SINA Documentation

Release 1.7.2

Elmar Pruesse

Dec 13, 2020

Contents

SINA allows incorporating additional sequences into an existing multiple sequence alignment (MSA) without modifying the original alignment. While adding sequences to an MSA with SINA is usually faster than re-computing the entire MSA from an augmented set of unaligned sequences, the primary benefit lies in protecting investments made into the original MSA such as manual curation of the alignment, compute intensive phylogenetic tree reconstruction and taxonomic annotation of the resulting phylogeny.

Additionally, SINA includes a homology search which uses the previously computed alignment to determine the most similar sequences. Based on the search results, a LCA based classification of the query sequence can be computed using taxonomic classifications assigned to the sequences comprising the reference MSA.

SINA is used to compute the small and large subunit ribosomal RNA alignments provided by the SILVA project and is able to use the ARB format reference databases released by the project here.

An online version of SINA is provided by the SILVA project.

Publication

If you use SINA in your work, please cite:

Pruesse E, Peplies J, Glöckner FO. SINA: accurate high-throughput multiple sequence alignment of ribosomal RNA genes. *Bioinformatics*. 2012;28(14):1823-9. doi:10.1093/bioinformatics/bts252

CHAPTER 1

Installing SINA

You can install SINA

1. *using Bioconda* (recommended)
2. from *pre-compiled tarballs* (alternate)
3. or build SINA *from source* (for developers)

1.1 Install using Bioconda

SINA is available as a Conda package in the Bioconda channel. Check the package info page for more information.

To install, follow these steps:

1. Install Miniconda (skip if you've got conda already)

   Download the Miniconda installer (links for MacOS and Linux), execute it and follow the instructions it shows in the shell:

   ```
   # if you are on MacOS
   wget https://repo.continuum.io/miniconda/Miniconda3-latest-MacOSX-x86_64.sh
   # if you are on Linux
   wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh

   # then
   sh Miniconda3-lastest-*-x86_64.sh
   ```

2. Add the Conda-Forge and Bioconda channels:

   ```
   conda config --add channels defaults
   conda config --add channels bioconda
   conda config --add channels conda-forge
   ```

3. Install SINA into a conda environment:

```
conda create -n sina sina
```

4. Activate and use environment:

```
conda activate sina
sina --help
```

SINA should also work fine if installed with conda into the `base` environment. If you encounter problems, try the separate environment.

1.2 Install from pre-compiled tarballs

Tar archives containing pre-compiled binaries and requisite libraries are available on the SINA releases page at Github. Head on over there and download the Linux or Macos one. Inside the folder created by unpacking the archive, you should find a *sina* executable:

```
tar xf ~/Downloads/sina-1.7.2-linux.tar.gz
~/Downloads/sina-1.7.2-linux/sina --help
```

To install SINA system wide, place the contents of the archive in */opt* and create symlinks into */usr/local/bin*:

```
wget https://github.com/epruesse/SINA/releases/download/v1.7.2/sina-1.5.0-linux.tar.gz
sudo tar xf sina-1.7.2-linux.tar.gz -C /opt
rm sina-1.7.2-linux.tar.gz
sudo ln -s /opt/sina-1.7.2-linux /opt/sina
sudo ln -s /opt/sina/bin/sina /usr/local/bin/sina
```

1.3 Build from source code

Building SINA from source can be challenging because SINA depends on the ARB development libraries. Pre-compiled versions of these are currently only available from Bioconda, so you may have to start by building ARB from source.

1.3.1 Prerequisites

When building from a source tar ball from the SINA releases page, you will need to install:

- ARB >= 6.0.0
- Boost >= 1.62 - Boost Thread - Boost Program Options - Boost IO Streams - Boost Filesystem - Boost Serialization (SINA < 1.5) - Boost System - Boost Unit Test Framework (when building / running tests)
- TBB >= 2017
- zlib

When building from the raw git source code directly, you will additionally need:

- Autoconf
- Automake
- Libtool
- pkg-config

To build the documentation, you need:

- Sphinx >= 1.8

The easiest way to get all requirements is using Conda:

```
if test $(uname) == Linux; then
  dist_extra="gcc patchelf coreutils"
else
  dist_extra="llvm"
fi
conda create -n sina_build automake autoconf libtool pkg-config boost arb-bio-devel \
                git tbb tbb-devel glib libiconv bc sed sphinx $dist_extra
```

1.3.2 Building

1. If you are building from git, start by checking out the source code and generating the *configure* script:

   ```
   git clone https://github.com/epruesse/SINA.git sina
   cd sina
   autoreconf --force --install
   ```

2. Run the configure script, pointing it at all required libraries as necessary and choosing features and build types:

   ```
   ./configure --prefix=install_location \
               --with-arbhome=path_to_arbhome \
               --with-boost=path_to_boost_install \
               --with-boost-libdir=path_to_boost_libs
   ```

   If you used conda to install your dependencies, this line should work:

   ```
   conda activate sina_build
   mkdir build
   cd build
   ../configure --prefix `pwd`/install \
                --disable-docs \
                --with-tbb=$CONDA_PREFIX \
                --with-boost=$CONDA_PREFIX \
                --with-boost-libdir=$CONDA_PREFIX/lib \
                --with-arbhome=$CONDA_PREFIX/lib/arb \
                LDFLAGS="$LDFLAGS -Wl,-rpath,$CONDA_PREFIX/lib"
   ```

   Essential options to **configure**:

   --prefix=PATH (/usr/local)
   : Set the folder under which `./bin/sina`, `./lib/libsina.so` (or `.dylib`), etc. will be installed.

