A user's manual for *shortran*

Released: 06-June-2012

Vikas Gupta and Stig U. Andersen
Bioinformatics Research Center and
Department of Molecular Biology and Genetics
Aarhus University, Denmark
Contact: vgupta@birc.au.dk, sua@mb.au.dk

1 OVERVIEW

1.1 Background

We found it complicated to combine small RNA (sRNA) profiling and annotation information using existing analysis tools. Here we describe the installation and use of *shortran,* which generates and combines expression profiling and annotation data from sRNA sequencing raw data. The output is provided as an easily accessible MySQL database format and provides graphical output to facilitate fast data analysis.

1.2 Summary of *shortran* modules

shortran is a combination of various modules, allowing a user to analyze data using either an interactive command line interface or a configuration file. A detailed description of the use of each module is provided in the application section of this manual. *shortran* is capable of performing filtering, read error correction, expression profiling, mapping, clustering analysis, miRNA/ta-siRNA prediction, pattern search, target prediction and annotation.

1.3 Implementation

Python is the primary language of the pipeline but Perl has been also used. The package contains multiple Python language scripts. *shortran.py* is the main, which invokes the other pipeline modules. *shortran* is developed using *python* (v2.7.2) and *perl* (v5.12.3), it has been tested on both Mac OS X (10.6.8) and Linux Ubuntu (10.04.3).

1.4 Availability

The *shortran* package including an example dataset is available at www.carb.dk/resources.asp.

2 Installation

The primary modules of the *shortran* package require no installation and can be invoked directly using python after the dependencies have been installed. To run *shortran*, change directory (cd) to the folder containing the *shortran.py* file and run the script:

```
> cd [path of shortran_folder]
> python shortran.py
```

The ">" indicates the command prompt and should not be typed. Brackets [] indicates text that needs to be adjusted. Brackets should not be typed. Comments are indicated by "#". To obtain the path of a file or folder for introducing it into a script, the icon of the respective item can be dragged into a termial window where the path will then be displayed.

Detailed instructions for installation and setup of core dependencies on MacOS and Linux platforms are included. Please refer to "install_osx.README" and "install_linux.README".

2.1 Python and Perl

Most operating systems have built-in Python and Perl and no installation is required. If that is not the case, Python can be downloaded from http://python.org/getit/, and Perl from http://www.perl.org/get.html. In addition, the Python libraries *numpy* (http://numpy.scipy.org/), *scipy* (http://www.scipy.org/) and *matplotlib* (http://matplotlib.sourceforge.net/) need to be installed.

2.2 Bowtie

Bowtie is required by a number of modules and should be installed and added to the system path (so that typing bowtie in the terminal activates the program regardless of the working directory). If installing on MacOS, Bowtie is installed and added to the path by the installation script. It can be found at http://bowtie-bio.sourceforge.net/index.shtml.

2.3 MySQL and setting up database

Installation of MySQL is optional but it is required for use of the *shortran* database functionalities. A detailed description on downloading and setting up MySQL can be found at http://www.mysql.com/.

MySQL setup for shortran is handled automatically by the MacOS setup script and can be set up on Linux systems using the enclosed MySQL setup script:

```
> sudo mysql < "mysql_setup.txt"
```

You can also set up MySQL manually by creating a database and giving permissions to your user(s):

```
> sudo mysql
mysql> CREATE DATABASE profiles;
mysql> CREATE USER 'user'@'localhost' IDENTIFIED BY '';
mysql> GRANT ALL PRIVILEGES ON profiles.* TO
  user'@'localhost';
```

This basic MySQL setup corresponds to the following settings in the configuration.txt file:

```
server='localhost'
user='user'
password=''
database='profiles'
```

2.4 ECHO

ECHO is used only if k-mer correction of sequencing reads is required. It can be downloaded from http://uc-echo.sourceforge.net/.

2.5 mirDeep

All the scripts required for mirDeep are supplied with the *shortran* package but its dependencies must be globally installed, see http://www.mdc-berlin.de/en/research/research_teams/systems_biology_of_gene_regulatory_elements/projects/miRDeep/. It is only required for module-4.

2.6 Blast

formatdb and *fastacmd* from the Blast package should be installed. A description can be found here:
http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/formatdb_fastacmd.html.

2.7 NiBLS

NiBLS (MacLean et al. BMC Bioinformatics. 2010 18;11:93) can optionally be used to detect small RNA loci instead of the simple cluster detection algorithm included with shortran. See https://github.com/danmaclean/NiBLS.

3 QUICK START GUIDE

3.1 Configuration

For the first run, we recommend that you open the "configuration.txt" file in a text editor and replace "/Users/test/Desktop/shortran_0.43/demo" with the path to the shortran demo folder on your system.

Open the terminal window and change to the shortran directory and run the program:

```
> cd [path_to_shortran_folder]
> python shortran.py
```

And hit "n" to run shortran according to the configuration file.

If all dependencies are installed correctly, your output folder should look like the "example_output" folder available for download, when the run is finished.

3.2 Explore the database

You can explore the MySQL database directly using the terminal window. For example, type:

```
> mysql -u user
mysql > show databases;
mysql > use profiles;
mysql > show tables;
mysql > describe `demo_2012-05-09_profile_19_24_sorted.cut-1_score_1`;
mysql > SELECT * FROM `demo_2012-05-09_profile_19_24_sorted.cut-1_score_1` LIMIT 10;
```

When logged into the database, you can experiment with the queries. For instance retrieve the 10 most highly abundant sRNA sequences with exon matches and sort them by abundance:

```
mysql > SELECT `#Sequence`,`sum_norm`,`genomic_region` FROM `demo_2012-05-09_profile_19_24_sorted.cut-1_score_1` WHERE `genomic_region` LIKE "%exonic%" ORDER BY `sum_norm` DESC LIMIT 10;
```

For further query examples, please see section 6-9 (Module 9)

4 CONFIGURATION

There are two ways of configuring the analysis. Either the file *configuration.txt* can be modified directly, or the interactive configuration interface can be used.

