

SINA: high throughput multiple sequence alignment

Elmar Pruesse

February 27, 2017

Version 1.3.0

Abstract

SINA is a tool for aligning sequences with an existing multiple sequence alignment (MSA) at high accuracy. It can also execute a homology search based on the computed alignment and generate a per sequence classifications from the search results.

This manual documents the command line usage of sina. Please see <http://www.arb-silva.de/aligner> for a reference to the scientific description of the employed algorithms.

Contents

1	Synopsis	2
2	Description	2
3	Options	3
4	Generated Meta Data Values	11
5	Examples	13
6	See Also	14
7	Version	14
8	License and Copyright	14

1 Synopsis

```
sina -i sequences.fasta|arb -o output.fasta|arb {--prealigned | --ptdb aligndb.arb}  
[--search --search-db searchdb.arb] [options]
```

2 Description

You can view SINA as a one-command pipeline composed of the following stages:

1. Read sequences from FASTA or ARB file.
2. Align sequences with reference MSA.
3. Search for most similar sequences in search MSA.
4. Classify sequences using search result.
5. Write sequences to FASTA or ARB file.

You can enable or disable the middle three stages as required. By default, only the alignment stage is enabled. Briefly, this is what those stages do (see section Options for details on the configuration options accepted by each of the stages).

Read: Reads sequences from a multi-FASTA file or an ARB database. If reading from ARB, additional meta-data can be read as key-value pairs. These key-value pairs are carried with each sequence throughout the pipeline and will be exported at the end.

Align: Sequences are aligned using the POA algorithm with a set of reference sequences drawn from the reference MSA. Reference sequence selection is based on a kmer search.

Search: Sequences are compared after alignment with the aligned sequences in the configured search database. Comparison is done either against all sequences or against the best matches from a kmer search. Identity is computed as the number of identical column/base pairs divided by the length of the query sequence.

Classify: Sequence classification uses least-common-ancestor (LCA) to derive a classification from the classifications of the sequences found during the search stage.

Write: Writes sequences and meta-data to a multi-FASTA file or an ARB database. See section Options for possible format options to export meta data when writing to multi-FASTA.

The default parameters are a pretty good starting point. They were optimized using a large SSU rRNA gene reference MSA. If you want to use SINA for other gene sequences, see section Examples on how to do some simple accuracy benchmarks on them. To improve the results, the parameters you will want to start with are **-fs-full-len** (set to the typical size of a full-length sequence) and **-fs-kmer-len** (setting this to 8 may help with more variable or shorter sequences).

3 Options

Options beginning with a single “-” must be separated from arguments with a space character. Options beginning with “--” can also be separated from arguments with an equal sign.

3.1 General Options

- h, --help** Print a summary of the available options and exit.
- version** Print the version information and exit.
- show-conf** Print a summary of all configuration settings before processing the input sequences.
- i *filename*, --in=*filename*** Specify the source file containing the sequences to be aligned. The special filename “.” can be used to access an open ARB database when starting SINA from a shell spawned from within ARB. The sequence data may already be aligned (and should be, if you supply *--prealigned*).
- intype {*fasta*|*arb*}** Specify the format of the source file. If the filename ends with “arb” or “fasta”, the type is automatically deduced.
- o *filename*, --out=*filename*** Specify the destination file for the aligned sequences. The special filename “.” can be used to access an open ARB database when starting SINA from a shell spawned from within ARB. If you want to discard the aligned sequences, you can set *filename* to */dev/null* and **--outtype** to *fasta*.
- outtype *fasta*|*arb*** Specify the format of the destination file. If the filename ends with “arb” or “fasta”, the type is automatically deduced.
- prealigned** If set, the alignment stage is disabled. Sequences are passed to search (if enabled) and output stage unmodified. Mandatory alignment parameters (**--ptdb**) are not required in this case. The input file should contain correctly aligned sequences.
- search** If set, the search stage is enabled.

