Version 1.5.1

- Documentation -

0. Change history

from	to	date	changes
1.0.0	1.0.1	08/05/2017	• updated internal links.
1.0.1	1.1.0	22/05/2017	• added `-riborase` option for most sensitive detection of rRNA fragments. • the option `-input` now accepts directories with input files in addition to standard file names. • automatic adjustment of FASTA headers with bad format in files provided via the option `-refseq`. • collapsed input FASTA files produced by unitas are sorted according to read counts. • minor improvements by adding the options -quick, -slow, -skip_ncRNA, skip_mapping • bug fixed: bad recognition of uncollapsed ELAND3 format. • bug fixed: bad detection of ping pong signature when not searching for phasiRNAs at the same time.
1.1.0	1.2.0	07/06/2017	• default value for estimated deviation when using only one replicate in differential expression analysis was changed from 0.15 to 0.05. • bug fixed: bad read out of total reads from runs without low complexity filter in differential expression analysis. • new precompiled executables that run on Windows 10 systems.
1.2.0	1.3.0	16/06/2017	• check for reads with length=0 in input files to avoid program termination • check for N-bases in 3' adapter prediction (do not allow them) • optionally allow polyA tails to be recognized as 3' adaper (option -trim_polyA) • bug fixed: bad recording of '-5p' or '-3p' origin during miRNA annotation
1.3.0	1.4.0	20/07/2017	• new output file that comprises annotations for each sequence (unitas.full_annotation_matrix.txt) • data from output file unitas.sncRNA_annotation is now split and saved in files unitas.annotation_summary.txt and unitas.hits_per_target.txt. • unitas can now parse GENSCAN ids from Ensembl cDNA files.
1.4.0	1.4.1	16/08/2017	• updated internal links. • bug fixed: errors when parsing sequence names that include special characters [?+*]
1.4.1	1.4.2	28/08/2017	• updated internal links. • unitas can now parse SNAP ids from Ensembl cDNA files.
1.4.2	1.4.3	04/09/2017	• bug fixed: error when using the option –species x
1.4.3	1.4.4	15/09/2017	• updated internal links. • bug fixed: multiple use of tRNA identifiers when parsing Ensembl ncRNA files lead to concatenation of distinct tRNA sequences.
1.4.4	1.4.5	02/11/2017	• implemented CCA-3' check/adding for tRNA reference sequences (disable with -noCCAcheck) • new option -CC_is_CCA allows identification of 3'-CCA-tRFs in poly-A trimmed libraries. • 3' tRFs no longer have to match the very 3' end of the tRNA. Now, up to 3 bp offset are allowed.
1.4.5	1.4.6	15/11/2017	• bug fixed: output file full_annotation_matrix.txt did only comprise one annotation per ncRNA class, even if the sequence read matched multiple reference sequences of one class (e.g. rRNA or tRNA).
1.4.6	1.5.0	21/12/2017	• allow optional skipping of mapping reads to non-species miRNAs with option -species_miR_only. • new output file when using option -pp (ppmatrix) allows identification of different (size) piRNA fractions. • new output file when using option -pp (ping-pong.fasta) lists all sequences with 10 nt 5' overlap. • running several multithreading-instances of unitas in the same working directory. • updated internal links.
1.5.0	1.5.1	11/01/2018	• bug fixed: bad output for differentially expressed sRNAs when skipping correction of alpha error.

1. Scope

`unitas` is a convenient tool for small non-coding RNA (sncRNA, typically ~18-40 nt in length) annotation using Next Generation Sequencing data. `unitas` uses latest sequence information from publicly available online databases to annotate user input sequences. No installation, no further prerequisites; it runs out-of-the-box on all popular platforms (Linux, MacOS, Windows) and can be started with one simple command from the command line (terminal).

2. Getting started

2.1 General remarks

All you need to start is a sequence file is FASTA or FASTQ format, or alternatively a map file in SAM or ELAND3 format (produced by SeqMap when using the option /output_all_matches, or by default by sRNAmapper), and a local copy of `unitas`. You can run the stand-alone `unitas` executable file (precompiled versions are available for Linux, MacOS and Windows) on your local machine without installation or any further requirements. Running the `unitas` Perl script on your local machine requires the installation of a Perl interpreter. Perl is pre-installed on common Linux and Mac systems. For Windows you can download and install a free Perl distribution such as StrawberryPerl (www.strawberryperl.com) or ActivePerl (www.activestate.com/activeperl/downloads). When

running `unitas` for the first time you need a connection to the internet in order to allow `unitas` to download a set of latest reference sequences. Subsequent runs can use previously downloaded data. You can start `unitas` from the command line (terminal) using the following command:

STAND-ALONE EXECUTABLES

Windows

```
unitas.exe -input sequence_or_map.file -species genus_species
```

MacOS or Linux

```
./unitas -input sequence_or_map.file -species genus_species
```

PERL SCRIPTS

Windows, MacOS or Linux

```
perl unitas.pl -input sequence_or_map.file -species genus_species
```

By default, `unitas` will not open more than one thread and multiple input files will be processed one after the other. We recommend to use the option `-threads [integer]` if more than on CPU core is available (which should be the case on most modern computers). During computation, `unitas` creates temporary files named, e.g., `-5388.unitas_pid` according to the process IDs of each forked process. `unitas` uses these temporary files to control the maximum number of parallel processes. Though this may not be the most sophisticated solution for this task, it is one that does not require the installation of additional Perl modules such as Parallel::ForkManager which are probably not part of a standard Perl distribution. You should not remove or rename these files while `unitas` is running. The files will be removed automatically when the annotation process for the input file is finished. You should also not run several instances of `unitas` in same directory.

2.2 Examples

2.1.1 Find (21nt and 24nt) phasiRNAs

You are a plant guy and have a small RNA map file in SAM or ELAND3 format. You want to find phasiRNAs and their source loci without doing all the other stuff (sequence annotation):

```
perl unitas.pl –skip_mapping -input map.file –phasi 21 –phasi 24
```

2.1.2 You want to annotate small RNAs from a 'non-Ensembl species'

You are lucky and there is a closely related species supported by unitas:

```
perl unitas.pl –input sequence_or_map.file –species related_species
```

```
or
```

Bad luck, there is no closely related species supported, but you have a set of reference sequences you want to use:

```
perl unitas.pl -input sequence_or_map.file –species x –refseq user_reference.fasta
```

2.1.3 You want to calculate ping-pong signatures

You are one of those piGuys and are interested in ping-pong signatures, particularly for those reads that are not fragments of any other non-coding RNA class:

```
perl unitas.pl –input map.file –species genus_species –pp
```

