SSU-ALIGN User's Guide

Structural alignment of 16S and 18S small subunit (SSU) ribosomal RNA sequences

eddylab.org/software/ssu-align
Version 0.1.1; Feb 2016

Eric P. Nawrocki
nawrocke@ncbi.nlm.nih.gov
http://eddylab.org/

Contents

1 Introduction

SSU-ALIGN identifies and aligns small subunit ribosomal RNA (SSU rRNA) genes in sequence datasets. It uses models of the primary sequence and secondary structure conservation of SSU rRNA as inferred by comparative sequence analysis and confirmed by crystal structure determination from the comparative rna website (crw) (`http://www.rna.ccbb.utexas.edu/`) (Cannone et al., 2002).

Overview

The SSU-ALIGN package contains six different programs: **`ssu-align`**[1], **`ssu-build`**, **`ssu-draw`**, **`ssu-mask`**, **`ssu-merge`**, and **`ssu-prep`**.

Suppose you have a dataset of SSU rRNA sequences from a PCR-based environmental survey. You can use **`ssu-align`** to create structure-based multiple sequence alignments of the SSU sequences using profile probabilistic models call covariance models (CMs). **`ssu-align`** will determine whether each sequence is archaeal, bacterial or eukaryotic and output a multiple alignment for each domain that was assigned at least one sequence. The **`ssu-mask`** program can then be used to identify and remove columns that are not reliably aligned. This step is important prior to using the alignments as input to phylogenetic inference tools that rely heavily on alignment accuracy. The **`ssu-draw`** program can be used to create secondary structure diagrams of your alignments. Diagrams that display per-column alignment statistics, such as information content and frequency of insertions and deletions, as well as those that display individual aligned sequences can be drawn.

The **`ssu-prep`** and **`ssu-merge`** programs allow users to divide up large alignment jobs into sets of parallel **`ssu-align`** jobs on clusters or multi-core computers.

`ssu-build` can be used to create different models than the default archaeal, bacterial, and eukaryotic ones. For example, you might want to build a model that represents a more specific phylogenetic range, like the *Firmicutes* bacterial phyla to create maximally precise *Firmicutes* SSU alignments. Or you can build truncated versions of the default models that model only a specific region of SSU, such as the V4 hypervariable region.

ssu-align's search and alignment stages

Given an input dataset of unaligned SSU sequences, **`ssu-align`** proceeds through two stages to generate alignments as depicted in figure 1. In the first stage, each target sequence is scored with each of the models based only on sequence conservation. This is done with a profile HMM derived from each CM, which is significantly faster than using the CM. The model whose profile HMM gives the highest score to each sequence is defined as the *best-matching* model for that sequence. If this highest score is not above a predefined threshold, the sequence is discarded and not evaluated further. The boundaries of the best-matching HMM alignment are used to truncate each target sequence if the alignment does not span the entire target length. In stage 2, each surviving, and possibly truncated, sequence is aligned to its best-matching model, this time using the CM which scores both sequence and conserved structure. Up to N alignments are created, one for each model that was the best-match to at least one target sequence.

What distinguishes SSU-ALIGN from other SSU alignment tools

- **Structural alignments are calculated at roughly one second per sequence.**

  CM alignment explicitly takes into account conserved sequence and well-nested structure. In the past, the slow speed of CM alignment has prevented their application to SSU alignment, but recent

[1] SSU-ALIGN is the name of the software package as well as the name of one of the programs within the package. In this guide, when written in lowercase bold-faced type (**`ssu-align`**) it refers to the executable program, and when written in all capital letters (SSU-ALIGN) it refers to the package.

sequence-based acceleration heuristics (Brown, 2000; Nawrocki, 2009) have made large-scale SSU CM alignment feasible.

- **Alignment-specific masks for removing ambiguously aligned columns are automatically generated.**

  CMs are probabilistic models that allow calculation of the posterior probability that each sequence nucleotide belongs in each column of the output alignment given the model. Regions of low posterior probabilities are indicative of high alignment ambiguity and should be pruned away (masked out) prior to phylogenetic inference. **`ssu-mask`** automatically detects and remove these columns for any alignment it generates. This is discussed in more detail in the section 3 of this guide.

- **Accurate alignments can be computed using seed alignments of less than one hundred sequences.**

  CMs built from small seed alignments of dozens of sequences can achieve comparable accuracy to nearest-neighbor based alignment methods that use seed (reference) alignments of thousands or tens of thousands of sequences. This reduces the level of manual curation necessary for creating useful seeds, making it easier to extend SSU-ALIGN by adding new seeds and corresponding CMs that cover specific phylogenetic ranges.

What is included in this guide

This user's guide is long. If you'd like to get started using the software right away, take a look at section 2, Installation and then section 4, Tutorial. You can come back to the other sections later if you'd like.

Section 2 describes how to install the package. Section 3 provides background information on how SSU-ALIGN aligns SSU sequences, the distinction between *consensus* and *insert* columns in its output alignments, and how it identifies and removes columns from (masks) those alignments. The tutorial in section 4 walks through examples of using the programs in SSU-ALIGN to create alignments, mask alignments, build models of a specific region of SSU, draw structure diagrams for alignments, and split up large alignment jobs into smaller jobs to run in parallel on a cluster. Section 5 describes how the three default models were derived from CRW. Section 6 details how the default masks for the models were determined. Section 8 contains timing statistics and output file sizes for aligning, masking and drawing various sized datasets. Section 7 describes the format of output files generated by SSU-ALIGN programs. Finally, the manual pages included at the end of this guide explain the command-line options of the six programs.

What is included in this package

The six SSU-ALIGN programs: **`ssu-align`**, **`ssu-build`**, **`ssu-draw`**, **`ssu-mask`**, **`ssu-merge`** and **`ssu-prep`** are PERL scripts. The package includes the three default (archaea, bacteria, and eukarya) SSU models and the *seed* alignments they were built from. These alignments were based on SSU structures and alignments from the Comparative RNA Website (CRW) (Cannone et al., 2002). A broader discussion of these models and alignments and the specific procedure used to create them from the CRW data is explained in section 5.

SSU-ALIGN bundles the INFERNAL software package, which is written in C and is installed along with SSU-ALIGN by following the directions in the Installation section of this guide. INFERNAL itself includes Sean Eddy's HMMER package and his sequence analysis library EASEL. All of the programs in INFERNAL as well as small-scale EASEL applications for sequence file manipulations are installed with SSU-ALIGN. Importantly, the versions of the INFERNAL and EASEL programs get renamed and given a **`ssu-`** prefix to avoid namespace clashing with programs of the same names if you have INFERNAL installed on your system. For example, **`cmsearch`** and **`esl-seqstat`** as installed with SSU-ALIGN are actually called **`ssu-cmsearch`** and **`ssu-esl-seqstat`**.

What this package depends on

SSU-ALIGN was developed and tested on Mac OS/X and Linux (Redhat), but it should work on most UNIX platforms. It requires a C compiler and PERL (Practical Extraction and Report Language, Larry Wall) interpreter package version 5.0 or later. No other programs or libraries are required. If the program **`ps2pdf`** is installed on your system and in your PATH, **`ssu-draw`** and **`ssu-mask`** will use it to convert postscript diagrams to PDF, but it is not required.

What this package does not do

SSU-ALIGN only creates alignments, masks them, and draws SSU secondary structure diagrams; it does not infer trees from the alignments it creates, nor does it classify sequences beyond reporting which model in the input CM file they score highest to.

How to cite SSU-ALIGN

SSU-ALIGN does not yet have an associated publication, so please cite the INFERNAL software publication ((Nawrocki et al., 2009a)) if you find the package useful for work that you publish. Additionally, because SSU-ALIGN's seed alignments were derived from the comparative rna website we ask that you cite that database as well: (Cannone et al., 2002).

Future development

We consider version 0.1.1 of SSU-ALIGN to be a prototype. Compute time for alignment is roughly one second per full length SSU sequence. We hope to speed up the program and add additional SSU models in future releases. Bug reports and feature requests are appreciated. Please send them to nawrocke@ncbi.nlm.nih.gov.

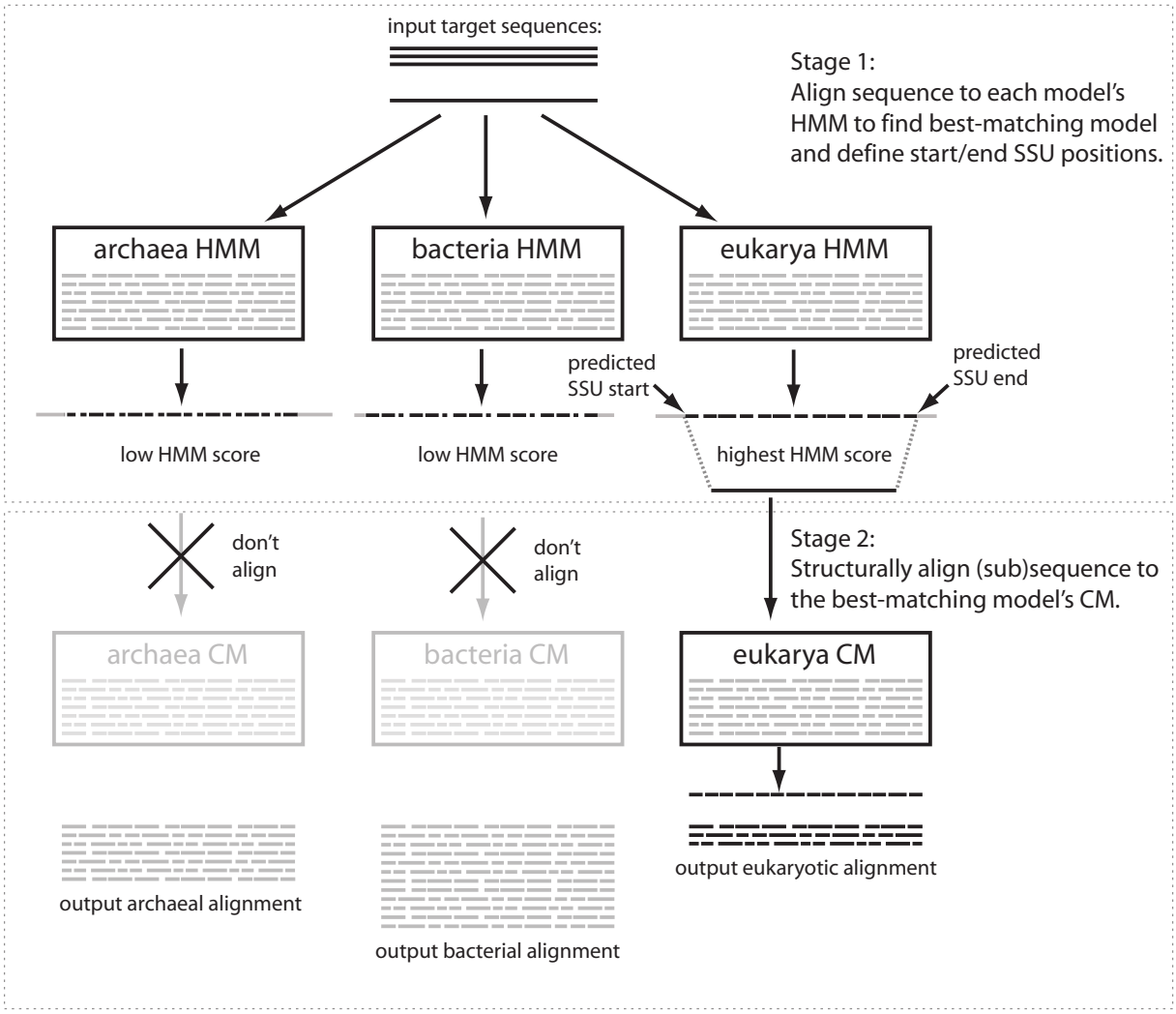

Figure 1: **Schematic of the `ssu-align` alignment pipeline.** Unaligned target sequences are input to the program. In stage 1, each sequence is independently aligned using only primary sequence scoring to each of N HMMs (by default, N is 3), one built from each model in the input CM file. The model whose HMM alignment has the maximum bit score is the "best-matching" model for that sequence, in this example "eukarya" is the best-matching model. In stage 2, the unaligned (sub)sequence from the best-matching model's HMM alignment (potentially with some sequence trimmed off the ends) is aligned to the best-matching model's CM which scores both sequence and conserved secondary structure. The CM aligned target sequence is added to that model's output alignment. After all target sequences are processed, the program has output up to N new structural alignments, one for each model that was the best-matching model for at least 1 target sequence.

2 Installation

Quick installation instructions

Download the source tarball (**ssu-align-0.1.1.tar.gz**) from `http://eddylab.org/software/ssu-align`, or directly from `eddylab.org/software/ssu-align/ssu-align-0.1.1.tar.gz`;
untar; and change into the newly created directory **ssu-align-0.1.1**:

```
> wget eddylab.org/software/ssu-align/ssu-align-0.1.1.tar.gz;
> tar xfz ssu-align-0.1.1.tar.gz
> cd ssu-align-0.1.1
```

Configure for your system, and build the programs:

```
> ./configure
> make
```

Install the man pages and programs in system-wide directories:

```
> make install
```

This will install three types of files: programs, man pages, and data files. By default, programs are installed in **/usr/local/bin**. Installed programs include the six main SSU-ALIGN scripts (**ssu-align**, **ssu-build**, **ssu-draw**, **ssu-mask**, **ssu-merge** and **ssu-prep**), the INFERNAL programs, each of which is prefixed with 'ssu-cm', e.g. **ssu-cmsearch**, and EASEL programs called *miniapps*, these are all named with a 'ssu-esl' prefix, e.g. **ssu-esl-seqstat**. The INFERNAL and EASEL programs are called internally by the SSU-ALIGN scripts. Normal users will probably never need to run them. By default, man pages for all programs are installed in **/usr/local/man/man1**. The file **ssu.1** in that directory is a special man page that summarizes the entire package. Finally, data files and a PERL module (**ssu.pm**) required by the programs are placed in **/usr/local/share/ssu-align-0.1.1**, by default. Data files include the default CM files, the seed alignments, mask files, and postscript secondary structure template files.

You might need special privileges to write to **/usr/local**. (You may need to execute **sudo make install**). You can change **/usr/local** to any directory you want using the **./configure --prefix** option, as in **./configure --prefix /the/directory/you/want**.

After installing, you'll see the following:

```
=====================================
SSU-ALIGN has been successfully built
=====================================

The final step is to update your environment variables:

If you are using the bash shell, add the following three:
lines to the '.bashrc' file in your home directory:

export PATH="\$PATH:/usr/local/bin"
export MANPATH="\$MANPATH:/usr/local/share/man"
export SSUALIGNDIR="/usr/local/share/ssu-align-0.1.1"

And then source that file to update your current
environment with the command:

source ~/.bashrc

If you are using the C shell, add the following three:
lines to the '.cshrc' file in your home directory:

setenv PATH "\$PATH:/usr/local/bin"
setenv MANPATH "\$MANPATH:/usr/local/share/man"
setenv SSUALIGNDIR "/usr/local/share/ssu-align-0.1.1"

And then source that file to update your current
environment with the command:

source ~/.cshrc

(To determine which shell you use, type: 'echo \$SHELL')
```

Follow these instructions to update your environment for the current and every subsequent shell session. This step is crucial because it ensures all the required executables are in your PATH. SSU-ALIGN internally calls INFERNAL and EASEL programs that were just installed, all of which must be in your PATH.