   --with-tbb=PATH
   : Set the location of libraries and headers for the Intel Threading Building Blocks library.

   --with-boost=PATH
   : Set the location of the boost header files (without the `include/` part).

   --with-boost-libdir=PATH
   : Set the location of the boost lib folder. Often, this is the value you used for *--with-boost* with `/lib` appended.

   --with-arbhome=PATH
   : Set the location of the ARB build directory. Not needed if you have `$ARBHOME` set to point to the place

you built ARB. When using the Bioconda package *arb-bio-devel*, use *$CONDA_PREFOX/lib/arb* where `$CONDA_PREFIX` is the root of the environment you installed ARB into.

`--with-buildinfo``=TEXT`
Set an additional string to be added to the version to identify your build.

`--enable-code-coverage`
Add compiler flags to collect code coverage statistics.

`--enable-debug`
Enable debug options (sets `-DDEBUG -O0 -ggdb3` instead of `-DNDEBUG -O2 -g`).

`--enable-asan`
Enable address sanitizer (sets *-fsanitize=address*).

`--enable-fat-tar`
Alters the build so that `make bindist-gzip` constructs a fully contained tar archive of the build.

`--enable-profiling`
Add compiler flags collecting profiling statistics (`-pg`).

`--disable-docs`
Do not build the documentation.

If you installed the dependencies in system wide, standard FHS locations, the **configure** script should detect the locations correctly. Otherwise you may have to use the `--with-something` options to point it at the right places. If things go wrong, the full error messages will be in `config.log`.

3. Build SINA (replace `<number of cpus` with however many cores you've got):

```
make -j<number of cpus>
make install
```

To build binary archives (see also *`--enable-fat-tar`*), use:

```
make bindist-gzip2
```

To run unit tests, call:

```
make check
```

To run only part of the tests, call:

```
make check-filtered P=pattern
```

where `pattern` matches the name(s) of the test you wish to (re)run.

To run unit tests collecting code coverage, call:

```
make check-code-coverage
```

To see the full command line of compiler and linker instead of the abbreviated display, append `V=1` to the `make` commandline.

CHAPTER 2

Commandline Reference

SINA adds sequences to an existing multiple sequence alignment (MSA). It can also execute a homology search based on the computed alignment and generate a per sequence classifications from the search results.

2.1 Synopsis

sina [*options*] `-i` <*unaligned*> `-r` <*reference*> `-o` <*aligned*>

sina [`-h` | `--help` | `--help-all` | `--version` | `--has-cli-vers`]

2.2 Description

`sina` aligns the sequences in the file `<unaligned>` to match the alignment in `<reference>` and places the aligned sequences in the file `<aligned>`.

Please refer to the publication for details on the algorithm(s).

2.3 General Options

`-h, --help`
Displays brief command line description.

`-H, --help-all`
Displays full command line description. If in doubt, refer to this as it will always be in sync with your installation of SINA.

`-V, --version`
Shows the version of SINA.

-i filename, **--in**=filename (-)
Specifies the file containing the input sequences. Allowable file formats are ARB (see *ARB Options*) and FASTA (see *FASTA Options, optionally gzipped). The format will be selected based on the file name unless overridden with :option:*–intype‘.

Special file names: “–” (dash) will read sequences from standard input. “**:**” (colon) will connect to a running ARB database (SINA must be a child process of ARB).

-o filename [filename [...]], **--out**=filename (-)
Specifies the output file(s) to which the aligned sequences and/or meta data will be written. Allowable file formats are ARB (see *ARB Options*) and FASTA (see *FASTA Options*, optionally gzipped) and CSV (see *CSV Options*). The format will be selected based on the file name unless overridden with *--outtype*. Specifying multiple names or specifying the option multiple times will output all data to each file.

See *-i* for special filenames – and **:**. Use -o /dev/null to disable output.

Not specifying this option at all will write sequences to stdout if the input is FASTA format. When reading from an ARB database, output is written back to the source database (specified with *-i*).

-r filename, **--db**=filename
Specifies the file containing the reference alignment. This file must be in ARB format.

To convert a reference alignment from FASTA to ARB format, run:

```
sina -i reference.fasta --prealigned -o reference.arb
```

-t [all], **--turn** [=all]
Enables turn check stage. Sequences not oriented in accordance with the reference database will be reverse complemented as needed.

If all is specified, sequences will also be tested for only reversal or only complemented (this should only be necessary if your data was mishandled).

-S, --search
Enables the search stage. See *Search & Classify Options* below for more information.

-P, --prealigned
Disables the alignment stage. This is useful if you have already aligned sequences you wish to pass directly into the search stage, or if you want to use SINA to convert between any of its supported file formats.

-v, --verbose
Increase logging verbosity. Can be specified multiple times.

-q, --quiet
Decrease logging verbosity. Can be specified multiple times.

--log-file=filename
Specify log file. The output written to the log file will always be verbose and is not affected by using *-v* or *-q*.