3.2 Logging Options

- show-diff** This flag enables visualization of alignment differences. This feature allows you to quickly assess where your alignment differs from the one SINA computed. By also showing you the alignment of the reference sequences used for aligning the sequence, you can get an idea of why SINA came to its conclusions. Many cases of “suboptimal” alignment can be attributed to inconsistent alignment of the reference sequences. To fix such problems, you could either correct the alignment of the reference sequences or add your corrected sequence to the reference alignment.

Alignment difference visualization requires that the input sequences be already aligned in a way compatible with the used reference alignment. For positions at which the original alignment and the alignment computed by SINA differ, output as shown below will be printed to the log:

```

Dumping pos 1121 through 1141:
-{}--{}--{}--{}--  4 14 16-17 21 24
G-C-AGUC-  40 <{}--(%% ORIG %%)
GCA-{}-GUC-  41 <{}--(## NEW ##)
GCA-AGUC-  0-3 5-13 15 18-20 22-23 25-27 29-39
GCAA-GUC-  28

```

In this case, the bases 'C' and 'A' were placed in other columns than as per the original alignment. The original alignment is marked with <---(%% ORIG %%). The new alignment is marked with <---(## NEW ##). The numbers to the right of the alignment excerpt indicate the indices of the sequences in the alignment reference (field *align_family_slv*) which the respective row represents. All-gap columns are not shown. The first line indicates the range of alignment columns displayed.

--show-dist This flag enables computing the values *sps*, *error*, *matches*, *mismatches*, *bps*, *cpm*, *idty* and *achieved_idty*. See section “Generated Meta Data Values” for an explanation of the individual values. All values except *bps* are computed by comparing the newly computed alignment with the original alignment of the sequences. If a database is configured using **--orig-db**, the original alignment is obtained from that database. Otherwise, the alignment of the input sequences is used.

--orig-db *arb database* The database *arb database* is used to retrieve a aligned sequences to be used as a reference for comparison by **--show-diff** and **--show-dist**. Sequences are retrieved based on the contents of the ARB field *name*. If FASTA is used as input format, the first word of the FASTA header will be used for matching.

--colors Enable color in the output of **--show-diff**.

--log-file *filename* Redirect the log output to *filename*.

3.3 Reading from ARB

--select-file *filename* If using an ARB database as sequence input file, only sequences with a *name* matching a line contained within *filename* will be passed into alignment and search stages.

--select-step *n* If using an ARB database as sequence input file, only every *n*th sequence will be passed into alignment and search stages. This may be combined with **--select-file**. In combination with **--select-skip** this option can be used transparently distribute processing of a single ARB database to multiple instances of SINA.

--select-skip *n* If using an ARB database as sequence input file, the first *n* sequences will be skipped. Combination with **--select-file** is possible. In combination with **--select-step** this option can be used transparently distribute processing of a single ARB database to multiple instances of SINA.

--extra-fields *fieldnames* Passing a colon separated list of field names will load the meta data contained within these database fields from ARB. The contents will be passed as key-value pairs through the internal SINA pipeline. They will be treated like meta data generated by SINA itself. That is, they will be printed to the log file and written to the output file.

3.4 Writing to ARB

--prot-level *n* Set the protection level used to write to the ARB database to *n*. If a field was set to have a protection level above *n*, SINA will (silently) fail to write to these fields. If your sequences have a protection level of for example 4 and you set *n* to 0, your sequence data will not be modified. If you use the same ARB database for input and output, this may be used in combination with **--show-diff** to inspect the effect of varying the alignment parameters without modifying the alignment.

3.5 Writing to FASTA

--meta-fmt {*none*|*header*|*comment*|*csv*} This option configures the format in which meta data will be exported if the output format is FASTA. *none* will discard all meta data (it will still be written to the log, however). *header* will export meta data values as bracket enclosed key value pairs on the FASTA header line. *comment* will export meta data values as key value pairs on FASTA comment lines, that is lines beginning with a semi-colon between the header and the sequence data. *csv* will export meta data values to a separate file in RFC4180 compatible comma separated value format. The filename will be generated from the output filename by appending “.csv”.

