.....ACGAGAGUCAUCUGUGACAG..... 1=20 a=2

.....ACGAGAGUCAUCUGUGACAGGA..... 1=22 a=13

.....ACGAGAGUCAUCUGUGACAGGc..... 1=22 a=2

.....ACGAGAGUCAUCUGUGACAGG..... 1=22 a=3

.....ACGAGAGUCAUCUGUGACAaG..... l=21 a=1

.....ACGAGAcUCAUCUGUGACAGG..... 1=21 a=1

.....ACGAGcGUCaucUGUGACAGG..... 1=21 a=1

.....AaGAGAGUCAUCUGUGACAGG..... 1=21 a=3

.....dCGAGAGUCAUCUGUGACAGG..... 1=21 a=3

.....GACGAGAGUCAUCUGUGACA..... 1=20 a=1

.....GAGGAGAGTCACTCTGTGACAG..... [2] a=1

☒ My file contains header

Separator

- ☐ Comma
- ☐ Semicolon
- ☒ Tab

Quote

- ☒ None
- ☐ Double Quote
- ☐ Single Quote

Choose miRNA File

Browse...

No file selected

☐ My file contains header

Separator

- ☐ Comma
- ☐ Semicolon
- ☒ Tab

Quote

- ☒ None
- ☐ Double Quote
- ☐ Single Quote

Choose gff File

Browse...

No file selected

Generate!

Click the button to generate the intersection table and check results in the output tab panel

Tables

input

output

**Upload** files and **select columns** of interest, in order, containing: **chromosomes, start position, end position** and **orientation**

# A

```
>Cluster 1
0    148nt, >pre-miRNA1... at +/91.22%
1    240nt, >sly-MIR9471b... *
2    181nt, >sly-MIR9471a... at +/95.58%
```

# B

```
>Cluster 2
0    113nt, >pre-miRNA2... *
```