3. Workflow

3.1. Prior to annotation

`unitas` creates a data dump folder on your local machine to store reference sequences downloaded from the internet. The folder will be named UNITASrefdump_genus_species (e.g. UNITASrefdump_homo_sapiens when downloading human reference sequences). If present, subsequent annotation runs will use the information stored in this folder. You can force `unitas` to download the latest reference sequences using the option `-latest_ref`. This folder will contain a file named `db_versions.info` that provides information on database version or release date:

```
SeqMap version/date: .................. 1.0.13
Genomic tRNA database version/date: ... 30.05.2017 (dd.mm.yyyy)
piRNA cluster database version/date: .. 30.05.2017 (dd.mm.yyyy)
Ensembl version/date: ................. Release 88
EnsemblGenomes version/date: .......... Release 35
tRF-1 sequence data version/date: ..... 30.05.2017 (dd.mm.yyyy)
tRNA-leader sequence data version/date: 30.05.2017 (dd.mm.yyyy)
SILVA rRNA (SSU) database version/date: Release 128.1
SILVA rRNA (LSU) database version/date: Release 128.1
miRBase database version/date: ........ Release 21
```

3.1.1 Input file check and conversion

`unitas` will check the format of the input file(s) which must be provided in FASTA, FASTQ, SAM or ELAND3 format. You should not use multiple files with identical names from different folders. Input files will be converted into FASTA format. `unitas` uses collapsed input files in FASTA format as this greatly reduces computation time. Collapsed input files contain only non-identical sequences with FASTA headers referring to read counts of each sequence. If necessary, `unitas` will automatically collapse the input files provided by the user, e.g.:

```
>SRR029124.2054760 WICMT-SOLEXA_309ETAAXX:2:43:1367:1703 length=18
AGCGTGTAGGGATCCAAA
>SRR029124.2054761 WICMT-SOLEXA_309ETAAXX:2:43:1151:254 length=22
TAGCAGCACGTAAATATTGGCG
>SRR029124.2054763 WICMT-SOLEXA_309ETAAXX:2:43:1785:135 length=18
AGCGTGTAGGGATCCAAA
```

will be collapsed to:

```
>2
AGCGTGTAGGGATCCAAA
>1
TAGCAGCACGTAAATATTGGCG
```

In addition to reference sequences from online databases, the user can provide an arbitrary number of additional sequence files in FASTA format that will be used for sequence annotation (use the option `-refseq [sequence.file]`). The FASTA headers should have the following format:

```
>ncRNA_class|ncRNA_name
```

Where the ncRNA class (e.g. tRNA) is separated from its name (e.g. tRNA-Gly-GGT) with a pipe symbol. Do not use identical FASTA headers for different sequences. `unitas` will convert FASTA headers with bad format automatically using 'refseq' as description for ncRNA class and the original FASTA header for the ncRNA name.

3.1.2 3' adapter recognition and trimming

In most cases, sequence reads from NGS datasets comprise 3' adapter sequences. When using datasets from NCBI's Sequence Read Archive it can be difficult to figure out which adapter was used to create the dataset in question. When using the option `-trimm`, `unitas` screens for the most frequent 3' sequence motifs (default length = 12 nt but is adjusted automatically when sequence reads in input file are shorter). A first round of adapter trimming is then performed based on the identified motif allowing 2 mismatches for 12 nt motifs, 1 mismatch for motifs ≤ 11 nt and 0 mismatch for motifs ≤ 8 nt. If the original motif is not found within a given sequence read, `unitas` truncates the motif sequentially by one 3' nt and checks for its occurrence at the very 3' end of the sequence read until the motif is found or the motif length falls below 6 nt. Following this first round of adapter trimming, `unitas` checks the positional nucleotide composition of the trimmed sequence reads and will remove further 3' nucleotide positions in case they exceed a specified nucleotide bias (default = 0.8, change this value with the option `-trim_maxfinalbias`). Recently, a nice tool for adapter prediction named `DNApi` has been published by Tsuji and Weng (http://dx.doi.org/10.1371/journal.pone.0164228). Noteworthy, `unitas` and `DNApi` reported identical adapter sequences for the datasets that we have tested. However, if you prefer to use `DNApi` you can simply copy the software into the `unitas` folder. `unitas` will check for the presence of the `DNApi` Phython script (`dnapi.py`) and call it with a system command. Note that there might be additional dependencies for `DNApi`, at least you should

have Python installed on your computer. If, for what reason ever, adapter prediction with `DNApi` fails, `unitas` will continue adapter prediction with its own algorithm.

Relevant command line options:

option	default	explanation
`-trim`	`off`	Apply adapter recognition and trimming
`-trim_ignore_5p [integer]`	`22`	Ignore first *n* basepairs for 3' adapter prediction
`-trim_skip_reads [integer]`	`10000`	Skip the first *n* reads when searching for frequent motifs.
`-trim_check_reads [integer]`	`500000`	Check *n* reads for 3' adapter prediction.
`-trim_polyA`	`off`	Allow polyA tails as 3' adapter sequence.
`-trim_minlength [integer]`	`10`	Minimum length of trimmed read for further processing
`-trim_maxlength [integer]`	`50`	Maximum length of trimmed read for further processing
`-trim_maxfinalbias [floating point, 0..1]`	`0.8`	Maximum allowed 3' nucleotide bias after trimming

3.1.3 Low complexity filter

`unitas` can filter out low complexity sequences from the input file(s). This filter is an improved implementation of the duster tool provided as a part of the NGS TOOLBOX (http://www.smallrnagroup.uni-mainz.de/software/TBr2.zip). By default, it removes sequences that consist for ≥75% (change default value with option `-dust_limit [fraction]`) of simple sequence repeats with a minimum motif size of 1 nt and a maximum motif size of 5 nt (change default value with option `-dust_max [integer]`). In addition, unitas will filter out reads that exhibit a high fraction of N positions (default = 0.5, change this value with the option The filtering step can be skipped using the option `-skip_dust`.

Relevant command line options:

option	default	explanation
`-skip_dust`	`off`	Do not filter low complexity reads
`-dust_limit [floating point, 0..1]`	`0.75`	Maximum allowed fraction of simple sequence motifs
`-dust_max [integer]`	`5`	Maximum length of a simple sequence motif
`-dust_maxN [floating point, 0..1]`	`0.5`	Maximum allowed fraction of N nucleotides
`-dust_ignore_2bases_bias`	`off`	Ignore bias for 2 bases in sequence reads

3.1.4 Reference sequence download

`unitas` uses publicly available reference sequences for the species in question. Automatic sequence download is supported for all species that are included in the current Ensembl release (744 species or strains, 13-MARCH-2017). For other species you should select a closely related species or use the option `-species x` and provide separate reference sequence file(s) with the option `-refseq [sequence.file]` (see 3.1.1 for correct format). `unitas` connects to the Mainz University server to get the latest list of supported species. If that fails, `unitas` uses an internal list that may vary depending on the `unitas` version you use. Latest download links are also fetched from the Mainz University server or alternatively taken from an internal list. If download of the reference sequence from the respective database fails (in case of server downtime or outdated links [links on the smallRNAgroup server are updated weekly]), `unitas` downloads the required reference sequences directly from the Mainz University server where they are updated weekly. The following online sources are used for annotation (BE FAIR: Please cite the according references in addition to `unitas` when making use of `unitas` sequence annotation):

database name	data	reference
GtRNAdb	genomic tRNA sequences	Chan PP, Lowe TM. GtRNAdb: A database of transfer RNA genes detected in genomic sequence. *Nucleic Acids Res*. 2009 **37**:D93-D97.
SILVA rRNA database	genomic rRNA sequences	Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, et al. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res*. **41**:D590-D596.
piRNA cluster database	genomic piRNA cluster sequences	Rosenkranz D. piRNA cluster database: a web resource for piRNA producing loci. *Nucleic Acids Res*. 2016 **44**:D223-D230.
Ensembl	ncRNA/cDNA collection (miRNA sequences will be	Yates A, Akanni W, Amode MR, Barrell D, Billis K et al. Ensembl 2016. *Nucleic Acids Res*. 2016 44:D710-716.