--line-length *n* If *n* is different from 0, sequence data will be line wrapped after *n* characters.

3.6 Alignment Options

--ptdb *filename* Specifies the ARB database to be used as alignment reference. This is a mandatory parameter. The file must be in ARB format. See section Examples below for an explanation how to generate such a database from a FASTA file using only SINA. The name of this parameter is historical and refers to the fact that a ARB PT server will be started using the configured database to search for the sequences having the least kmer distance to the input sequences.

--ptport *socket* Configures the socket which will be used for communication with the ARB PT server. SINA will attempt to contact a running PT server via this port. If no PT server can be contacted, SINA will attempt to start one itself.

socket may either be of the format *hostname:port*, specifying a TCP socket, or of the format *:filename*, specifying a Unix socket. If no running PT server could be contacted and a Unix socket is specified or hostname is “localhost”, a PT server will be started locally. If *hostname* is “__SGE__”

SINA will start and contact a PT server on a cluster node using *qrsh*(1). Otherwise, *ssh*(1) will be used to start a PT server on the configured host.

The default is to use port “localhost:4040”.

CAUTION: If a PT server is already running on the configured socket, but its database does not match the database configured with **--ptdb** the results will be undefined. The search result retrieved from the PT server identifies sequences using the *name* field. For completely different databases, this will usually result in SINA being unable to find reference sequences. It may, however, also result in SINA retrieving the wrong sequences.

--turn {none|revcomp|all} Using this option, SINA can be configured to automatically reorient input sequences. If set to *none*, automatic reorientation is disabled. If set to *revcomp* only the reversed and complemented orientation of the input sequences is considered. If set to *all* all four combinations of reversing and complementing the sequence are considered. The default is *all*. Turning this feature off or reducing its scope will improve performance.

To determine which orientation is most likely, SINA uses the PT server to search for the sequence in the configured orientations. If an orientation different to the original yields a higher scoring best match, the sequence is modified accordingly.

--realign Configures SINA not to copy alignment information from identical reference sequences or reference sequence of which the input sequence is a substring.

Normally, SINA will compare the input sequence with all reference sequences found via the PT server search. If the input sequence is a substring of any of the reference sequences, the alignment of the reference sequence of which the input sequence is a substring will be directly transferred to the input sequence.

If the input sequence is found to be an exact match to a reference sequence, this will be noted in the field *align_log_slv* with the string “copied alignment from identical template sequence”. If the input sequence is found to be a substring of a reference sequence, this will be noted with the string “copied alignment from (longer) template sequence”. In both cases, the contents of the fields *acc* and *start* will also be logged to identify the reference sequence.

If suitable sequences for alignment copying are found, but **--realign** is set, the sequences will be removed from the alignment reference. This will be noted in the log with the message “sequences [acc list] containing exact candidate removed from family;”.

--overhang {attach|remove|edge} If the reference sequences used for alignment do not cover the input sequence completely, e.g. because it contains bases beyond the gene boundary, these bases cannot be aligned. This option configures how SINA handles such unaligned bases at the end of the input sequences. If set to *attach*, the bases will be placed in consecutive columns outwards from the last aligned base, i.e. they will be

“attached” to the outer most aligned base. If set to *remove*, these bases will be omitted from the output. If set to *edge*, these bases will be placed in consecutive columns inwards from the first and last alignment column, i.e. “moved to the edge of the alignment”. The default is *attach*.

--lowercase {*none*|*original*|*unaligned*} Use this option to configure which bases you wish to be in lower case in the output. The default setting is *none*, which will output all bases in upper case. If set to *original*, the original cases will not be modified. If set to *unaligned*, the case will be used to convey which bases of the input sequences remained unaligned by setting aligned bases to upper case and unaligned bases to lower case in the output. Unaligned bases are either overhang (see *--overhang* above) or result from insertions which could not be found in any of the reference sequences. If large insertions required shifting aligned bases (see *--insertion* below), the shifted bases will also be considered unaligned and shown in lower case.

