sRNAtoolboxVM: small RNA analysis in a virtual machine

Cristina Gómez-Martín, Ricardo Lebrón, Antonio Rueda, José L. Oliver, and Michael Hackenberg

Abstract/summary

High-Throughput Sequencing (HTS) data for small RNAs (non-coding RNA molecules that are 20–250 nucleotides in length) can now be routinely generated by minimally equipped wet laboratories; however the bottleneck in HTS-based research has shifted now to the analysis of such huge amount of data. One of the reasons is that many analysis types require a Linux environment but computers, system administrators and bioinformaticians suppose additional costs that often cannot be afforded by small to mid-sized groups or laboratories. Web-servers are an alternative that can be used if the data is not subjected to privacy issues (what very often is an important issue with medical data). However in any case they are less flexible than standalone programs limiting the number of workflows and analysis types that can be carried out.

We show in this protocol how virtual machines can be used to overcome those problems and limitations. sRNAtoolboxVM is a virtual machine that can be executed on all common operating systems through virtualization programs like VirtualBox or VMware, providing the user with a high number of preinstalled programs like ***sRNAbench*** for small RNA analysis without the need to maintain additional servers and/or operating systems.

Introduction

Given the broad interest in small RNA molecules existing in many different fields, the primary and downstream analysis of small RNA sequencing data is nowadays a highly demanded task. Over the last 15 years many basic research has been conducted to understand the functions that microRNAs can play in numerous vital processes like cell differentiation [1] and its role in conferring robustness to biological processes [2]. More recently, microRNAs have exited expectations both as prognostic and diagnostic biomarkers and therapeutic agent in medicine [3] or as transgenes in plant science [4]. Furthermore, they seem to play vital roles in host-pathogen or host-symbiont interactions [5, 6].

Over the last years, the generation of small RNA sequencing data was becoming constantly cheaper and assessable to a broader audience. Consequently the generated data volume was continually rising converting the data analysis as the main task in HTS-based small RNA research. There are several reasons why the data analysis is currently the main bottleneck in HTS-based research. First, many important programs like

samtools [7] are only available for Linux operating systems or would need to be compiled by the user in order to make them run on other OS like Windows. This however is frequently not a straight forward process and requires skilled personal. The asset costs and administration of Linux machines together with the need for personal trained in HTS-data analysis raises the cost for HTS-based research enormously bringing especially smaller groups or laboratories into difficulties. Webservers like miRanalyzer [8] can be used alternatively for basic small RNA data analysis, however, the usage of web-servers in general is very often prohibitive due to privacy issues.

In this protocol, we will show how virtual machines can be a solution to the above outlined problems. Basically a virtual machine (VM) is an emulation of a given computer system. In this way, a Linux OS can be easily executed within a windows environment. Those virtual Linux machines can then be prepared for its usage (like small RNA analysis) by a developing team lifting from the user the burden to administer a separate Linux OS and to install all needed programs. Furthermore, the maintenance is likewise carried out by the developer team and the user needs to download only the newest versions without need for specialized personal in Linux administration and maintenance.

We show the strength of this concept by means of the virtualization of sRNAtoolbox [9]. This collection of programs for small RNA research that includes programs like sRNAbench [10] allowing for many important analysis types like i) expression profiling from NGS data, prediction of novel microRNAs, detection of other small RNAs, differential expression of different RNA types and several downstream analyses like target gene prediction and functional analysis of those genes (pathways, GO-terms). Furthermore, sRNAtoolboxVM includes helper tools that automatically populate the local database and keep the machine up to date.

In this chapter we will show how to run the virtual machine on any OS, how to customize the sRNAtoolboxVM, i.e. how to install the species annotations needed by the user, and how to carry out a basic NGS data analysis.

Install sRNAtoolboxVM

Requirements

- Supported Operating Systems: Windows (7 or higher), Mac OS X (10.9 or higher), Linux or Solaris.
- Virtualbox (5 or higher) and Virtualbox Extension Pack: https://www.virtualbox.org/wiki/Downloads. Official User Manual: https://www.virtualbox.org/manual/UserManual.html.
- Available Memory: at least 4GB (recommended: 8GB or more).
- Free disk space: at least 50GB (recommended: 250GB or more).

Run sRNAtoolboxVM

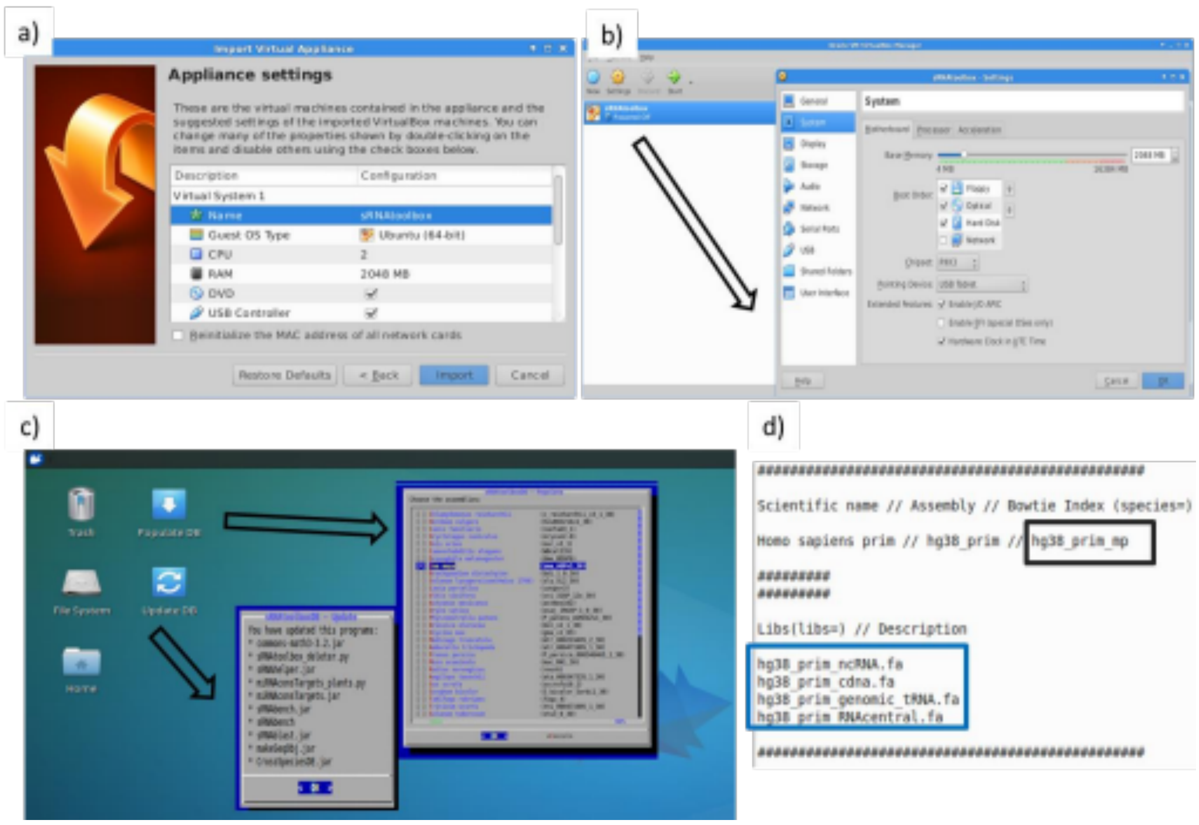
- Download the sRNAtoolbox OVA file, available at http://bioinfo2.ugr.es/sRNAtoolboxDB/sRNAtoolbox.ova .

- Open Virtualbox. In the menu, click on ***File → Import Appliance***. Select the OVA file and click on **Next**. This should display the appliance settings shown in Figure 1a. · In the appliance settings the number of CPUs and memory available for the VM can be selected. For a decent performance, the RAM memory should be at least 4GB, and the number of processors 2.
- After clicking on '**Import**' the Virtual Machine will be prepared on your host computer. This step will take a couple of minutes!
- Once imported, select the virtual machine and click on Start (green arrow).
- Now the virtual machine is running (Figure 1c).
- The default user is 'srna' and all relevant passwords are also 'srna'. The password can be changed by the user. Note that this is not a security issue as the VM can only be assessed from the host machine.
- All parameters can be changed any time by means of the 'Settings' tab (see Figure 1b) when the VM is shutdown off.
- To shutdown Click on ***ACPI Shutdown***.

Share data with the Virtual Machine

An important issue is how to pass data between the host and the virtual machine. There are basically two possibilities.

- Share a folder between host and virtual machines. Click on **Settings** (Figure 1b) and then on ***Shared Folder***s tab. The shared folder will appear in the user's home directory. · Use a ***Dropbox client*** inside the virtual machine. It is already installed and opens automatically when you start the virtual machine. You only need to provide once you login credentials. The Dropbox folder can be found as well in the user's home.



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Description	Configuration
Virtual System 1	
Name	sRNAtoolbox
Guest OS Type	Ubuntu (64-bit)
CPU	2
RAM	2048 MB
DVD	✓
USB Controller	✓

Reinitialize the MAC address of all network cards

```
##################################################

Scientific name // Assembly // Bowtie Index (species=)

Homo sapiens prim // hg38_prim // hg38_prim_mp

#########
#########

Libs(libs=) // Description

hg38_prim_ncRNA.fa
hg38_prim_cdna.fa
hg38_prim_genomic_tRNA.fa
hg38_prim_RNAcentral.fa

##################################################
```

Figure 1: a) the appliance settings allow to adjust the performance of the VM to the means of the host machine (number of CPUs and memory), b) by means of 'Settings', the user can add shared folders (with the host machine)

or change other important parameters, c) after launching the VM, a typical 'XFCE' desktop appears with a number of additional links like i) 'Populate DB' (add data to the local DB), ii) 'Update DB' (keep the programs up-to-date) and iii) 'protocol.txt' (the small RNA data analysis protocol of this chapter); d) the 'info.txt' file on the desktop holds the names of the installed libraries: within the black square the name of the assembly that has to be specified in 'genome mode' (see protocol), and within the blue square the available libraries that can be specified by means of the libs= parameter (see protocol). The annotations exist both, in fasta format and as bowtie indexes (which have the same names without the '.fa' extension).