--meta-fmt=[none|header|comment|csv]
Configures how meta data (such as alignment score or sequence classification results) are to be exported.

none No output other than in the log is generated.

header Appends meta data as [key=value] pairs to the FASTA header line

comment Appends meta data as ; key: value lines between the FASTA header and the sequence data.

csv Writes meta data into a CSV side car file.

Deprecated since version 1.7.0: Use -o output.csv instead.

-f fields, **--fields**=fields
Configures the set of fields written to the output file. See *Field Reference* for a description of the available fields.

-p, --threads (automatic)
Override automatic detection of the number of threads used by SINA. This is usually only necessary if you need to constrain SINA to a lower number of threads. According to the Intel engineers whose *Threaded Building Blocks* library does the thread number detection for SINA, the only reason to use this parameter should be scalability testing.

--num-pts (1)
Set the maximum number of ARB PT server instances used by SINA. See also *--fs-engine* below. If you are using the **pt-server** engine, this setting will be the limiting factor in your throughput. Be aware, however, that each PT server will occupy additional system memory. Choosing a too high value may cause SINA to fail with out-of-memory errors.

--add-relatives=n (0)
Add up to *n* reference sequences for each query sequence to the output file. If *Search & Classify* is enabled via *--search*, the reference sequences are selected from the search result. Otherwise, they are selected from the query's alignment reference set.

If the source set is smaller than *n*, no further sequences are added to the output. Sequences already included are skipped, but count towards the *n* of the query sequence.

2.4 Reference Selection Options

These options configure how the set of reference sequences used during alignment is selected from the configured reference database.

--fs-engine=[internal|pt-server]
Selects the search engine used to find closely related reference sequences for the alignment stage.

pt-server Uses the ARB PT server to execute the k-mer search. The ARB PT server is a truncated suffix trie implementation implemented as part of the ARB package.

internal Uses an internal k-mer search implementation.

--fs-kmer-len=k (10)
Set the size of *k* for the reference search. For SSU rRNA sequences, the default of 10 is a good value. For different sequence types, different values may perform better. For 5S, for example, 6 has shown to be more effective.

--fs-min=n (15)
Set the minimum number of reference sequences used for each query.

--fs-max=n (40)
Set the maximum number of reference sequences used for each query.

--fs-msc=n (0.7)
Set the minimum similarity reference sequences are required to have with the query sequence. This affects the range between *--fs-min* and *--fs-max*.

--fs-req=n (1)
Set the minimum number of reference sequences that must be found in order to attempt alignment. If fewer sequences than indicated here are found, the respective query sequence will be discarded.

--fs-req-full=n (1)
Set the minimum number of *full length* (see *--fs-full-len*) reference sequences that must be included in the selected reference set. The search will proceed regardless of other settings until this setting has been satisfied. If it cannot be satisfied by any sequence in the reference database, the query sequence will be discarded.

This setting exists to ensure that the entire length of the query sequence will be covered in the presence of partial sequences contained within your reference database.

Note: If you are working with sequences other than 16S, you need to adjust this value or the value of `--fs-full-len` accordingly. In particular when working with short reference sequences, this setting may prevent any acceptable reference sequences from being found, leading to no sequences being aligned.

--fs-full-len=n (1400)
Set the minimum length a sequence is required to have to be considered *full length*.

--fs-req-gaps=n (10)
Set the minimum number of gaps a reference sequence is required to contain to be considered. This setting ensures that unaligned sequences contained within the reference database are not used as reference (this may happen when SINA is used from within ARB).

--fs-min-len=n (150)
Set the minimum length reference sequences are required to have. Sequences shorter than this will not be included in the selection.

Note: If you are working with particularly short reference sequences, you will need to lower this settings to allow any reference sequences to be found.

2.5 Search & Classify Options

When enabled via `--search`, SINA will execute a homology search. Unlike most homology search tools, SINA uses the inferred multiple sequence alignment to determine the similarity of each query with the reference sequences, rather than computing pairwise optimal alignments. **The similarity values will therefore be generally lower than the results of a pairwise alignment based homology search**.

Based on the search results, SINA can be instructed to compute a lowest common ancestor (LCA) based classification of the input sequences. For this, your reference database must include a field containing taxonomic classifications for each reference sequence. The field contents must be in the format t *Domain;Phylum;...*. SINA will compute query classifications as the deepest classification shared by at least the fraction `--lca-quorum` of the search result.

--search-db=filename (=db)
Specify an alternate reference database to use for search and classify. This can be useful if you have a specially curated alignment reference, but wish to search a larger set of sequences for classification purposes.

--search-engine=[internal|pt-server]
Override the value of `--fs-engine` for use within the search module.

--search-min-sim=id (0.7)
The minimum fractional identity each result sequence must have with the query.

--search-max-result=n (10)
The maximum number of search results to return for each query sequence.

--lca-fields=names
Enables the classification stage. The parameter *name* must be a colon or comma separated list of field names in the search database containing the classification reference data. Use `--arb-list-fields` to show a list of the fields available in a given ARB database. When using a SILVA ARB database as reference, the fields *tax_slv*, *tax_embl* (renamed to *tax_embl_ebi_ena* in newer releases) and *tax_ltp* contain the reference classifications according to the SILVA, EMBL-EBI/ENA and LTP taxonomies, respectively. When using a SILVA SSU ARB database, the fields *tax_gg* (only older databases) and *tax_rdp* are available additionally, containing the reference classifications according to RDP II and Greengenes, respectively. Newer SILVA databases also contain the field *tax_gtdb* containing classifications from the Genome Taxonomy Database.