For instance, if you use the bash shell you'd open your **.bashrc** file in your home directory (e.g. **Users/nawrockie/.bashrc**, or just **/.bashrc**) and add the three lines:

```
export PATH="$PATH:/usr/local/bin"
export MANPATH="$MANPATH:/usr/local/share/man"
export SSUALIGNDIR="/usr/local/share/ssu-align-0.1.1"
```

Note that these will only work for the current configuration (that is, they'll be different if you change the installation locations with **--prefix** or something like it). Then, source this file to update the current session:

```
> source ~/.bashrc
```

After doing this, you can test if your environment is setup correctly by checking the following two commands:

```
> which ssu-align
```

should return the path to the **ssu-align** script you just installed (**/usr/local/bin/ssu-align** if you configured with default settings), and

```
> echo $SSUALIGNDIR
```

should return the path to the your data directory (**/usr/local/share/ssu-align-0.1.1** if you configured with default settings).

If you want to, at this point you can run the automated testsuite. (You can't run the testsuite before installation because it requires all the INFERNAL and EASEL programs be in your path.) This step is optional. It takes about 10 minutes, and all these tests should pass:

```
> make check
```

That's it. You can keep reading if you want to know more about customizing a SSU-ALIGN installation, or you can skip ahead to section 4, the tutorial.

More detailed installation notes

The six main programs in the SSU-ALIGN package: **ssu-align**, **ssu-build**, **ssu-draw**, **ssu-mask**, **ssu-merge** and **ssu-prep** are PERL scripts. The package depends on, and includes, INFERNAL which itself includes HMMER and the EASEL sequence analysis library; all of which are distributed as ANSI C source code.

SSU-ALIGN is designed to be built and used on UNIX platforms. It is developed mainly on Intel GNU/Linux systems and Mac OS, and intermittently tested on a variety of other UNIX platforms. It is not currently tested on Microsoft Windows, but it should work there; it should be possible to build it on any platform with an ANSI C compiler. The software itself is vanilla POSIX-compliant ANSI C and PERL. You may need to work around the configuration scripts and Makefiles to get it built on a non-UNIX platform.

The GNU configure script that comes with SSU-ALIGN has a number of options. You can see them all by doing:

```
> ./configure --help
```

Be warned that some of these options (such as **--enable-mpi**) will *not* affect SSU-ALIGN performance. These options affect only how INFERNAL or HMMER is built underneath SSU-ALIGN, but have no impact on the toplevel SSU-ALIGN programs.

All customizations can and should be done at the **./configure** command line, unless you're a guru delving into the details of the source code.

Setting installation targets

The most important options are those that let you set the installation directories for **make install** to be appropriate to your system. What you need to know is that SSU-ALIGN installs three types of files: programs,

man pages, and data files. It installs the programs in `--bindir` (which defaults to `/usr/local/bin`), the man pages in the `man1` subdirectory of `--mandir` (default `/usr/local/man`) and the data files in a newly created subdirectory `ssu-align-0.1.1` of `--datadir` (which defaults to `/usr/local/share`. Thus, say you want `make install` to install programs in `/usr/bioprogs/bin`, man pages in `/usr/share/man/man1` and data files in `/usr/share/ssu-align-0.1.1`; you would configure with:

```
> ./configure --mandir=/usr/share/man --bindir=/usr/bioprogs/bin --datatdir=/usr/share
```

That's really all you need to know, since SSU-ALIGN installs so few files. But just so you know; GNU configure is very flexible, and has shortcuts that accomodates several standard conventions for where programs get installed. One common strategy is to install all files under one directory, like the default `/usr/local`. To change this prefix to something else, say `/usr/mylocal` (so that programs go in `/usr/mylocal/bin`, man pages in `/usr/mylocal/man/man1`, and data files in `/usr/mylocal/share/ssu-align-0.1.1`), you can use the `--prefix` option:

```
> ./configure --prefix=/usr/mylocal
```

In summary, a complete list of the `./configure` installation options that affect SSU-ALIGN:

Option	Meaning	Default
`--prefix=PREFIX`	all files	`/usr/local`
`--bindir=DIR`	programs	PREFIX/bin/
`--mandir=DIR`	man pages	PREFIX/man/
`--datadir=DIR`	data files	PREFIX/share/

Setting compiler and compiler flags

By default, `configure` searches first for the GNU C compiler `gcc`, and if that is not found, for a compiler called `cc`. This can be overridden by specifying your compiler with the cc environment variable.

By default, the compiler's optimization flags are set to `-g -O3` for `gcc`, or `-g` for other compilers. This can be overridden by specifying optimization flags with the `CFLAGS` environment variable.

For example, to use an Intel C compiler in `/usr/intel/ia32/bin/icc` with optimization flags `-O3 -ipo`, you would do:

```
> env CC=/usr/intel/ia32/bin/icc CFLAGS="-O3 -ipo" ./configure
```

which is the one-line shorthand for:

```
> setenv CC /usr/intel/ia32/bin/icc
> setenv CFLAGS "-O3 -ipo"
> ./configure
```

If you are using a non-GNU compiler, you will almost certainly want to set `CFLAGS` to some sensible optimization flags for your platform and compiler. The `-g` default generated unoptimized code. At a minimum, turn on your compiler's default optimizations with `CFLAGS=-O`.

Makefile targets

`all` Builds everything. Same as just saying `make`.

`check` Runs the automated test for SSU-ALIGN.

`clean` Removes all files generated by compilation (by `make`). Configuration (files generated by `./configure`) is preserved.

`distclean` Removes all files generated by configuration (by `./configure`) and by compilation (by `make`).

Note that if you want to make a new configuration you should do a `make distclean` (rather than a `make clean`), to be sure old configuration files aren't used accidentally.

3 Background

Profile versus nearest neighbor alignment strategies

There are several existing software packages dedicated to SSU alignment, including, but not limited to, NAST (DeSantis et al., 2006a) (used by the GREENGENES database (DeSantis et al., 2006b)), PYNAST (Caporaso et al., 2010), SINA (used by the SILVA database (Pruesse et al., 2007)), and MOTHUR (Schloss, 2009). Most of these methods use a *nearest-neighbor* (NN) based alignment strategy: given a new target sequence to align, each sequence in a reference alignment is evaluated to find one or more nearest-neighbor template sequences that are most similar to the target. An alignment of the target sequence to its template(s) is then computed, and this alignment is used to place the target within the context of the full reference alignment (figure 2).

In contrast, SSU-ALIGN uses a profile-based alignment strategy: each target sequence is aligned independently to a statistical model called a profile (figure 2). The profiles SSU-ALIGN uses for alignment are called covariance models (CMs) [2]. A comprehensive explanation of CMs is outside the scope of this guide. Instead, a brief discussion of some of their relevant features for using SSU-ALIGN and understanding its output is included below. For more information on CMs see (Eddy and Durbin, 1994; Durbin et al., 1998; Eddy, 2002; Nawrocki and Eddy, 2007; Nawrocki et al., 2009a; Nawrocki, 2009; Kolbe and Eddy, 2009).

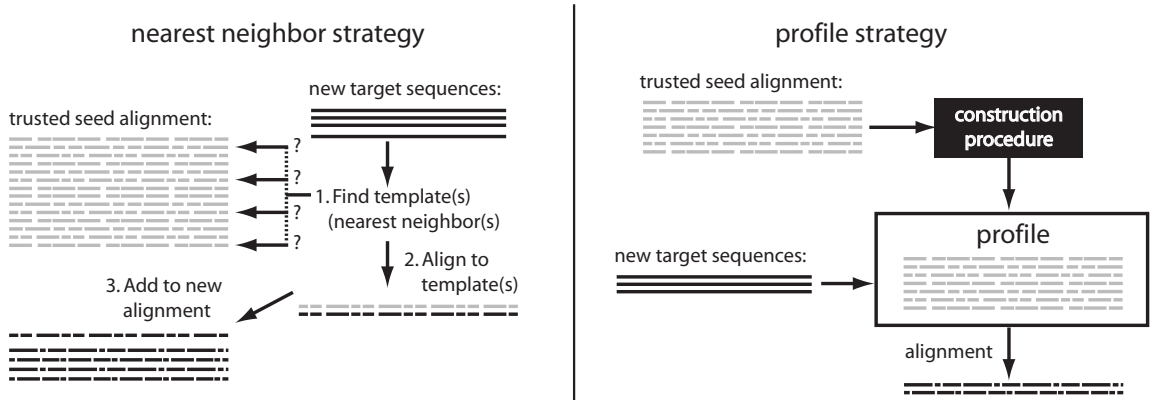

Figure 2: Schematic of nearest-neighbor and profile based alignment strategies.

Covariance models: profiles of RNA sequence and secondary structure

A CM is a consensus model of the conserved sequence and secondary structure of an RNA family that is built from a multiple sequence alignment. This alignment is called the *seed* alignment of the model. The seed alignment should include a representative set of sequences from the family being modeled and must include consensus structure annotation indicating which positions in the alignment are basepaired to each other. Given a new target sequence, its best-scoring alignment to the CM can be computed using a dynamic programming algorithm. Both primary sequence conservation and secondary structure conservation between the target and the model contribute to the alignment score. Figure 3 depicts the building of a CM for a small, made-up RNA family and the alignment of sequences using that CM.

[2] SSU-ALIGN actually uses two types of profiles, CMs and profile HMMs, which model only conserved primary sequence. Profile HMMs are used to determine which domain each sequence belongs to, and then the corresponding domain-specific CM is used to align the sequence, as depicted in figure 1.

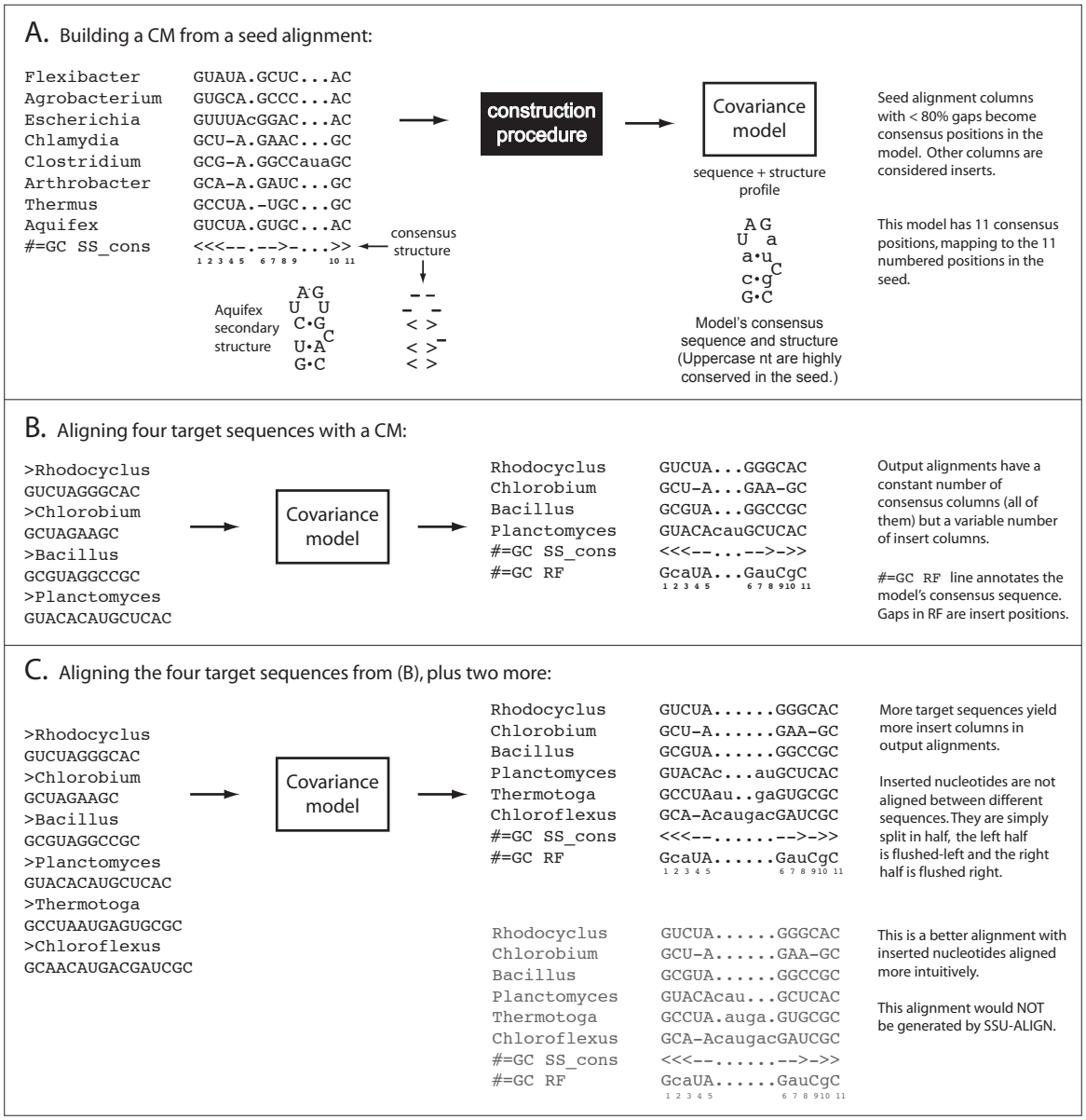

Figure 3: **Building and aligning sequences to an example CM.** (A) The made-up seed alignment of 8 homologs of a small stem-loop RNA family is sent through the black-box construction procedure to create a CM that models the family. The **`#=GC SS_cons`** line annotates the consensus structure as shown. Columns that are . characters in this line that have more than 80% gaps and so are defined as insert columns. All other columns become consensus positions of the model. (B) Four unaligned target sequences are aligned to the CM to generate a new structure annotated alignment that includes all 11 consensus columns and 3 insert columns. (C) Six unaligned target sequences, including the same 4 from (B), are aligned to the CM to generate a new alignment that includes all 11 consensus columns and 5 insert columns. Nucleotides in insert columns are not aligned between different sequences, but are simply split and flushed left and right. The bottom alignment has the inserted nucleotides aligned more reasonably, but would not be created by SSU-ALIGN

.

Profiles use position-specific scores

The scores used when aligning sequences to a CM are position-specific and are derived from the observed nucleotide distributions in the seed alignment. As a simple example, consider the small structural RNA family in figure 3. This RNA forms the simple stem-loop structure shown in part A of the figure. The seed alignment is in Stockholm format. The `#=GC SS_cons` line maps the secondary structure onto the alignment. Basepairs in this line are indicated by matching < with >, just like matching parentheses in a mathematical formula. When a CM is built from this seed alignment, each of the columns that include fewer than 80% gaps will become consensus positions of the model. There are 11 such positions in the seed alignment, which are numbered. Other columns are considered insert columns; these are marked with a `.` in the `#=GC SS_cons` line, and `.` or lowercase nucleotides in the sequences. Basepaired positions in the structure will become consensus basepairs, and other positions will be single-stranded consensus positions.