Customize and populate the local database

Some of the installed programs rely on a local database. This database holds for example the bowtie indexes of the genome assemblies, microRNA sequences or other small RNA annotations among other needed data. By default, the VM is shipped without this data as otherwise it would grow too big. Therefore the first step will be to populate the database with the species needed by the user. The local database is installed in the following path: **'/opt/sRNAtoolboxDB'.**

Populate and update tools

The easiest way to populate the local database and to maintain the VM updated (install the most recent versions of the programs) is by means of two small tools installed in the VM. On the desktop, two links can be found, '***Populate DB***' and '***Update DB***' (Figure 1c).

To populate the local database with any of the species contained in the sRNAtoolboxDB:

- Double-click on the icon 'Populate DB'
- Select the species that should be downloaded by activating the 'check-box': **for the following protocol, please choose 'hg38_prim'** (this is the 'most recent, primary human assembly, i.e. without alternative sequences)
- Already installed species will appear as 'checked'; those can be removed from the local DB by unchecking the boxes.
- Click on 'ok' to start the processes, which will take a while depending on network conditions.

Important: after installing a species, the file ***info.txt*** on the desktop will contain the information about the names of the bowtie indexes that have to be used for the profiling (see below) and the names of the available annotations (Figure 1 d).

Update the sRNAtoolbox programs:

- Double-click on the 'Update DB' icon
- First, the program will check if a new version of the entire sRNAtoolboxVM exists. If a new version is available, the corresponding download link will be provided.
- Second, the update program will check if for any of the sRNAtoolbox programs a new version exists. If so, those will be downloaded and the old versions will be replaced.
- A short summary of the whole process is shown to the user including a brief description of the changes.
- **Perform an update after importing sRNAtoolboxVM!**

Note that the local sRNAtoolbox database can be populated also manually. Later in this chapter we will give one example on how to do this.

Small RNA analysis

The following practical protocol for small RNA data analysis is meant for users with only very basic knowledge in Linux and analysis of HTS data. The practical session consists in a number of different analyses which are carried out by running different programs in the Linux command line. Those commands will be highlighted in **grey shaded boxes** and they can be found also in the file 'protocol.txt' on the desktop. The commands can be directly copy/paste from this file into the terminal which can be opened by clicking on the 'Terminal' icon on the desktop. Finally, most of the programs and analysis types are highly parameterized processes. In the beige 'Boxes' located at the end of many sections, we will briefly discuss those parameters.

Retrieving data

To illustrate the basic features of sRNAtoolboxVM, we will use a publically available dataset from mucosal epithelium of the nasopharynx with two different states: nasopharyngeal carcinoma and chronic nasopharyngitis [11]. First we need to retrieve this data from the public repository SRA and to store it into a previously generated folder (in the protocol file, this is shown step by step). We recommend to name this folder exactly as proposed in the protocol.txt file which will allow copy/paste all following commands, otherwise those need to be adapted by the user.
After having navigated into the data folder (with 'cd' command), the following commands are executed to download a subset of the whole study which will be used to illustrate the function of sRNAtoolboxVM (note that they can be pasted together into the terminal and will be processed subsequently one after one:

```
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105948/SRR2105948.sra
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105949/SRR2105949.sra
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105950/SRR2105950.sra
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105951/SRR2105951.sra
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105941/SRR2105941.sra
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105940/SRR2105940.sra
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105939/SRR2105939.sra
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-
instant/reads/ByRun/sra/SRR/SRR210/SRR2105938/SRR2105938.sra
```

After downloading the data files in ***SRA*** format, those have to be converted into ***fastq*** format by means of the ***fastq-dump*** program of the SRA Toolkit [12]. In order to save space, we will convert into compressed files. Note that the '*' will apply the program to all file names starting with

‘SRR’.

```
fastq-dump --gzip SRR*
```

After the conversion process, the original SRA files (those with sra extension) can be removed with the command:

```
rm *.sra
```

Pre-processing of the data
A crucial step in the analysis of high-throughput small RNA sequencing data is the detection of the 3’ adapter sequence. Many small RNAs (or RNA fragments) have lengths between 21/22 nt (mature microRNAs) and 33 nt (tRNA halves), and therefore the adapter or part of it will be sequenced as well at nowadays typical read length of 36-100nt. As a consequence, the pre processing consists normally of two basic steps: i) adapter trimming (detect and remove the adapter sequence) and ii) read collapsing (determine the unique reads and assign a read count to them, i.e. the number of times they have been sequenced). The following command will perform the pre-processing generating additionally length distribution files.

```
sRNAbench input=/home/srna/data/nasopharyngeal_carcinoma/SRR2105938.fastq.gz
output=/home/srna/results/SRR2105938 adapter=TGGAATTCTCGGGTGCCAAGGG
graphics=true
```

Explanation of the command:

- ***input=’filename path’***: several input formats are accepted including fastq, sra, fasta, and read/count as well as those formats compressed with the gzip algorithm (e.g. ‘gz’ extension).
- ***output=’folder’:*** the output folder. If this parameter is not given, then the output will be written into ***/opt/sRNAtoolboxDB/out/***
- ***adapter=’sequence’:*** the adapter sequence that should be trimmed from the reads. If this parameter is no used, then the program assumes adapter trimmed input. · ***graphics=true:*** the length distributions are plotted and the graphics are copied into the ‘***graphs’*** folder

Results of the command:

- The adapter trimmed reads can be found in the file ***‘reads_orig.fa’***. Those reads will be used for the subsequent analyses.
- In the ***‘stat’*** folder, two files reflecting the length distribution can be found o ***‘readLengthFull.txt’***, the length distribution of all raw reads after adapter trimming
    o ***‘readLengthAnalysis.txt’*** the length distribution of all adapter trimmed reads in the analysis (i.e. after quality and length filtering)

Box 1: Preprocessing

Several parameters can influence the results of this step:

- **holdNonAdapter=true:** the reads for which the adapter was not found are **also** used for the analysis. This might be indicated if many RNA molecules longer than the read length can be expected.
- **adapterMM (default 1) and adapterMinLength (default 10):** manipulate the stringency of adapter detection, i.e. more stringent if the number of mismatches is lower (adapterMM) and the detected adapter length is longer (adapterMinLength).
- **removeBarcode='number of bases':** the given number of nucleotides will be clipped of the 5' end of the read (if a barcode sequence was used to distinguish between samples) · **remove3pBases='number of bases':** after adapter trimming, the given number of nucleotides is trimmed of the 3' end of the read. This might be important if random adapters are used to avoid sequencing bias.
- **guessAdapter=true:** if the adapter is not known, sRNAbench will try to guess it. · **recursiveAdapterTrimming=true:** The adapter is recursively detected and trimmed. If ***adapterMinLength*** bases of the adapter are not found within the read sequence, then the adapter is recursively detected only at the 3' end of the read.
- **minRC='number':** a read must have at least the specified read count in order to be used in the analysis. For example minRC=2 deletes all singleton reads.
- **minReadLength and maxReadLength** allow to set the maximum and minimum read length. Reads outside this range are filtered out (default **minReadLength=15**)
- **sep='separator string':** if the input is in fasta format, by default sRNAbench expects this format: **>ID#300**, being the number after the '#' the read count. By means of the **sep=** parameter, other separators can be chosen. Note that this is necessary if other programs are used for adapter trimming and collapsing.

Profile the expression of miRBase microRNAs

First we will see how we can determine the expression of known microRNAs. In general, to profile the expression of known RNAs we will require its reference sequences. By default, sRNAbench uses microRNA annotations from miRBase [13] . The reference precursor (***'hairpin.fa'***) and mature (***'mature.fa'***) sequences must be located in the '**libs**' folder of the local sRNAtoolbox database (***'/opt/sRNAtoolboxDB/libs'***). The VM contains the sequences of release 21. If miRBase releases a new version, both files can be downloaded from here: http://mirbase.org/ftp.shtml

Command:

```
sRNAbench input=/home/srna/data/nasopharyngeal_carcinoma/SRR2105938.fastq.gz
output=/home/srna/results/SRR2105938_miRBase adapter=TGGAATTCTCGGGTGCCAAGGG
graphics=true microRNA=hsa
```

<u>Explanation of the command:</u>

- ***microRNA='short species name':*** In miRBase, the nomenclature makes reference to the species in the first 3-4 letters. For example, **hsa**-mir-21 refers to microRNA 21 in *Homo sapiens* while **mmu**-mir-21a would be the microRNA in *Mus musculus*. This first part of the microRNA name is used to parse out the reference sequences of the species that should be analysed.