4.1 Configuration using the *configuration.txt* file

Parameters and variables can be defined during configuration. Here we describe the common parameters for the pipeline. Module-specific parameters will be explained in section 4 of the manual. We start by defining a name (i.e. date) for the folder as:

```
date = '[a string]'       # i.e. 2012-02-21
```

The range of small RNA lengths to be analyzed is defined by entering the minimum and maximum lengths:

```
min_seq_size= '[an integer]'          # i.e. 19
max_seq_size= '[an integer]'          # i.e. 24
```

Rare reads can be filtered by using a minimum threshold (*cutoff*), which removes all sequences which do not occur a minimum of *cutoff* times when summed across all libraries. Filtering using a variation score (*score_regulation_cutoff*) can be used to filter for sequences that show inter-library variation in abundance:

```
cutoff = '[an integer]'
score_regulation_cutoff= '[an integer]' #stdev/sqrt(average)
```

Library headers, which are used in the plots and in the MySQL tables, can be defined:

```
genotype = '[strings separated by commas]' #i.e. 'wt','mut'
```

The MySQL server should be configured using the following options:

```
server = '[string]'                   # i.e. localhost
user = '[string]'                     # i.e. user
password = '[string]'                    # i.e. ''
database = '[string]'                 # i.e. profiles
```

Bowtie mapping parameters such as “max mismatches in seed”, “max sum of mismatch quality scores across alignment” and “seed length”, can be modified using the follwoing parameters:

```
mismatches = '[an integer]'
max_maq_err = '[an integer]'
```

```
seedlen = '[an integer]'
```

Each library should have its own folder containing sequencing data in fastq format. The path for the directory containing all folders and folder name should be configured (for details see the Application section).

```
fq_file_dir = '[path to directory with all data folders]'
lib = '[all the data folder names separated by commas]'
```

The variable *filtering_file* in the configuration file can refer to any fasta file, against which the reads should be filtered. There is an option *use_only_mapped_reads,* which if True allows discard all reads not mapping to the *filtering_file* and if False discards only the mapped reads. *filtering_file* can be left empty, if no filtering is required.

```
filtering_file = '[file_path]'
use_only_mapped_reads = '[True/False]'
```

The reference genome file, annotation feature file (describing for example gene models), and conserved miRNA files are defined using the following options:

```
genome_file = '[file_path]'
gtf_file = '[file_path]'
miRNA_database = '[file_path]'
```

The parameters *igv_infile* and *cluster_infile* are optional and can be used to input mapping or clustering results from other sources into the pipeline. For file formats, please see the example data. If left empty, the files generated from earlier modules are used by default.

4.2 Interactive configuration

The *shortran* pipeline also offers an interactive configuration mode, where the user is prompted for all options and parameters. The options are the same as described in section 3.1, but an additional help function is provided in the interactive version.

5 Modules

5.1 Module-1: Adapter trimming, error correction, filtering and profiling

This module offers pre-processing of the raw data obtained from sequencing (fastq format). Sequencing errors can be corrected by k-mer correction using

ECHO. This step is computationally expensive and is not critical for use of the other modules. It may, however, improve analysis accuracy.

```
Raw_data_filtering = True
ErrorCorrection = True
Path_2_ECHO = '[full path to echo]'
```

If raw fastq files are used, adapters should be clipped off and the files should be split by size:

```
adapter_filtering = True
keep_with_adapter_only = True
# If True, discards reads if no adapter was detected
adapter='[3’ adapter sequence'   # i.e. 'AGCCTTCTA'
# 3’ adapter sequence. Multiple sequences can be entered as a comma-separated list in the same order as the sequencing libraries in the “lib” variable
trim=0            # number of bases to trim from 5’ end
```

Reads can be filtered based on homology to a user-supplied multifasta file. For instance, the user could remove reads matching ribosomal RNA sequences or choose to focus only on reads mapping to the reference genome.

```
homology_filtering='[full path and filename]' #multifasta file
use_only_mapped_reads=False   #removes reads matching the fasta file
```

Following error correction and filtering, profile tables with raw and normalized read counts can be generated.

```
Make_no_cut_profile_files=True
```

Once the profile tables are available, additional filters for abundance and inter-library variation score (stdev/sqrt(average)) can be applied.

```
cutoff='a digit'  # the minimum read count sum across all libraries
score_regulation_cutoff=‘[a digit]’ # the minimum variation score
```

The final step in Module-1 adds the filtered profile table to a MySQL database:

```
Add_output_to_mysql_database=True
```

Summary Module-1:

Input:

- Raw Fastq files

Output:

- Adapter filtered Ffastq files - *$output*/module-1/adapter_filtered_data/
- Fastq files split by size - *$output*/module-1/size_fractionated_data/
- Homology filtered Fastq files - *$output*/module-1/filter/
- Profile tables - *$output*/module-1/profiles/
  - Normalized count tables with abundance cutoff
  - Normalized count tables with abundance cutoff and variation score cut-off
  - Similar files for all the size fractions
- Files with raw abundance total counts
  - Redundant counts - *$output*/module-1/*libraries_abundance*
  - Non-redundant counts – *$output*/module-1/libraries_abundance_unique
- File for selecting control for libraries - *libraries_reference*
- File for selecting pairs to analyzed for correlation - *compare_libraries*
- Graphical output
  - Plots for raw abundances across libraries
  - Sequence size variation across the libraries
  - Sequence size fraction variation within the library
- MySQL table

5.2 Module-2: Genomic Mapping

Reads are mapped to a given reference genome and alignment .igv files are generated. These files contain information about mapping positions and expression values and can be displayed using the IGV browser (http://www.broadinstitute.org/igv/).

```
genome_file = '[full path and filename'   #reference genome file
Genome_mapping = True
```

The variable *infile* allows using an alternative profile table. If left empty, the profile table generated by Module-1 is used by default.

```
infile='[full path and filename]'   #use alternative profile table
```

Summary Module-2:

Input:

- File with all the reads
  - By default taken from module-1 output
  - Optional: The user can submit any file where the first column contains sequences for genome mapping using the *infile* parameter.