--lca-quorum=fraction (0.7)
Sets the fraction of the search result that must share the same classification. Using the default parameters `--search-max-result`=10 and `--lca-quorum`=0.7, this means that the deepest classification shared by 7 out of the top 10 search results is chosen for the query sequence.

2.6 Advanced Options

--show-conf
Print the values of all configuration options (including defaults) at startup.

--intype=[auto|arb|fasta] (auto)
Set the file format for *--in*. If set to *auto* (default), the type is selected based on the file extension.

--outtype=[auto|arb|fasta|csv|none] (auto)
Set the file format for *--out*. If set to **auto** (default), the type is selected based on the file extension. The option can be specified multiple times. It applies to all files listed in the next *--out* option. If no output files are specified and this option is set to **none**, no output is produced at all.

--preserve-order
Preserve the order of the input sequences in the output.

--max-in-flight=n (2 * number of CPU cores)
Set the maximum number of sequences "in flight", i.e. processed in parallel.

--has-cli-vers=cliversion
Verify that this version of SINA supports the CLI version **cliversion**. Exits immediately with exit code 0 if true and 1 if false.

--no-align
Backwards compatibility alias for *--prealigned*.

2.7 Logging Options

--show-diff
Show differences between the inferred alignment and the original alignment. Requires either aligned sequences to be passed into sina via *--in* or that a database with matching names is specified using *--orig-db*.

Todo: Fix text below

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 "sub-optimal" 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 'texttt{C}' and 'texttt{A}' where placed in other columns than as per the original alignment. The original alignment is marked with texttt{<-{}–(%% ORIG %%)}. The new alignment is marked

with texttt{<-{}–(## NEW ##)}. The numbers to the right of the alignment excerpt indicate the indices of the sequences in the alignment reference (field

--show-dist
Show distance to original alignment

Todo: describe values generated

--orig-db=filename
Specify a database containing the original alignments for use with *--show-dist* and *--show-diff*. The sequence names in the input file and in the reference database must match exactly.

--colors
Use ANSI codes to show alignments dumped by *--show-diff* in color.

2.8 ARB Options

These options configure behavior supported only by the ARB backend for input and output sequences.

--arb-list-fields=FILE
Show the per-sequence meta-data fields available in the specified ARB database and exit.

--markcopied
Set *Mark* on sequences copied from the reference.

--markaligned
Set *Mark* on sequences updated or added by alignment stage.

--prot-level=n (4)
Set the *protection level* to use when writing sequences to the output database.

--select-file=filename
Instead of iterating over the entire input database, process only the sequences listed in *filename*. The names must match the ARB *name* field and be separated by newlines. Use "**-**" to read from standard input.

--select-step=n (1)
Process only every *n*th sequence. Can be combined with *--select-file* and *--select-skip*.

--select-skip=n (0)
Do not process the first *n* sequences. Can be combined with *--select-file* and *--select-step*.

2.9 FASTA Options

These options configure behavior supported only by the FASTA backend for input and output sequences.

--line-length=n (0)
Output sequences using at most *n* characters per line. Set to 0 to place the entire output sequence on one line.

--min-idty=id
Exclude sequences sharing less than *id* fractional identity with any of the alignment reference sequences from the output. Implies *--calc-idty*.

--fasta-write-dna
Write output sequences as DNA, rather than the default RNA. (I.e. use T and t rather than U and U).

--fasta-write-dots
Use dots (".") rather than dashes ("-") for gaps that indicate missing data rather than an actual insertion/deletion. Most often, those are only the terminal gaps at the ends of the alignment.

Todo: Check whether internal dots are handled correctly.

--fasta-idx=n
Only process sequences *starting* withing the *n*th block of bytes within the input FASTA file. The first block has index 0.

--fasta-block=size
Sets the size in bytes for the blocks used by *--fasta-idx*.

2.10 CSV Options

These options configure the CSV output backend.

--csv-crlf
Enables RFC4180 compliant CSV output using CRLF as line separator. This was the default behavior for the old, deprecated CSV writer enabled using *--meta-fmt*.

--csv-sep=sep (,)

Specifies the string used to separate fields. May be multple characters. By default, a comma is used when writing to STDOUT or files ending in `.csv` or `.csv.gz` and a `TAB` character is used when writing to files ending in `.tsv` or `.tsv.gz`.

--csv-id=id (name)
Specifies the field name used for the always present ID column. For sequences read from FASTA, this is the first word of the header. For sequences read from ARB, this is the `name` field. By default, the column name is `name`.

2.11 Alignment Options

--realign
Forces computing the alignment of query sequences even if a reference sequence containing the exact sequence was found. Without this flag, SINA will copy the alignment from the reference sequence.

--overhang=[attach|remove|edge] (attach)
Configures how unaligned bases at the edge of the alignment (overhanging bases) should be handled.

attach Overhang bases will be placed next to the last aligned base consecutively.

remove Overhang bases will be deleted.