Each single-stranded consensus position of the model will include scores for aligning to each of the four possible nucleotides: `A`, `C`, `G`, and `U`, based on their frequency in the corresponding position of the alignment. For example, column 5 includes an `A` in all 8 sequences in the seed alignment and so would assign high scores to `A` and low scores to the `C`, `G`, and `U`. In contrast, column 7, which includes 2 cases of each of the 4 nucleotides would assign equal, marginal scores to all 4.

Importantly, basepaired positions will assign scores to two nucleotides at once and so will include scores for each of the 16 possible pairs of nucleotides. For example, the pair between positions 1 and 11 is 100% `GC` in the seed alignment and so `GC` pairs in the target would receive high scores, while the other 15 possible pairs would receive low scores. In contrast, the pair between positions 3 and 8 includes two of each of the four possible Watson-Crick basepairs: `AU`, `UA`, `CG`, and `GC`, and so these four basepairs would receive high scores, while the other 12 possible pairs would receive lower scores.

Additionally, position-specific scores for inserting nucleotides are derived from the alignment. The only insertions in the seed alignment occur after positions 5 and 9, and so the score for inserts after those positions would be higher than for the other positions. Similarly, the only deletions, i.e. gaps in consensus positions, occur at positions 4 and 6, with one occuring in 6, and three in 4. Consequently, the deletion score for position 6 would be the highest, 4 would be second highest, and all other positions would receive lower scores. (I should point out that this discussion of CM parameterization is oversimplified for brevity. For formulas and more details, see (Eddy and Durbin, 1994; Durbin et al., 1998; Nawrocki et al., 2009a).)

In contrast, nearest-neighbor based methods often do not use position-specific scores. For example, if a single template sequence is used, aligning an `A` in the target to any `A` in the template sequence will receive an identical score. This is because, given one template sequence, there is no way for the method to distinguish the relative level of conservation at different positions. This type of pairwise alignment with position-independent scores is accurate when the template is nearly identical to the target, but becomes more difficult as that identity decreases. As a result, most nearest-neighbor based methods use very large reference alignments (thousands of sequences) to increase the probability that a nearly identical template exists for any potential target. Ensuring the accuracy of such large reference alignments is critical but time-consuming and requires expert curators. Seed alignments for profile-based methods on the other traditionally include less than one hundred sequences (Finn et al., 2010; Gardner et al., 2009).

Two types of alignment columns: consensus and inserts

Alignments created by SSU-ALIGN will contain two types of columns: consensus columns, that correspond to a consensus position of the model, and insert columns, which include nucleotides inserted between consensus positions of the model. These columns can be distinguished based on the `#=GC RF` line in the alignment files. Continuing with the small RNA example, figure 3 shows an alignment of 4 target sequences (part B) and 6 target sequences (part C) to the CM. Notice that for both alignments, the `#=GC RF` line contains 11 nucleotides, one for each of the consensus columns in the model. These are the highest scoring nucleotides at each position. Uppercase letters in this line were highly conserved in the seed alignment, lowercase letters were less well conserved. Positions that are gaps in the `#=GC RF` line contain inserted nucleotides that don't align to any of the consensus positions. SSU-ALIGN alignments created

with the same CM will always include the same number of consensus columns, but the number of insert columns will vary depending on the input sequences.

There are two important caveats regarding how CMs and SSU-ALIGN treat inserts:

1. Nucleotides in insert columns are simply inserted, not aligned.

   You might have noticed that in part (C) of figure 3, the nucleotides in the insert columns after consensus position 5 obviously misaligned with respect to each other. This is because SSU-ALIGN does not align inserted nucleotides between different sequences. Instead, inserted nucleotides are simply split in half, the left half is placed flush-left next to nearest consensus column on the left (**`au`** in **`Thermatoga`**), and the right half is placed flush-right next to the nearest consensus column to the right (**`ga`** in **`Thermatoga`**).

   This is an important limitation of profile-based alignment (at least as implemented in INFERNAL and SSU-ALIGN). In contrast, a nearest-neighbor based method might be able to correctly align nucleotides that would be inserts for a profile. However, inserted positions should be rare. Remember that consensus positions were defined as any position that had a nucleotide in at least 20% of the seed sequences, so if the seed was representative the most common inserts should only appear in about 20% of sequences. And if the alignment will eventually be used for phylogenetic inference, inserted columns in SSU-ALIGN alignments should be removed first anyway, and so are unimportant. It could be argued that the corresponding columns in NN-based alignments could be kept for phylogenetic inference, but inserts tend to occur at highly variable regions that are inherently difficult to correctly align, and so including them might confound phylogenetic inference that rely on correct alignments. This is discussed further below in the section on masking alignments.

2. Deeper alignments tend to have more insert columns.

   Because the likelihood of a large insertion in at least one sequence increases as more sequences are added, alignments of large numbers of sequences (deep alignments) tend to have a lot of insert columns. In fact, the number of insert columns can be much greater than the number of consensus columns. Table 5 in section 8 provides examples of the dominance of insert columns in very deep alignments containing more than 100,000

   This contrasts markedly with the popular nearest-neighbor based aligners NAST and SINA, which generate alignments of a fixed with (7,682 columns for NAST and 50,000 for SINA). However, to guarantee a fixed width, NAST must deliberately introduce errors in the alignment in some cases (DeSantis et al., 2006a). One reason for doing this is consistency across datasets: column x from dataset A is directly comparable to column x from datasets B and C. Importantly, SSU-ALIGN alignments have a fixed number of *consensus* columns, which facilitates this type of cross-dataset comparison for consensus positions.

Alignment posterior probabilities are confidence estimates

When a sequence is aligned to a CM, the posterior probability that each nucleotide aligns to each position of the model is calculated. The posterior probability values for the output alignment are annotated in the alignment file. These probabilities are *confidence estimates* that each nucleotide is correctly aligned, given the parameters of the CM. The estimates are useful for identifying and removing columns of the alignment that are most likely to contain errors. This procedure is called alignment *masking*. Masking is especially important prior to using the alignment as input to a phylogenetic inference program because those programs typically assume that all nucleotides in a given alignment column are homologous and so are confounded by alignment errors. **`ssu-align`** output alignments include posterior probabilities that can be used to automatically construct and apply masks using the **`ssu-mask`** program. The tutorial (section 4, page 20) includes an example of alignment masking.

Inserted columns should always be excluded during masking

Importantly, any CM alignment mask should automatically exclude every insert column of the alignment. This is because, as discussed earlier, CMs do not actually align inserted nucleotides between different sequences, but rather simply insert them between the appropriate consensus columns in the alignment (figure 3, parts B and C). This means that sequence nucleotides appearing in the same insert alignment columns are not aligned with respect to each other and consequently should be removed prior to phylogenetic analysis which assumes aligned nucleotides are homologous. **`ssu-mask`** always automatically removes all insert columns from alignments.

Consensus columns with low alignment confidence should also be removed during masking

It is necessary but not sufficient to remove insert columns during masking. Additionally, some consensus columns may include a significant number of nucleotides aligned with low confidence estimates and these should be removed as well.

Low alignment confidence tends to occur at positions with low sequence conservation where insertions and deletions are common, corresponding to regions of the molecule that exhibit high length heterogeneity across different species. In such cases, the correct alignment is often ambiguous. Take for example the CM alignment of the `GUAU` subsequence of the *Desulfovibrio desulfuricans* SSU target sequence to a loop region depicted in figure 4. The reference sequence (this is the majority-rule consensus sequence from the seed alignment) for the loop (`AUUCAAC`) differs from `GUAU` in both sequence and length. Consequently, the CM alignment for this loop is not well defined, and two alternative alignments are given posterior probabilities of 0.4 or higher. In contrast, the surrounding helix region, for which higher sequence similarity exists between the reference and target sequence, is aligned confidently with high posterior probabilities.

Automated probabilistic masking with `ssu-mask`

The default masking strategy used by **`ssu-mask`** is to remove all insert columns and any consensus column *x* for which fewer than 95% of the sequences that are nongaps in *x* have a posterior probability of less than 0.95 (i.e. any value in the **`PP`** annotation that is not **`*`**). Columns that are 100% gaps are also removed. These thresholds were chosen based on good performance using a simple SSU alignment benchmark. See chapter 9 of (Nawrocki et al., 2009a) for benchmark details. The thresholds can be changed with the **`--pf`** and **`--pt`** options to **`ssu-mask`**. Additionally, columns that include greater than a given fraction of gaps can also be removed using the **`--gapthresh`** option. See the **`ssu-mask`** manual page at the end of this guide for more details.

Alternatively, it can be useful to apply a preset mask that is not alignment-specific so that alignments of different datasets can be directly compared. SSU-ALIGN includes a default preset mask for each of the default archaeal, bacterial, and eukaryotic alignments. The determination of these is discussed in section 6. They can be applied within the **`ssu-mask`** program using the **`-d`** option.

Other useful references

For more information on CM parameterization and alignment, see the INFERNAL user's guide (Nawrocki et al., 2009b). My Ph.D. thesis (`http://eddylab.org/publications.html`), includes more background on SSU databases, alignment methods, CM acceleration heuristics used by SSU-ALIGN, and the masking benchmark mentioned earlier. Other useful references on CMs include (Eddy and Durbin, 1994; Eddy, 2002; Nawrocki and Eddy, 2007; Nawrocki et al., 2009a; Kolbe and Eddy, 2009). Finally, publications related to the RFAM database (`http://rfam.sanger.ac.uk/`), which uses INFERNAL to search for and align structural RNAs using more than 1000 different CMs, may be of interest (Griffiths-Jones et al., 2003; Griffiths-Jones et al., 2005; Gardner et al., 2009).

```
00904::Desulfovibrio_desulfuricans-1          GAUGUCGGGGA--GUAU---UCUUCGGUGUC
#=GR 00904::Desulfovibrio_desulfuricans-1 PP  ***********--6666---9**********

00904::Desulfovibrio_desulfuricans-2          GAUGUCGGGGA---GUAU--UCUUCGGUGUC
#=GR 00904::Desulfovibrio_desulfuricans-2 PP  ************---4444--9**********

#=GC SS_cons                                  <<<<<<<<<<<<.......>>>>>>>>>>>>
#=GC RF                                       GGUGUuGGgggcAuUcaACgcccUCaGUGCC
```

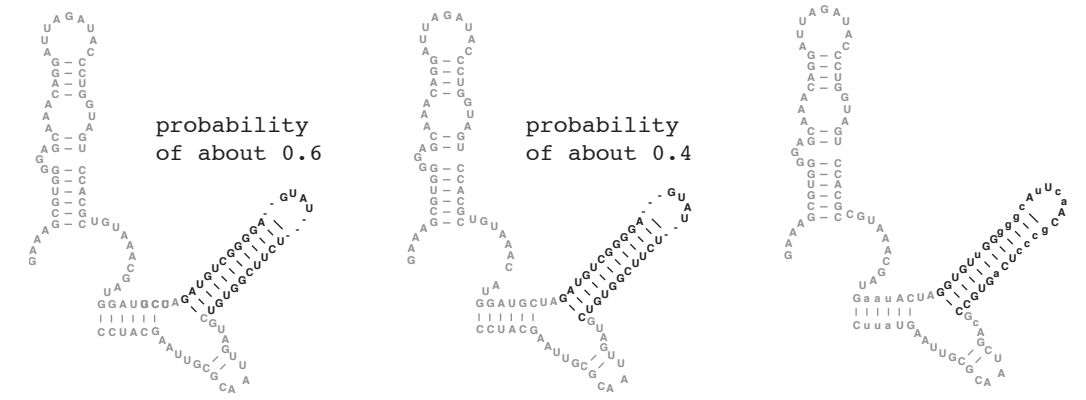

Figure 4: **Example of alignment ambiguity in a hairpin loop.** Top: An alignment fragment of two different alignments of the *Desulfovibrio desulfuricans* (sequence accession M34113) sequence from the SSU-ALIGN bacterial seed alignment for the region between consensus columns 861 and 881. Each alignment is annotated with its posterior probability in the #=GR PP rows. Characters in PP rows have 12 possible values: "0-9", "*", or ".". If ".", the position corresponds to a gap in the sequence. A value of "0" indicates a posterior probability of between 0.0 and 0.05, "1" indicates between 0.05 and 0.15, "2" indicates between 0.15 and 0.25 and so on up to "9" which indicates between 0.85 and 0.95. A value of "*" indicates a posterior probability of between 0.95 and 1.0. Higher posterior probabilities correspond to greater confidence that the aligned nucleotide belongs where it appears in the alignment. For example, the second A in the first aligned sequence has a PP value of '6' indicating a posterior probability of between 0.55 and 0.65 of being aligned in its current position. The probability this A aligns in the next position over, as it does in the second alignment, is between 0.35 and 0.45 as indicated by the 4 in that position. The #=GC SS_cons and #=GC RF rows correspond to the consensus secondary structure and sequence respectively. Bottom: The secondary structures corresponding to the two possible alignments of the *D. desulfuricans* and the reference alignment. Nucleotides in the actual alignment are black. Nucleotides surrounding the alignment fragment are gray.

4 Tutorial: Creating, masking and visualizing SSU alignments

Here is a tutorial walk-through of some small projects with SSU-ALIGN. To follow along with this tutorial, move into the **tutorial/** subdirectory of the ssu-align-0.1.1/ directory where you unpacked and built the package. The instructions in this tutorial assume that you have already installed the package (see the Installation section) and all of the SSU-ALIGN executable programs in your PATH. For example, you should be able to run **ssu-align** by simply typing **ssu-align**.

Aligning SSU sequences with ssu-align

We'll use a small dataset to demonstrate how the package works. The file **seed-15.fa** contains five archaeal sequences, five bacterial sequences and five eukaryotic sequences from the SSU-ALIGN v0.1 seed alignments (there was no change to the seed alignments in v0.1.1 relative to v0.1). These seed alignments were derived from alignments from the CRW database (Cannone et al., 2002) as described in section 5.

Pretend that **seed-15.fa** is a set of SSU sequences obtained from an environmental sampling survey and that we want to analyze. First, we can run the **ssu-align** program to classify each sequence by its domain, and create an alignment for each domain by executing the command:

```
> ssu-align seed-15.fa myseqs
```

ssu-align takes two command line arguments. The first is the target sequence file. The second is the name of a directory that the program will create and place its output files into. This directory should not yet exist[3].

The program will first print a header describing the program version used, command used, current date, and some other information:

```
# ssu-align :: align SSU rRNA sequences
# SSU-ALIGN 0.1.1 (Feb 2016)
# Copyright (C) 2016 Howard Hughes Medical Institute
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# command: ssu-align seed-15.fa myseqs
# date:    Mon Feb 22 09:44:56 2016
#
# Validating input sequence file ... done.
#
# Stage 1: Determining SSU start/end positions and best-matching models...
```

In stage 1, the program scans the input sequences with each of the three default SSU models. This has two purposes. First, it classifies each sequence by determining which model in the input CM file is its "best-matching" model, defined as the model that gives the sequence the highest primary sequence-based alignment score using a profile HMM. Secondly, it defines the start and end points of the SSU sequences based on the best-matching model's alignment.