Output:

- Mapped files
  - Sam format
  - IGV format

- Fasta file
  - Contains all the reads

5.3 Module-3: Clustering analysis

In module-3, a clustering algorithm is invoked where mapped reads are separated into different clusters based on the genomic positions. With default settings, a cluster is a genomic region, which contains at least five unique reads containing no gap longer than 50 nucleotides between consecutive reads. Also the abundances of these reads should sum to a total of at least 75. Parameters can be changed to adjust the stringency. An external IGV file to be used for clustering can be specified using the *igv_infile* parameter. If left blank, the IGV file produced by Module-2 is used by default.

```
Cluster_prediction=True
divide_by_size=False #do not treat size fractions separately in
                      cluster analysis
abundance_cutoff = 75 #minimum sum of read abundances per cluster
Add_cluster_to_mySQL = True
igv_infile=''
```

If preferred, the NiBLs algorithm (MacLean et al. BMC Bioinformatics. 2010 18;11:93) can be used to detect clusters instead of the simple approach described above:

```
Cluster_prediction=True
NiBLS=True
```

Finally, clusters can be annotated according to their overlap with the genomic features specified using the *gtf_file* parameter.

```
Cluster_genomic_region_prediction=True
```

Summary Module-3:

Input:

- Mapped file to genome
  - By default mapped file from Module-2
  - Optional: IGV format mapped file provided externally (*igv_infile*)

Output

- Cluster file
  - BED format file for cluster consisting all size classes
  - Size fractionated clusters
  - Clusters summary
  - Clusters in fasta format file
- Genomic annotation of clusters
  - Figure with genomic distribution of clusters

- Counts in the file *cluster_genomic_region.txt*
- File with all the sequences and their corresponding genomic annotation (file extension .region)

- MySQL table

5.4 Module-4: miRNA predictions

Modules 4 and 5 are designed for categorizing small RNAs. Module-4 uses miRDeep or miRDeep-P for miRNA prediction. The miRDeep-P module is only applicable for small RNA datasets obtained from plants. For animal data, please use miRDeep instead. Predicted miRNAs are mapped against the a fasta file containing conserved miRNA sequences (*miRNA_database*) to check the accuracy of the prediction. A summary on mapped conserved miRNA/miRNA*/miRNA-families is generated.

```
miRNA_database = '[file name]'

#fasta file with conserved miRNA sequences

MiRNA_prediction=True

mi_sequence_file = '[file name]' #optional external miRDeep format
input file

gff_file = '[file name]'      #annotation   file   only   for   miRNA
predictions
```

The variable *mi_sequence_file* is the file path to the properly formatted (miRDeep standard input) fasta file. If left empty, a miRDeep format input fasta file is generated from the Module-1 output profile table by default. miRDeep can use an annotation file to filter tRNA and rRNA sequences from its prediction results (*gff_file*).

Summary Module-4:

Input

- Genome Fasta file (*genome_file*)
- RNA classification annotation file (*gff file*)
- Sequence fasta file in MiRDeep format (*mi_sequence_file*)
  - By default it is taken from Module-1
- Conserved miRNA (*miRNA_database*)
  - By default it is taken from demo (miRbase v.17)

Output

- Potential miRNA candidates - *miRNA_filter_P_prediction_miRNA.fasta*
- miRNA precursors – *miRNA_precursors*
- miRNA structures - *miRNA_structures*
- Conserved miRNAs, family and homolog counts - *miRNA_filter_P_prediction_miRNA.miRNA_mapped.miRNAfamily.txt*

5.5 Module-5: ta-siRNA predictions

Trans-acting small interfering RNAs is another category of small RNAs generated by plants. Here we predict ta-siRNA using a modified version of Chen's algorithm (Ho-Ming Chen et al., 2007). A fasta file containing ta-siRNA clusters is produced and supplemented with another file describing the probability that an sRNA is associated with a tas-locus.

```
TasiRNA_prediction=True
```

Summary Module-5:

Input

- Genome fasta file (*genome_file*)
- Cluster file (from Module-3)

Output

- Predicted ta-siRNA sequences with p-valve < 0.001 - *p0.001_sRNA21nt_out.txt*
- P-value for all sRNA sequences - all_sRNA21nt_out.txt
- Predicted tas-loci clusters - *tasi_cluster.fa*

5.6 Module-6: Annotation

This module requires the MySQL server configuration. A user can provide multiple files in multifasta format. Reads in the profile will be mapped and the output will be sent to the MySQL server. Annotations will be added to the profile table, and an additional option can be enabled, which allows adding genomic annotation for all the reads, for example 'exonic', 'intronic' etc.

```
Add_annotation_to_mysql_database=True

annotation_file='[full paths and filenames, comma separated]'
                #multifasta files

gtf_file='full path and filename'

                # gtf format file with genomic annotations

add_genomic_region=True
```

Summary Module-6:

Input

- Multiple fasta formatted files (*annotation_file*)

Output

- Annotation appended to the MySQL table

5.7 Module-7: Expression Plots

Here we generate plots for the expression values for comparisons between libraries. Correlation coefficients are also calculated to find co-regulated

libraries. These libraries can be directly compared against one another or they can be compared after being normalized to a given reference library. Modify the files *libraries_reference* and *compare_libraries* in the Module-1 folder to set up the library comparisons.

```
Pattern_plot=True
pattern_plot_file='' #external file with profile table, default is
                      profile table from Module-1
```

Summary Module-7:

Input

- Profile table
  - By default from module – 1 (*pattern_plot_file*)

Output

- Graphical output
  - Normalized expression values plot
  - Log of normalized expression values plot
  - Relative fraction plot when provided control libraries
- Files with correlation coefficient values - **.spearman_values_**

5.8 Module-8: Pattern Search

This module calculates the Markov probability for the most 5' two and three nucleotides. An output file with probabilities is generated to allow the user to identify sequence composition biases. Given a pattern in the sub-module, all the sequences with the pattern can be retrieved and further analysis can be performed on this set.

```
Find_pattern=True
infile_for_pattern='' #profile table from Module-1 or other file with
                       sequences in first column
```

Summary Module-8:

Input

- File containing sequences
  - By default profile table from module-1 (*infile_for_pattern*)

Output

- Markov probabilities
  - For first three nucleotides - *markov_prob.txt*
- File containing sequences with the targeted pattern - **.fasta.**
- Mapping counts for sequences with pattern - *pattern_read_stats.txt*

5.9 Module-9: Queries

Once the MySQL table is created, the data can be accessed using SQL queries.

```
MySQL_query=True

query_infile='query_infile'  #Text  file  containing  MySQL  queries.
Should be located in the shortran main directory.
```

Summary Module-9:

For example queries, please see the application section, Module-9.