	removed from ncRNA data)	
miRBase	miRNA (mature/precursor) sequences	Kozomara A, Griffiths-Jones S. miRBase: annotating high confidence microRNAs using deep sequencing data. *Nucleic Acids Res*. 2014 **42**:D68-D73.

3.1.5 SeqMap download and compilation

`unitas` uses SeqMap to perform many (not all) probe to reference mapping steps. The SeqMap source code (C++) is downloaded and compiled to an executable file via the g++ compiler which is part of most operating systems. The executable file is named `seqmap.exe` and will appear in the same directory as `unitas`, the source code is stored in the data dump folder (see 3.2.3). If compilation fails, `unitas` will download precompiled SeqMap executable from the Mainz University sever (On linux and Mac you may have to manually change file permissions in order to allow execution of the SeqMap file!). You can also manually download the files from here: http://www-personal.umich.edu/~jianghui/seqmap/. Save a local copy of SeqMap to the same directory as `unitas` and rename it into `seqmap.exe` (always use the extension .exe since `unitas` will do a system call). Please do not forget to cite the SeqMap paper when making use of `unitas` sequence annotation: Jiang H, Wong WH SeqMap: Mapping Massive Amount of Oligonucleotides to the Genome. 2008 *Bioinformatics* **24**:2395-2396.

3.2 The annotation process

`unitas` can process an arbitrary number of sequence files in parallel. By default, multiple input files are processed one after another. For multithreading use the option `-threads [integer]`. For each input dataset, `unitas` creates a results dump folder, which is named according to the current date and input file name (plus consecutive numbering), e.g.: `UNITAS_01-01-2017_input.fasta_#1`. Result files and temporary files are stored in this folder. By default, temporary files (SeqMap reports, SeqMap input/output files other reference sequence files) are removed after annotation. To keep temporary files use the option `-keep_temp`.

3.2.1 miRNA annotation

Mature miRNA sequences from miRBase are loaded into memory. Matches to miRNAs from the species in question and matches to miRNAs from different species will be reported separately. In the first step, `unitas` searches for perfect full length matches to mature miRNAs of the species in question (canonical miRNAs). Subsequently, non-matching sequences are mapped to miRNA precursor sequences of the species in question in order to identify non-canonical miRNAs. Still not matching sequences are trimmed at the 3' end to check whether they represent 3' tailed canonical or non-canonical miRNAs. By default, `unitas` allows up to 2 non-template 3' nucleotides. This value can be changed using the option `-tail [integer]`. Then, the same procedure is performed for the remaining sequences and miRNAs from other species. In summary, the order of miRNA annotation is:

1. untailed canonical miRNA from the species in question
2. untailed non-caninical miRNA from the species in question
3. 3' tailed canonical miRNA from the species in question
4. 3' tailed non-canonical miRNA from the species in question
5. untailed canonical miRNA from other species
6. untailed non-caninical miRNA from other species
7. 3' tailed canonical miRNA from other species
8. 3' tailed non-canonical miRNA from other species

Examples:

```
1.    UUCAAGUAAUCCAGGAUAGGCU
2.    UUCAAGUAAUCCAGGAUAGGCUUU
3.     UCAAGUAAUCCAGGAUAGGCU--G
4.    UUCAAGUAAUCCAGGAUAGGCU--GA
      G      U          C          --G  CA
5' GUG CCUCGU CAAGUAAUC AGGAUAGGCU   UG  G
   ||| |||||| ||||||||| ||||||||||   ||   G
3' CGC GGGGCA GUUCAUUGG UCUUAUCCGG   AC  U
      A      C          U          GUA  CC
```

This means that, e.g., when using human input sequences, a sequence is annotated as 3' tailed canonical human miRNA instead of untailed canonical mouse miRNA, even though both alternatives are possible.
`unitas` outputs tab-delimited miRNA annotation tables (e.g. `unitas.miR-table_Human.txt` and `unitas.miR-table_non-Human.txt`) that contains the following information: miRNA name (e.g. miR-376b-3p), miRNA sequence, total number of reads, number of reads without tailing (+N0-3'), number of reads with one non-template 3' nucleotide (+N1-3') [... number of additional columns depends on the number of allowed non-template 3' nucleotides], number of A-tailed reads, number of T-tailed reads, number of G-tailed reads, number of C-tailed reads [... number of additional columns depends on the number of allowed non-template 3' nucleotides, e.g. AA-tailed reads etc.].
In addition, `unitas` outputs files (e.g. `unitas.miR-modifications_Human.txt` and `unitas.miR-modifications_non-Human.txt`) that summarize information on non-template 3' ends in the following simple format (tail, read counts):

```
3'-tailings (non-template nucleotides)
A       33596
AA      3143
AAA     855
AAC     114
AAG     347
AAT     547
[...]
```

This file also summarizes internal modifications that were identified during miRNA annotation in the following format:

```
Internal modifications (?->[ATGCN] = mapped sequence exceeds precursor sequence)
G->T    53898
C->T    13998
T->G    13514
T->C    10506
A->G    8916
T->A    6270
[...]
```

It further gives you information on the positions where modifications occur like this:

```
Positions were internal modifications occur
1       2481
2       1638
3       3030
4       1589
5       1706
6       3580
[...]
```

Finally, it provides a table that combines modifications and positions in the following style:

```
Modifications per position
        G->T    C->T    T->G    T->C    A->G    T->A    A->C    G->A    C->A    G->C
1       425     108     251     403     103     247     153     110     184     258
2       332     42      51      109     156     40      360     193     43      202
3       1047    100     69      36      90      11      175     771     167     499
4       177     263     85      39      104     17      153     60      495     67
5       607     116     216     194     62      74      70      124     84      98
6       398     211     1351    414     218     101     97      139     133     72
[...]
```

Relevant command line options:

option	default	explanation
-skip_miR	off	Skip miRNA annotation
-species_miR_only	off	Skip mapping to non-species miRNAs
-tail [integer]	2	Maximum number of non-template 3' nucleotides
-intmod [integer]	1	Maximum number of internal modifications

3.2.2 ncRNA/mRNA fragments

Input sequences that cannot be annotated as miRNA sequence are mapped to ncRNA/cDNA sequences in sense orientation to identify fragments of larger ncRNA classes or fragments of mRNA. By default the number of allowed mismatches is 1 and the number of allowed insertions/deletions is 0. You can change the default value with the option `-mismatch [integer]` and `-insdel [integer]`, respectively. The maximum possible number of allowed mismatches is 5, the maximum possible number of allowed insertions/deletions is 3. Note that increasing the number of allowed mismatches and insertions/deletions will significantly increase memory usage and computation time. When allowing mismatches, only the best alignments in terms of mismatch counts will be considered for sequence annotation. Read counts of sequences that match different reference sequences (with equal alignment quality) are apportioned accordingly. As a result, the values for read counts per class may be decimal numbers rather than integers.