Todo: This feature appears to be broken.

edge Overhang bases will be placed next to the outer edge of the alignment.

--lowercase=[none|original|unaligned]
Configures which bases should be written using lower case characters.

none All bases will use upper case characters

original All bases will be written using the case they had in the input data.

unaligned Aligned bases will be written in upper case; unaligned bases will be written in lower case. This serves to mark sections of the query sequences that could not be aligned because they were insertions (internal or edge) with respect to any of the reference sequences.

--insertion=[shift|forbid|remove]
Configures how the alignment width is preserved.

shift The alignment is executed without constraining insertion sizes. Insertions for which insufficient columns exist between the adjoining aligned bases are force fitted into the alignment using NAST. That is, the minimum number of aligned bases to the left and right of the insertion are moved to accommodate the insertion.

This mode will add warnings to the log for each sequence in which aligned bases had to be moved.

forbid The alignment is executed using a scoring scheme disallowing insertions for which insufficient columns exist in the alignment.

This mode causes less "misalignments" than the **shift** mode as it computes the best alignment under the constraint that no columns may be added to the alignment. However, it will not show if the computed alignment suffered from a lack of empty columns.

remove The alignment is executed without constraining insertion sizes. Insertions larger than the number of columns between the adjoining aligned bases are truncated.

While this mode yields the most accurate alignment for sequences with large insertions, it should be used with care as it modifies the original sequence.

--fs-no-graph
Instructs SINA to use a profile vector instead of a DAG to perform the alignment. That is, the base frequencies for all selected reference sequences are collected into a vector and the query is aligned to this vector weighting the alignment scores according to the respective frequencies.

This feature was added in response to the requests of a reviewer of the original SINA publication and only intended to demonstrate that the DAG/POA approach is superior to the profile vector approach. Do not use this other than for testing.

--fs-weight=weight (1)
Adjust the weight factor for the frequency at which a node was observed in the reference alignment. Use 0 to disable weighting.

This feature prefers the more common placement for bases with inconsistent alignment in the reference database.

--match-score=n (2)
Configures the score given for a match (should be positive).

--mismatch-score=n (-1)
Configures the score given for a mismatch (should be negative).

--pen-gap=n gap open penalty (5)
Configures the penalty subtracted from the score for opening a gap (should be positive).

--pen-gapext=n
Configures the penalty subtracted from the score for extending a gap (should be positive).

--debug-graph
Writes the DAG computed from the reference sequences for each query sequences to disk in dot format.

--use-subst-matrix
Weights the match and mismatch scores according to the overall base frequencies observed in the database.

This feature is experimental and does not currently improve the results.

--write-used-rels
Writes the names of the alignment reference sequences into the field *used_rels*. This option allows using the ARB *mark used rels* feature to highlight the reference sequences used to align a given query sequence.

--calc-idty
Computes the highest similarity the aligned query sequence has with any of the sequences in the alignment reference set. The value is written to the field *align_ident_slv*.

2.12 Advanced Reference Selection Options

--ptdb=filename
Alias of *--db* for backwards compatibility.

--ptport=port_or_socket (:/tmp/sina_pt_<pid>)
Configures the port or socket on which the ARB PT server for the reference alignment is expected or started. To use a TCP port, specify <*hostname*>:<*port*>. If <*hostname*> is not *localhost*, the PT server must be launched externally. To use a Unix socket, specify *:*<*filename*>.

When *--num-pts* is greater than 1, the additional PT servers port names are generated by appending the respective number. Using port numbers greater of equal to 10000 will therefore not work.

By default, the file */tmp/sina_pt_*<*pid*> is used, where <*pid*> is replaced by the process ID of the SINA instance.

--fs-kmer-no-fast
Use all k-mers occurring in the query sequence in the search. By default, only k-mers starting with an A are used for extra performance.

--fs-kmer-mm=n (0)
Allow k-mer matches in the reference database to contain *n* mismatches. This feature is only supported by the **pt-server** search engine and requires substantial additional compute time (in particular for $n > 1$).

--fs-kmer-norel
Use absolute (number of shared k-mers) match scores in the kmer search rather than relative (number or shared k-mers divided by length of reference sequence) match scores.

--fs-msc-max=id (2)
Overriding all other options, reference sequences having a similarity with the query higher than this value are excluded from the alignment reference.

This option artificially increases the difficulty of the alignment by increasing the distance of a query to any reference found in the database. It's purpose of this option is to generate a sufficiently large *N* of test cases for statistical analysis of SINA's accuracy for sequences distant to the reference alignment.

--fs-leave-query-out
Excludes sequences from the alignment reference sharing the same name as the respective query sequence. (For testing and evaluation).

--gene-start=n (0)
Sets the beginning of the gene within the reference alignment. See *--fs-cover-gene*.

--gene-end=n (0)
Sets the end of the gene within the reference alignment. See *--fs-cover-gene*.

--fs-cover-gene=n (0)
Similar to *--fs-req-full*, this option requires a total of *n* sequences to cover each the beginning and the end of the gene within the alignment. This option is more precise than *--fs-req-full*, but requires that the column numbers for the range in which the full gene is expected be specified via *--gene-start* and *--gene-end*.