Input

- MySQL format query (*query_infile*)

Output

- MySQL query result - *mysql_query_output*

6 Application

Here we explain the application of *shortran* for the analysis of the dataset supplied with the package.

A sample dataset is available with the *shortran* package. It consists of a subset of four libraries from *Lotus japonicus* sRNA-seq data. Outputs from all the modules are included in the package, and we recommend that users go through the demo data examples before proceeding with their own data. Here we briefly explain the usage and output of each module. All *shortran* settings used in the analysis are presented in the *configuration.txt* file, which can be used to analyze the sample data after changing directory and MySQL server settings to reflect your local environment.

6.1 Module-1

Raw data for processing are fastq files. In the demo, these fastq files are separated in four different folders named GHD-5_sample, GHD-6_sample, GHD-9_sample and GHD-10_sample. These four folders refer to libraries mock_wild_type, wild_type, mock_NFR1 and NFR1 respectively. Each contains reads obtained from Illumina RNA-seq reads, which are mapped to chromosome 2 (35-40 MB) on the genome.

We pre-processed the reads in module 1.1 and an error correction algorithm was invoked. There were 627,147 reads in the raw dataset (Figure 1) and when parsing sequences through ECHO on the dataset approximately 3.6% reads were corrected for each library.

The two files *lib_abundance* and *lib_abundance_unique* contain abundances for each size and library for non-redundant and redundant reads respectively, data is plotted (Figures 2 and 3). Figures 2a and 2b are example figures showing read length distribution (in %) among different libraries for size classes 21 nt and 24 nt. Plots for all size are available in the plots folder. Figure 3a and 3b are plots for size distribution within a library for mock_NFR1 and NFR1 libraries, respectively.

Homology filtering was performed to remove all the reads coming from lotus repeat data, leaving 605,329 reads. Both filtered reads and reads left after filtering are provided as fastq files that can be used in separate analyses.

In module 1.2, all the reads are processed to calculate abundances of all small RNA unique reads within each library, and normalization is performed to take into account differences in library sizes. There were 42,448 unique reads in the dataset and 25,329 were left after applying an abundance cut-off of 2. An additional variation score-cutoff (stdev/$\sqrt{}$ average > 1) was applied to identify 12,814 differentially regulated sequences (unique reads). The profile tables are stored in the Module-1 output folder as well as in the MySQL output table.

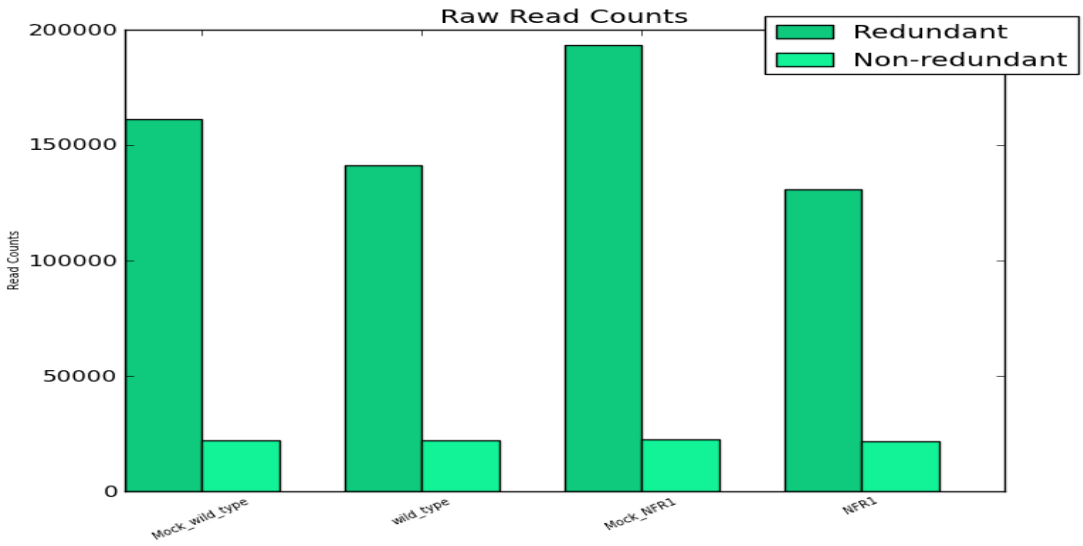

Figure 1. Raw read counts for each library.

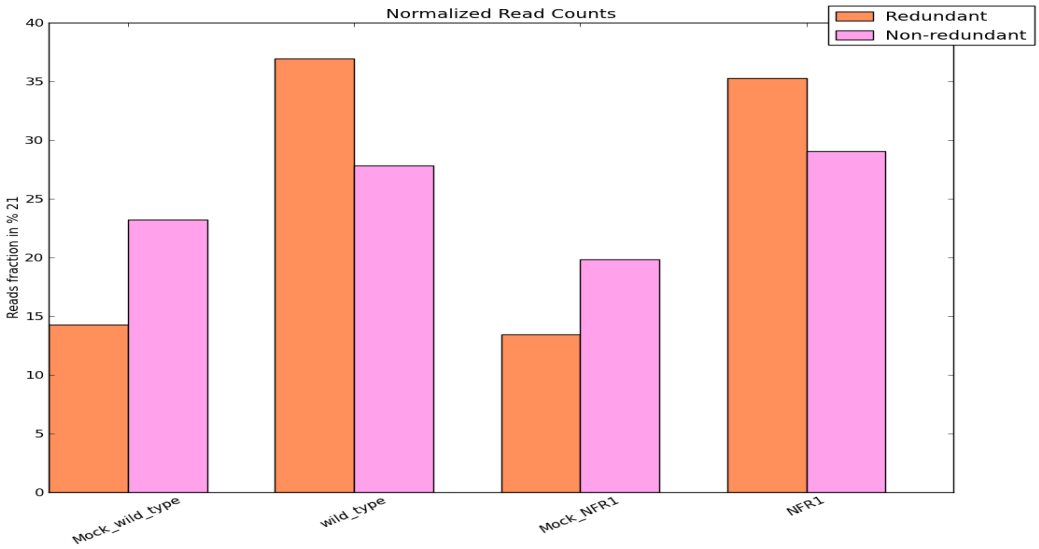

Figure 2a. Read length distributions among all libraries. 21 fraction.