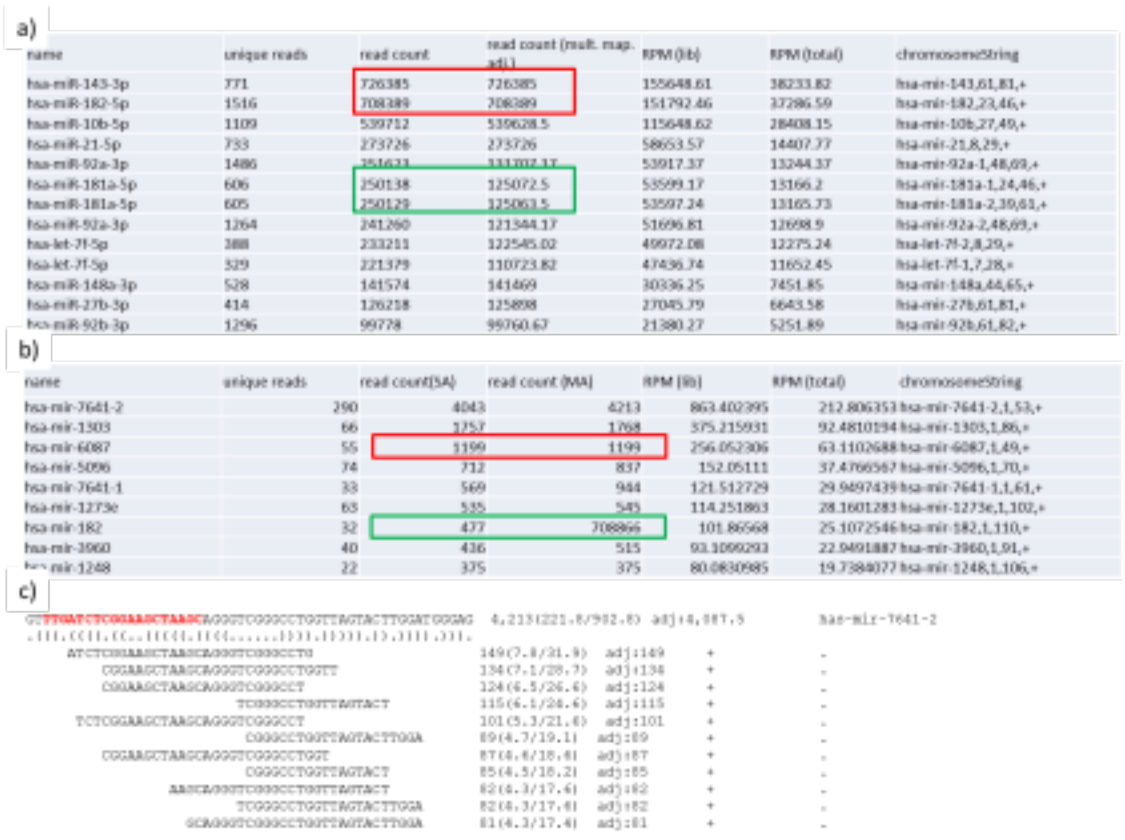
Results of the command:

- Several expression files that contain the information shown in Table 1 (the '***.grouped**' files)
- '***mature_sense.grouped***' holds the expression values of the mature sequences (see Figure 2a). The format is the same for all expression files and has the columns shown in Table 1.
- ***'mature_sense_SA.grouped';* the single assignment expression file.** Multiple mapping of reads is a well-known problem in HTS data analysis. MicroRNAs are frequently members of broader families that contain several genes with highly similar mature sequences. sRNAbench addresses this problem in two ways: i) a simple adjusted expression value is calculated dividing the read count of each read by the number of reference sequences or genome loci to which they map (Figure 2a) and ii) by determining a 'single assignment' expression value, i.e. each read is only assigned to one sequence or loci, i.e. to the one with the highest expression value (for details please see [10]). Therefore, for each expression file, like ***'mature_sense.grouped'***, one with 'single assignment' expression values are generated.
- '***hairpin_sense.grouped***'; the expression of the precursor sequences including the mature sequences
- '***hairpin_sense_SA.grouped***'; the expression values of this file are based only on those reads that map to the precursor, BUT NOT to the mature sequences (see Figure 2b). This is because in the single assignment, the reads are first assigned to the mature sequences and then to the precursors. For most microRNAs, the number of reads uniquely mapped to the precursor but not to the mature sequence should be close to cero. Therefore, those precursors with high number of uniquely mapped reads could be incorrect microRNAs (false positive annotations) (see Figure 2c).
- ***hairpin*** folder; which holds two types of files: i) '****.align'*** files that show all reads mapped to the sequence (Figure 2c) and ***'*.countArray'***, which gives the expression values per sequence position, i.e. the percentages based on unique reads (UR) and total read count (RC) and the RPM (read per million) expression per base.

Table 1: The file format of all expression files (i.e. *.grouped files)

name	The name of the reference sequence.
unique reads	The number of unique reads mapped, i.e. in the case of microRNAs these sequences are isomiRs.
read count	The total number of reads mapped.
read count (mult. map. adj.)	Adjusted for multiple mappings, i.e. if a read maps to N different reference sequences, the read count is divided by N.
RPM (lib)	The Read Per Million expression value using all reads mapped to this library for normalization. Note, the RPM value is calculated using the 'read count' and not the adjusted number.

RPM (total)	The Read Per Million expression value using all genome mapped reads (genome mode) or all adapter cleaned reads (library mode) for normalization. Note, the RPM value is calculated using the 'read count' and not the adjusted number.
chromosomeString	In genome mode, this string reflects the chromosome coordinates and the name of the reference sequence, while in library mode, the name of the reference sequence and the coordinates (within the reference sequence)


a)

name	unique reads	read count	read count (mult. map. adj.)	RPM (lib)	RPM (total)	chromosomeString
hsa-miR-143-3p	771	726385	726385	155648.61	38233.82	hsa-mir-143,61,81,+
hsa-miR-182-5p	1516	708389	708389	151792.46	37286.59	hsa-mir-182,23,46,+
hsa-miR-10b-5p	1109	539712	539628.5	115648.62	28408.15	hsa-mir-10b,27,49,+
hsa-miR-21-5p	733	273726	273726	58653.57	14407.77	hsa-mir-21,8,29,+
hsa-miR-92a-3p	1486	251623	131707.17	53917.37	13244.37	hsa-mir-92a-1,48,69,+
hsa-miR-181a-5p	606	250138	125072.5	53599.17	13166.2	hsa-mir-181a-1,24,46,+
hsa-miR-181a-5p	605	250129	125063.5	53597.24	13165.73	hsa-mir-181a-2,39,61,+
hsa-miR-92a-3p	1264	241260	121344.17	51696.81	12698.9	hsa-mir-92a-2,48,69,+
hsa-let-7f-5p	388	233211	122545.02	49972.08	12275.24	hsa-let-7f-2,8,29,+
hsa-let-7f-5p	329	221379	110723.82	47436.74	11652.45	hsa-let-7f-1,7,28,+
hsa-miR-148a-3p	528	141574	141469	30336.25	7451.85	hsa-mir-148a,44,65,+
hsa-miR-27b-3p	414	126218	125898	27045.79	6643.58	hsa-mir-27b,61,81,+
hsa-miR-92b-3p	1296	99778	99760.67	21380.27	5251.89	hsa-mir-92b,61,82,+

b)

name	unique reads	read count(SA)	read count (MA)	RPM (lib)	RPM (total)	chromosomeString
hsa-mir-7641-2	290	4043	4213	863.402395	212.806353	hsa-mir-7641-2,1,53,+
hsa-mir-1303	66	1757	1768	375.215931	92.4810194	hsa-mir-1303,1,86,+
hsa-mir-6087	55	1199	1199	256.052306	63.1102688	hsa-mir-6087,1,49,+
hsa-mir-5096	74	712	837	152.05111	37.4766567	hsa-mir-5096,1,70,+
hsa-mir-7641-1	33	569	944	121.512729	29.9497439	hsa-mir-7641-1,1,61,+
hsa-mir-1273e	63	535	545	114.251863	28.1601283	hsa-mir-1273e,1,102,+
hsa-mir-182	32	477	708866	101.86568	25.1072546	hsa-mir-182,1,110,+
hsa-mir-3960	40	436	515	93.1099293	22.9491887	hsa-mir-3960,1,91,+
hsa-mir-1248	22	375	375	80.0830985	19.7384077	hsa-mir-1248,1,106,+

c)

```
GTTTGATCTCGGAAGCTAAGCAGGGTCGGGCCTGGTTAGTACTTGGATGGGAG  4,213(221.8/902.8) adj:4,087.5        has-mir-7641-2
.(((.((((.((..((((.((((......)))).)))).)).)))).))).
     ATCTCGGAAGCTAAGCAGGGTCGGGCCTG                  149(7.8/31.9)  adj:149      +          .
         CGGAAGCTAAGCAGGGTCGGGCCTGGTT               134(7.1/28.7)  adj:134      +          .
         CGGAAGCTAAGCAGGGTCGGGCCT                   124(6.5/26.6)  adj:124      +          .
                        TCGGGCCTGGTTAGTACT          115(6.1/24.6)  adj:115      +          .
      TCTCGGAAGCTAAGCAGGGTCGGGCCT                   101(5.3/21.6)  adj:101      +          .
                         CGGGCCTGGTTAGTACTTGGA      89(4.7/19.1)  adj:89        +          .
         CGGAAGCTAAGCAGGGTCGGGCCTGGT                87(4.6/18.6)  adj:87        +          .
                         CGGGCCTGGTTAGTACT          85(4.5/18.2)  adj:85        +          .
                 AAGCAGGGTCGGGCCTGGTTAGTACT         82(4.3/17.6)  adj:82        +          .
                        TCGGGCCTGGTTAGTACTTGGA      82(4.3/17.6)  adj:82        +          .
                   GCAGGGTCGGGCCTGGTTAGTACTTGGA     81(4.3/17.4)  adj:81        +          .
```

Figure 2: sRNA expression and alignment files. a) excerpt from a typical *mature_sense.grouped* file. The read count and the adjusted read count will have same values if none of the reads maps to several reference sequences like those marked with the red box. The green box shows hsa-miR-181a-5p which can be obtained from two different gene copies (181a-1 and 181a-2) and therefore the adjusted read count is approximately half of the read count. b) excerpt of a typical *hairpin_sense_SA.grouped* file. In the single assignment files (_SA), instead of the adjusted read count column, the single assignment read count is shown (read count (SA)). Note that the read count (MA) is the total read count (without adjusting or single assignment). hsa-mir-6087 (red box) has the same read count for MA (multiple assignment) and SA. This means, that none of the reads mapped to the precursor was assigned to the mature sequence. Marked with the green box we can see that for mir-182, 708866 reads are mapped to the precursor sequences and most of them except 477 are assigned to the mature sequences. c) the alignment file for hsa-mir-7641-2 which shows that none of the shown reads coincides with the mature sequence (red subsequence of the precursor sequence). This might indicate that this is an incorrect microRNA.

Box2: Profiling of known microRNAs

There are several alignment parameters that can influence the results.