--filter=name
Chooses an *ARB posvar filter* to use for weighting alignment positions by their variability.

--auto-filter-field=name
Configures a database field using which the value of *--filter* is determined by majority vote from the selected reference sequences. Since the filters are usually computed at domain level, this approach is usually sufficient to select an appropriate filter. For SILVA database, the field *tax_slv* contains appropriate data.

--auto-filter-threshold arg
Sets the minimum quorum required for automatic filter selection. See *--lca-quorum* for information on how the value is interpreted.

--fs-oldmatch
Use the pre-1.6.0 implementation for composing the alignment family. Requires *--fs-engine* = pt-server.

2.13 Search & Classify Options

--search-port=port_or_socket (:/tmp/sina_pt2_<pid>)
See *--ptport*. This option sets the port for the search database. It is only used if *--search-db* is specified and its value differs from the one given by *--db*.

--search-all
Calculate the similarity of the query sequences with **all** reference sequences. Normally, SINA will only calculate the similarity for the sequences returned by a k-mer based similarity search. See also *--search-kmer-candidates*.

--search-no-fast don't use fast family search
See *--fs-kmer-no-fast*. This option configures the same behavior for the search stage.

--search-kmer-candidates=n (1000)
Configures the number of candidate reference sequences retrieved from the k-mer based search. For each candidate, the MSA based similarity is calculated and the search result based on these numbers. A value for *n* one or two orders larger than *--search-max-result* is usually quite sufficient.

--search-kmer-len=n (10)
See *--fs-kmer-len*. Sets *k* for the kmer based candidate search.

--search-kmer-mm arg
See *--fs-kmer-mm*. Sets the number of allowed mismatches within each kmer. Only available with the **pt-server** search engine.

--search-kmer-norel
See *--fs-kmer-norel*. Configures the candidate search to use absolute rather than length-relative scores for ordering the results.

--search-ignore-super
Omit reference sequences of which the query is an exact sub-string from the result. Useful for testing and evaluation of the classification feature.

--search-copy-fields=fields
Specifies a (colon or comma separated) list of meta-data fields to be copied from each search result sequence into the output sequence. In the output sequence, the field names will each be prefixed with *copy_<acc>_* where *<acc>* is the value of the *acc* field in the reference.

Use *--arb-list-fields* for listing the fields available in a given ARB database.

--search-iupac=[pessimistic|*optimistic|exact] (optimistic)
Configures how ambiguous bases are matched when computing the scores for the search results.

pessimistic Ambiguous bases do not match anything because they *could* always be a mismatch.

optimistic Ambiguous bases are considered matches if a match with the other (potentially also ambiguous base) is possible. That is, *N* will match everything, including *Y*.

exact Matches on character level. *N* matches exactly *N*.

--search-correction=[none|jc] (none)
Apply distance correction to search result scores.

none Leave score unmodified.

jc Apply Jukes-Cantor correction.

--search-cover=[abs|query|target|min|max|avg|overlap|all|nogap] (query)
Compute sequence similarity as the fraction of the number of matches and

abs the number 1: yields the absolute number of matching bases

query the length of the query sequence. Yields the fraction of the query covered by the reference sequence.

target the length of the target sequence. Yields the fraction of the result sequence covered by the query sequence.

min the length of the shorter of the sequences compared.

max the length of the longer of the sequences compared.

avg the average length of the two sequences compared.

nogap the number of columns in which both sequences have bases. Yields the equivalent of *matches / (matches+mismatches)*.

all the number of columns in which either sequence has a bases. Similar to **nogap**, but does not ignore indel events.

overlap the length of the overlapping portion of the two sequences.

--search-filter-lowercase
Ignore lowercase bases when scoring result sequences. This can be used in conjunction with *--lowercase*=unaligned to ignore unaligned bases during the search and classification stage.

CHAPTER 3

Field Reference

SINA generates a number of named meta-data values for each processed sequence. By default, all values will be written to ARB output files, while CSV output requires that each meta-data field is specified using *-f/--fields*. The fields generated by SINA and the typical fields present in ARB databases are described below.

3.1 Basic Fields

The below fields are standard ARB meta data fields describing each sequence.

name
: The `Id` of the sequence. In FASTA input and output, this value is mapped to the first word of the header line.

full_name
: The textual description of the sequence. In FASTA input and output, this value is mapped to all but the first word of the header line.

acc
: The sequence accession number. This field is relevant for ARB input/output as together with *start* it defines the unique identity of the sequence when regenerating sequence names. For sequences read from FASTA and written to ARB, SINA will generate a pseud-accession as `ARB_` followed by a 8 character hexadecimal CRC32 checksum. This matches the behavior of ARB during FASTA import.

version
: The version part of the accession number reference.

start
: The start position of the gene sequence with the sequence referenced by the accession number.

stop
: The stop position of the gene sequence with the sequence referenced by the accession number.