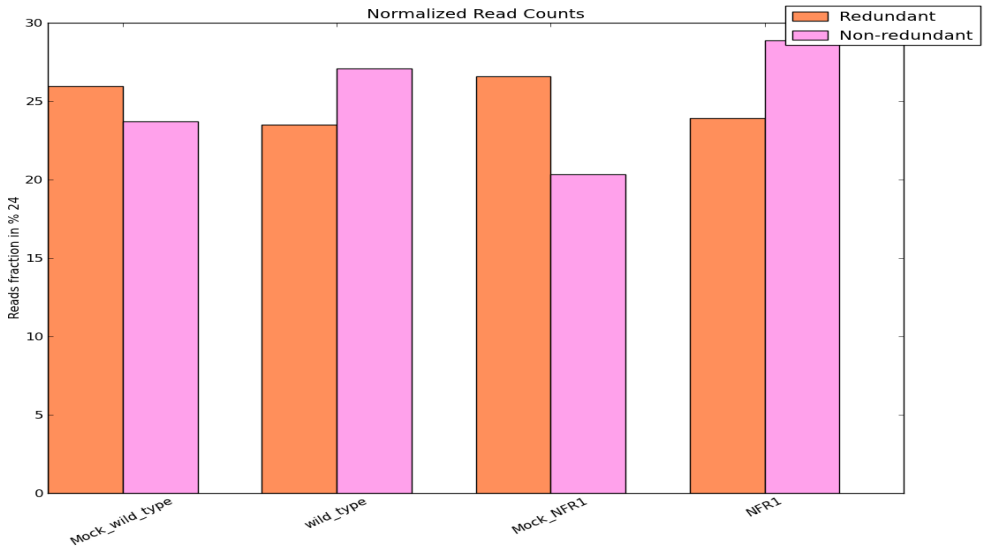

Figure 2b. Read length distributions among all libraries. 24 fraction.

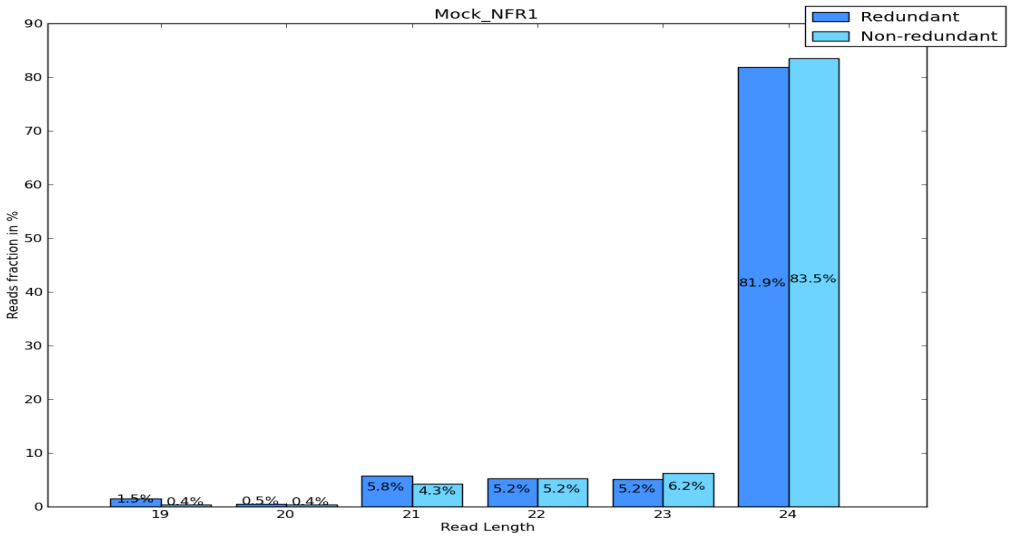

Figure 3a. Size distribution for the mock_NFR1 library

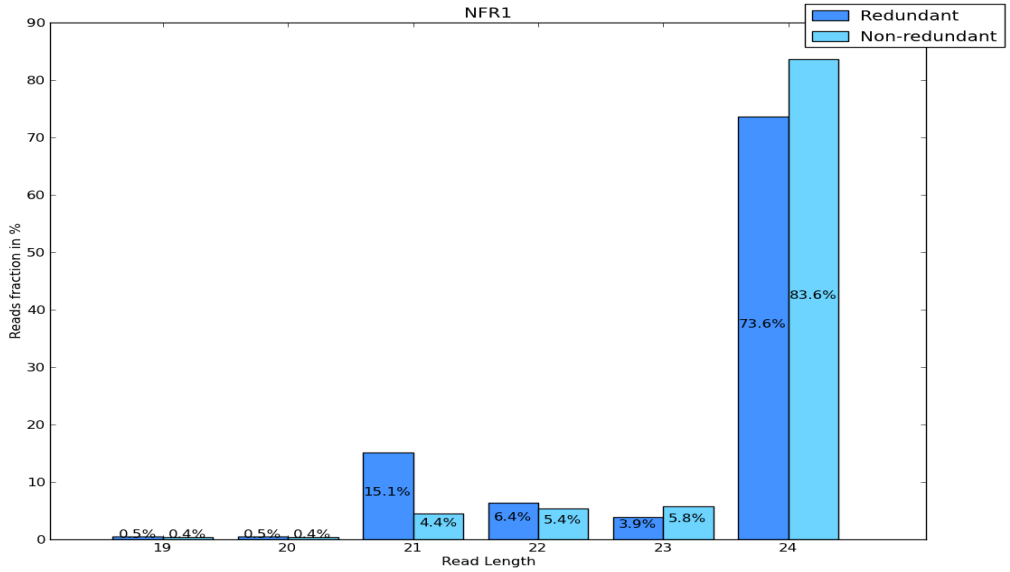

Figure 3b. Size distribution for the NFR1 library

6.2 Module-2

All the reads were mapped to genome using Bowtie mapping software. Mapping parameter options were:

Mismatches = 2

max_mac_err = 70

seedlen = 25

All the reads were mapped back to the genome on 102,356 places. As the demo data was constructed with the mapped reads only, all the reads are expected to map back. An IGV format file is generated and placed in Module-2. This IGV file also contains all the normalized expression values and these can visualized in the IGV browser.