- **noMM='number':** the number of allowed mismatches (default = 0)
- **alignType=[v/n]:** v (the whole read is aligned), n (only the first L nucleotides are aligned, i.e. the seed)
- **seed='number':** the length of the seed if seed alignment is chosen (alignType=n)
- **winUpMiR='number of nucleotides' and winDownMiR='number of nucleotides':** these parameters define the windows around the reference mature sequence used to assign a read to this mature microRNA. That means that a read is assigned to a mature sequence if it starts at most **winUpMiR** nt upstream and ends at most **winDownMiR** downstream of the reference sequence.
- **isoMiR=true:** the program performs the classification of isoMiRs and generates an expression file only for the canonical sequences (canonical.txt). If isoMiR=true, then **alignType=n** should be set!

Profile the expression of 'other' microRNA references

By default, sRNAbench takes the 'mature.fa' and 'hairpin.fa' files from miRBase for microRNA detection. However, other microRNA annotations can also be used like for example miRGeneDB [14]. This database proposes a uniform system for the annotation and nomenclature of miRNA genes together with stricter criterion on what should be considered a microRNA. The microRNA names use a very similar species nomenclature, starting however with upper case letters.

```
sRNAbench input=/home/srna/data/nasopharyngeal_carcinoma/SRR2105938.fastq.gz
output=/home/srna/results/SRR2105938_miRGeneDB
adapter=TGGAATTCTCGGGTGCCAAGGG graphics=true microRNA=Hsa
mature=miRGeneDB_mature.fa hairpin=miRGeneDB_pre.fa
```

<u>Explanation of the command:</u>

- **microRNA=Hsa:** the microRNA names in miRGeneDB start with upper-case letters, i.e. we indicate the usage of human microRNAs.
- **mature=miRGeneDB_mature.fa:** sets the name of the file in the 'libs' folder that contains the mature (guide and passenger) sequences.
- **hairpin=miRGeneDB_pre.fa:** the name of the file with the precursor sequences.

<u>Results of the command:</u>

- **'results.txt'**: summary of the preprocessing and mapping statistics. In this case it can be used to compare some basic features between miRBase and miRGeneDB annotations.
    - Mature miRBase sequences have a total of 4,666,826 mapped reads (readsRCmatureSense=4666826) while miRGeneDB mature sequences are mapped by 4,167,527 reads (readsRCmatureSense=4167527). The miRGeneDB picks up 89% of the total miRBase expression. However, there are 2.5 times more mature sequences in miRBase release 21 (2588) compared to miRGeneDB (1,046).
    - The fact that miRBase might contain a considerable number of false positives can be also seen in the number of reads mapped to the precursor but not the mature sequence. This number is 15,811 in miRBase but only 1329 in

miRGeneDB. Like mentioned before, a high number of unique precursor reads which however are not mapped to the mature sequences can indicate false positives like the one shown in Figure 2c.

Profile microRNA expression in 'genome mode'

sRNAbench allows to map directly to the reference sequences ('library mode') but also to map first to the genome, quantifying the expression by mean of a genome annotation in bed/gff format (genome mode). The advantages and disadvantages have been discussed before [10].

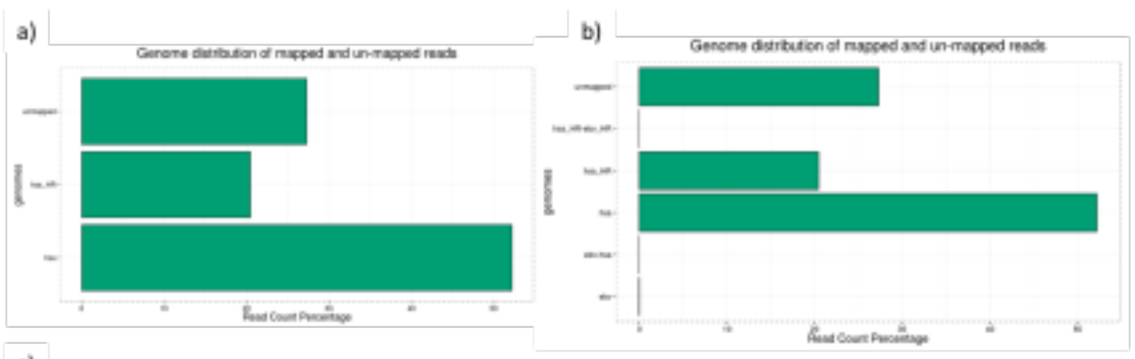
```
sRNAbench input=/home/srna/data/nasopharyngeal_carcinoma/SRR2105951.fastq.gz
output=/home/srna/results/SRR2105951_miRBase_genome
adapter=TGGAATTCTCGGGTGCCAAGGG graphics=true microRNA=hsa
species=hg38_prim_mp
```

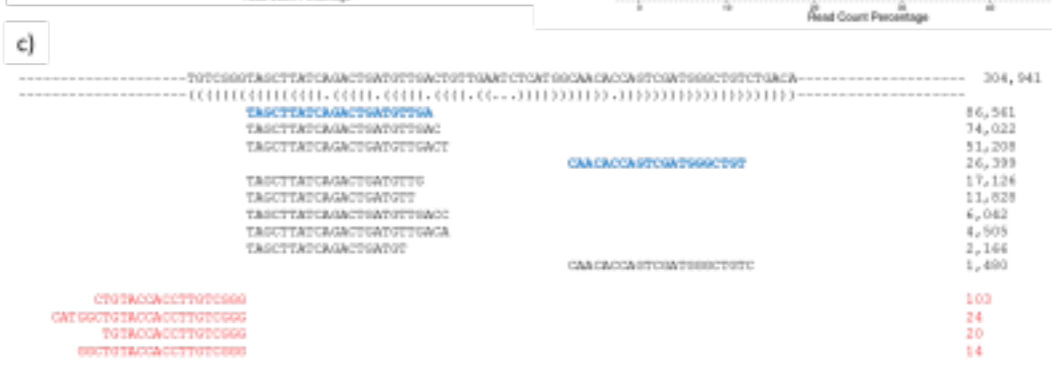
Explanation of the command:

- ***species='name of the bowtie index'***: this parameter allows specifying the name of the bowtie index that should be used to map the reads first against a genome.

Results of the command:

- The same expression files as when using the 'library mode'
- The file '***genomeDistribution.txt***' in the '***stat***' folder: This file shows the percentages of unique reads and read count mapped to the genome. Figure 3a shows the read count percentage of 'unmapped reads', 'highly redundant reads' (hsa_HR in the graphic; i.e. reads mapped to more than 10 loci by default but see box 3 on how to change) and mapped reads ('hsa' in the graphic). It can be seen that both, a rather high number of redundant reads and unmapped reads do exist. In the section 'Characterize unmapped reads: sRNAblast' we will show how to explore this finding in more detail.
- The '***reads.fa***' file: This file holds all unmapped reads at the end of the analysis. Note that highly redundant reads are also removed, i.e. not contained in this file. · The '***unassigned.fa***' file: in library mode, this file is equal to '***reads.fa***', however in genome mode it holds the reads that map to the genome, but that do not overlap with any used annotation.
- The '***assigned.fa***' file: the reads that have been mapped and assigned (note that in library mode, mapped and assigned is the same)





```
--------------------TGTCGGGTAGCTTATCAGACTGATGTTGACTGTTGAATCTCATGGCAACACCAGTCGATGGGCTGTCTGACA--------------------  304,941
--------------------((((((((((((((.((((.((((.((((.((...)))))))))).)))))))))))))))))))))--------------------
                           TAGCTTATCAGACTGATGTTGA                                                             86,561
                           TAGCTTATCAGACTGATGTTGAC                                                            74,022
                           TAGCTTATCAGACTGATGTTGACT                                                           51,208
                                                                 CAACACCAGTCGATGGGCTGT                        26,393
                           TAGCTTATCAGACTGATGTTG                                                              17,126
                           TAGCTTATCAGACTGATGTT                                                               11,828
                           TAGCTTATCAGACTGATGTTGACC                                                           6,042
                           TAGCTTATCAGACTGATGTTGACA                                                           4,505
                           TAGCTTATCAGACTGATGT                                                                2,166
                                                                 CAACACCAGTCGATGGGCTGTC                       1,490

         CTGTACCACCTTGTCGGG                                                                                   103
    CATGGCTGTACCACCTTGTCGGG                                                                                   24
          TGTACCACCTTGTCGGG                                                                                   20
       GGCTGTACCACCTTGTCGGG                                                                                   14
```

Figure 3: a) the genome distribution of sample SRR2105951. Rather high percentages of both, unmapped and highly redundant reads (those that map against many different loci) exist. b) genome distribution when adding Epstein-Barr virus in the profiling. It can be seen that EBV sequences can be detected, but at very low frequency. c) Alignment file of microRNA hsa-mir-21 when using up and downstream windows for profiling (the windows of 20nt are marked with '-' in the sequence and secondary structure above). The reads marked in red can only be detected (will only be assigned) when adding the flanking windows as otherwise, part of the read will lie outside the analyzed region (the precursor sequence)

The mapping in genome mode does offer another advantage; it allows setting windows or flanking regions around the reference sequences. For example, we can define a window around the precursor sequences in order to detect putative moRNA sequences [15]. All reads that lie completely within the reference sequence & the specified flanking regions are assigned (see figure 3c).

```
sRNAbench input=/home/srna/data/nasopharyngeal_carcinoma/SRR2105938.fastq.gz
output=/home/srna/results/SRR2105938_miRBase_genome_win20
adapter=TGGAATTCTCGGGTGCCAAGGG graphics=true microRNA=hsa
species=hg38_prim_mp winUpPre=20 winDownPre=20
```

Explanation of the command:

- ***winUpPre='number of nt'***: the flanking window upstream of the microRNA precursor that will be used to assign a read to the precursor.
- ***winDownPre='number of nt'***: the flanking window downstream of the microRNA precursor.