3.2 SINA specific fields

align_quality_slv
The alignment "quality". The alignment score, normalized to remove weighting effects and scaled as integer between 0 and 100. If the alignment for the sequence was copied from an identical match to a reference sequence, the value is set to 100.

align_cutoff_head_slv
The number of unaligned basepairs at the beginning of the sequence.

align_cutoff_tail_slv
The number of unaligned basepairs at the end of the sequence.

aligned_slv
The time and date at which the sequence was aligned.

align_startpos_slv
The position of the first base of the sequence within the reference alignment.

align_stoppos_slv
The position of the last base of the sequence within the reference alignment.

align_ident_slv
The highest fractional identity of the aligned sequence with any of the used reference sequences. The value is computed using optimistic IUPAC comparison (N matches anything) over the overlapping region of each pair of sequences.

nuc_gene_slv
The number of basepairs aligned within the gene. (Currently not computed).

align_bp_score_slv
A score indicating the average binding strength of basepairs aligned into helix regions. Each pair of bases aligned to opposing sides of a helix specified in the reference database is assigned a score (`AG` = 0.5, `AU` = 1.1, `CG` = 1.5, `GG` = 0.4, `GU` = 0.9), the sum of scores divided by the number of helix positions with bases on either side and multiplied by 100.

align_family_slv
The reference sequences used to align the query sequence. Each reference is listed as `ACC.START:SCORE` where `ACC` and `START` are the contents of the reference sequence's respective *acc* and *start* fields and `SCORE` is the score assigned by the sequence search engine (ARB PT server or internal kmer search).

align_log_slv
A log of events that occurred during the alignment of a query sequence.

align_filter_slv
The weighting filter selected for the query sequence, if any.

nearest_slv
The results from the sequence search. Available only when the search stage is enabled (*-S/--search*).

Each matched sequence is given as `ACC.VERSION.START.STOP~SCORE` where `ACC`, `VERSION`, `START`, and `STOP` are the contents of the matched sequence's respective *acc*, *version*, *start* and *stop* fields and `SCORE` is the score calculated according to the search settings.

3.3 SILVA taxonomy fields:

The SILVA SSU and LSU databases in ARB format contain taxonomic meta data suitable for generating taxonomic assingments using the :option:`--lca-fields` option. Each of the following fields contains the taxonomic assign-

ment as a "materialized path" (`Domain; Phylum; ...`). The `_name` field contains the sequence name assigned by the respective taxonomy.

tax_slv
: The SILVA taxonomy.

tax_embl
: The EMBL-EBI/ENA taxonomy. Note that the name was changed to *tax_embl_ebi_ena* in newer releases of SILVA.

tax_embl_ebi_ena
: The EMBL-EBI/ENA taxonomy.

tax_ltp
: The Living Tree Project (LTP) taxonomy.

tax_gg
: The Greengenes taxonomy. (Discontinued)

tax_rdp
: The RDP II taxonomy.

tax_gtdb
: The Genome Taxonomy Database Taxonomy.

3.4 Additional standard ARB fields:

ali_16s/data
: The actual sequence alignment. This field type always has the form `ali_<name>/data`, with `<name>` indicating the alignment (ARB databases may contain multiple alignments).

ARB_color
: A number indicating in which color the sequence should be highlighted inside of ARB.

used_rels
: The *`names`* of the reference sequences used during alignment separated by spaces. This field is generated only if *`--write-used-rels`* is given. It allows selecting the reference sequences via a special menu item from within ARB.

nuc
: The length of the sequence.

CHAPTER 4

Changelog

4.1 Version 1.7.2:

- Fix building without TBB Malloc library (#98)

4.2 Version 1.7.1:

- Fix rounding error in lca classifier (#93)

4.3 Version 1.7.0:

- allow multiple output types at once
- add dedicated CSV/TSV output (#10)
- fix loading reference database from running ARB (#76)
- report errors when sequence can't be read from ARB (#73)
- add `--arb-list-fields` listing fields available in ARB database

4.4 Version 1.6.1:

- fix progress bar not honoring verbosity (#85)

4.5 Version 1.6.0:

- make internal kmer engine the default (#23)

- add pretty progress monitor
- run search stage in parallel (#32)
- `--num-pts` defaults to number of cores available (previous: 1)
- add `--search-engine` setting search engine for search module
- always run internal engine without thread limit
- split num pt servers evently between search and align
- use fixed point format for logging (instead of scientific format)
- rewrote family selection (use `--fs-oldmatch` for old implementation)
- replace boost::mutex with std::mutex (c++11)
- fix `--show-dist` if alignment width don't match
- fix race starting pt servers (library code not threadsafe)
- fix engine type not shown in `--show-conf`
- fix writing to ARB sequence cache not threadsafe
- use lock free map for ARB sequence cache (speedup)
- add pod buffer to replace std::vector (speedup)
- add FIFO cache for kmer search results (speedup for `--search` and `--turn`)

4.6 Version 1.5.0:

- update documentation (#20)
- reinstate `--show-dist`
- reinstate `--fs-msc-max`
- add choice `exact` to `--search-iupac`
- change default for `--search-kmer-len` to match `--fs-kmer-len`
- parallelize launch of background PT servers
- lower memory usage: - avoid redundant sequence caching by libARBDB - use compact aligned base (50% on internal sequence cache)
- improve internal kmer search performace - add caching of kmer index on disk - parallelize kmer index construction - add presence/absence optimization
- fix field *align_ident_slv* added for 100% matches even when not enabled
- fix crash on overhang past alignment edge
- fix libARBDB writing to stdout, clobbering sequence output
- fix out-of-bounds access on iterator in NAST implementation
- remove dependency on boost serialization library
- build release binaries with GCC 7 and C++11 ABI
- add integration tests watching for accuracy regressions (#25)