Stage 1 takes about 30 seconds on this dataset (on an Intel Xeon 3.0 GHz processor, which I'll use for all of the example runs in this guide). When it finishes you'll see:

```
# Stage 1: Determining SSU start/end positions and best-matching models...
#
# output file name         description
# ----------------------   ------------------------------------------
  myseqs.tab               locations/scores of hits defined by HMM(s)
  myseqs.archaea.hitlist   list of sequences to align with archaea CM
  myseqs.archaea.fa              5 sequences to align with archaea CM
  myseqs.bacteria.hitlist  list of sequences to align with bacteria CM
  myseqs.bacteria.fa             5 sequences to align with bacteria CM
  myseqs.eukarya.hitlist   list of sequences to align with eukarya CM
  myseqs.eukarya.fa              5 sequences to align with eukarya CM
```

[3] If the directory does exist, **ssu-align** will print an error and exit. To overwrite an existing directory, use the **-f** option.

This lists and briefly describes the seven output files the program created in the newly created **myseqs/** subdirectory of the current directory. The content and format of these files are described in detail in section 7, but I'll briefly describe them here. The first file **myseqs.tab** is output from INFERNAL's **cmsearch** program. The other six files are model-specific: two files for each model that was the best-matching model for at least one sequence in the input target sequence file **seed-15.fa**. The **.hitlist** suffixed files contain a list of the sequences that match best to the model, and the **.fa** suffixed files are those actual sequences. If any of the models had not been the best-matching model to at least one target sequence, there would be no **.hitlist** or **.fa** files for that model.

The program will now proceed to stage 2, the alignment stage. This stage serially progresses through each model that was the best-matching model for at least one sequence and aligns the best-matching sequences to that model. The alignments are computed by scoring a combination of both sequence and secondary structure conservation (see section 3 for further discussion). As the alignment to each model finishes, three new lines of text, one for each of three newly created files, will appear on the screen. For this example, alignment to all three models takes about 20 seconds. When it finishes you'll see:

```
# Stage 2: Aligning each sequence to its best-matching model...
#
# output file name         description
# ----------------------   ---------------------------------------
  myseqs.archaea.stk       archaea alignment
  myseqs.archaea.cmalign   archaea cmalign output
  myseqs.archaea.ifile     archaea insert info
  myseqs.bacteria.stk      bacteria alignment
  myseqs.bacteria.cmalign  bacteria cmalign output
  myseqs.bacteria.ifile    bacteria insert info
  myseqs.eukarya.stk       eukarya alignment
  myseqs.eukarya.cmalign   eukarya cmalign output
  myseqs.eukarya.ifile     eukarya insert info
  myseqs.scores            list of CM/HMM scores for each sequence
```

The newly created alignments are the **.stk** suffixed files. These were created by INFERNAL's **cmalign** program. The **.cmalign** and **.ifile** suffixed files were also output by **cmalign**. As in stage 1, these files were created in the **myseqs/** subdirectory of the current directory. We'll go over the key features of the alignment files next. For more detail on all other types of output files, see section 7.

Description of alignments

The alignment files are the most important important type of output file. Take a look at the archaeal alignment we just created in **myseqs/myseqs.archaea.stk**. The alignment includes consensus secondary structure annotation and is in *Stockholm format*. Stockholm format, the native alignment format used by the HMMER and INFERNAL packages and by the pfam and rfam databases. The key features of the alignment file are:

- Each sequence occurs on a single line and each line consists of the sequence name followed by the aligned sequence[4].
- Gaps are indicated by the characters `.`, or `-`. The `.` characters occur in insert columns, while `-` characters occur in consensus columns. Page 14 in section 3 explains the difference between these two types of columns. In short, insert columns are not well conserved and typically are gaps in most sequences while consensus columns are typically non-gaps in most sequences. However, many SSU alignments will have large regions of 100% gaps in both insert and consensus columns at the beginning and ends of the alignment. This will happen if the sequences are partial SSU sequences, such as those obtained with PCR primers that target well conserved regions within the SSU molecule.

[4] Alternatively, interleaved alignment files can be created using the **-i** option. See the **ssu-align** manual page at the end of this guide for more information.

- Special lines starting with `#=GR` followed by a sequence name and then `PP` contain posterior probabilities for each aligned nucleotide for the sequence they correspond to. These are confidence estimates in the correctness of the alignment. Page 15 in section 3 introduces posterior probabilities with an example in figure 4. Characters in **`PP`** rows have 12 possible values: **`0-9`**, **`*`**, or **`.`** If **`.`**, the position corresponds to a gap in the sequence. A value of **`0`** indicates a posterior probability of between 0.0 and 0.05, **`1`** indicates between 0.05 and 0.15, **`2`** indicates between 0.15 and 0.25 and so on up to **`9`** which indicates between 0.85 and 0.95. A value of **`*`** indicates a posterior probability of between 0.95 and 1.0. Higher posterior probabilities correspond to greater confidence that the aligned nucleotide belongs where it appears in the alignment. These confidence estimates can be used to mask the alignment to remove columns with significant fractions of ambiguously aligned nucleotides as demonstrated below.
- A special line starting with **`#=GC SS_cons`** indicates the secondary structure consensus. Gap characters annotate unpaired (single-stranded) columns. Basepairs are indicated by any of the following pairs: **`<>`**, **`()`**, **`[]`**, or **`[]`**.
- A special "RF" line starting with **`#=GC RF`** indicates the consensus, or ReFerence, model. Gaps in the RF line are *insert* columns, where at least 1 sequence has at least 1 inserted nucleotide between two consensus positions. Uppercase nucleotides in the RF line are well conserved positions in the model; lowercase nucleotides are less well conserved.

Masking (removing columns from) alignments with ssu-mask

If your goal is to use a phylogenetic inference program to build trees from alignments created by **`ssu-align`**, you should mask out columns of the alignment that likely include a significant number of misaligned nucleotides first, and then only run the inference on the remaining columns where you're confident the alignment is correct.

The **`ssu-mask`** program uses the posterior probabilities (PP values) in the alignments to determine which alignment columns contain a significant fraction of nucleotides that are aligned with low confidence. It takes a single command-line argument, the name of the directory created by **`ssu-align`**. The directory must exist within the current working directory. To run it for our example, do:

```
> ssu-mask myseqs
```

As with **`ssu-align`**, the program will print information to the screen about what it is doing and the files it is creating:

```
# Masking alignments based on posterior probabilities...
#
#                                                mask
#                                            ------------
# file name                 in/out  type  #cols  incl.  excl.
# ------------------------  ------  ----  -----  -----  -----
  myseqs.archaea.stk         input   aln   1511      -      -
  myseqs.archaea.mask       output  mask   1508   1449     59
  myseqs.archaea.mask.pdf   output   pdf   1508   1449     59
  myseqs.archaea.mask.stk   output   aln   1449      -      -
#
  myseqs.bacteria.stk        input   aln   1597      -      -
  myseqs.bacteria.mask      output  mask   1582   1499     83
  myseqs.bacteria.mask.pdf  output   pdf   1582   1499     83
  myseqs.bacteria.mask.stk  output   aln   1499      -      -
#
  myseqs.eukarya.stk         input   aln   2009      -      -
  myseqs.eukarya.mask       output  mask   1881   1634    247
  myseqs.eukarya.mask.pdf   output   pdf   1881   1634    247
  myseqs.eukarya.mask.stk   output   aln   1634      -      -
```

The **`file name`** column includes file names, either input or output files, as listed in the **`in/out`** field, with type specified by the **`type`** column: **`aln`** for alignment, **`mask`** for mask file, and **`pdf`** or **`ps`** for structure diagram file. The **`cols`** column gives the number of columns for the file. The **`incl.`** and **`excl.`** columns

list the number of consensus columns of the alignment that are included (not removed) and excluded (removed), respectively, by the mask. For example, the **myseqs.archaea.stk** input alignment was initially 1511 total columns, 1508 of which were consensus columns and 3 of which were insert columns; the 3 insert columns and 59 of the 1508 consensus columns were deemed unreliable and removed by the mask based on the PP values in those columns, the remaining 1449 were saved as a new alignment to **myseqs.archaea.mask.stk**.

Importantly, all insert columns are always removed by **ssu-mask** regardless of the PP values in those columns. This is because the nucleotides in these columns are not actually aligned between different sequences, but rather simply inserted between the appropriate consensus columns in the alignment. See figure 3 (page 13) in section 3 for an example. Insert columns should therefore be removed prior to phylogenetic analysis which assumes all nucleotides in the same column are homologous.

In addition to alignments, the command above has generated two more files for each of the domains: mask file (e.g. **myseqs.archaea.mask**), and a structure diagram file showing the masked positions on the consensus secondary structure: **myseqs.archaea.mask.pdf** or **myseqs.archaea.mask.ps** [5].

The mask file is a single line of text of length 1508 characters, one per consensus position, containing only **0**s and **1**s. A **0** at position *n* indicates that consensus position *n* was excluded when the mask was applied, and a **1** at position *n* indicates that consensus position *n* was kept and included in **myseqs.archaea.mask.stk**. In this mask file there are 1449 **1s** and 59 **0s**.

The secondary structure diagram file highlights which positions on the consensus structure are excluded by the mask. The **myseqs.archaea.mask.pdf** is included in this guide as figure 5.

Using pre-calculated masks for consistent masking of multiple datasets

In the **ssu-mask** example above, we calculated a mask specifically based on our **myseqs** alignments. Given a different dataset of SSU sequences, the generated masks would likely be different (i.e. exclude different columns). This per-alignment-specificity can be undesirable. For example, if you'd like to directly compare alignments from multiple runs of **ssu-align** you probably want to make sure all of the alignments being compared contain an identical set of consensus positions [6]. SSU-ALIGN contains a default, pre-calculated mask for each of the three models. These masks were derived from large alignments as described in section 6. As we'll see in the next example, you can use these masks by supplying the **-d** option to **ssu-mask**. We'll also use the **--key-out** option to add the string "default" to the names of the output files so we do not overwrite the files we created in the previous example. This option enables you to save multiple sets of alignments and masks in a single directory:

```
> ssu-mask -d --key-out default myseqs

# Masking alignments using pre-existing masks...
##                                                        mask
#                                                    ------------
# file name                          in/out  type  #cols  incl.  excl.
# ---------------------------------  ------  ----  -----  -----  -----
  myseqs.archaea.stk                  input   aln   1511      -      -
  archaea-0p1.mask                    input  mask   1508   1376    132
  myseqs.archaea.default.mask.pdf    output   pdf   1508   1376    132
  myseqs.archaea.default.mask.stk    output   aln   1376      -      -
#
  myseqs.bacteria.stk                 input   aln   1597      -      -
  bacteria-0p1.mask                   input  mask   1582   1376    206
  myseqs.bacteria.default.mask.pdf   output   pdf   1582   1376    206
  myseqs.bacteria.default.mask.stk   output   aln   1376      -      -
#
  myseqs.eukarya.stk                  input   aln   2009      -      -
  eukarya-0p1.mask                    input  mask   1881   1343    538
  myseqs.eukarya.default.mask.pdf    output   pdf   1881   1343    538
  myseqs.eukarya.default.mask.stk    output   aln   1343      -      -
```

[5] If the program **ps2pdf** is in your PATH, a PDF will be created, otherwise a postscript **.ps** suffixed file will be created.

[6] This would also allow the alignments to be combined easily; see the **ssu-merge** manual page.

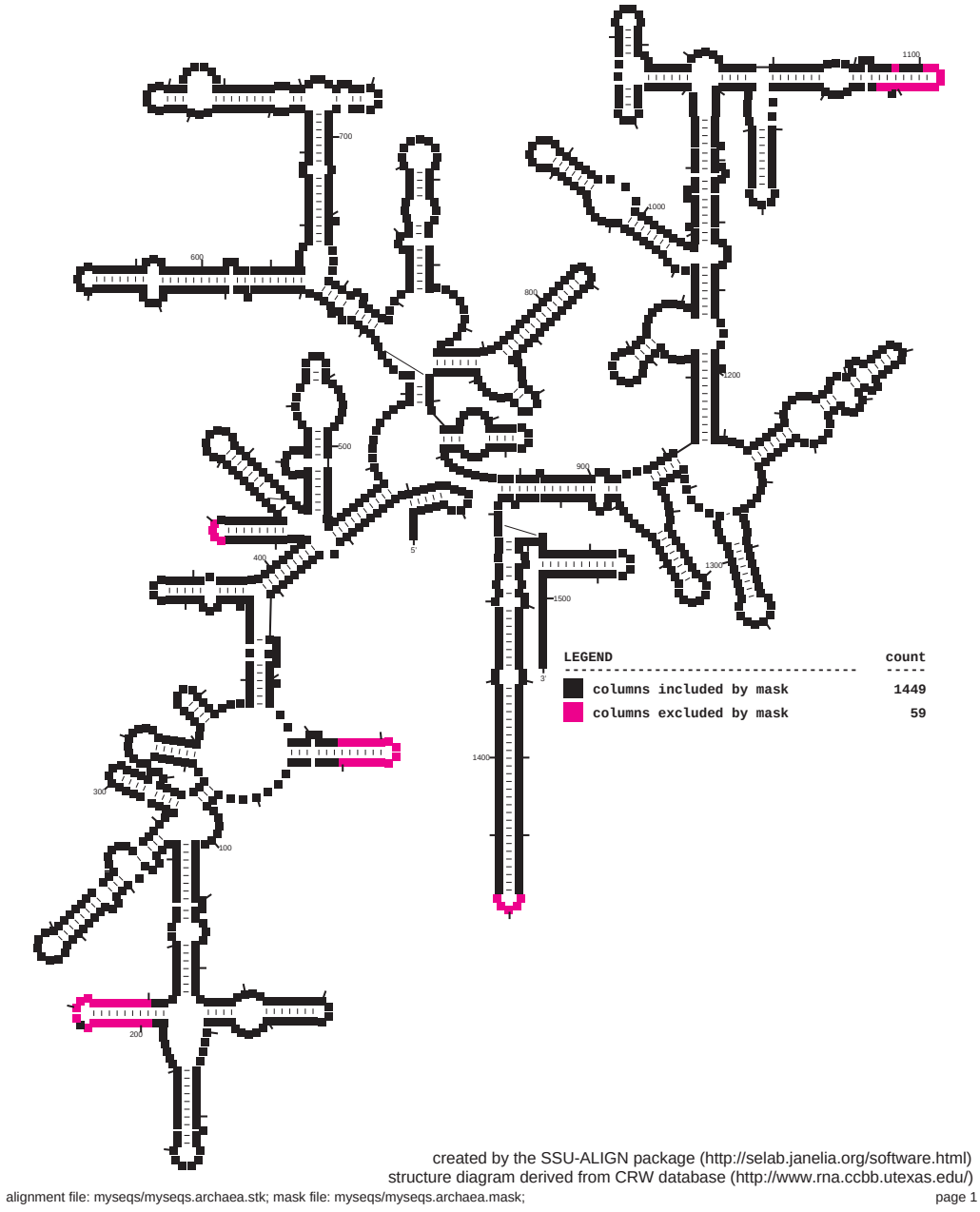

Figure 5: **Archaeal posterior probability-based mask for the tutorial example.** Pink positions are excluded by the mask. Black positions are included.