Mapping to cDNA is performed by default, but can be skipped using the option `-skip_cDNA`. Considering tRNA-derived sequences, `unitas` further classifies sequences into 5' tRFs, 5' tR-halves, 3' tRFs, 3' CCA-tRFs, 3' tR-halves, tRF1, tRNA-leader and misc-tRFs based on alignment position and sequence length (read counts for genomic and mitochondrial tRNA-derived sequences are listed separately). In addition, unitas creates a table (`unitas.tRF-table.txt`) that lists which fragments are processed from which tRNAs.

`unitas` outputs FASTA files for each class of annotated sequences (miRNAs, tRNA, rRNA etc.). Note that one sequence can occur in more than one output file if the sequence maps to different reference sequences. Sequences in the FASTA output files are sorted according to read counts in descending order, e.g.:

```
>5044
TCCCTGGTGGTCTAGTGGTTAGGATTCGGCGCT
>781
GCCCGGATAGCTCAGTCGGTAGAGCATCAGAC
>774
CGCGGGAGACCGGGGTTCGATTCCCCGACGGG
>647
GCGCCGCTGGTGTAGTGGTATCATGCAAG
```

[...]

For sequences that map to genomic piRNA clusters, the FASTA headers contain additional information that refers to the genomic coordinates of the piRNA producing locus, e.g.:

```
>46:Chr6 4863485-4881004
TAGGCGAATCTAGGGTATTTCAACGATGCA
```

The FASTA files are stored in the results dump folder (see 3.2) and are named e.g. `unitas.miR.Human.fas` when using human input sequences. Each FASTA file is accompanied by a file that contains information on length distribution and positional nucleotide composition for the sequence reads assigned to this class, e.g.:

```
LENGTH DISTRIBUTION
length  reads
18      271
19      263
20      173
21      902
22      1675
23      729
24      311
```

[...]

```
POSITIONAL NUCLEOTIDE COMPOSITION
pos.    A       C       G       T
1       1963    2406    11144   7520
2       472     16896   4775    890
3       2277    13295   5735    1726
4       1613    12796   3578    5046
5       667     4171    7075    11120
6       2124    1965    17564   1380
```
[...]

The output file `unitas.annotation_summary.txt` provides a table that summarizes sequence annotation (normalized reads per class) which looks like this:

```
low_complexity                          1244
miRNA                                   80727
   miRNA:homo_sapiens                   76338
   miRNA:other                          4389
rRNA                                    100575.918129203
   genomic_rRNA                         100367.418129203
   Mt_rRNA                              208.5
tRNA                                    22867.0863247866
   genomic_tRNA                         22846.0863247866
      5'tR-halves                       12672
      5'tRFs                            4800.75299145299
      3'tR-halves                       64
      3'tRFs                            23.7142857142857
      3'CCA-tRFs                        67
      tRF-1                             21
      tRNA-leader                       4
      misc-tRFs                         5218.61904761903
   Mt_tRNA                              21
      5'tR-halves                       1
      5'tRFs                            4
      3'tR-halves                       1
      3'tRFs                            2
      3'CCA-tRFs                        0
      tRF-1                             0
      tRNA-leader                       0
      misc-tRFs                         13
pseudogene                              2.18972986748216
ribozyme                                8
antisense                               73.9023104237751
snoRNA                                  849.666666666667
lincRNA                                 1202.5266023332
misc_RNA                                807.009174311925
protein_coding                          4682.66446764933
snRNA                                   4349.3477084787
scaRNA                                  45
no annotation                           37381
   mapped to piRNA producing loci       1265
```

Another file lists read counts per reference transcripts (`unitas.hits_per_target.txt`):

```
TRANSCRIPT_CLASS            TRANSCRIPT_NAME       NORMALIZED_READ_COUNTS
3prime_overlapping_ncRNA    CTD-2196E14.9         3.57337335772603
3prime_overlapping_ncRNA    RP11-473I1.9          2.0014450867052
3prime_overlapping_ncRNA    RP11-373L24.1         1.30392156862745
3prime_overlapping_ncRNA    RP11-134G8.10         0.333333333333333
3prime_overlapping_ncRNA    CTD-2651B20.1         0.0206557649767618
3prime_overlapping_ncRNA    LINC00846             0.00144508670520231
IG_V_pseudogene             IGHVII-28-2           0.0606060606060606
IG_V_pseudogene             IGHVII-30-43          0.0417693316175626
IG_V_pseudogene             IGHVII-20-3           0.0113937832958541
IG_V_pseudogene             CH17-60O17.11         0.00377588629090349
IG_V_pseudogene             IGHVII-44-2D          0.00377588629090349
Mt_rRNA                     MT-RNR1               7953.66666666667
Mt_rRNA                     MT-RNR2               3363.69603174603
Mt_tRNA                     MT-TP                 8870.83333333333
```
[...]

This list is sorted alphabetically according to transcript classes (capital letters first). The order of transcripts within one class depends on the number of normalized read count.

Relevant command line options:

option	default	explanation
-mismatch [integer]	1	Maximum number of mismatch to reference
-insdel [integer]	0	Maximum number of insertions/deletions
-riborase	off	Map sequences to a complete collection of rRNA sequences from NCBI nucleotide database. This is recommended to identify rRNA degradation products in species with low quality ncRNA annotation.
-CC_is_CCA	off	Treats CC-3' ends as CCA-3' ends when matching to tRNAs. Use this option when you removed poly-A tails from your raw sequences.
-noCCAcheck	off	Use this option if you want to map against tRNA sequences as they are deposited in the databases. By default unitas will build CCA-3' ends to reference tRNA sequences.

3.2.3 Searching piRNA candidates

Piwi interacting RNAs (piRNAs) typically pass the described pipeline without producing a match to any known ncRNA or cDNA. Thus, non-annotated sequences (saved in the output file `unitas.no-annotation.fas`) are subsequently mapped to known piRNA producing loci of the species in question (if available). This step is performed by default, but can be skipped using the option `-skip_piR`. Searching for piRNA candidates is particularly insightful when handling data derived from germ line tissues, or tissues that were shown to contain noteworthy amounts of piRNA-like molecules (e.g. brain or epididymis). In this case, we further recommend using sequences without annotation (saved in the output file `unitas.no-annotation.fas`) for subsequent piRNA cluster prediction with proTRAC.