--insertion {*shift*|*forbid*|*remove*} Since SINA aligns sequences to match a given fixed column reference alignment, insertions in the input sequences may have occurred that cannot be accommodated by the reference alignment. While the only correct way of dealing with this is certainly inserting further columns into the reference alignment to create sufficient room, this may not always be feasible.

The default setting is to *shift* the bases surrounding such a large insertion aside as required. This is done by iteratively choosing the nearest free column to the left or right until sufficient columns have been found. Each time bases are encountered between the insertion and the free column, these bases are added to the insertion. The main benefit of this naive approach is that the position and size of insertions that could not be accommodated are known. The message “**shifting bases to fit in N bases at pos X to Y**” will be logged each time an insertion of length *N* is attempted between positions *X* and *Y* with $Y - X < N$. The affected bases can be marked as unaligned by exporting them in lower case letters using the *--lowercase* option described above. A summary giving the total number of shifted bases and the longest insertion is also logged for each sequence.

The option *forbid* configures SINA to instead disallow insertions that will not fit the reference alignment during the dynamic programming stage of sequence alignment. While this option constitutes a loss of optimality of the alignment algorithm if the gap extension penalty (see *--pen-gapext* below) is different from the gap open penalty (see *--pen-gap* below) it results in slightly less damage to the alignment accuracy.

The option *remove* configures SINA to omit bases from insertions as necessary to fit these insertions into the alignment without moving surrounding aligned bases. This option should be handled with care as the original sequence is altered. If the alignment is subjected to column masking or column sampling (such as during tree reconstruction with bootstrapping), omitting bases is safe, as these methods interpret the resulting MSA from a column perspective.

Which option is the most suitable should be carefully considered for each use case. Whenever possible, circumventing the necessity to handle insertions that do not fit into the alignment by simply adding gap columns into the reference alignment is the preferred solution.

- filter *filtername*** Using this option it is possible to configure using statistical information on positional variability during the alignment. “Filter” is a colloquial term used for “sequence associated information” or SAIs as used by ARB. Filters/SAIs of the type “positional variability by parsimony” (PVP) are eligible for use via this parameter. Please consult the SINA publication and the ARB documentation for more information.
- auto-filter-field *fieldname*** This option allows automatically selecting a PVP filter based on strings contained in an ARB database field. If the configured field contains a shared prefix over all selected reference sequences, the ARB database configured with **--ptdb** is searched for a matching filter. A filter is considered matching if the part of its name following the first colon is itself a prefix of the shared prefix described above. As an example, if *tax_slv* is chosen and all reference sequences share the prefix “Bacteria;Proteobacteria;” then the filter “silva.108:Bacteria” will match. If **--filter** is also provided, the *filtername* must match the part of the filter name before the first colon.
- auto-filter-threshold *value*** The term “shared prefix of all reference sequence” can be relaxed to “longest prefix shared by *value* of the reference sequence” using this parameter. (Default: 0.8)
- fs-min *value*** The minimum number of reference sequences that should be used. If less matches are returned by the kmer search, less sequences will be used. (Default: 40)
- fs-max *value*** The maximum number of reference sequences that should be used. (Default: 40)
- fs-msc *value*** The minimal kmer score reference sequences should have with the input sequence. At least as many sequences as configured by **--fs-min** will be used. Up to **--fs-max** sequences will be used **if** they have a kmer score higher than configured by **--fs-msc**. (Default: 0.7)
- fs-msc-max *value*** Limits sequence selection to sequences having kmer score no higher than *value*. (Default: 2, that is, disabled)
- fs-leave-query-out** Setting this option will remove the query sequence from the reference sequences based on its *name*. This is sensible for evaluation in comparison to other tools where leave-query-out style evaluation can only be done by excluding the exact query sequence from the reference. If the alignment must not be directly derived from any reference sequence, even if the reference dataset contains redundant data, **--realign** should be used.
- fs-req *value*** The minimum number of reference sequences that must be used. If less matches are returned by the kmer search, alignment is refused. The sequence will not be contained in the output. (Default: 1)