6.3 Module-3

This module processes the IGV file from Module-2 and predicts clusters with a minimum abundance of 75. 4315 clusters were predicted. A .bed format file is supplied containing all the clusters with embedded sequences. The file can be visualized using the IGV browser. Genomic annotation was performed and 48 clusters were exonic, 46 intronic, 3855 were intergenic and 2 clusters overlapped intron/exon boundaries (Figure-4).

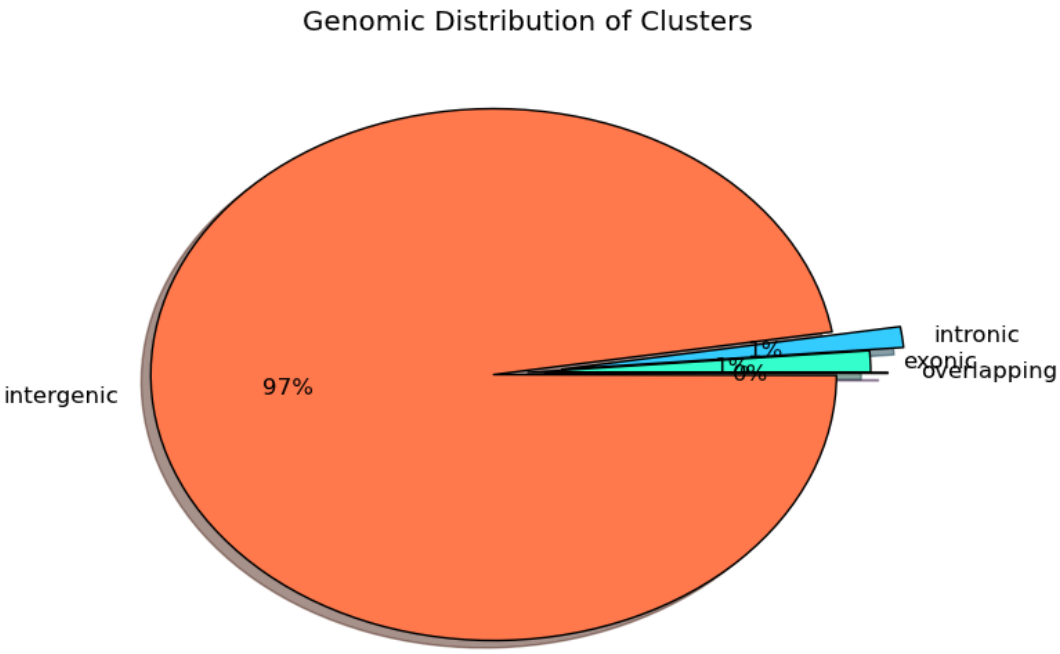

Figure 4. Genomic distribution of clusters

6.4 Module-4

miRNA prediction was done using the plant-adjusted version of miRDeep, miRDeep-P, and 29 miRNAs were predicted, including 1 conserved and 28 novel miRNAs. These contain 1 miRNA families and details of homologs for each family. All details are given in the Module-4 output folder, mapping is performed against mIRBASE release 17. Alternative miRNA databases can be compared against as defined by the user.

6.5 Module-5

30 individual ta-siRNA sequences were predicted based on the relative positioning (phasing) of the sRNA loci on the genome. The majority of these loci was contained in 6 clusters (Module-3). A cluster fasta file is available, as is the list of all the small RNA reads with associated p-value evaluating the relative positioning of the loci. A read should have p-value lower than 0.001 to be considered a phased RNA.

6.6 Module-6

This module is implemented to add annotation to all the sequences in the profile table. This module requires MySQL installation to be executed. As input, a variety of fasta files can be provided by the user. Sequences will be mapped against each file, and where sequences map to the respective reference the associated mapping positions are be reported.

6.7 Module-7

This module creates expression pattern plots for the libraries that need to be compared. Pairs to be considered can be defined in the file provided in the Module-1 folder, *compare_libraries*. By default plots are generated for all possible sample pairs, including the plot against sum and score values from the profile table. As target libraries will usually be sequenced along with corresponding controls, the latter can be specified as "references" in the file named *libraries_reference*, also provided in the Module-1 folder.

The file *libraries_reference* contains two columns of which the first refers to the target library. Libraries are annotated by numbers in the same order as on the user-provided input list for libraries. The second column is filled by 0s by default, unless a control library is defined. Target/reference relationships are defined by replacing the 0s with the number of the appropriate control library.

Four different kinds of plots can be generated. When no control library is defined, normalized counts and log values of normalized counts are plotted (Figures 5a and 5b). If reference libraries are defined, expression scores are plotted according to equations 1a and 1b. The expression plots are visualized in Figures 5c and 5d.

Equation 1a. log-odd-score = log(Ti/Ci)

Equation 1b. norm_control = (Ti-Ci)/(Ti+Ci)