The output is similar to the previous example, but note that the **.mask** suffixed files were input rather than output (these files were placed in your $SSUALIGNDIR directory during installation), and the number of columns included and excluded has changed.

There are several other options that can be supplied to **ssu-mask**, including **-s** and **-k** which let you use your own pre-calculated masks. See the **ssu-mask** manual page at the end of this guide for more information.

Converting Stockholm alignments to FASTA format

Once the ambiguously aligned regions of the alignment are removed you may want to use the alignments as input to a phylogenetic inference program. Not many of those programs accept Stockholm formatted alignments as input. You can convert the Stockholm alignments to aligned FASTA using the **ssu-mask** program by specifying the **--stk2afa** option on the command line:

```
> ssu-mask --stk2afa myseqs
```

After running, three **.afa** suffixed files will have been created in the **myseqs** directory.

Visualizing alignments with ssu-draw

Because SSU rRNA is a long RNA (about 1500 nucleotides in archaea and bacteria, and about 1800 in eukarya) alignments of even a few sequences are difficult to view in a meaningful way. This is especially true if the alignments contain thousands of sequences. The **ssu-draw** program can be used to generate secondary structure diagrams for displaying alignment statistics, such as location and frequency of insertions and deletions, as well as individual aligned sequences. **ssu-draw** can only be used for alignments created using the three default models; it cannot be used for alignments to models created using **ssu-build**. To draw alignment statistic diagrams of our **myseqs** example dataset, do:

```
> ssu-draw myseqs
```

Two new files will be created for each of the alignments, as reported to the screen:

```
# Drawing secondary structure diagrams...
#
# alignment file name  structure diagram file  # pages  tabular data file
# -------------------  ----------------------  -------  ----------------------
  myseqs.archaea.stk   myseqs.archaea.pdf            9  myseqs.archaea.drawtab
  myseqs.bacteria.stk  myseqs.bacteria.pdf           9  myseqs.bacteria.drawtab
  myseqs.eukarya.stk   myseqs.eukarya.pdf            9  myseqs.eukarya.drawtab
```

The structure diagram files will be PDF format only if you have the program **ps2pdf** installed and in your PATH, otherwise they will be postscript files (see the **ssu-draw** manual page for more information).

Take a look at the nine page **myseqs.archaea.pdf** or **myseqs.archaea.ps** file [7]. Page 2 of this file is included in this guide as Figure 6.

Each page shows the same secondary structure template, it is 1508 consensus positions for archaea and includes 471 basepairs as indicated at the top of each page. Each of the nine pages displays a different alignment statistic on each of these 1508 consensus positions. Page 1 shows the alignment consensus sequence which is defined by the most common nucleotide present in the alignment **myseq.archaea.stk** at each consensus position. Page 2 shows the information content per consensus position in a heatmap-like color scheme; blue positions are highly conserved, red positions are higly variable. Page 3 shows the mutual information from basepaired nucleotides in a similar heatmap style: paired positions that are red exhibit many covariations across the sequences, blue positions exhibit few covariations. Page 4 and 5 show the frequency and average length of insertions, respectively, that occur after each consensus position. An inserted nucleotide is one that does not align to a consensus position. Importantly, these nucleotides are not aligned by **ssu-align** but rather simply inserted between the two appropriate consensus columns (see refsec:background-columns for more detail). Pages 6 shows the frequency of all deletions (gaps) in the

[7] For Mac OS/X users: the standard Mac Preview application should be able to view either type of file.

alignment. Page 7 shows the frequecy of *internal* deletions, these are deletions that occur after the first nucleotide and before the final nucleotide in their corresponding sequence. Page 8 shows the average posterior probability (confidence estimate) per consensus position. Finally, page 9 shows the fraction of sequences that span each position, i.e. that have a nongap nucleotide occuring at or before the position and a nongap nucleotide occuring at or after the position.

You may have noticed that while the majority of positions are drawn as square cells, some of the positions are open circles. These are the positions that are excluded by the mask we created earlier using the command **`ssu-mask myseqs`**. As it has done here, **`ssu-draw`** will automatically use masks to show excluded positions if **`ssu-mask`** has already been executed for the directory. To use the default masks discussed in the masking section of this tutorial, use the **`-d`** option (**`ssu-mask -d myseqs`**), but note that this will overwrite the files we've just created above unless you also use the **`--key-out`** option (e.g. **`ssu-mask --key-out default -d myseqs`**) as shown previously in the masking section.

By default, alignment summary diagrams are drawn as solid color cells per position, but the consensus sequence can be overlaid on the colored cells using the **`--cnt`** option to **`ssu-draw`**.

The **`.drawtab`** suffixed files include tabular delimited data corresponding to the statistics shown in the structure diagrams. In these files, lines that start with **`#`** are comment lines which are not tabular, but rather explain each tab-delimited field. The comments also explain how the statistics, such as information content and mutual information, are calculated.

Drawing individual sequences

In addition to drawing alignment summary diagrams, **`ssu-draw`** can be used to create secondary structure diagrams of individual aligned SSU sequences using the **`--indi`** option:

```
> ssu-draw --indi myseqs
```

As before, this command will create the nine-page alignment summary diagram files, but it will also create **`indi.pdf`** suffixed files which display each sequence in the alignment.

Take a look at **`myseqs.archaea.indi.pdf`** or **`myseqs.archaea.indi.ps`**. This file has 10 pages, two per sequence. Pages 7 and 8, which display a Thermococcus sequence, are included as Figures 7 and 8.

The first page for each sequence is the aligned sequence with nucleotides colored by basepair type, and background cells colored by number of inserts after each nucleotide (blank (white) backgrounds predominate, indicating 0 inserted nts after those positions). Additionally, nucleotides that differ from the most frequent nucleotide at each position are outlined, with thicker outlines indicating the most frequent nucleotide at the position occurs in more than 75% of the aligned sequences. The legend section explains the coloring and other formatting of each cell.

The second page for each sequence occurs immediately after the first, and displays the posterior probability of each position using a heatmap scheme: blue positions have high posterior probability meaning the program is confident these positions are correctly aligned, while red indicates low confidence and high ambiguity.

Be warned that these files can grow very large for large alignments, especially if they are being created as postscript files [8]. In general, the files will be about 1 Mb per 20 sequences for PDFs and 1 Mb per 2 sequences for postscripts. By default, **`ssu-draw`** will exit with an error if the output postscript file you're requesting would exceed 100 Mb in size, to override this warning and create the file anyway, use the **`-f`** option.

You might not want to draw all the sequences in the alignment, especially since the output image files are large. To select a subset of sequences to draw, use **`ssu-mask`** with the **`--seq-k <listfile>`** option to create a new alignment with only sequences listed in the file **`<listfile>`** [9]. In a list file, each sequence name appears on a separate line. An example list of 2 bacterial sequences from the **`myseqs`** dataset is in

[8] When **`ssu-draw`** creates PDF files it first creates postscript files, then converts them to PDF with **`ps2pdf`**, and finally removes the postscript files.

[9] It might be helpful to use **`ssu-mask --list myseqs`** to create lists of all the sequences in each alignment, and then remove those you don't want.

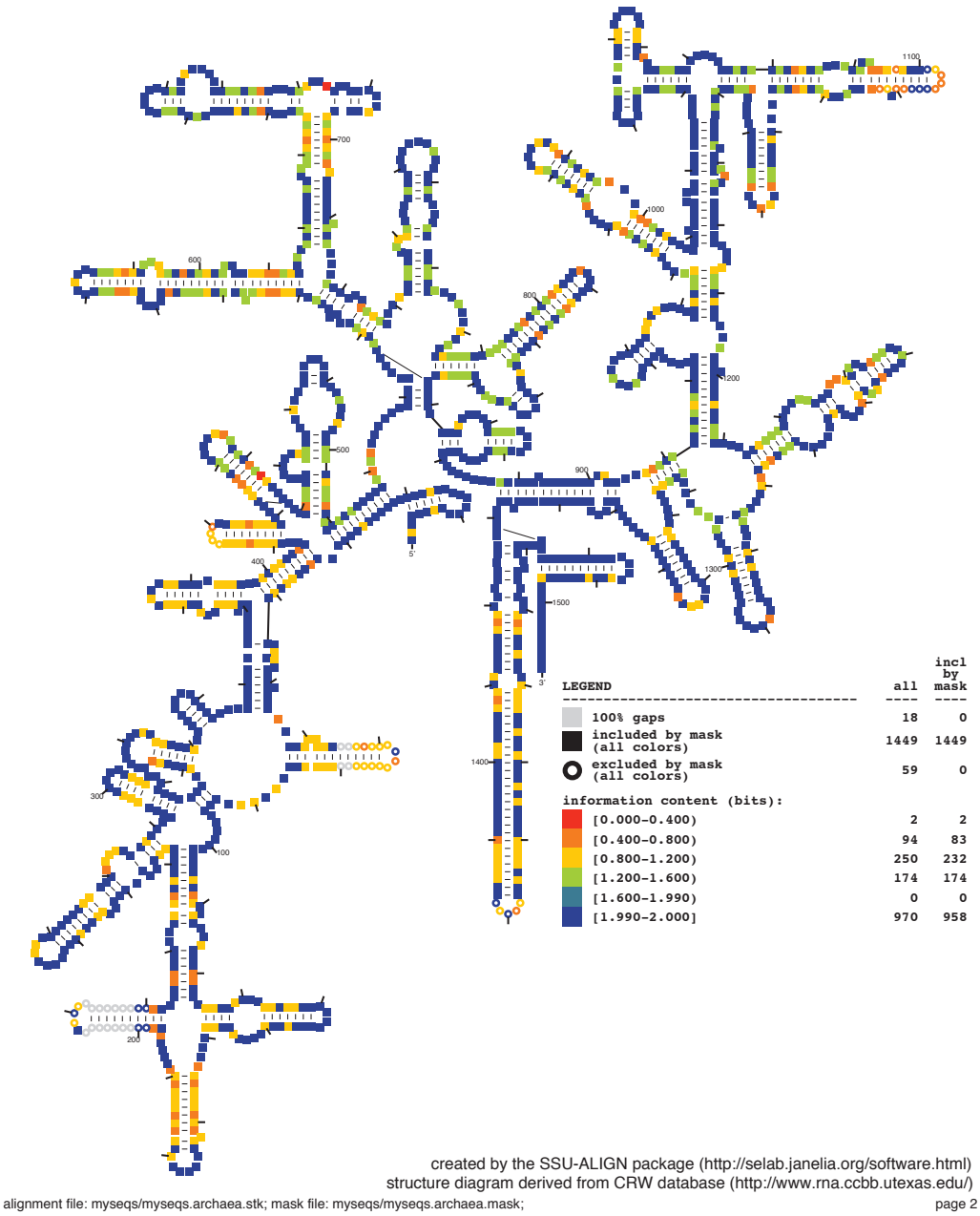

Figure 6: **Information content per consensus position of the archaeal alignment created in the tutorial.** Coloring reflects level of conservation: highly conserved positions are blue, highly variable positions are red, as indicated in the legend. Solid square positions are included by the alignment mask created by **`ssu-mask`**; open circle positions are excluded. See text for more details.

tutorial/mytwobac.list. To draw just these two sequences, you would execute the following two commands:

```
> ssu-mask -a --seq-k mytwobac.list myseqs/myseqs.bacteria.stk
> ssu-draw -a --indi myseqs.bacteria.seqk.stk
```

Notice that the **-a** option was also used and instead of using **myseqs** as the command-line argument we specified the bacterial alignment. The **-a** option tells **ssu-draw** that the command-line argument is an alignment file, not a **ssu-align**-created directory.

There are several other options that can be supplied to **ssu-draw**. See the **ssu-draw** manual page at the end of this guide for more information.

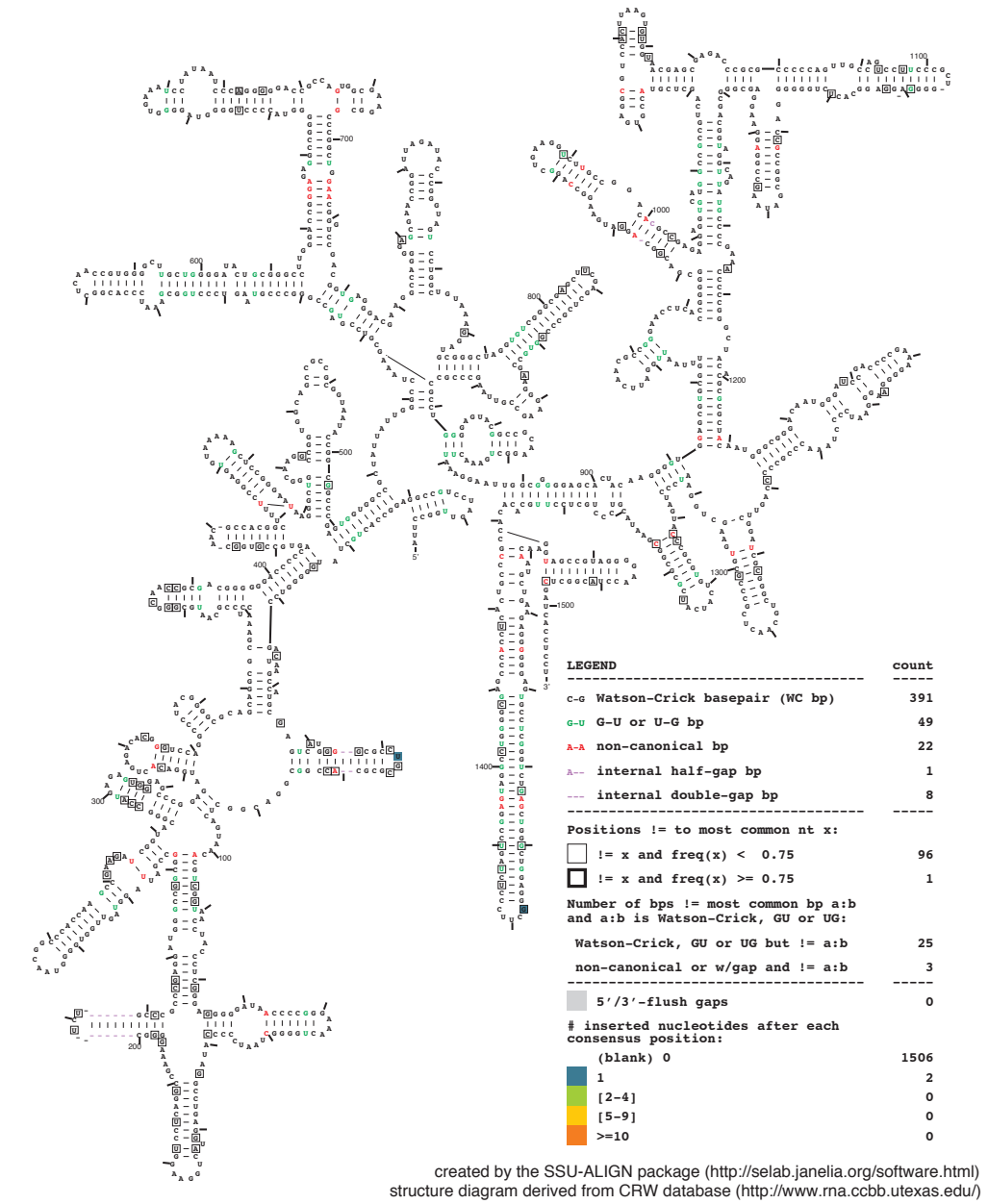

Figure 7: **Sequence and predicted secondary structure of a *Thermococcus celer* SSU sequence as it aligns to the default archaeal model.** Colors and other annotations of the nucleotides are listed in the legend and discussed in the text.