When using map files as input you can also search for ping-pong signatures within your sequence data. When using the option `-pp`, `unitas` will check and report the 5' overlaps of mapped sequence reads and calculate a Z-score for the enrichment of 10 bp overlaps according to Zhang et al. 2011, *Mol Cell* 44:572-584.

3.2.4 Searching phased small RNAs (phasiRNAs)

When using map files as input you can search for phased small interfering RNAs (phasiRNAs). Use the option `-phasi` [*n*] to search for phasiRNAs with length *n*. You can use the option multiple times in one command, like this:

```
-phasi 21 -phasi 24
```

If you only want to search for phased RNAs without doing all the other annotation steps before you should make use of the otions -skip_dust, -skip_miR, -skip_piR, -skip_cDNA and skip_ncRNA.

`unitas` will scan the map file with a sliding window (default size = 1 kb) and save coordinates and read counts of putative phasiRNAs and all other mapped sequence reads. The coordinates of putative phasiRNAs (that is all reads with correct length) from plus and minus strand are unified to check for phasing on both strands. Each sliding window is subsequently analyzed for the presence of phasiRNAs with significant read count over the background. Significance is calculated based on the following formula:

$$p = 1 - \left(\sum_{k=0}^{j} \binom{n}{k} q^k (1-q)^{n-k} \right)$$

in which j refers to the observed number of reads with length i in a specified phase, n refers to the total number of reads with length i and q is given by 1/i and refers to the probability of a read to be located in a given phase, assuming that a sequence read can map to any position within the sliding window with equal probability. `unitas` employs strict Bonferroni correction for multiple testing by means of the number of analyzed sliding windows, that is the number of sliding windows with at least one mapped sequence. Adjacent sliding windows (default < 1 kb)

with significant enrichment for phasiRNAs are merged to phasiRNA clusters. Besides significant enrichment of phasiRNAs there is a series of further thresholds (see below for command line options) that aim to reduce the false positive prediction rate. `unitas` produces two output files in addition to the obligatory FASTA and .info file. A file named `unitas.phasiRNA.sorted.fas` comprises the same sequences that are stored in the FASTA file, but sorted according to the identified phasiRNA clusters. Note, that in this file sequences may occur multiple times when they can be assigned to more than one phasiRNA cluster. A file named `unitas.phasiRNA.align` comprises information on predicted phasiRNA clusters with aligned phasiRNAs. The alignment file will look like this:

```
cluster_ID:1  location:chr01  start:2301  end:2386  size:86 bp  non-identical_sequences:5  reads:61  reads_normalized:50
 2/2                                                                 ATCGATAACTTGGTTCAGCAA
 6/1                                            CAGCATCGGTTTGAAGGTGCG
22/1                       GACACTGGAGATTCCACCGGA
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
11/1 ACACAGGTGTTGCTTTCGAAA
20/2                     AGCTGTGACCTCTAAGGTGGC
```

At the beginning of each cluster you can see some information on the cluster location and aligned sequences and reads. For each sequence of the cluster, `unitas` gives information on read count and genomic hits at the very left. In this example, the 2nd sequence from the top has six reads and one genomic hit. This results in a normalized read count of six. For the 5th sequence from the top (20 reads, two genomic hits) the normalized read count is 10. This is why the value for the normalized read count is below the absolute read count in most cases. In the depiction of the cluster, both genomic strands are displayed with a stretch of > for plus-, and a stretch of < for minus strand. Importantly, sequences that map to the minus strand (below the stretches of > and <) are displayed in reverse (3'-5') orientation.

Relevant command line options:

option	default	explanation
-phasi [integer]	off	Expected length of phasiRNAs
-na_for_phasi	off	Only consider sequences without annotation for prediction of phasiRNAs
-min_phasi_fraction [floating point, 0..1]	0.25	Minimum fraction of phasiRNAs in relation to all mapped reads within a sliding windows
-min_phasi_reads [floating point]	0	Absolute minimum of phased reads within a sliding window
-min_phasi_loci [integer]	5	Minimum number of different phasiRNA loci within a sliding window
-min_phasi_in_line [integer]	0	Minimum number of phasiRNAs directly adjacent to each other
-min_distance_between_clusters [integer]	1000	Minimum distance between two phasiRNA clusters [bp]
-max_allowed_strandbias [floating point, 0..1]	0.95	Maximum allowed strand bias of phasiRNA reads
-phasi_sw_size [integer]	1000	Size of the sliding window during phasiRNA prediction [bp]
-phasi_p [floating point, 0..1]	0.05	Critical value for significant enrichment of phasiRNAs
-calc_big_factorials	off	Allow `unitas` to calculate factorials >100. Will increase memory usage and computation time. May increase sensitivity.

3.3 Searching differentially expressed sequences

`unitas` allows to search for differentially expressed small RNAs after sequence annotation. To test for differentially expressed small RNAs, you have to define two groups that you want to compare (e.g. two groups of replicates). For all the datasets there must be a `unitas` output folder. The command would look something like this:

```
perl unitas.pl -diffexpr -g1 UNITAS_01-01-2017_file1.fas_#1 -g2 UNITAS_01-01-2017_file2.fas_#1
```

With the option `-g1` and `-g2` you can assign the different output folders to group 1 (`-g1`) and group 2 (`-g2`), respectively. You can use and arbitrary number of output folders for each group like this:

```
-g1 UNITAS_01-01-2017_fileA.fas_#1 -g1 UNITAS_01-01-2017_fileB.fas_#1
```

By default the results will be written into the files `unitas.diff-expr.up1` and `unitas.diff-expr.up2` where the first file lists sequences with significantly higher abundance in group 1 and the second file lists sequences with significantly higher abundance in group 2. Both files will have the following format:

```
sequence           abs_folder1  abs_folder2  rpm_folder1  rpm_folder2  avg_group1  avg_group2  p-value  corr_p-value  class
AAAAGAACTTTGAAGAGA 36           10           16.98        5.39         16.98       5.39        0.000034 0.02992343    rRNA
```

where the first column gives the small RNA sequence. The next columns refer to absolute read counts for this sequence in the different samples (`unitas` output folders) followed by the corresponding rpm values. The number of these columns depends on the number of input samples (`unitas` output folders). The next two columns refer to the average rpm values for samples in group 1 and samples in group 2. In this case the values are identical because there is only one sample per group (note that you should really use more than one replicate per group for this kind of analysis). The next columns refer to the raw p-value and the alpha-corrected p-value. The final column gives you information to which ncRNA class this small RNA belongs (can be more than one class).
The statistical test performed for detection of differentially expressed small RNAs in principal bases on the assumption that the number of sequence reads observed in biological replicates for one condition (variance across samples, individuals, RNA extraction- or library preparation procedures) is roughly normally distributed. To calculate the variance for one condition you should not use less than two replicates for one condition (one group). Otherwise, there is no possibility to get a reasonable estimation of the biological variance occurring between samples of the same condition. However, `unitas` will also perform tests with only one replicate, in this case using predefined minimum values of variance (standard deviation) depending on dataset size and the observed rpm value for each sequence. The predefined minimum values correspond to the standard deviation one would expect to observe when the identical dataset would be resampled randomly (e.g. doing resequencing of the same library). They are calculated according to the formula:

$$\sqrt{\frac{r}{10^6}}^{-1} \cdot a^{-1}$$

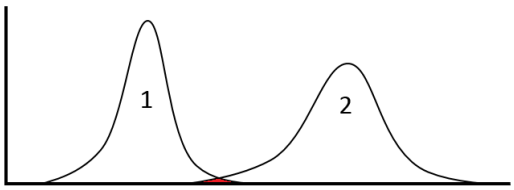
where *r* represents the total number of reads for a replicate and *a* represents the rpm value for the sequence in question. These minimal values for standard deviation are also employed when providing more than one replicate per group, which is relevant for those cases where the rpm values for one sequence in different replicates are by hazard (almost) identical. When you have no possibility to use more than one replicate for a condition (group), we recommend that you give an estimation on the deviation that you *would* expect across replicates (rather than relying on the minimum standard deviation values). You can do this using the option `-diffexpr_estdev [0..1]`. As an example, you may estimate that when you had two additional replicates, the rpm value for a given small RNA sequence could be 20% higher or 20% lower compared to the replicate that you have. In this case, you would have to use the option like this:

```
-diffexpr_estdev 0.2
```

Doing so, `unitas` will assume that rpm values can be 20% higher or lower. You can think of it as adding two virtual replicates to the group that has only one replicate, one comprising read counts 20% higher, one comprising read counts 20% lower).
Having the expectation value (average read counts across replicates) and standard deviation of rpm values across replicates, the probability density function of rpm values can be displayed according to the formula:

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the expectation (or mean) value and σ is the standard deviation. When we now compare the probability density function for the same sequence across two different conditions (1 and 2) we should observe two graphs that look like this:



In theory, these graphs represent the distribution of rpm values that one would expect when we have an infinite number of replicates for each condition (x-axis: reads or rpm, y-axis: number of replicates). The shared area under both graphs (red), in relation to the remaining integrals of both graphs, represents the probability to observe the same rpm values in both conditions. Accordingly, we can assume that read counts for one sequence are significantly different across two conditions when the shared area is below a critical value (e.g. 0.05, with an overall integral of 1).

Relevant command line options:

option	default	explanation
-diffexpr	off	Search for differentially expressed small RNAs
-diffexpr_p [floating point, 0..1]	0.05	Critical p-value for differential expression
-diffexpr_acorrection [0;1;2]	1	Chose how to correct alpha error. 0: no correction. 1: Bonferroni-Holm procedure. 2: Strict Bonferroni correction.
-diffexpr_estdev [floating point, 0..1]	0.05	Estimate standard deviation when you have only one replicate for a condition.
-g1 [folder name]	n.a.	Specify `unitas` output folder for group 1
-g2 [folder name]	n.a.	Specifiy `unitas` output folder for group 2

4. Special output files

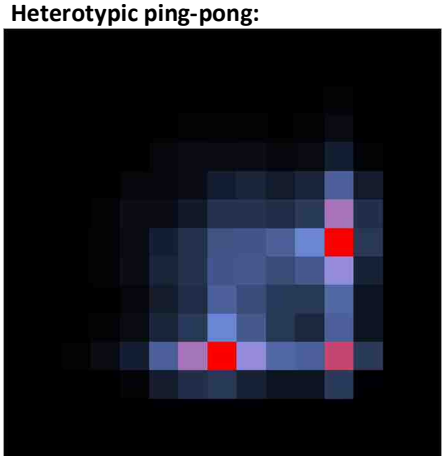
4.1 unitas.tRF-table.txt

This file lists all tRNAs that were found to produce tRFs. The columns report sequence read counts for the different types of tRFs. For each tRF-type you will see fractionated and absolute values. Fractionated values account for the fact that one sequence may map to multiple mature tRNA sequences.

4.2 unitas.[RNAclass].ppmatrix

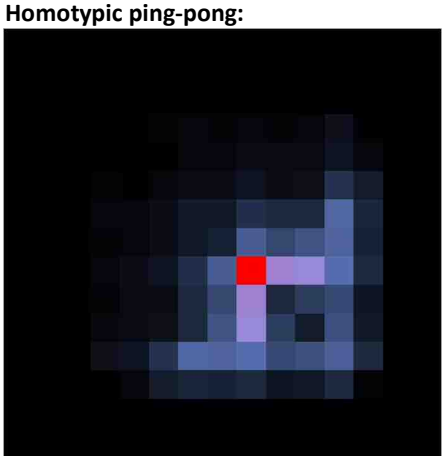
This file represents a matrix that gives information on scores for 10 nt 5' overlaps separated by sequence length. This means that you will see which size of sequences is responsible for a putative ping-pong signal. Converted into an image, you may observe one of the following scenarios (x-axis and y-axis correspond to sequence length [22nt-30nt]):

Heterotypic ping-pong:



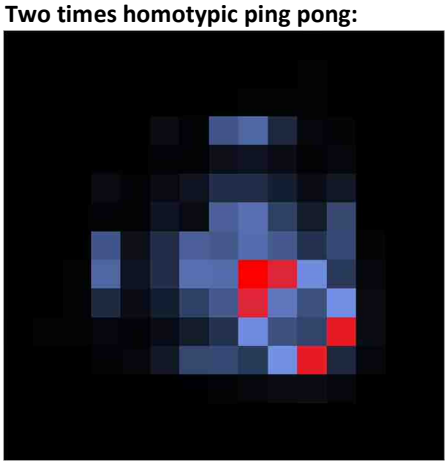
In this case, a strong ping-pong signal is produced by 24nt sequences and 29nt sequences (and vice versa). The different (*bona fide*) piRNA populations are presumably bound to different PIWI paralogs.

Homotypic ping-pong:



In this case, a ping-pong signature is produced by sequence pairs that are equal in size. Putative piRNA are presumably bound to the same PIWI paralog.

Two times homotypic ping pong:



In this case we can observe a strong 26nt-26nt ping-pong signature and a strong 28nt-29nt ping-pong signature. We can assume that, although two different populations of piRNAs are present, ping-pong processing bases on the interaction of identical PIWI paralogs.

4.3 unitas.ping-pong.fas

This file lists all sequences that were found to produce a 10 nt 5' overlap with any other mapped sequence.