Where Ti is the expression values in the target library and

Ci is the expression values in the control library

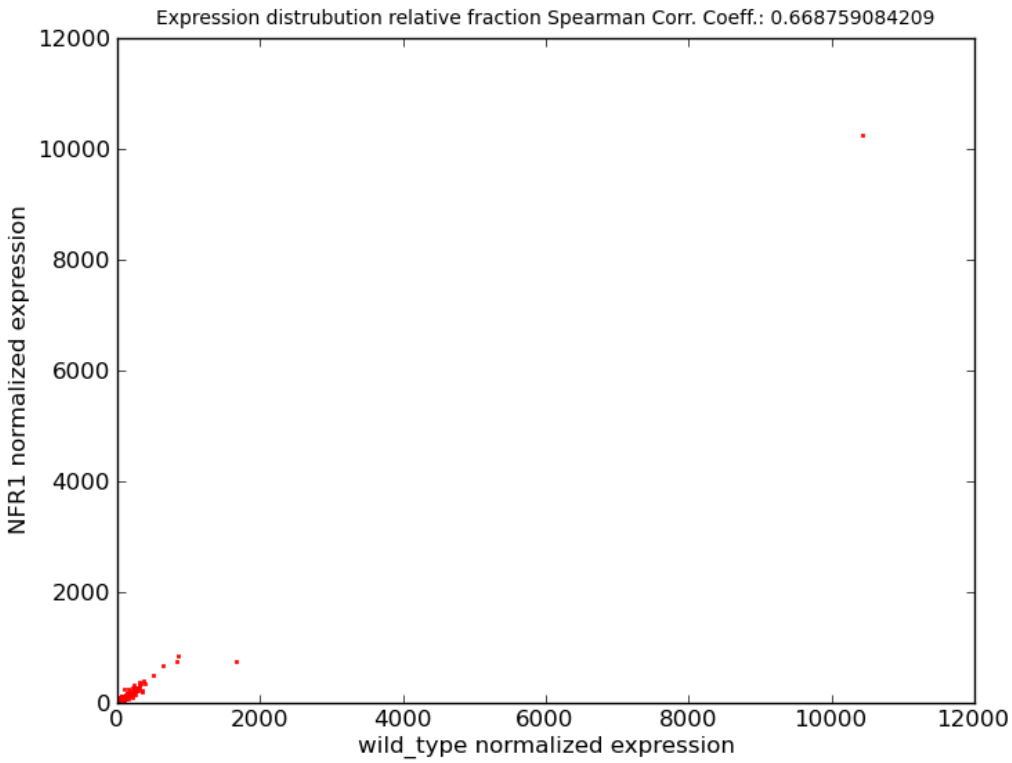

Figure 5a. Normalized counts are plotted.