Results of the command:

- No new files will be generated by these parameters, however, the expression counts will change. Figure 3c shows the alignment of reads to the precursor sequence of hsa mir-21 plus the 20nt up-and downstream flanking windows specified in the command line. **Note that the secondary structure is calculated based only on the reference sequence, i.e. without the flanking sequences.**

Box 3: Genome mapping

By default, only reads that map at most to 10 different loci are used for any further analysis. All other reads are detected as genome mapped reads but are not considered for expression profiling. This behavior can be changed by means of two parameters:

'bowtieReportType=-a -m': This parameter must be given within single quotation marks on the command line and regulates the bowtie reporting behaviour. The default combination **–a –m** indicates that only reads that map at most **m** times to the genome are reported. Note that all type of parameters can be passed to bowtie, for example **'-k'** would be another possibility (see the bowtie manual for more details)

bowtieReportCount='number': this parameter specifies the corresponding number to the '**bowtieReportType**' parameter. For example,

- **'bowtieReportType=-a –m' bowtieReportCount=20** would pass to bowtie the following parameters **'-a –m 20'**
- **'bowtieReportType=-a -k' bowtieReportCount=5** would pass **'-a –k 5'** · If all mappings should be reported, **'bowtieReportType=-a ' 'bowtieReportCount=** ' needs to be specified on the command line (with a space after the =)

chunkmbs=number: the memory reserved for each chunk by bowtie. For longer and highly redundant reads, this number should be increased (default 256).

p='number': the number of threads used by bowtie and the parallelized parts of sRNAtoolbox programs.

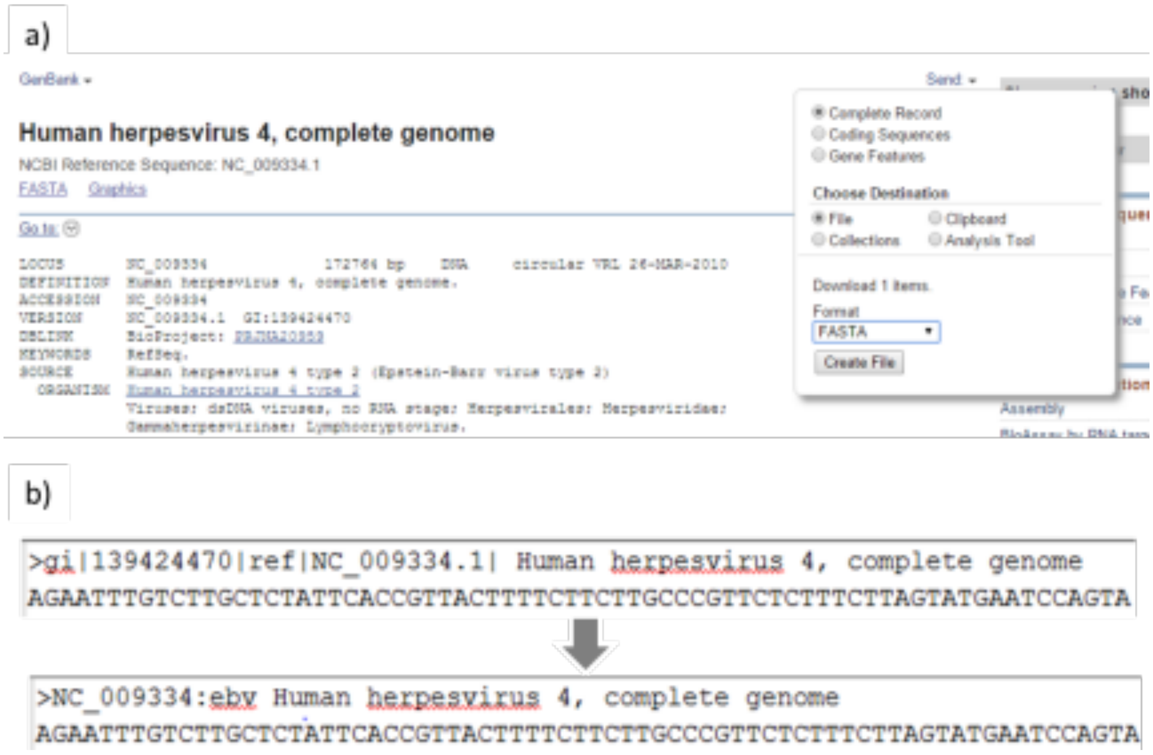
Set up a multi-species profiling: profile EBV sequences

It was reported that EBV virus infection might be related to nasopharyngeal cancer [11, 16], and therefore it would be interesting to quantify the content of EBV derived molecules in the samples. Although the Epstein-Barr virus is in our local database and could be easily downloaded by means of the **'Populate DB'** tool, we will show briefly how assemblies and annotations can be added to the local database.

- In http://www.ncbi.nlm.nih.gov/assembly/ we introduce the term 'Epstein-Barr virus' or 'Human herpesvirus 4'
- Above, in 'Global assembly definition' we can access the link 'NC_009334.1' http://www.ncbi.nlm.nih.gov/nuccore/NC_009334.1
- Download the genome sequence in fasta format (Figure 4a) into ***'/opt/sRNAtoolboxDB/index'*** (we will assume that the file name is NC_009334.fa) · Edit the header as shown in Figure 4b. The reason is that the '|' is used as internal separator by sRNAbench and should therefore not be used as sequence names. Furthermore, we add the **':ebv'** tag which will be used in the genome distribution statistics.

After obtaining the sequences, we need to generate the genome index and the required sequence object file (those need to be copied into the **'seqOBJ'** folder):

```
cd /opt/sRNAtoolboxDB/index
bowtie-build NC_009334.fa NC_009334
makeSeqObj NC_009334.fa
mv NC_009334.zip ../seqOBJ/
```



Human herpesvirus 4, complete genome

NCBI Reference Sequence: NC_009334.1

FASTA Graphics

Go to:

```
LOCUS       NC_009334             172764 bp    DNA     circular VRL 26-MAR-2010
DEFINITION  Human herpesvirus 4, complete genome.
ACCESSION   NC_009334
VERSION     NC_009334.1  GI:139424470
DBLINK      BioProject: PRJNA20959
KEYWORDS    RefSeq.
SOURCE      Human herpesvirus 4 type 2 (Epstein-Barr virus type 2)
  ORGANISM  Human herpesvirus 4 type 2
            Viruses: dsDNA viruses, no RNA stage; Herpesvirales; Herpesviridae;
            Gammaherpesvirinae; Lymphocryptovirus.
```

[✓ Complete Record] Coding Sequences Gene Features

Choose Destination

[✓ File] Clipboard Collections Analysis Tool

Download 1 items.

Format: FASTA

Create File

```
>gi|139424470|ref|NC_009334.1| Human herpesvirus 4, complete genome
AGAATTTGTCTTGCTCTATTCACCGTTACTTTTCTTCTTGCCCGTTCTCTTTCTTAGTATGAATCCAGTA
```

```
>NC_009334:ebv Human herpesvirus 4, complete genome
AGAATTTGTCTTGCTCTATTCACCGTTACTTTTCTTCTTGCCCGTTCTCTTTCTTAGTATGAATCCAGTA
```

Figure 4: a) download the EBV genome sequence from NCBI nucleotides database. b) the fasta headers should be adapted to sRNAbench, by i) eliminating the '|' and ii) by adding a species tag ':ebv'

Box 4: Populate the local database

General recipe to add annotations to the database:

Genome assemblies: make the bowtie indexes with ***bowtie-build*** (place them into the 'index' folder) and the java sequence object file with ***makeSeqObj*** (place the zip file into the seqOBJ folder)

Sequence annotation: those can be given in fasta format, as bowtie indexes or even in bed or gtf format (in genome mode ONLY). They are placed always in the ***'libs'*** folder.

sRNAhelper: This program can assist the user in setting up the local database for species which are not covered by our sRNAtoolbox database. For example, it can help to extract the sequences of a species (o whole genus, family ect) from RNAcentral [30] or genomic tRNA database [31] and format them for use with sRNAbench. Note that these annotations can be also generated using the appropriate helper tools from sRNAtoolbox web-server: http://bioinfo5.ugr.es/srnatoolbox

Now we can detect the reads that originate from Epstein-Barr virus:

```
sRNAbench input=/home/srna/data/nasopharyngeal_carcinoma/SRR2105951.fastq.gz
output=/home/srna/results/SRR2105951_miRBase_genome_EBV
adapter=TGGAATTCTCGGGTGCCAAGGG graphics=true microRNA=hsa:ebv
species=hg38_prim_mp:NC_009334
```

Explanation of the command:

- **microRNA=has:ebv** in order to profile microRNA from the Epstein-Barr virus and human simultaneously we only need to add its short species name (ebv) separated by ':'

- **species=hg38_prim_mp:NC_009334:** Add the EBV index**.** Just like above, in general we can specify as many genomes as we want. The reads are mapped simultaneously to all and only the best mappings are retained.

Results of the command:

- **genomeDistribution.txt:** it can be seen (figure 3b) that EBV sequences can be detected, but at very low levels.
- **mature_sense.grouped** and **mature_sense_SA.grouped:** EBV microRNAs can be detected, but and very low levels.

Given the low levels of EBV derived molecules, those cannot explain the high number of unmapped reads that we observed before. We will use another tools, ***sRNAblast*** in order to obtain more knowledge about the rather high number of unmapped reads.