5. Command line options

```
Allowed values for options: [s]=string, [i]=integer, [f]=floating point number

 -i OR -input [s]          Name of the input file. You can use multiple input
                           files at once, e.g.:
                           -i file1.fas -i file2.fq -i file3.sam
                           Alternatively you can provide the name of a
                           directory that contains all of your input files,
                           e.g:
                           -i path/to/your/input/files
 -s OR -species [s]        Species name. Use the binominal nomenclature with
                           genus and species seperated by an underline, e.g.:
                           -s mus_musculus
 -latest_ref               Force unitas to download the latest database
                           sequences. Otherwise unitas will use previously
                           downloaded sequences (if available) from the
                           UNITAS_refdump folder.
 -refdump                  Create a reference sequence dump folder, download
                           species-specific reference sequences and QUIT.
                           This folder and sequences herein can be used for
                           later offline unitas runs.
 -refseq [s]               Use an arbitrary number of additional files that
```

```
                        contain reference sequeces in FASTA format. FASTA
                        headers should have the following format:
                        >ncRNA_type|ncRNA_name
-riborase               Find all rRNA-like sequences in your input file
                        by using a complete NCBI rRNA nucleotide collection
                        in addition to the available ncRNA data for your
                        species.
-trim                   Use this option to search for and remove any 3'
                        adapter sequence in input files. There is no need
                        to provide a specific sequence.
-trim_minlength [i]     Minimum allowed read length [nt] after removal of
                        adapter sequence. Default=10.
-trim_maxlength [i]     Maximum allowed read length [nt] after removal of
                        adapter sequence. Default=50.
-trim_ignore_5p [i]     Ignore first n basepairs for 3’ adapter prediction.
                        Default=22.
-trim_skip_reads [i]    Skip the first n reads when searching for frequent
                        motifs. Default=10000
-trim_check_reads [i]   Check n reads for 3’ adapter prediction.
                        Deafult=500000
-trim_polyA             Allow polyA tails to be recognized as 3' adapter.
                        Useful when library was prepared with polyA-tailing
                        rather than 3' adapter ligation.
-trim_maxfinalbias [f]  Maximum allowed 3' nucleotide bias after trimming.
                        If this value is exceeded, unitas will trim further
                        3' positions. Default=0.8.
-quick                  Allow less mismatch. Equivalent to:
                        -tail 1 -intmod 0 -mismatch 0 -insdel 0
                        Faster, less memory usage, less sensitive.
-slow                   Allow more mismatch. Equivalent to:
                        -tail 3 -intmod 2 -mismatch 2 -insdel 1
                        Slower, more memory usage, more sensitive.
-tail [i]               Maximum number of allowed non-template 3'
                        nucleotides when mapping user sequences to reference
                        miRNA (miRNA-precursor) and piRNA cluster sequences.
                        Default=2. Summed values for -tail and -intmod must
                        not exceed 5.
-intmod [i]             Maximum number of allowed internal mismatches when
                        mapping user sequences to reference miRNA (miRNA-
                        precursor). Default=1. Summed values for -tail and
                        -intmod must not exceed 5.
-species_miR_only       Do not use miRNA (-precursor) sequences from other
                        species when searching for miRNAs in input dataset.
                        Default=off.
-mismatch [i]           Maximum number of allowed mismatches when mapping
                        user sequences to reference ncRNA sequences (does
                        not apply to miRNAs and putative piRNAs). Default=1.
-insdel [i]             Maximum number of insertions/deletions when mapping
                        user sequences to reference ncRNA sequences (does
                        not apply to miRNAs and putative piRNAs). Default=0.
-threads [i]            Maximum number of parallel threads. Applies only if
                        using more than one input file. Default=1.
-memory [i]             Limit memory usage [GB] for SeqMap mapping. Default
                        is no limit. When using a limit, unitas will quit
                        before the limit will be exceeded.
-less_memory            Will use modified parameters for SeqMap mapping to
                        reduce memory usage. Will increase running time.
-even_less_memory       Will use modified parameters for SeqMap mapping to
                        reduce memory usage even more. Will increase running
                        time.
-phasi [i]              Search for phased siRNAs (phasiRNAs or tasiRNAs)
                        with specified length = [integer]. When using this
                        option, the input file must be a map file in either
                        SAM or ELAND3 format. You can use this option
                        multiple times, e.g: -phasi 21 -phasi 24.
-na_for_phasi           Only sequences without annotation will be considered
                        for phasiRNA prediction.
-pp                     Analyze 5' overlaps of mapped reads to identify so-
                        called ping-pong signatures (preference for 10 bp
                        overlaps) which is a footprint of secondary piRNA
                        biogenesis. When using this option, the input file
                        must be a map file in SAM or ELAND3 format.
-keep_temp              Do not remove temporary files from data dump folder.
```

```
-skip_mapping             Skip mapping to reference sequences. Equivalent to:
                          -skip_miR -skip_cDNA -skip_ncRNA -skip_piR
                          Use this when you want to trim 3' adapters and/or
                          want to search for phasiRNAs only.
-skip_miR                 Skip search for miRNA reads, search for other ncRNAs
                          only.
-skip_cDNA                Skip search for reads that map to Ensembl cDNA
                          sequences.
-skip_ncRNA               Skip search for reads that map to ncRNA collection.
-skip_piR                 Skip search for reads that map to known piRNA
                          producing loci.
-skip_dust                Skip filtering low complexity sequences.
-dust_limit [f]           Threshold for fraction of a sequence that consists
                          of simple sequence repeats When filtering low
                          complexity reads from input files. Default=0.75.
-dust_max [i]             Maximum motif size for simple sequence repeats.
                          Default=5.
-dust_maxN [f]            Maximum fraction of Ns in a read. Default=0.5.
-noCCAcheck               Mature tRNAs carry a CCA-3' which is enzymatically
                          added to tRNA transcripts unless it is present
                          already. unitas will check 3' termini of tRNAs and
                          add the CCA-3' unless this option is turned on.
-CC_is_CCA                Treat 3' tRFs with CC-3' as CCA-3'. Makes sense
                          when protocols for library construction use poly-A
                          tailing with subsequent A(n)-3' trimming.
-no_html                  Do not output html files with visualization of
                          sncRNA annotation and analysis.
-diffexpr                 Search for differentially expressed sRNAs accross
                          different probes. Define 2 groups with a recommended
                          minimum of 2 probes per group using the options -g1
                          and -g2 as described below. When using this option
                          unitas will do the comparison and quit. Change the
                          name of the output file using the option
                          -diffexpr_out.
-diffexpr_p [f]           Critical p value for differential expression
                          analysis. Default=0.05.
-diffexpr_acorrection [i] Correction of alpha error (multiple testing).
                          Allowed values are 0 (=no correction), 1
                          (=Bonferroni-Holm, default) and 2 (=Bonferroni).
-diffexpr_estdev [f]      If one group comprises only one replicate, the
                          variance can be estimated by means of this factor
                          which is 0.05 by default. That means if sequence A
                          has 100 reads in the given replicate, unitas
                          virtually adds two replicates with read count values
                          105 and 95 (+- 5%).
-g1 [s]                   unitas output folder for probe assigned to group 1.
                          Use this option multiple times for every probe that
                          you want to assign to group 1. E.g:
                          -g1 UNITAS_04-01-2017_SRR123456.fasta_#1
-g2 [s]                   Probes assigned to group 2 (see above).
-diffexpr_out [s]         Specifies file names were the results of the
                          differentially expressed RNAs analysis will be
                          written to. Extensions .up1 and .up2 will be added
                          to the name. Default file names are:
                          'unitas.diff-expr.up1' and
                          'unitas.diff-expr.up2'.
                          (~.up1->higher expression in group 1, ~.up2->higher
                          expression in group 2).
-silent                   Less console (STDOUT) output during annotation.
-supp_spec                Prints the complete list of supported species and
                          quits.
-h OR -help               Shows usage information and quits.
-show_options             Shows available options and quits.
```