- fs-req-full *value*** The minimum number of full length sequences that should be included in the reference. The matches from the kmer search are parsed until, beyond the limits given by *--fs-max* and *--fs-msc*, at least ***value*** such sequences have been found and added to the reference sequences.
- fs-full-len *value*** The minimum number of bases constituting a full-length sequence.
- fs-kmer-no-fast** Disable the PT server fast search. The fast kmer search considers only kmers beginning with 'A'.
- fs-kmer-len *k*** Configures the length *k* of the kmers used for the kmer similarity search.
- fs-kmer-mm *value*** Configures the number of mismatching bases a kmer may have to be considered matching.
- fs-kmer-norel** Computes the kmer score using the length of the query sequence only. If not set, the kmer score is computed as the number of shared kmers between query and match candidate divided by the length of the shorter.
- fs-min-len *value*** Minimal length sequences found via the kmer search must have to be considered for inclusion into the reference sequences.
- fs-weight *value*** Factor with which the frequency at which a base occurs within the reference sequences will be used to weight match and mismatch scores between the base and bases from the input sequence.
- gene-start *value*** Position within the alignment corresponding to the first base of the aligned gene.
- gene-stop *value*** Position within the alignment corresponding to the last base of the aligned gene.
- fs-cover-gene *value*** Minimum number of times the gene-start and gene-stop positions are at least touched by one of the reference sequences. If the above rules did not result in sufficient such sequences, further sequences covering the respective position are added until the condition is met.
- match-score *value*** The match score used during the dynamic programming stage of partial order alignment (POA).
- mismatch-score *value*** The mismatch score used during the dynamic programming stage of partial order alignment (POA).
- pen-gap *value*** The gap open penalty used during the dynamic programming stage of partial order alignment (POA).
- pen-gapext *value*** The gap extension used during the dynamic programming stage of partial order alignment (POA).

--debug-graph Enables dumping of graph data in graphviz format suitable for processing with e.g. *dot(1)*. For each aligned sequence, the DAG used as alignment template is dumped. Subsections of the dynamic programming graph/mesh, each covering the same fractions as shown with *--show-diff*, are also dumped. Please be aware that the output will be huge.

--use-subst-matrix Experimental. Do not use!

3.7 Search and Classification Options

--search-db *filename* Configures the name of the ARB database which should be used for sequence search. Unless *--search-all* is also set, a PT server will be started for this database. The same rules as for *--ptdb* apply. It is permissible to use the same file as in *--ptdb*. In this case, the database will be loaded only once.

--search-port *socket* Configures the port on which SINA should communicate with the PT server used for kmer searching. The same rules as for *--ptport* apply. If *--search-all* is set, no PT server will be used and this setting will be ignored.

--search-all Configures SINA to compare the aligned input sequence with **all** sequences contained in the database given by *--search-db*. No PT server will be used.

--search-no-fast Disable the PT server fast search. The fast kmer search considers only kmers beginning with 'A'.

--search-kmer-candidates *n* Configures the number of best matching results from the kmer search that should be compared with the input sequences based on the alignment.

--search-kmer-len *arg* Configures the length *k* of the kmers used for the kmer similarity search.

--search-kmer-mm *arg* Configures the number of mismatching bases a kmer may have to be considered matching.

--search-kmer-norel Computes the kmer score using the length of the query sequence only. If not set, the kmer score is computed as the number of shared kmers between query and match candidate divided by the length of the shorter.

--search-min-sim *value* Minimal identity a sequence must have with the input sequence to be included in the search result.

--search-ignore-super Exclude sequences of which the input sequence is a substring from the search result.

--search-max-result *value* Limit the maximum number of search results per input sequence.