Characterize unmapped reads: sRNAblast

This program allows using all ‘BLAST’ databases hosted at NCBI to obtain more information about the ‘unmapped reads’ which could reflect contamination, sequencing errors but also correspond to biologically important information like virus or bacterial infections.

```
sRNAblast input=/home/srna/results/SRR2105951_miRBase_genome/reads.fa
output=/home/srna/results/SRR2105951_blast maxReads=50
```

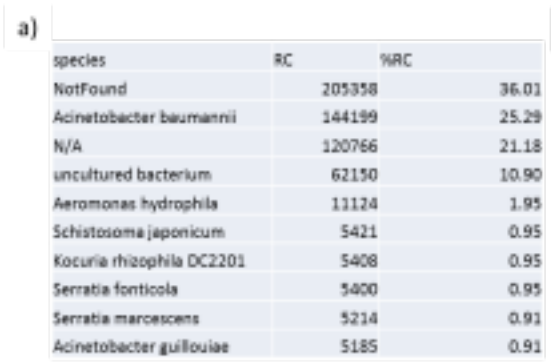
Explanation of the command:

- **input=’file’:** sRNAblast accepts the same input files as sRNAbench. In this example we use the ‘reads.fa’ file which holds in genome mode the reads that were not mapped to the genome
- **maxReads=’number’**: number of reads that will be ‘blasted’ (taken from an expression value ordered list). Blast is a very time intensive procedure. The query is executed in the NCBI servers, and therefore the execution time depends very much on the server load. Therefore, this number should not be too high.

Results of the command:

- **blast.out:** this file contains all reads mapped to a sequence in the database. Only the hits with the highest score are preserved.
- **species.out:** this file contains the read count (RC) and percentage per species. Note that if a read maps to several species, the read count is distributed between the species.
- **speciesSA.out:** the single assignment file of the species. The species.out file is sorted, and then each read that maps to several species is assigned only once; to the species with the highest number of mapped reads (Figure 5).
- **tax.out:** The read distributions per domains (superkingdoms)

a)



species	RC	%RC
NotFound	205358	36.01
Acinetobacter baumannii	144199	25.29
N/A	120766	21.18
uncultured bacterium	62150	10.90
Aeromonas hydrophila	11124	1.95
Schistosoma japonicum	5421	0.95
Kocuria rhizophila DC2201	5408	0.95
Serratia fonticola	5400	0.95
Serratia marcescens	5214	0.91
Acinetobacter guillouiae	5185	0.91

b)

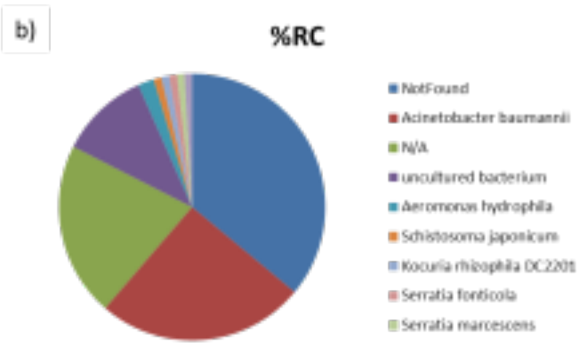


Figure 5: a) First lines of the *speciesSA.out* file and its graphical representation (b). 'NotFound' represents those reads that could not be mapped ('blasted') to this database. Note that these numbers are only based on the 50 most expressed reads that could not be mapped to the human genome.

The **speciesSA.out** file and its visualization can be seen in the figure 5a/5b. It can be seen that around 25% of the input reads (50 most expressed, unmapped reads) were assigned to *Acinetobacter baumannii*. This gram-negative bacterium behaves frequently as an opportunistic pathogen in humans, affecting people with weak immune systems. Moreover it is becoming increasingly important as a hospital-derived infection. In this case, it would be important to investigate further if this high number is a consequence of the pathology, or if it might be even a cause.

Box 5: sRNAblast

Several parameters can be passed to the **blastn** algorithm

blastDB='name of the NCBI blast database': by default the database 'nr' is used, but all NCBI blast databases can be used.

word_size='number' from NCBI manual: *Word size is roughly the minimal length of an identical match an alignment must contain if it is to be found by the algorithm (default 9)*
maxEvalue='value': the maximum E-value which will be reported in the output

Profile other smallRNAs

The annotations to profile other small RNAs are also downloaded by means of the populate tool and include cDNA and ncRNA from Ensembl, tRNAs from the genomic tRNA database and all RNA types from RNA central:

```
sRNAbench input=/home/srna/data/nasopharyngeal_carcinoma/SRR2105948.fastq.gz
output=/home/srna/results/SRR2105948_libs adapter=TGGAATTCTCGGGTGCCAAGGG
graphics=true microRNA=hsa libs=hg38_ncRNA libs=hg38_RNAcentral
libs=hg38_genomic_tRNA libs=hg38_cdna
'libsStringTypes=mature#sense;hairpin#sense|snRNA#sense|snoRNA#sense|tRNA#sense|rRN
A#sen
se|ncRNA#sense|protein_coding#sense;cdna#sense;cds#sense|protein_coding#antisense;cdn
a#ant isense;cds#antisense|piRNA#sense|repeat#sense'
'libsStringNames=miRBase(sense)|snRNA|snoRNA|tRNA|rRNA|ncRNA|cdna(sense)|mRNA(an
tisens e)|piRNA|repeats(sense)'
```

Explanation of the command:

- **libs='library name':** in library mode the annotations should be given either as fasta files or directly as bowtie indexes (index base names). If only the file name is given (without absolute path), sRNAbench supposes to find the files in '/opt/sRNAtoolDB/libs', i.e. in the **'libs'** folder of the local database (**'dbPath'**) · **libsStringTypes='string':** this parameter is ONLY important for the summary files. It specifies the categories that should be grouped in the summary files.

- **libsStringNames='string':** The names of the categories, i.e. this number of names must fit the number of types!

<u>Results of the command:</u>

- **'library name'_'orientation'.grouped files:** For each library, 4 expression files are produced: 2 for each orientation (sense and antisense) and 1 for each multiple mapping 'treatment' (read count adjusted and single assignment). For example '**hg38_ncRNA_sense.grouped**' holds the expression profiles for non-coding RNA annotations obtained from Ensemble mapped in sense direction (with adjusted read counts) while '**hg38_ncRNA_sense_SA.grouped'** contains the single assignment expression values.
- **'mappingStat.txt' (in 'stat' folder):** the summary file of the mapping process (the corresponding graphic is in the 'graphs' folder). See Figure 6 a/b.
- **The read length files in the 'stat' folder:** for each category (like tRNA, snoRNA) the length distribution of the reads is generated (see Figure 6 c).
- **rnaComposition_readLength.txt:** the RNA composition as a function of read length, i.e. allows answering questions like: which percentage of 33nt long reads are of tRNA origin?

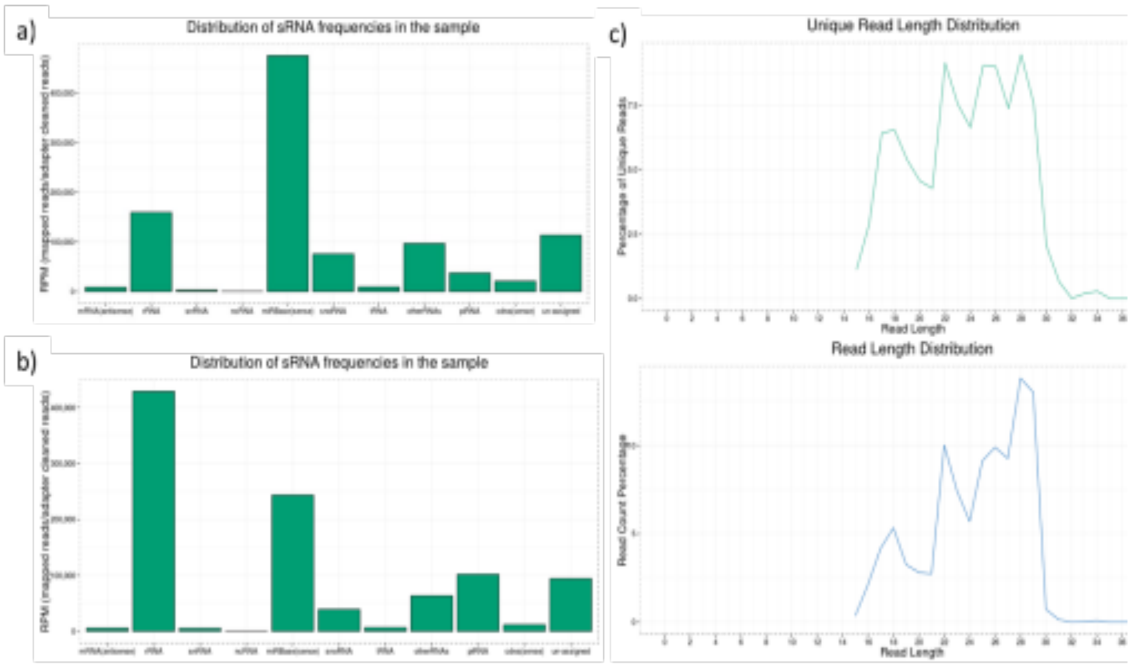


Figure 6: a) the mapping statistic as a function of RNA type using 'genome mode' and b) using 'library mode'. One clear difference between both distributions is the frequency of rRNA (ribosomal RNA). This is due to the default thresholds for multiple mappings. While in the genome mode, only reads with at most 10 mappings are reported, in the library all mappings are reported. Given that rRNA derived reads map normally to a high number of loci, in genome mode those highly redundant reads are not considered, hence the notable difference in frequency between both modes. c) the read length distribution for snRNA (small nuclear RNAs) fragments, above for unique reads and below for the read count. It can be seen that fragments from 18 nt to 30 nt exist.

Box 6*: other small RNAs*

Differences between library and genome mode:

Annotation types in library mode: only bowtie indexes or fasta files can be provided. Fasta files will be first converted to bowtie indexes internally.

Annotation type in genome mode: fasta annotations will be converted into an internal bed format (by mapping of the annotations against the genome); bed and gff/gff3 will be converted into an internal bed format; bowtie indexes are **NOT** converted into bed format, **BUT** the genome mapped reads are mapped again against those indexes.