4.7 Version 1.4.0:

- process sequences in parallel (#17, #31)
- add support for gzipped read/write (#29)
- add support for "-" to read/write using pipes
- remove internal pipeline in favor of TBB
- add `--add-relatives`; adding search result to output (#19)
- add logging with variable verbosity (#14)
- be smart about locating arb_pt_server binary (#30)

4.8 Version 1.3.5:

- report number of references discarded due to configured constraints
- fix crash if no acceptable references found for a query
- fix `--search` causes a program option error (#28)
- fix race condition in terminating PT server

4.9 Version 1.3.4:

- build binary releases for macOS and Linux (#26)
- fix "search.h" missing in source tar ball (#27)

4.10 Version 1.3.3:

- add option `--fasta-write-dots`; writes dots on edges
- add option `--fasta-write-dna`; writes T/t instead of U/u (#24)
- fix PT server fails to build if ARBHOME not set (#15)
- fix psina not installed to $bindir
- fix tab character in sequence causes sequence to be skipped (#21)
- fix last line of input FASTA ignored if missing newline (#16)
- fix `--db` parameter demanded even if not required due to use of `--prealigned`
- fix SIGPIPE race on PT server shutdown (#11)

4.11 Version 1.3.2:

- split `--help` into "common" and advanced options (`--help-all`)
- add psina wrapper script (runs parallel instances of SINA to align a single FASTA file)

- fix memory access failure in cseq
- fix memory access failure in mseq
- fix crash on all references removed by filters
- don't exit(1) on `--help` (#9)
- added README.md (#5)

4.12 Version 1.3.1:

- add OSX support
- change license to GPL
- remove limitation on ARB integration mode
- move revisioning to git
- fix compilation with CLANG

4.13 Version 1.3.0:

- dropped support for ARB 5.x

4.14 Version 1.2.13:

- uppercase aligned bases if lowercase=unaligned
- fix manual typos (thx to Mohamed El-hadidi)
- search-db defaults to pt-db
- search-port defaults to pt-port if search/align DBs are identical fixes unnecessary start of two PT servers (thx to Christian Wurzbacher)
- change default for lca-quorum to 0.7
- change default for search-min-sim to 0.7
- be smarter about recoginizing FASTA format files and creating output FASTA name (".frn", ".fna", ".fas", "/dev/stdin" as input, ".fasta.aligned" and "/dev/stdout" as output)
- write sequence ID in first column of CSV output
- add fasta-block and fasta-idx options allowing to process only specific smaller blocks of larger fasta files (for parallelization)

4.15 Version 1.2.12:

- use same ARB field type for align_ident_slv as SILVA uses
- skip sequences with non-IUPAC characters when building reference and when loading sequences to be aligned from ARB file (complaint is issued on stderr)

4.16 Version 1.2.11:

- fix `--fs-req` was ignored
- added option `--calc-idty` Computes the minimum identity of the aligned query sequence with any of the reference sequences used for alignment. The value is exported in align_slv_idty.
- added option `--min-idty` IDTY Excludes sequences with align_slv_idty < IDTY from FASTA output. Implies `--calc-idty`.

4.17 Version 1.2.10:

- added option `--fs-no-graph` Uses a column profile with PSP score as template (instead of the POA method) This feature is merely for completeness sake and evaluation. With SILVA SSU the POA based method is much more accurate.
- changed default for `--fs-cover-gene` to 0 (faster) The cover-gene feature only makes sense if *:option:*–gene-start‘ and `--gene-end` are set such that the reference actually contains sequences touching these boundaries. If this is not the case, the reference selection algorithm wastes time with a futile search.
- use unix socket as default for `--ptport` and `--search-port` Using “/tmp/sina_<PID>.socket” is a more suitable default than “localhost:4040”, as it runs less risk of accessing a different PT server than intended.
- fix inconsistencies in generated meta data fields and log output
- updated ARB components to SVN revision 8225
- added option `--write-used-rels` The field used_rels is interpreted by ARB as the field containing the reference sequences that were used during alignment.
- no longer write full_name content when exporting meta data encoded in the FASTA header
- re-add clamped align_quality_slv
- fix score normalization (scores > 1 were possible when fs-weight > 0)
- fix calculation of bp score when orig-db no set (default ptdb)
- added option `--fs-req-gaps` n Ignores reference sequences having less than n gaps before the last base. I.e.: Ignores “unaligned” sequences. This is useful when running SINA out of ARB to prevent accidental alignment against unaligned sequences.
- added options `--search-iupac`, `--search-correction` and `--search-cover` These options configure how the “distance” (identity, similarity, . . .) is calculated.
- skip FASTA input sequences that contain invalid characters (i.e. not IUPAC encoded bases, ‘.’, ‘-‘ or white space)

4.18 Version 1.2.9:

- fixed sequence not filled with gap characters after copying full alignment

4.19 Version 1.2.8:

- made –extra-fields actually load multiple fields from arb file

- fixed sequence not filled with gap characters after copying subalignment
- updated ARB components to SVN revision 7985
- added changelog :)

Index

Symbols

A

C

F

N

S

T

U

V