- search-copy-fields *fieldnames*** Configures a colon separated list of ARB fields which will be copied into the input sequence. The field be prepended with “copy_<accession>_” in the output to indicate from which search result the data came.
- lca-fields *fieldnames*** Derives a LCA classification of the input sequence from the classifications of the sequences found in the search. This feature requires the reference database to contain a field specifying the sequence classifications in materialized path format (i.e. “Bacteria;Proteobacteria;...”). The “least common ancestor” is the shared prefix of these strings. Prefixes must always end with a semicolon. Depending on the desired rank up to which the sequences should be classified, appropriate sequence similarity cutoffs should be configured with *--search-min-sim*. It is possible to specify multiple source taxonomies as **fieldnames** by passing colon separated list. Derived LCA classification will be stored in fields named “lca_<fieldname>”.
- lca-quorum *value*** Relaxes LCA classification from “shared by **all** search results” to a fraction **value** of the search results.

4 Generated Meta Data Values

align_bp_score_slv This is a score calculated from the aligned sequence and the HELIX SAI. If the reference database contains no HELIX SAI the score will be NaN. Otherwise, the score is computed as follows. For each pair of columns covered by the aligned sequence a score of 1 is awarded if the pair is AU, GU or GC; a score of 0 is awarded if the pair is AG or GG; a score of -1 is awarded if the pair is AA, AC, CC, CU or UU or if one of the columns contains a gap character; the sum of these scores is divided by the number of considered columns. The value is scaled to match the range between 0 and 100.

This value is likely to change or disappear in future versions.

align_cutoff_head_slv This is the number of bases at the beginning of the sequences that remained unaligned.

align_cutoff_tail_slv This is the number of bases at the end of the sequences that remained unaligned.

align_family_slv This is a list of the sequences that were used to build to align the input sequence.

align_filter_slv If a PVP filter was applied, the name of that filter will be stored in this field.

align_log_slv Messages generated during the alignment process will be logged here.

align_startpos_slv This is the alignment position (column number) of the first aligned base.

align_stoppoos_slv This is the alignment position (column number) of the last aligned base.

aligned_slv This is the current date.

full_name If FASTA is chosen as input format, this field will contain the part of the FASTA header lines after the first space character.

nearest_slv This field contains a space separated list of the results from the homology search stage. Each search result is given in the following form: “<accession>.<version>:<start>:<stop> <identity>”

nuc The number of nucleotides in the input sequence.

nuc_gene_slv The number of nucleotides in the sequence aligned to be within the gene borders.

turn_slv Documents actions taken by the automatic reorientation of sequences. Possible values are “disabled”, “none”, “reversed”, “complemented” and “reversed and complemented”.

sps This field contains the fractional identity of the aligned input sequence with the input sequence in its original alignment. The number of identical base/column pairs is divided by the number of nucleotides.

error The number of differing base/column pairs divided by the number of nucleotides. The sum of *error* and *sps* may be larger than 1 because of gap characters. If in the new alignment, a base ends up in what should be a gap position and a gap is placed where the base was in the original alignment, two misaligned positions are found.

matches Number of identical base/column pairs in SINA aligned sequence and input alignment.

mismatches Number of differing base/column pairs in SINA aligned sequence and input alignment.

bps The same as *slv_bp_score* but unscaled and not rounded to integer.

cpm Correctly placed mutations, or rather, an attempt at calculating such a measure intended to be used as a measure of alignment accuracy independent of the identity an input sequence as with its closest reference sequences. The value is the number of base/column pairs the aligned sequence shares with its original alignment **more** than the sequence in its original alignment shares with the closest found reference sequence divided by the number of base/column pairs in original alignment that are not matched by the closest reference.

This value is likely disappear or change in future versions.

idty The highest fractional identity of the input sequence with any of the selected reference sequences calculated as the number of matches (see above) divided by the length of the input sequence.

achieved_idty Identical to *idty* but using the SINA alignment rather than the original alignment.

*lca_** These fields contain the classifications derived via LCA.

*copy_** These fields contain the data copied from the search results.