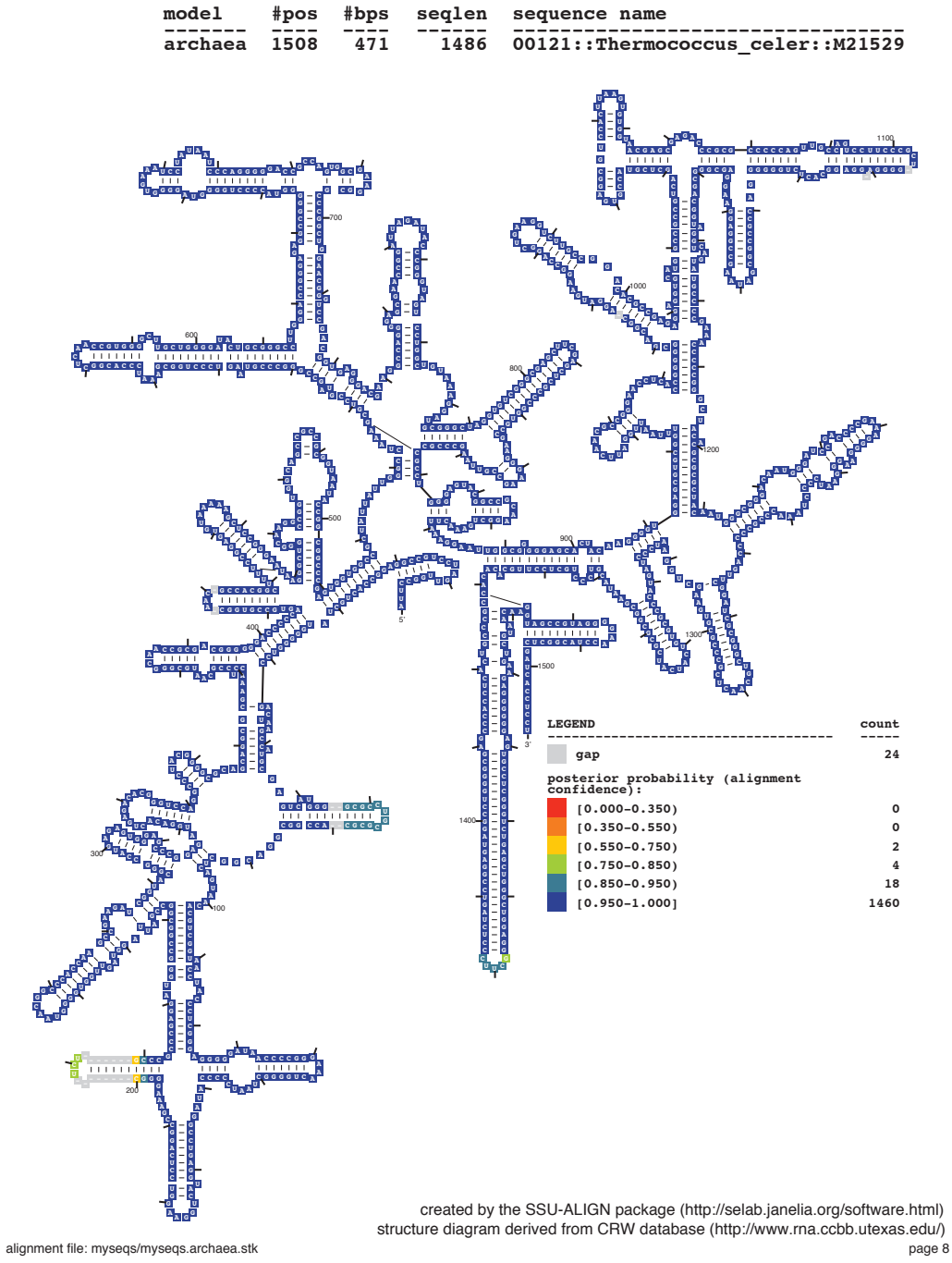

Figure 8: **Sequence and posterior probabilities for a *Thermococcus celer* SSU sequence as it aligns to the default archaeal model.** Colors and other annotations of the nucleotides are listed in the legend and discussed in the text.

Faster alignment of large datasets through parallelization with ssu-prep

ssu-align runs on a single processor and does not support MPI or multi-threading. However, if you have access to a compute cluster or multi-core computers, simplistic parallelization is possible with the **ssu-prep** program.

ssu-prep will split up a large input sequence file into *n* smaller files and create a shell script that will execute *n* **ssu-align** jobs in parallel, each processing one of the small sequence files. The results of all jobs will automatically be merged together by the final job, ultimately yielding the same results as if a single **ssu-align** job was run for the original large input sequence file. Parallelizing **ssu-align** in this way can drastically reduce the actual time required for aligning large datasets. A job that would have required 100 hours on 1 processor can be done in a little more than 1 hour on 100 processors. See table 5 in section 8 for example timing statistics.

In this section we'll walk through an example of how to do this for a small dataset. The sequence file **tutorial/seed-30.fa** includes 30 randomly chosen sequences from the complete set of seed sequences from the three default models of SSU-ALIGN.

ssu-prep has three different usage modes as explained by the program if it is run without any command-line arguments:

```
> ssu-prep

Incorrect number of command line arguments.
Usage: ssu-prep    [-options] <seqfile> <output dir> <num jobs> <prefix/suffix file>
Usage: ssu-prep -x [-options] <seqfile> <output dir> <num jobs>
Usage: ssu-prep -y [-options] <seqfile> <output dir> <num jobs>

ssu-prep splits up <seqfile> into <num jobs> smaller files and creates a shell
script that will execute <num jobs> ssu-align jobs in parallel, each processing
one of the small sequence files. The results of all jobs will automatically be
merged together by the final job, giving the same results as if a single
ssu-align job was run for the complete <seqfile>.

The 3 different usages control how the prefix and suffix are defined for the jobs
in the output shell script, allowing, for example, the user to wrap the ssu-align
commands in a cluster submission command (such as Sun Grid Engine's "qsub"):

Default: (neither -x nor -y enabled) prefix and suffix for ssu-align jobs in
         output shell script are defined in <prefix/suffix file>. First line is
         the prefix, second line is the suffix.
With -x: do not specify <prefix/suffix file>; output shell script will run all
         <num jobs> jobs in parallel on one machine with <num jobs> cores/cpus.
With -y: do not specify <prefix/suffix file>; user will manually add the desired
         prefix/suffix to ssu-align commands after ssu-prep finishes.

To see more help on available options, do ssu-prep -h
```

By default, if neither **-x** nor **-y** options are used, the program reads a **<prefix/suffix file>**, a simple two line file that a prefix string and a suffix string to prepend and append respectively to the **ssu-align** commands it generates.

The strings in the **<prefix/suffix file>** will likely be specific to your parallel computing environment. At Janelia, we use Sun Grid Engine's **qsub** program for submitting jobs to a large compute cluster. The relevant prefix/suffix file for our specific computing environment is included in **tutorial/janelia-cluster-presuf.txt**. Take a look at this file:

```
qsub -N ssu-align -o /dev/null -b y -j y -cwd -V "
"
```

The first line is the prefix string, containing the name and appropriate command line arguments of the **qsub** program which submits jobs to the Grid Engine queuing system at Janelia. The second line is the suffix, and contains only a double quote, which complements the double quote at the end of the prefix line as we'll see below.

Now, let's run **ssu-prep** as if we were going to create parallel **ssu-align** jobs for the Janelia cluster. Move into the **tutorial/** directory and execute the command:

```
> ssu-prep seed-30.fa my30 5 janelia-cluster-presuf.txt
```

About forty lines of text are output to the screen. We'll step through and discuss this output:

```
# Validating input sequence file ... done.
#
# Preparing 5 ssu-align jobs ...
# Partitioning seqs with goal of equalizing total number of nucleotides per job ...
#
# output file name   description
# -----------------  --------------------------------------------------
  my30/seed-30.fa.1  partition 1 FASTA sequence file (6 seqs; 10463 nt)
  my30/seed-30.fa.2  partition 2 FASTA sequence file (6 seqs; 10240 nt)
  my30/seed-30.fa.3  partition 3 FASTA sequence file (6 seqs;  9922 nt)
  my30/seed-30.fa.4  partition 4 FASTA sequence file (6 seqs; 10358 nt)
  my30/seed-30.fa.5  partition 5 FASTA sequence file (6 seqs;  9397 nt)
  my30.ssu-align.sh  shell script that will execute 5 ssu-align jobs
#
```

First, **ssu-prep** reports that it has validated the formatting of the sequence file, partitioned it into 5 new files and placed each file into the newly created **my30/** subdirectory. Each of these files has 6 of the original 30 sequences in it. Additionally, **my30.ssu-align.sh**, a shell script that will execute 5 jobs, one per sequence file, was created in the current working directory. After this, you'll see:

```
#################################################################################
# To execute all 5 ssu-align jobs, run the shell script with the command:
#        "my30.ssu-align.sh"
# (it is an executable file)
#################################################################################
```

These are instructions for how to execute the shell script. Take a look at the shell script **my30.ssu-align.sh**:

```
#!/bin/bash
# Bash shell script created by ssu-prep for running 5 ssu-align jobs.
# Each job will process a separate partition of the sequence file:
# 'seed-30.fa'.
#
# The final job is special, after computing its alignments it will wait for all
# other jobs to finish and then merge the output of all jobs together.
# The merged output files will be in the directory: '/my30/'
#
# The for loop below will execute/submit the first 4 of 5 jobs.
# The final ssu-align job is executed separately because it does the merging.
#
for (( i=1; i<=4; i++ ))
do
        echo "# Executing: qsub -N ssu-align -o /dev/null -b y -j y -cwd -V " ssu-align my30/seed-30.fa.$i my30/my30.$i ""
        qsub -N ssu-align -o /dev/null -b y -j y -cwd -V " ssu-align my30/seed-30.fa.$i my30/my30.$i "
done
echo "# Executing: qsub -N ssu-align -o /dev/null -b y -j y -cwd -V " ssu-align --merge 5 my30/seed-30.fa.5 my30/my30.5 ""
qsub -N ssu-align -o /dev/null -b y -j y -cwd -V " ssu-align --merge 5 my30/seed-30.fa.5 my30/my30.5 "
```

This is a bash shell script file[10]. The **#**-prefixed lines are explanatory comments. The remainder of the file consists of a for loop that will submit the first four **ssu-align** jobs to the cluster using **qsub**. The lines beginning with **qsub** are the actual job submission commands. The lines beginning with **echo** cause updates to be printed to STDOUT before each job is submitted. Note that the **qsub** command line is composed of the prefix string from the **janelia-cluster-presuf.txt** file, followed by a **ssu-align** command, followed by the suffix string. The end result is that the **ssu-align** command is contained within the quotes from the prefix/suffix strings.

The comments explain that the final job is special. It will merge the results of all jobs once they are finished. Consequently, it requires special command-line options and so is executed outside the for loop, as the final line of the script.

Because your specific compute system is likely different from Janelia's, the **my30.ssu-align.sh** script will probably not work. To make **ssu-prep** generate shell scripts you can run on your system, create a file like **janelia-cluster-presuf.txt** but with prefix and suffix strings specific to your system.

[10]The **--no-bash** option can be used to make a non-bash-specific script, see the **ssu-prep** manual page for more information.

This simple prefix/suffix string method may not work for your compute system. For example, if your cluster requires using **ssh** to remote login to different nodes, a single fixed prefix line will probably not be sufficient. If this is the case, run **ssu-prep** with the **-y** option. In this mode no prefix/suffix file will be read, and the output shell script will contain only the raw **ssu-align** commands. For example, do:

```
> ssu-prep -f -y seed-30.fa my30 5
```

(We had to also specify **-f** so the program would overwrite our existing **my30** directory.) Much of the output is the same as the prior example, but the **WARNING** section is new:

```
################################################################################
# WARNING: -y was set on the command line.
# This means that 'my30.ssu-align.sh' will simply run the 5 ssu-align jobs in
# succession, one after another, not in parallel. If you want to run the jobs in
# parallel you'll either have to manually edit that file or rerun ssu-prep using
# the options listed above to specify prefix and/or suffix strings for the
# ssu-align commands, so they are, for example, submitted to run on a cluster
# using a queing system manager like SGE. Or you can run <n> jobs in parallel on
# a single <n>-core machine by rerunning ssu-prep using '-x'.
# Do 'ssu-prep -h' or see the User Guide for more information.
################################################################################
```

As explained, you'll have to modify the newly created **my30.ssu-align.sh** to make the jobs run in parallel in this case, either manually or using your own script. The file will still contain a for loop submitting all but the final job. If you'd rather all jobs were placed on their own line, use the **--no-bash** option to **ssu-prep**.

For now, let's run the unmodified **my30.ssu-align.sh** script to demonstrate how the jobs are merged together automatically. As noted in the warning above, this will simply run the 5 jobs in succession instead of in parallel, but for our purposes here this is okay. To run the script, execute the command:

```
> my30.ssu-align.sh
```
[11]

The script executes **ssu-align** five times in succession, and these jobs will being reporting what they're doing to the screen. Once all five jobs are run, the **ssu-merge** program will automatically be called to merge their output. You'll see the following output at that point:

```
# Merging files from 5 ssu-align runs...
#
#                                    # files     # seqs
# merged file name       CM name      merged     merged
# ---------------------  --------  -------  ---------
  my30.tab               -               5          -
  my30.scores            -               5          -
  my30.ssu-align.sum     -               5          -
  my30.ssu-align.log     -               5          -
#
  my30.archaea.fa        archaea         2          5
  my30.archaea.hitlist   archaea         2          5
  my30.archaea.cmalign   archaea         2          5
  my30.archaea.ifile     archaea         2          5
  my30.archaea.stk       archaea         2          5
#
  my30.bacteria.fa       bacteria        5          8
  my30.bacteria.hitlist  bacteria        5          8
  my30.bacteria.cmalign  bacteria        5          8
  my30.bacteria.ifile    bacteria        5          8
  my30.bacteria.stk      bacteria        5          8
#
  my30.eukarya.fa        eukarya         5         17
  my30.eukarya.hitlist   eukarya         5         17
  my30.eukarya.cmalign   eukarya         5         17
  my30.eukarya.ifile     eukarya         5         17
  my30.eukarya.stk       eukarya         5         17
```

Two of the five small partition files included archaeal sequences, and all five included bacterial and eukaryotic sequences. The total number of archaeal, bacterial and eukaryotic sequences was 5, 8 and 17,

[11] If this command fails, you're probably not using the BASH shell or the **ssu-prep** was unable to make the file executable. In the former case, rerun the **ssu-prep** command above with the **--no-bash** option and try again. In the latter, execute **bash my30.ssu-align.sh**

respectively. The final merged alignments are in the files **my30.archaea.stk**, **my30.bacteria.stk**, and **my30.eukarya.stk**.