6. Troubleshooting

We really tried hard to design `unitas` in a way that ensures compatibility with most platforms. Although we have tested `unitas` thoroughly on several different systems, we cannot guarantee that it will run on every machine on this planet. Here are some solutions for problems that may occur or have been reported by some users:

```
Command perl is not available.
```

You try to run the `unitas` Perl script but Perl is not installed on your computer (most likely you are on a Windows machine). You should try one of the following:

1. Use the precompiled executable file for your system instead of running the Perl script.
2. Download and install either Strawberry Perl or ActivePerl, both are freely available.

```
Can't locate LWP/Simple.pm in @INC [...]
Can't locate Archive/Extract.pm in @INC [...]
Can't locate File/Copy.pm in @INC [...]
Can't locate File/Path.pm in @INC [...]
Can't locate Getopt/Long.pm in @INC [...]
```

You try to run the `unitas` Perl script but the module LWP::Simple (or one of the other modules) is missing on your computer for some reasons. You should try one of the following:

1. Since version 1.3.0 we provide unitas within a zip compressed folder that contains all the required modules. Try to run version 1.3.0 or later inside the supplied folder.
2. Use the precompiled executable file for your system instead of running the Perl script.
3. Upgrade your Perl distribution with the following two commands:
   ```
   cpan
   upgrade
   ```
4. Install the LWP::Simple module(or any other missing module) with the following two commands:
   ```
   cpan
   install LWP::Simple
   ```

```
Failed to load current species list from smallRNAgroup-SERVER. Use internal list.
```

This message will occur if you are not connected to the internet or the smallRNAgroup-SERVER is down. In the latter case, `unitas` will use an internal list of supported species which is maybe not up to date. If you are not connected to the internet `unitas` can only use previously downloaded sequence data and will quit in case that there is no reference dump folder for the species in question in the current working directory.

```
Unable to create temporary file with list of data dump folders.
Unable to create dump folder for reference sequences.
```

This is most likely because you are on a Mac or Linux machine and do not have permission to write into the current working directory. You can change permissions with the following command (may require password):

```
chmod -R 755 /path/to/folder
```

```
No success when building SeqMap from source. Will download pre-compiled SeqMap version from
smallRNAgroup-SERVER.
```

`unitas` tried to compile the downloaded SeqMap source code with g++ but failed. Maybe the g++ compiler is missing on your computer. unitas will download a precompiled SeqMap executable that should run on your system. However, it is likely that you have to change file permissions for that executable file manually (allow execution). Check subsequent `unitas` messages.

```
Permission denied. You have to change file permission for file 'seqmap.exe' manually.
```

unitas failed to compile the SeqMap source code and downloaded a precompiled version for your system. To execute this file, you have to change file permission manually (make this file executable).

Some people reported crabbed error messages when running one of the executable files that can look like this:

```
My-MacBook-Pro:mirna_unitas me$ ./unitas_1.2.0 -input unitas.fa -species hsa
Can't load '/var/folders/54/7cnshs5s2rq2scdmfk_y_m_r0000gn/T//par-506965727265/cache-
92735cedf1974b74e0b9c74cd3c7a5bca9e4c4b2/cfcc21dc.bundle' for module IO:
dlopen(/var/folders/54/7cnshs5s2rq2scdmfk_y_m_r0000gn/T//par-506965727265/cache-
92735cedf1974b74e0b9c74cd3c7a5bca9e4c4b2/cfcc21dc.bundle, 1): no suitable image found.  Did find:
        /var/folders/54/7cnshs5s2rq2scdmfk_y_m_r0000gn/T//par-506965727265/cache-
92735cedf1974b74e0b9c74cd3c7a5bca9e4c4b2/cfcc21dc.bundle: code signature invalid for
'/var/folders/54/7cnshs5s2rq2scdmfk_y_m_r0000gn/T//par-506965727265/cache-
92735cedf1974b74e0b9c74cd3c7a5bca9e4c4b2/cfcc21dc.bundle'

        /var/folders/54/7cnshs5s2rq2scdmfk_y_m_r0000gn/T//par-506965727265/cache-
92735cedf1974b74e0b9c74cd3c7a5bca9e4c4b2/cfcc21dc.bundle: code signature invalid for
'/var/folders/54/7cnshs5s2rq2scdmfk_y_m_r0000gn/T//par-506965727265/cache-
92735cedf1974b74e0b9c74cd3c7a5bca9e4c4b2/cfcc21dc.bundle'
 at /System/Library/Perl/Extras/5.18/PAR/Heavy.pm line 75.
```

```
Compilation failed in require at /System/Library/Perl/5.18/darwin-thread-multi-
2level/IO/Handle.pm line 269.
BEGIN failed--compilation aborted at /System/Library/Perl/5.18/darwin-thread-multi-
2level/IO/Handle.pm line 269.
Compilation failed in require at /System/Library/Perl/5.18/darwin-thread-multi-
2level/IO/Seekable.pm line 101.
BEGIN failed--compilation aborted at /System/Library/Perl/5.18/darwin-thread-multi-
2level/IO/Seekable.pm line 101.
Compilation failed in require at /System/Library/Perl/5.18/darwin-thread-multi-2level/IO/File.pm
line 133.
BEGIN failed--compilation aborted at /System/Library/Perl/5.18/darwin-thread-multi-
2level/IO/File.pm line 133.
Compilation failed in require at -e line 351
```

We believe that this comes from a conflict of modules packed into the executables and modules that are installed on the computer. In all reported cases, no error occurred when running the Perl script rather than the precompiled executable.

7. Citation policy

Sequence annotation with `unitas` is not possible without having reference sequences. `unitas` uses reference sequences from different public online databases. When you use `unitas` for sequence annotation, please cite the papers listed in 3.1.4. as well as the SeqMap paper from Jiang and Wong (see 3.1.5) in addition to `unitas`.

You can cite `unitas` as follows:
Gebert D, Hewel C, Rosenkranz D (2017). unitas: the universal tool for annotation of small RNAs. *BMC Genomics* 18(1):644.

8. Contact

If you have any questions or comments or found any bugs in the software, please do not hesitate to contact us:

David Rosenkranz
Institute of Organismic and Molecular Evolutionary Biology,
Anthropology, small RNA group
Johannes Gutenberg University Mainz, Germany
Email: rosenkranz@uni-mainz.de
Web: http://www.smallRNAgroup-mainz.de