5 Examples

Aligning some sequences To align sequences, you need to get a suitable reference alignment in ARB format. If you have LSU or SSU sequences to align, the Ref or RefNR datasets from www.arb-silva.de work well. Otherwise, check below for an example on how to convert your own multi-fasta reference alignment to ARB format.

```
./sina -i mysequences.fasta -o alignedsequences.fasta \
      --ptdb reference.arb
```

The first time you run this, a PT server will be started and will begin building its index. The index is stored in **reference.arb.pt** and will only be computed again if **reference.arb** changes (the decision is made based on file timestamps only). The PT server will also continue to run once it has been started. Subsequent **sina** runs will be much faster therefore. Nonetheless, start-up time may be long if **reference.arb** is large.

Classifying some sequences If you are using a reference database that has a field containing classifications, you can use **SINA** to classify your sequences. The SILVA Ref database contain several taxonomies in fields beginning with “**tax_**”. To classify sequences based on the SILVA taxonomy, you can use this command line:

```
./sina -i mysequences.fasta -o alignedsequences.fasta \
      --meta-fmt csv \
      --ptdb reference.arb \
      --search --search-db reference.arb --lca-fields tax_slv
```

The classifications will be (among the other values) written to **alignedsequences.fasta.csv** to the column labeled “**lca_tax_slv**”.

Converting FASTA to ARB By disabling all stages, **SINA** can be used to convert between ARB and FASTA format (in a limited fashion, use ARB if you want to do more fancy stuff):

```
./sina -i mysequences.fasta -o mysequences.arb --prealigned
```

This will generate an ARB file from your aligned sequences suitable for use as a reference MSA. The first word of each FASTA header will be written to the ARB field “**name**”. Make sure they are unique for each sequence. ARB uses this field to identify sequences, duplicates will overwrite the previous sequence with the same name. The remainder of the fasta header will be written to the field “**full_name**”.

Running a leave-query-out accuracy benchmark You can run a quick check on the accuracy achieved by SINA with your reference MSA by having it align each of those sequences (ignoring the same sequence in the process) and log the accuracy with which it could reproduce the original alignment.

```
./sina -i myreference.arb --ptdb myreference.arb \  
-o /dev/null --outtype fasta \  
--fs-leave-query-out --show-dist
```

The average accuracy will be printed at the end of the SINA run.

Converting FASTA output from RNA to DNA SINA writes IUPAC encoded RNA as its output. If you require DNA, you can simply convert the file using this command:

```
sed '/^[^>]/ y/uU/tT/' rna.fasta > dna.fasta
```

6 See Also

ARB, <http://www.arb-home.de>
SILVA, <http://www.arb-silva.de>

7 Version

Version: 1.3.0 of February 27, 2017.

8 License and Copyright

Copyright © 2006-2011 Elmar Priesse (epruesse@mpi-bremen.de)

License This copy of SINA is licensed under the SINA PUEL (see below).

The author of SINA reserves all copyrights and other intellectual property rights. All further rights are at Ribocon GmbH (the "Owner") in legal agreement with the author of SINA and all third parties involved.

If you are interested in commercial use of the SILVA stand-alone software contact sina@ribocon.com.

Personal Use and Evaluation License (PUEL) for SINA Stand-Alone Software

This license applies if you download the SINA Stand-Alone Software Package (the "Product") from www.arb-silva.de. In summary, the license allows you to use the Product free of charge for academic Personal Use or, alternatively, for non-academic, time-limited Evaluation.

Overview: Personal Use (academic) is when you install the Product yourself and you make use of it. You can use the Product within an academic study to process as much data as you like and publish the processed data as long as you follow the terms below. If you deploy the Product to a single or multiple computers for colleagues within your institution, e.g. in

the capacity as a system administrator, this would no longer qualify as Personal Use.

Personal Use does NOT include (1) any redistribution of the Product, (2) any kind of Product-based data analysis service for third parties, or (3) integration of the Product into another software.

License Agreement: You should have received a copy of the license agreement with this software in the file LICENSE.txt. If you did not, please visit <http://www.arb-silva.de/aligner/sina>.