Using ssu-prep to parallelize ssu-align on multi-core machines

The third and final **ssu-prep** usage mode is for parallelizing jobs to run on a single multi-core machine. This mode can be thought of as a simple substitute for multi-threading, and is enabled with the **-x** option to **ssu-prep**. As with **-y**, using **-x** obviates the need for a prefix/suffix file. As an example, imagine you are using a quad-core machine. In this case, execute:

```
> ssu-prep -f -x seed-30.fa my30 4
```

(We had to also specify **-f** so the program would overwrite our existing **my30** directory.) Much of the output is the same as before. However, the **my30.ssu-align.sh** file will be different; it is included below:

```
#!/bin/bash
# Bash shell script created by ssu-prep for running 4 ssu-align jobs.
# Each job will process a separate partition of the sequence file:
# 'seed-30.fa'.
#
# This script will execute all 4 jobs at once, in parallel. It is only
# meant to be executed on a system with 4 cpus/cores. The first 3 jobs
# will run in the background and output to /dev/null. The final job will
# output to STDOUT, allowing you to follow its progress.
#
# The final job is special, after computing its alignments it will wait for all
# other jobs to finish and then merge the output of all jobs together.
# The merged output files will be in the directory: '/my30/'
#
# The for loop below will execute/submit the first 3 of 4 jobs.
# The final ssu-align job is executed separately because it does the merging.
#
for (( i=1; i<=3; i++ ))
do
        echo "# Executing: ssu-align my30/seed-30.fa.$i my30/my30.$i > /dev/null &"
        ssu-align my30/seed-30.fa.$i my30/my30.$i > /dev/null &
done
echo "# Executing: ssu-align --merge 4 my30/seed-30.fa.4 my30/my30.4"
ssu-align --merge 4 my30/seed-30.fa.4 my30/my30.4
```

Note that the **ssu-align** commands in the for loop include **&** at the end, which cause them to be run simultaneously.

Using ssu-build to create a truncated model of a specific region of SSU rRNA

Many SSU sequencing studies target a specific region of the SSU rRNA molecule using PCR primers at the boundaries of that region. For such studies, you may want to build a new CM that only models the region of the molecule targeted by the study. There are two reasons for doing this. The first is speed: the running time of **ssu-align** decreases as the model size it is using decreases (see table 5 on page 77 and the associated text in section 8). The second reason is that it should slightly increase alignment accuracy because the uncertainty regarding what region of the full molecule the subsequence should align to is eliminated [12]. In this section I'll demonstrate how to create a CM of a specific region of SSU and use it to create alignments.

In this example, we'll build three new CMs targetted to the so-called V4 hypervariable region, one each for archaea, bacteria and eukarya. We'll do this using the SSU-ALIGN default seed alignments that were used to create the default full-length CMs.

The first step is to identify the region of each of the three default models we want our new models to represent. The secondary structure diagrams in the **models/** subdirectory of the **ssu-align-0.1.1/** toplevel directory where you unpacked the package. In particular, the files suffixed with **.rf.pdf** and **.info.pdf** are probably most useful. These figures are included in this guide in section 5 as figures 12 and 13 (archaea), 18 and 19 (bacteria), 24 and 25 (eukarya). In these figures, every 100th position is numbered, and every 10th position is indicated by a tick mark.

Based on the bacterial primer sites used by a recent environmental study that targetted the V4 region (Claesson et al., 2009), I determined the V4 region in the default bacterial model spans positions 584 through 824 via manual inspection of the bacterial secondary structure diagrams. By comparison to the archaeal and eukaryal diagrams, the corresponding region in archaea is from positions 525 to 765, and in eukarya is 620 to 1082.

Given these coordinates, we can use **ssu-build** to create V4 specific models. First, create and move into a new directory where we'll build the new models. For example, from the **tutorial/** directory, do:

```
> mkdir myV4; cd myV4
```

The models must be built one at a time. First, we'll create the archaeal V4 model:

```
> ssu-build -d --trunc 525-765 -n arc-V4 -o V4.cm archaea
```

The **-d** option tells **ssu-build** to use a default model, **--trunc 525-765** specifies the positions of the model, **-n arc-V4** specifies the name of the model should be **arc-V4** and **-o V4.cm** specifies the name of the CM file be **V4.cm** (we'll add two more models to this file next). Finally, **archaea** tells the program to use the default archaeal model. As with the other programs, **ssu-build** reports on what it's doing:

```
# Truncating alignment and default mask; saving only columns between predefined consensus
# columns 525 and 765...
#
# output aln file
# --------------------------
  archaea-0p1-sb.525-765.stk
#
# truncated mask file name
# ---------------------------
  archaea-0p1-sb.525-765.mask
#
# Building CM(s)...
#
#                            num  alignment  consensus   num
# CM file name  CM name    seqs     length     length   bps
# ------------  -------  ------  ---------  ---------  ----
  V4.cm         arc-V4       23        242        241    69
#
# structure diagram file
# -------------------------
  archaea-0p1-sb.525-765.pdf
#
```

Several files have been created as listed below:

[12] Though recommended, building truncated models for PCR studies is not required. **ssu-align** is designed to be able to handle truncated sequences deriving from any region of the SSU molecule

archaea-0p1-sb.525-765.stk the region of the default archaeal seed alignment spanning consensus columns 525 to 765. This alignment was used to build the new archaeal V4 model.

archaea-0p1-sb.525-765.mask the region of the default archaeal mask (see section 6) spanning consensus columns 525 to 765. This mask file could potentially be useful to mask alignments created using the new archaeal V4 model with the **ssu-mask** option and the **-s** option.

V4.cm the CM file with the new archaeal V4 model named **arc-V4**. As indicated by the output, the model was built from an alignment of width 242 positions that included 69 basepairs of 23 archaeal sequences. This alignment was saved as **archaea-0p1-sb.525-765.stk**.

archaea-0p1-sb.525-765.ps a structure diagram of the archaeal consensus model highlighting the region from positions 525 to 765 represented by the new model in **V4.cm**. This diagram is included as figure 9.

Next we'll build bacterial V4 and eukaryal V4 models and append them to the **V4.cm** file:

```
> ssu-build -d --trunc 584-824 -n bac-V4 --append V4.cm bacteria
> ssu-build -d --trunc 620-1082 -n euk-V4 --append V4.cm eukarya
```

The **--append** option tells **ssu-build** to append the new model onto the existing file **V4.cm**. After executing these two commands, the **V4.cm** file will contain three V4 models. This CM file can be used to create domain-specific SSU V4 alignments from a set of SSU sequences using **ssu-align**. An example dataset this would be particularly useful for would be a set of SSU sequences generated from an environmental sequencing survey that used three sets of PCR primers: one each for the archaeal, bacterial and eukaryal V4 regions. Alternatively, it could be used to create V4-only alignments from a set of full length SSU sequences. We'll demonstrate the latter application here using our sample dataset from earlier in the tutorial in the file: **tutorial/seed-15.fa**, which contains 15 full or nearly-full length archaeal, bacterial and eukaryotic SSU sequences:

```
> ssu-align -m V4.cm ../seed-15.fa myV4seqs
```
[13]

The **-m** option specifies that the model **V4.cm** be used. Using this model, processing **seed-15.fa** takes only about 4 seconds, as opposed to the 30 seconds it took with the default, full-length models earlier in the tutorial. Take a look at **myV4seqs/myV4seqs.bacteria.stk**, it contains only the V4 regions of the 5 bacterial sequences from **seed-15.fa**.

As with the earlier example, these alignments can be masked with the **ssu-mask** program, though I won't step through that here. However, the **ssu-draw** program cannot be used to draw diagrams of alignments created using any models other than the default three, so it is not possible to create drawings of these alignments.

[13] This assumes your current working directory is in a subdirectory of **tutorial/**. If not, specify the correct path to the **seed-15.fa** file in the **tutorial/** subdirectory of **ssu-align-0.1.1/**.

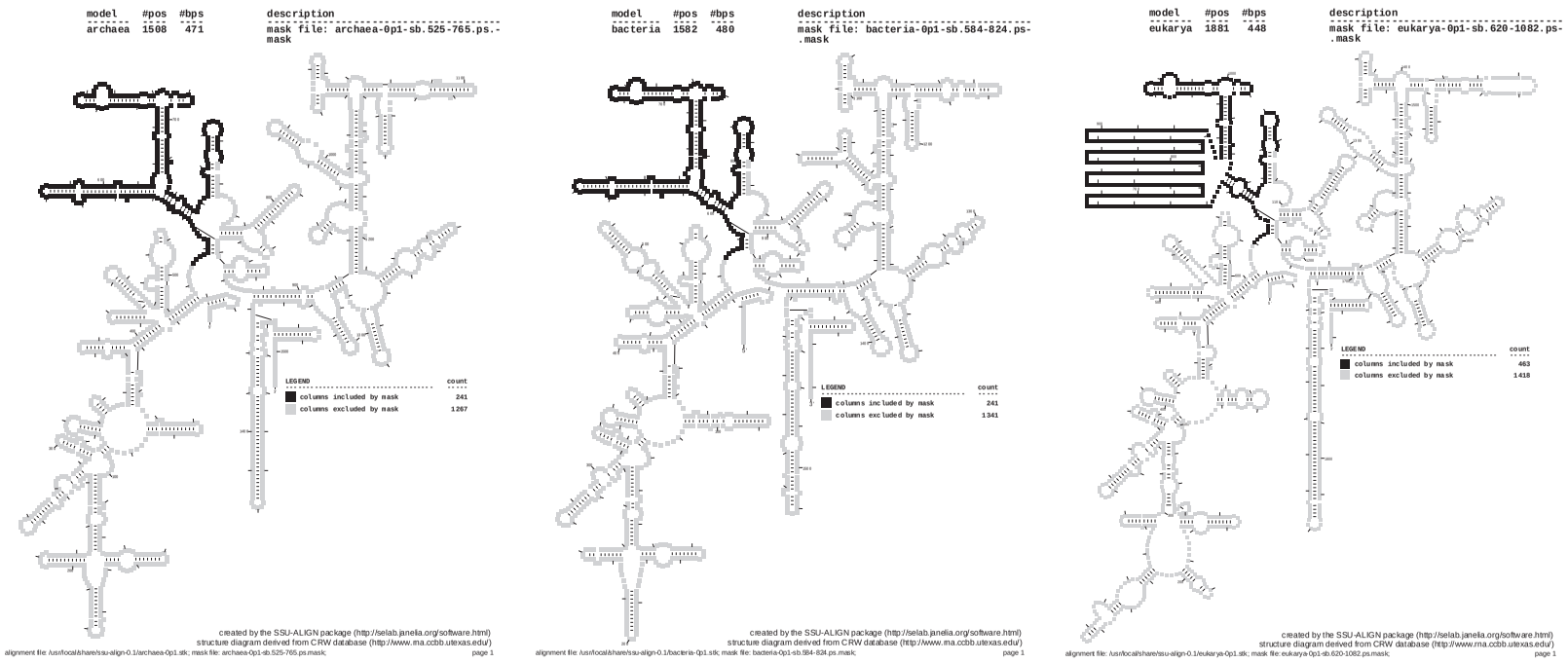

Figure 9: **V4 regions in archaea (left), bacterial (middle) and eukarya (right).** These diagrams are generated by `ssu-build` in the tutorial.

Using ssu-build to create a Firmicutes-specific SSU model

The default archaeal, bacterial and eukaryotic models provided with SSU-ALIGN are meant to be general models that represent the SSU sequence diversity in their respective domains. You might want to create more specific models that cover a tighter or broader phylogenetic range, for example a particular bacterial phyla of interest, or a single model that covers all SSU sequences. The **ssu-build** program allows users to create such models.

ssu-build takes as input a multiple sequence alignment, called a *seed* alignment, of SSU sequences and creates a CM file that includes a model that represents the diversity from the seed alignment. This CM file can be used individually to align sequences, or combined, through simple file concatenation, with other CM files and used to classify and create clade-specific SSU alignments.

Typically, a seed alignment is carefully constructed to minimize errors and ensure appropriate representation from the phylogenetic clade being modeled. Errors in the seed alignment will lead to errors in the output alignments from the CM built from that seed. Aligning sequences to models that are outside the range of diversity in the seed will also lead to errors. However, in this section, for demonstration purposes, we'll build quick and dirty models from two partitions of the default bacterial seed alignment included with SSU-ALIGN.

We'll create two bacterial SSU models, a *Firmicutes* specific model, and a model that represents all other non-*Firmicutes* bacteria. We'll then combine these two models with the default archaeal and eukaryotic models and use them to classify and align sequences from the **seed-15.fa** sample dataset used previously in this tutorial.

Partitioning the default bacterial seed alignment

The first step is to identify which of the sequences in the default bacteria seed are *Firmicutes*. To do this, I ran the FASTA file with the unaligned bacterial seed sequences in **models/bacteria-0p1.fa** through the RDP-CLASSIFIER tool using default parameters (Wang et al., 2007) on the RDP website: `http://rdp.cme.msu.edu/classifier`[14]. Seven of the 93 sequences are classified as *Firmicutes* with an 80% bootstrap threshold. These seven sequences are:

```
01088::Bacillus_halodurans.::AB013373
01106::Bacillus_subtilis::K00637
01382::Clostridium_perfringens::M69264
01375::Clostridium_tetani::X74770|g509282
01295::Lactococcus_lactis_subsp._lactis.::AE006456
01252::Staphylococcus_aureus::L36472|g567883__bases_8528_to_10082_
01305::Streptococcus_pyogenes::AE006473
```

We'll use these sequences to build a *Firmicutes* specific model. We'll use six of the seven sequences, omitting the *Bacillus subtilis* sequence for reasons that will become clear shortly. The remaining 86 seed sequences are non-*Firmicutes* and we'll build a separate model from these.

To build the models, we first need to extract the relevant aligned sequences from the bacterial seed alignments using **ssu-mask**. We'll use the list files **firm-7.list** and **firm-6.list** in **ssu-align-0.1.1/tutorial/** to do this. The **firm-7.list** file includes the names of the seven *Firmicutes* sequences, one per line, and **firm-6.list** includes the same names but without the *Bacillus subtilis* sequence.

From the **tutorial/** directory, we can create the two alignments that we want with two commands:

```
> ssu-mask -a --seq-k firm-6.list --key-out firm-6 ../models/bacteria-0p1.stk
> ssu-mask -a --seq-r firm-7.list --key-out nonfirm-86 ../models/bacteria-0p1.stk
```

The first command creates the six sequence *Firmicutes*-only alignment in the file **bacteria-0p1.firm-6.seqk.stk** and the second command creates the non-*Firmicutes* alignment in the file **bacteria-0p1-nonfirm-86.seqr.stk**. In the first command, the **--seq-k** option tells **ssu-mask** to keep only the six sequences listed in **firm-6.list** In the second command, the **--seq-r** option tells **ssu-mask** to remove only the seven sequences listed in **firm-7.list** and keep all other sequences. These

[14] Version 2.2 of RDP-CLASSIFIER was used with RDP training set 6.

commands only remove selected sequences from the default seed in **../models/bacterial-0p1.stk**, none of the positions of the aligned nucleotides in the remaining sequences have been changed.