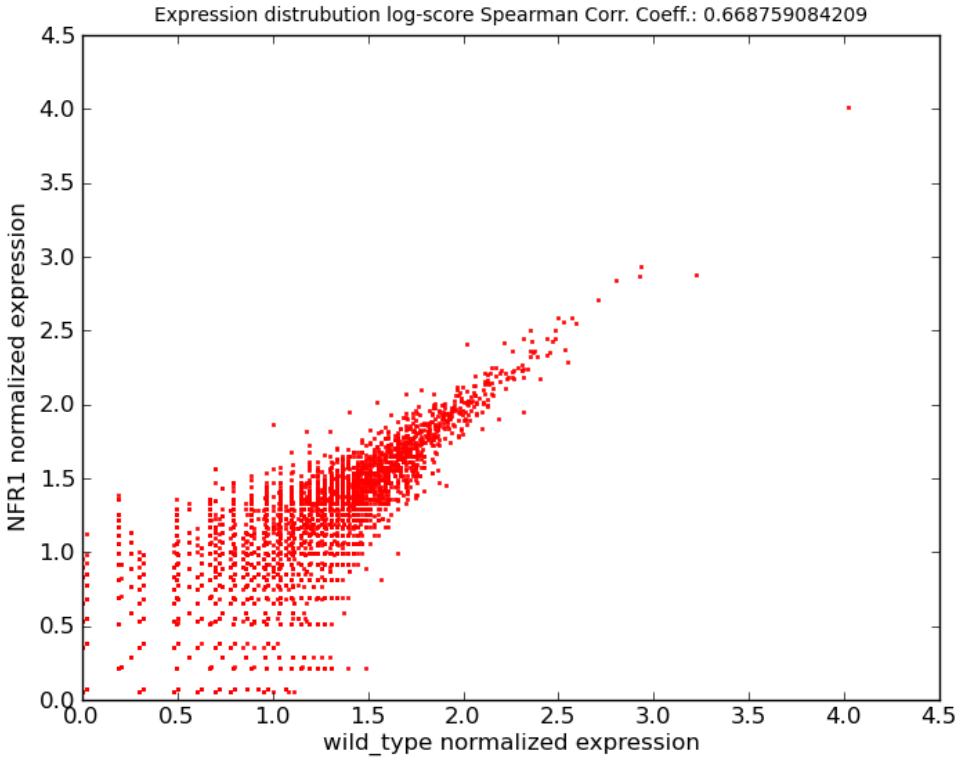

Figure 5b. log(normalized counts) are plotted

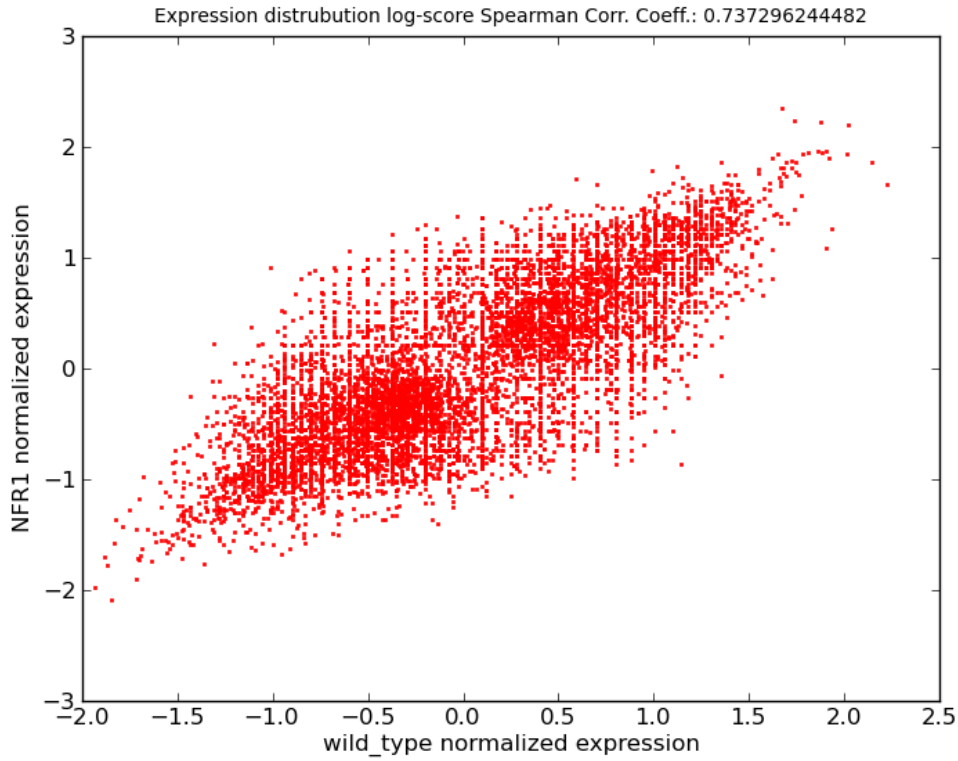

Figure 5c. .log-odd-score = log(Ti/Ci) plotted

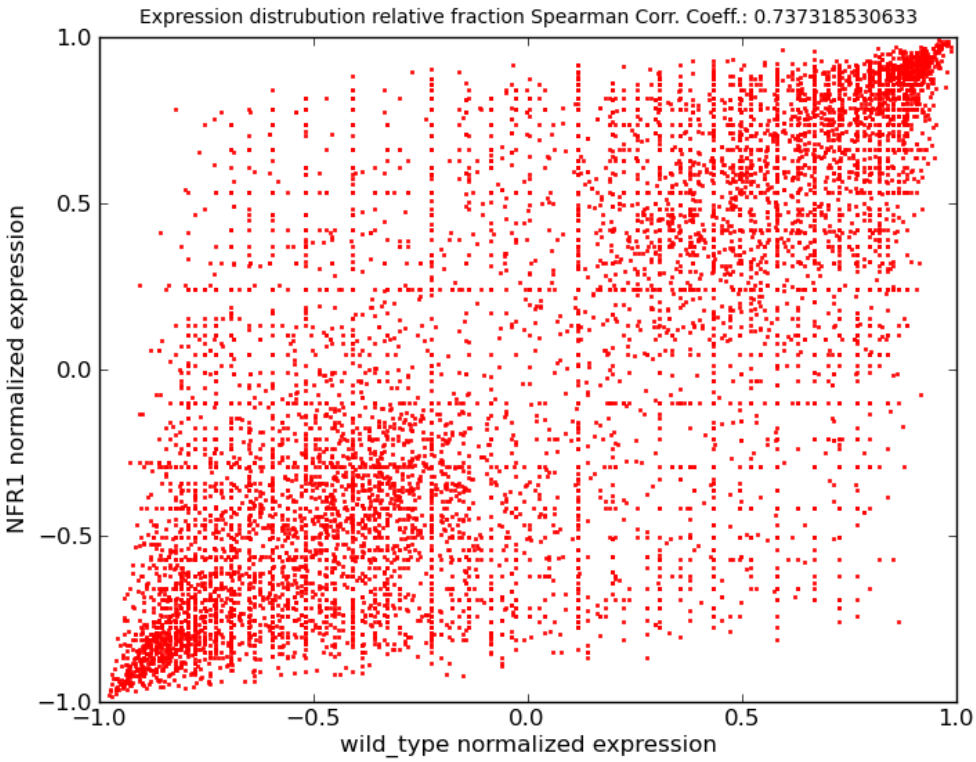

Figure 5d. norm_control = (Ti-Ci)/(Ti+Ci) plotted.

6.8 Module-8

This module is used to find sequence composition biases in the 5' end of the sequences. Di- and tri-nucleotide Markov probabilities are calculated for each nucleotide combination. In the provided sample dataset, for example, an over-representation of GAA starting triplets is observed. Sequences with particular motifs are extracted and mapped back to the genome. If a bias is caused by technical artifacts, the proportion of sequences with the motif in question mapping to the reference genome might be lower than the average.

6.9 Module-9

Using this module, the MySQL table can be queried. Complex queries concerning particular biological questions can be performed. The table below shows examples of queries.

Purpose	Syntax
Retrieve variation scores and miRBase annotations for predicted miRNAs.	SELECT `#Sequence`, `miRBase.fa`, `score` FROM `2012-03-04_profile_19_24_sorted.cut-1_score_1` WHERE `miRNA_prediction` = "predicted as miRNA";
Retrieve average response values for sRNAs up-regulated in the wild only for sRNAs mapping to miRNAs annotated in miRBase.	SELECT COUNT(score) AS n, AVG(LOG(wild_type_norm/Mock_wild_type_norm)) AS wt, AVG(LOG(NFR1_norm/Mock_NFR1_norm)) AS nfr1 FROM `2012-03-04_profile_19_24_sorted.cut-1_score_1` WHERE LOG(wild_type_norm/Mock_wild_type_norm) > 0 AND `miRBase.fa` !="0";
Retrieve expression information for sRNAs mapping to miRNAs annotated in miRBase and sort in descending order by the wild type expression response.	SELECT `#Sequence`AS sequence, `read_size` AS length, LOG(wild_type_norm/Mock_wild_type_norm) AS wt, LOG(NFR1_norm/Mock_NFR1_norm) AS nfr1, `miRBase.fa` FROM `2012-03-04_profile_19_24_sorted.cut-1_score_1` WHERE `miRBase.fa` != 0 ORDER BY wt DESC;

6.10 Computation summary

Module	Process	Time (s)	CPU %	Memory %
1.1	Adapter filtering/trimming	15.4	0.4	0.3
1.1	Error correction	5827.5	73.4	0.9
1.1	Size splitting	7.1	49.4	0.4
1.1	Homology filtering	116.0	113.5	0.5
1.2	Making profile tables	73.5	54.1	1.3
1.3	Adding data to MySQL database	1.9	37.9	0.2
2	Genomic mapping	129.0	131.9	5.4
3.1	Cluster prediction only	9.6	98.5	1.6
3.1	- including sequence extraction	1393.4	149.3	0.8
3.2	Cluster genomic region annotation	26.7	97.8	0.4
4.1	miRNA predictions-mirDeep2	2476.3	100	40.4
4.1	miRNA predictions-mirDeepP	11875.1	1.8	3.9
4.2	miRBase mapping	5.0	3.3	0.8
5	ta-siRNA prediction	79.7	98.9	0.9
6	Homology annotation - mapping	6.5	16.7	0.2
6	Homology annotation - MySQL	1.1	0.1	0.1
7	Expression profile plotting	107.8	97.6	1.1
8.1	Find pattern	3.7	100	0.2
8.2	Map pattern	155.2	99.8	5.9
9	MySQL querying	0.0	0.9	0.2

The total number of reads processed was 627147. Data was analyzed on a Macintosh iBook with 4 GB memory and a four core 2.7 GHz Intel i7 processor.