Up-downstream windows (winUpLibs= and **winDownLibs=**): these parameters allow adding some flanking regions to the annotation for profiling which is not possible in library mode (note that this is similar to the flankings applied to the microRNA precursors shown above).

plotLibs=true: writes out the alignment files for all annotations given by **libs=**

Differential expression

To calculate differential expression, we will need to run first all samples (see DE section in protocol.txt). Then we can use the sRNAde program to generate the expression matrix calculating the differential expression statistics.

```
sRNAde input=/home/srna/results/ output=/home/srna/results/diffExpr_libs_genome
grpString=SRR2105938_libs_genome:SRR2105939_libs_genome:SRR2105940_libs_genome:SR
R2105
941_libs_genome#SRR2105948_libs_genome:SRR2105949_libs_genome:SRR2105950_libs_ge
nome: SRR2105951_libs_genome grpDesc=nasopharyngitis#carcinoma minRCexpr=5
sampleDesc=nasopharyngitis_38:nasopharyngitis_39:nasopharyngitis_40:nasopharyngitis_41:c
arcin oma_48:carcinoma_49:carcinoma_50:carcinoma_51
```

Explanation of the command:

- **grpString='the grouping instruction':** group string must contain the names of the different sRNAbench output folders in the following way: f1_1:f2_1#f1_2:f2_2 being f1_1 the first folder (sample) of the first group (controls in a case/control study), f2_1 the second folder of the first group, f1_2 the first folder of the second group (cases), etc…
- **grpDesc='string' (optional):** the groups (different experimental conditions, phenotypes, etc) that exist in the study. In our case we have two groups, individuals with **nasopharyngitis** and with **carcinoma**.
- **sampleDesc='string' (optional):** The names of the samples. The user can give each sample a name which will be used in the heatmaps. Otherwise, the folder names of the individual sRNAbench runs are used.
- **minRCexpr=5:** the minimum read count that ALL samples in at least one condition (group) must have.

Results of the command:

- **mature_sense_5_2.mat:** the expression matrix based on the ***mature_sense.grouped*** file (indicated by the base name 'mature_sense'). This file is used by default for differential expression analysis. The first number is determined by the **minRCexpr** parameter which was set to 5. The second number indicates the column that was used for the expression value. In this case the read count (second column) which is the required input for edgeR, DESeq and noiSeq.

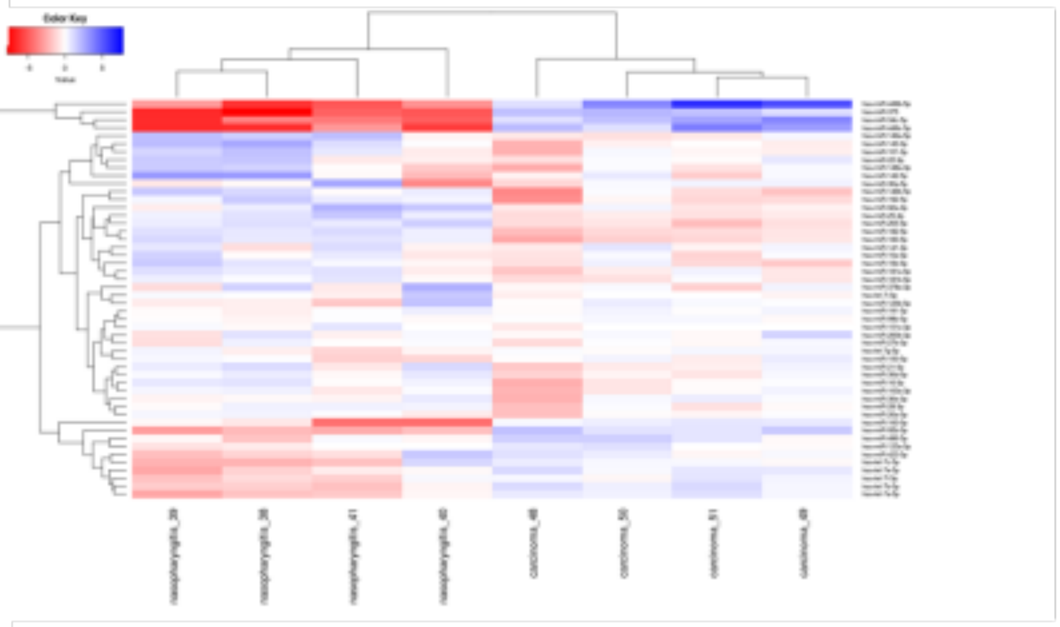
- **mature_Sense_5_4.mat:** the expression matrix based on the 4'th column (starting with 0), i.e. the library normalized RPM expression values.
- **mature_sense_5_2_nasopharyngitis_vs_carcinoma.xlsx:** the results when applying edgeR, DESeq and noiSeq to the expression matrix (mature_sense_5_2.mat). · **mature_sense_5_2_heatmap_perc0.78.png and mature_sense_5_2_heatmap_perc0.78_median_normalized.png (figure 7a) :** the heatmaps based by default in the 50 most expressed microRNAs (or annotations in general). The second is using a median normalization in order to obtain a clearer colour code.
- **mature_sense_5_2_TMM_normalized.tsv:** the expression matrix with the TMM normalized read counts (from edgeR)
- **sequencingStat.txt:** a summary of the sequencing, pre-processing and genome mapping process (Figure 7b).

Box 7: Differential expression

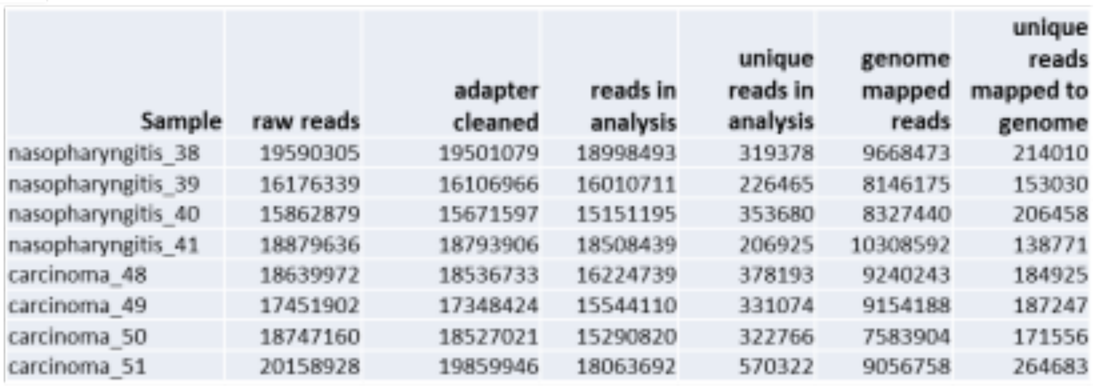
rmpCol='column of expression file 0-based': this value should be either 4 (library normalized RPM values) or 5 (total read count normalized RPM values) **diffExprFiles='expression file':** default is 'mature_sense.grouped', but any file generated with the **libs=** tag can be specified here.
makeSingleAssignDE=true: uses the single assignment expression values. **annotName='name of the category':** only needed if makeSingleAssignDE=true. The name of the category must be specified (for example, mature for mature_sense.grouped)

Warning!
Don't use the single assignment files for differential expression (like **diffExprFiles=mature_sense_SA.grouped**). If single assignment should be used, then **makeSingleAssignDE=true** needs be set.

a)



b)



Sample	raw reads	adapter cleaned	reads in analysis	unique reads in analysis	genome mapped reads	unique reads mapped to genome
nasopharyngitis_38	19590305	19501079	18998493	319378	9668473	214010
nasopharyngitis_39	16176339	16106966	16010711	226465	8146175	153030
nasopharyngitis_40	15862879	15671597	15151195	353680	8327440	206458
nasopharyngitis_41	18879636	18793906	18508439	206925	10308592	138771
carcinoma_48	18639972	18536733	16224739	378193	9240243	184925
carcinoma_49	17451902	17348424	15544110	331074	9154188	187247
carcinoma_50	18747160	18527021	15290820	322766	7583904	171556
carcinoma_51	20158928	19859946	18063692	570322	9056758	264683

Figure 7: a) a heatmap generated by the sRNAde program. It can be seen that both groups can be clearly separated. b) the study summary of sequencing and mapping generated by sRNAde.

Make study summary files

```
sRNAde input=/home/srna/results/ output=/home/srna/results/diffExpr_libs_genome_stat
grpString=SRR2105938_libs_genome:SRR2105939_libs_genome:SRR2105940_libs_genome:SR
R2105
941_libs_genome#SRR2105948_libs_genome:SRR2105949_libs_genome:SRR2105950_libs_ge
nome: SRR2105951_libs_genome grpDesc=nasopharyngitis#carcinoma minRCexpr=5
sampleDesc=nasopharyngitis_38:nasopharyngitis_39:nasopharyngitis_40:nasopharyngitis_41:c
arcin oma_48:carcinoma_49:carcinoma_50:carcinoma_51 diffExpr=false seqStat=false
stat=true statFiles=mappingStat.txt colData=6 minRCdata=0 folderData=stat
```

The sRNAde program can generate different summary files of a whole study. By default, the ‘sequencingStat.txt’ file is generated. In a very similar way, sRNAbench can summarize any file generated by sRNAbench. In the ‘stat’ mode sRNAde parses for each sample a given column out of the analysed file, and generates an ‘expression matrix’ type of file. This means that we will have a ‘key’ in the first column, and the ‘value’ for each sample in the other columns.

<u>Explanation of the command:</u>

- **stat=true:** activates the general study summary function.

- **statFiles='mappingStat.txt':** the file that should be summarized
- **colData=6:** the column that should be parsed out, i.e. the sixth column (starting counting with 0!) is the RPM value including the un-mapped/un-assigned reads (unmapped in library mode; genome mapped but not assigned to any annotation in genome mode)
- **minRCData=0:** the minimum expression value. In this case, we want no to set any limits on the RPM values.
- **folderData=stat:** the folder where the 'statFiles' are located. This will be normally either 'stat' or 'genomeDistribution'. If the folderData parameter is not set, then the 'statFiles' are expected in the root of the output folder.

Results of the command:

- **mappingStat_0_6.mat:** The matrix summarizing the sixth column of the mappingStat.txt file over all samples. The first column contains the RNA category, and the other columns the % of reads mapped to these categories in the different samples.
- **mappingStat_0_6_1_2.ttest:** the within and between groups statistics using the corresponding 'matrix file' (mappingStat_0_6.mat) applying the grouping provided by the user. The output has the following format
    - name: the name of the category (tRNA, snoRNA in this case)
    - mean_1: the mean value in group 1; nasopharyngitis in the example
    - stdev_1: the standard deviation in group 1
    - mean_2: the mean value in group2, carcinoma in the example.
    - stdev_2: the standard deviation in group 2.
    - P: the exact p-value using Fisher exact test.
    - FDR: the q-value (the corrected p-value)

Determine consensus target genes

sRNAtoolbox contains two programs to calculate consensus target gene predictions, one for animals and one for plants. These programs are especially useful for cross-species predictions (microRNA of species A vs. transcriptome of species B) or novel microRNAs (novel microRNA of species A vs. transcriptome of species A). In those cases, webservers of pre-calculated target predictions like TargetScan [17] or experimentally verified interaction databases like TarBase [18] cannot be used. One example would be to estimate the impact of EBV microRNAs in a human cell.

First we need to extract ebv microRNA sequences from the miRBase database. For this we use the sRNAhelper tool which includes several small helper tools.

```
sRNAhelper input=/opt/sRNAtoolboxDB/libs/mature.fa
output=/home/srna/results/BART1_mir search=BART1- mode=FA
```

Explanation of the command:

- **search='the search term':** all sequences that contain this search term in the name will be extracted.
- **mode=FA:** indicates which of the implemented small tools will be used; 'FA' means that certain sequences will be extracted from a bigger fasta file.

Results of the command:

- **mature.txt:** contains the extracted sequences. The output file is named like the original fasta file used to extract the sequences

```
miRNAconsTargets /home/srna/results/BART1_mir/mature.txt
/opt/sRNAtoolboxDB/utr/Homo_sapiens3p_UTR.fa /home/srna/results/BART1_targets/ 2
'TS:MIRANDA:PITA' '-s: : '
```

Explanation of the command:

The input data needs to be provided in specific order:

- **microRNA file**: the microRNA sequences in fasta format
- target sequences: i.e. the 3'UTR sequences in fasta format
- output_directory: the output directory
- p(threads): the number of CPUs that should be used (2 in the example) o program string: the programs that should be used; TS=TargetSpy, PITA=pita, MIRANDA=miranda. Note that pita is the program with the longest running time
- program parameters: the parameters that should be passed to the programs. Must be given in the same order than the program string. In this example, only TargetSpy receives a parameter, '-s' which enforces a seed region.

Results of the command:

- **pita.txt; ts.txt and miranda.txt:** the predicted targets of the three algorithms · **consensus.txt:** the consensus file giving the microRNA, the target sequence, the number and names of programs that predict this interaction

Note that for plant target predictions the program miRNAconsTargets_plants.py can be used in a similar way. This program uses Tapir and psRobot for target gene prediction.

Appendix 1: Installed programs

Analysis type	**Program(s)**	**Reference**
Sequence alignment	Bowtie, BLAST+	[19], [20]
sRNAtoolbox standalone programs	sRNAbench, sRNAde, sRNAblast, miRconsTargets, sRNAhelper	[10], [9]
Vienna package	prediction and comparison of RNA secondary structures	[21]
Target gene prediction (animals)	TargetSpy, pita and miranda	[22], [23], [24]
Target gene prediction (plants)	psRobot, Tapir	[25], [26]
Differential expression analysis	edgeR, DESeq and NOIseq	[27], [28], [29]
NGS related programs	SRA Toolkit	[12]

References

1. Ivey KN, Srivastava D: **MicroRNAs as regulators of differentiation and cell fate decisions.** *Cell Stem Cell* 2010, **7**:36–41.

2. Ebert MS, Sharp PA: **Roles for microRNAs in conferring robustness to biological processes.** *Cell* 2012, **149**:515–24.

3. Iorio M V, Croce CM: **MicroRNA dysregulation in cancer: diagnostics, monitoring and therapeutics. A comprehensive review.** *EMBO Mol Med* 2012, **4**:143–59.

4. Zhou M, Luo H: **MicroRNA-mediated gene regulation: potential applications for plant genetic engineering.** *Plant Mol Biol* 2013, **83**:59–75.

5. Bernal D, Trelis M, Montaner S, Cantalapiedra F, Galiano A, Hackenberg M, Marcilla A: **Surface analysis of Dicrocoelium dendriticum. The molecular characterization of exosomes reveals the presence of miRNAs.** *J Proteomics* 2014.

6. Buck AH, Coakley G, Simbari F, McSorley HJ, Quintana JF, Le Bihan T, Kumar S, Abreu Goodger C, Lear M, Harcus Y, Ceroni A, Babayan SA, Blaxter M, Ivens A, Maizels RM: **Exosomes secreted by nematode parasites transfer small RNAs to mammalian cells and modulate innate immunity**. *Nat Commun* 2014, **5**:5488.

7. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R: **The Sequence Alignment/Map format and SAMtools.** *Bioinformatics* 2009, **25**:2078–9.

8. Hackenberg M, Rodríguez-Ezpeleta N, Aransay AM: **miRanalyzer: an update on the detection and analysis of microRNAs in high-throughput sequencing experiments.** *Nucleic Acids Res* 2011, **39**(Web Server issue):W132–8.

9. Rueda A, Barturen G, Lebrón R, Gómez-Martín C, Alganza Á, Oliver JL, Hackenberg M: **sRNAtoolbox: an integrated collection of small RNA research tools**. *Nucleic Acids Res* 2015, **43**:W467–W473.

10. Barturen G, Rueda A, Hamberg M, Alganza A, Lebron R: **sRNAbench : profiling of small RNAs and its sequence variants in single or multi-species high-throughput experiments**. 2014, **1**:21–31.

11. Cai L, Ye Y, Jiang Q, Chen Y, Lyu X, Li J, Wang S, Liu T, Cai H, Yao K, Li J-L, Li X: **Epstein-Barr virus-encoded microRNA BART1 induces tumour metastasis by regulating PTEN-dependent pathways in nasopharyngeal carcinoma.** *Nat Commun* 2015, **6**:7353.

12. Staff SRAS: **Using the SRA Toolkit to convert .sra files into other formats**. 2011.

13. Kozomara A, Griffiths-Jones S: **miRBase: annotating high confidence microRNAs using deep sequencing data.** *Nucleic Acids Res* 2014, **42**(Database issue):D68–73.

14. Fromm B, Billipp T, Peck LE, Johansen M, Tarver JE, King BL, Newcomb JM, Sempere LF, Flatmark K, Hovig E, Peterson KJ: **A Uniform System for the Annotation of Vertebrate microRNA Genes and the Evolution of the Human microRNAome.** *Annu Rev Genet* 2015.

15. Langenberger D, Bermudez-Santana C, Hertel J, Hoffmann S, Khaitovich P, Stadler PF: **Evidence for human microRNA-offset RNAs in small RNA sequencing data.** *Bioinformatics* 2009, **25**:2298–301.

16. Chu EA, Wu JM, Tunkel DE, Ishman SL: **Nasopharyngeal carcinoma: the role of the Epstein Barr virus.** *Medscape J Med* 2008, **10**:165.

17. Agarwal V, Bell GW, Nam J-W, Bartel DP: **Predicting effective microRNA target sites in mammalian mRNAs**. *Elife* 2015, **4**.

18. Vergoulis T, Vlachos IS, Alexiou P, Georgakilas G, Maragkakis M, Reczko M, Gerangelos S, Koziris N, Dalamagas T, Hatzigeorgiou AG: **TarBase 6.0: capturing the exponential growth of miRNA targets with experimental support.** *Nucleic Acids Res* 2012, **40**(Database issue):D222–9.

19. Langmead B, Trapnell C, Pop M, Salzberg SL: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.** *Genome Biol* 2009, **10**:R25.

20. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL: **BLAST+: architecture and applications.** *BMC Bioinformatics* 2009, **10**:421.

21. Lorenz R, Bernhart SH, Höner Zu Siederdissen C, Tafer H, Flamm C, Stadler PF, Hofacker IL: **ViennaRNA Package 2.0.** *Algorithms Mol Biol* 2011, **6**:26.

22. Sturm M, Hackenberg M, Langenberger D, Frishman D: **TargetSpy: a supervised machine learning approach for microRNA target prediction.** *BMC Bioinformatics* 2010, **11**:292.

23. Kertesz M, Iovino N, Unnerstall U, Gaul U, Segal E: **The role of site accessibility in microRNA target recognition.** *Nat Genet* 2007, **39**:1278–84.

24. John B, Enright AJ, Aravin A, Tuschl T, Sander C, Marks DS: **Human MicroRNA targets.** *PLoS Biol* 2004, **2**:e363.
25. Wu H-J, Ma Y-K, Chen T, Wang M, Wang X-J: **PsRobot: a web-based plant small RNA meta analysis toolbox.** *Nucleic Acids Res* 2012, **40**(Web Server issue):W22–8.

26. Bonnet E, He Y, Billiau K, Van de Peer Y: **TAPIR, a web server for the prediction of plant microRNA targets, including target mimics.** *Bioinformatics* 2010, **26**:1566–8.

27. Robinson MD, McCarthy DJ, Smyth GK: **edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.** *Bioinformatics* 2010, **26**:139–40.

28. Anders S, Huber W: **Differential expression analysis for sequence count data.** *Genome Biol* 2010, **11**:R106.

29. Tarazona S, García F, Ferrer A, Dopazo J, Conesa A: **NOIseq: a RNA-seq differential expression method robust for sequencing depth biases**. *EMBnet.journal* 2012, **17**(B):18.

30. The RNAcentral Consortium: **RNAcentral: an international database of ncRNA sequences.** *Nucleic Acids Res* 2015, **43**(Database issue):D123–9.

31. Chan PP, Lowe TM: **GtRNAdb: a database of transfer RNA genes detected in genomic sequence.** *Nucleic Acids Res* 2009, **37**(Database issue):D93–7.