Now, we can use **ssu-build** to build models from the alignments we've just created with two commands:

```
> ssu-build -n firmicutes -o firmicutes.cm bacteria-0p1.firm-6.seqk.stk
> ssu-build -n bac-nonfirm -o bac-nonfirm.cm bacteria-0p1.nonfirm-86.seqr.stk
```

The first command creates the file **firmicutes.cm** which includes a CM named *firmicutes*, and the second command creates the file **bac-nonfirm.cm** which includes a CM named *bac-nonfirm*.

We can use both of these models within **ssu-align** simultaneously to differentiate *Firmicutes* SSU sequences from other bacterial SSU sequences by concatenating them together:

```
> cat firmicutes.cm bac-nonfirm.cm > my.cm
```

Since we'd also like to be able to differentiate archaeal and eukaryotic sequences from bacterial sequences, we can also append those default models:

```
> cat ../models/archaea-0p1.cm ../models/eukarya-0p1.cm >> my.cm
```

Now the file **my.cm** includes four CMs. Let's run **ssu-align** using this model on the **seed-15.fa** dataset we've been using throughout this tutorial:

```
> ssu-align -m my.cm seed-15.fa myseqs2

# Stage 1: Determining SSU start/end positions and best-matching models...
#
# output file name             description
# ---------------------------  ---------------------------------------------
  myseqs2.tab                  locations/scores of hits defined by HMM(s)
  myseqs2.firmicutes.hitlist   list of sequences to align with firmicutes CM
  myseqs2.firmicutes.fa              1 sequence  to align with firmicutes CM
  myseqs2.bac-nonfirm.hitlist  list of sequences to align with bac-nonfirm CM
  myseqs2.bac-nonfirm.fa             4 sequences to align with bac-nonfirm CM
  myseqs2.archaea.hitlist      list of sequences to align with archaea CM
  myseqs2.archaea.fa                 5 sequences to align with archaea CM
  myseqs2.eukarya.hitlist      list of sequences to align with eukarya CM
  myseqs2.eukarya.fa                 5 sequences to align with eukarya CM
```

Earlier in this tutorial, we used the default models to classify and align these sequences and found that 5 sequences were assigned to each of the three domains. In this search, 1 of the 5 bacterial sequences has been classified as a *Firmicutes* sequence. The **myseqs2.firmicutes.hitlist** file lists the name of the sequence:

```
# target name                        start    stop     score
# ---------------------------------  ------  ------  --------
  01106::Bacillus_subtilis::K00637       1    1552   2304.65
```

This is the *Bacillus subtilis* sequence from the bacterial seed that we purposefully omitted from the *Firmicutes* seed. **ssu-align** has correctly identified it as a *Firmicutes* sequence. We omitted the sequence to make this task more challenging. If we had included this sequence in the seed, its correct identification would have been easier for the program.

In this section we've crudely split the bacterial seed alignment into two smaller seed alignments and built a separate model from each, without modifying either seed. A better, but more labor intensive, strategy would be to manually examine and modify the seeds before constructing a CM from them. Given the limited context of the sequences in the seed, it may be clear that certain alignment regions should be refined and that basepairs in the consensus secondary structure should be added or removed. Also, more sequences could be added to make the model more robust and more fully represent the *Firmicutes* phyla. While six sequences is probably too few, many more than 100 sequences in a seed is typically not necessary.

5 SSU-ALIGN's SSU rRNA sequence and structure models[15]

CMs model both the conserved sequence and secondary structure of an RNA family. CM construction requires as input a multiple sequence alignment with well-nested consensus secondary structure annotation. An important question in the design of the SSU-ALIGN program was where to obtain these alignments from. I decided to use the Comparative RNA Website (CRW) (Cannone et al., 2002) as the source because it has the largest amount of high quality structural data of any of the SSU databases (see Chapter 7 of (Nawrocki, 2009)).

The structure models used by crw are based on nearly thirty years of comparative analysis. The first secondary structures of SSU were created in the early 1980s, by Carl Woese, Harry Noller, Robin Gutell and others using comparative sequence analysis to identify covarying positions indicative of structural relationships such as basepairs (Woese et al., 1980; Noller and Woese, 1981; Woese et al., 1983). Since then, Gutell and his colleagues have continued to refine those models. Their comparative approach was validated in 2000, when the crystal structure of the small subunit of *Thermus thermophilus* was solved (Wimberly et al., 2000) and 97% of the predicted basepairs in the then current bacterial secondary structure model were confirmed (Cannone et al., 2002).

Over the past twenty years, Gutell and coworkers have constructed SSU alignments of thousands of sequences using a combination of automated techniques and manual curation. As part of this process, they have singled out novel (phylogenetically distinct) SSU sequences and manually predicted their secondary structures. The alignment and structural data is publicly available in the crw database (Cannone et al., 2002).

SSU secondary structure data from the Comparative RNA Website

CRW includes separate SSU alignments for archaea, bacteria, chloroplasts, eukarya, and mitochondria. I concentrated only on the archaeal, bacterial, and eukaryotic alignments for the initial version of SSU-ALIGN. Unfortunately, the crw alignments are not structurally annotated, so they cannot be used directly to build CMs. However, crw curators have predicted structures for a subset of the sequences in each alignment. To create structure annotated alignments I mapped the structural data onto the alignments through a series of steps as described below. Statistics on the crw alignments and structural data as of May 13, 2009 are given below:

	# of aligned sequences		
family	primary	seed	# of structures
archaea	788	132	25
bacteria	35998	1266	231
eukarya	1937	N/A	259

The *primary* alignments are the largest, most complete alignments in crw. The *seed* alignments are smaller and "highly refined". Because CM alignment accuracy is highly dependent on the quality of the seed alignment used to build the model, I decided to concentrate on the seed alignments for archaea and bacteria and the primary alignment for the eukarya (because no eukaryotic seed exists).

In an effort to ensure high accuracy, I decided not to use the full crw alignments as my seed alignments but rather to use subsets of the alignments containing only the sequences for which individual structure predictions exist. There were two main reasons for this. First, the sequences that were chosen for individual structure prediction were those that represented the major phylogenetic groups and "reveal the major forms of sequence and structure conservation and variation" (Cannone et al., 2002). This suggests they would constitute a good seed alignment from which to build a profile. (A good seed alignment should be representative of the family and generally does not benefit from redundancy (Durbin et al., 1998).) Secondly,

[15]This is a slightly modified version of a section from chapter 9 of my Ph.D. thesis (Nawrocki, 2009). The full thesis is available from (`http://eddylab.org/publications.html`)

after predicting an individual structure, the crw curators use the structure to revise the larger alignments, which suggests that these particular sequences are the most reliably aligned because they have received the most expert attention.

Defining consensus structures from individual structures

Obtaining the consensus structure annotated alignments needed to build CMs required combining the individual structural data and the alignments. One simple approach would be to define a single individual structure x as the consensus structure and impose it on the entire alignment. However, this is not ideal because it means the consensus *will not* include structural features that are absent from x but present in other individual structures, and *will* include structural features that may be unique to x, or at least uncommon in other individual structures. A better approach is to combine or average the individual structures in a reasonable way to determine the consensus structure, and then impose it on the alignment. The procedure I used for deriving consensus structure annotated seed alignments from the crw data is shown in Figure 5.

The first step was to extract the aligned sequences that matched to the individual structures from the master alignments. An individual structure sequence i and an aligned sequence a qualified as match if: a and i both had the same sequence accession, and the unaligned sequence of a was either identical to i or an exact subsequence of i. Not all of the individual structures i had a matching aligned sequence a per these criteria. The number of matches per alignment is shown below:

family	# of aligned sequences	# of structures	# of matches (overlap)
archaea	132	25	23
bacteria	1266	231	95
eukarya	1937	259	148

This defines three sets of aligned sequences in which each sequence has its own predicted structure. CMs can only model well-nested basepairing interactions, so all pseudoknotted basepairs were removed from the individual structures. A well-nested structure is a set of basepairs for which no two pairs between positions i:j and k:l exist such that $i < k < j < l$. I used the program KNOTTED2NESTED.PY by (Smit et al., 2008) to remove pseudoknots using the `-m OSP` option which maximizes the number of basepairs in the resulting nested structure.

From this set, any sequences with more than 5 ambiguous bases were removed because ambiguous bases in a seed alignment inject noise into the parameters of a CM (equation 1.4 of (Nawrocki, 2009)). Additionally, sequences less than 80%, or more than 120% the median length of the alignment were removed[16]..

At this stage, basepairing *conflicts* between the aligned individual structures were identified. A conflict exists between two basepairs in different structures, one between alignment columns *i* and *j* and the other between columns *k* and *l*, if $i = k$ and $j \neq l$, or $j = l$ and $i \neq k$. An example of two conflicting basepairs is shown in Figure 5. Conflicting basepairs are problematic because a consensus basepair between columns i and j in a CM seed alignment is assumed to exist (or be deleted) between the nucleotides in i and j in *all* sequences of the alignment. Columns involved in conflicting basepairs violate this assumption by specifying that a nucleotides in a single column is involved in different basepairs in different sequences.

Next, I removed conflicts using an iterative alignment refinement procedure that eliminates sequences with more than 15 conflicts after each iteration. The alignments at each stage are determined using a CM. The initial consensus structure used to build the CM for the first iteration was defined as the set of consensus base pairs between alignment positions i and j that exist as paired in more than *x* fraction of the individual structures. The value for *x* was determined as the minimum value for which there were no conflicting basepairs in the consensus set.

[16] 33 eukaryotic sequences were less than 80% the median length (table 1). I plan to build an additional eukaryotic model of these shorter sequences for a future release of SSU-ALIGN

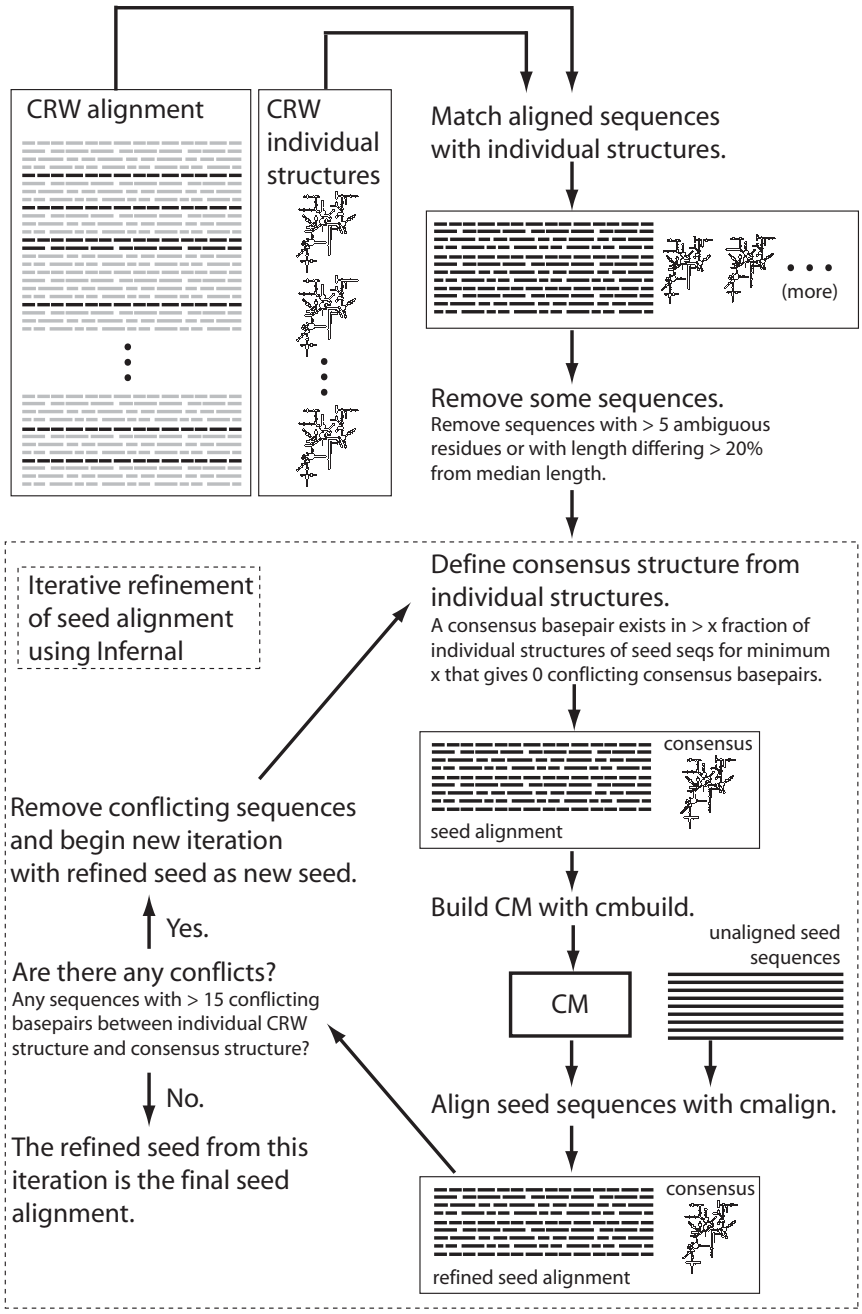

Figure 10: **The procedure for converting SSU alignments and individual structures from crw to seed alignments for SSU-ALIGN**. crw individual structures only exist for a small subset of the sequences from the crw alignments (the black sequences among the majority of gray ones). Each conversion step is explained in more detail in the text. Figure 5 demonstrates an example of conflicting basepairs. Table 1 includes alignment statistics and number of sequences surviving each step for each of the three seed alignments derived from crw.

```
                                          .........i....j.......
00560::Xylella_fastidiosa                 GCAGGGGACCUUAGGGCCUUGU
#=GS 00560::Xylella_fastidiosa SS         <<<<<<..<<....>>>>>>>>
00018::Thermomicrobium_roseum             GGCGCA--G-GCGAC-UGUGCU
#=GS 00018::Thermomicrobium_roseum SS     <<<<<<..<.....>.>>>>>>
                                          ........k.....l.......
```

Figure 11: **Example of conflicting basepairs between two aligned individual SSU structure predictions from CRW.** The individual basepair between aligned columns i and j in *Xylella fastidiosa* (sequence accession M34115) conflicts with the *Thermomicrobium roseum* (accession AE003861) basepair between columns k and l as defined in the text (because $i \neq k$ and $j = l$).

This provided me with an initial alignment that I then iteratively refined using INFERNAL. Each iteration consists of a build step, an alignment step, and a sequence removal step. First, a CM is built from the current alignment (in iteration 1 this is a subset of the crw alignment). Then all of the seed sequences are aligned to the CM to generate a new alignment. The individual structures are mapped onto the new alignment and a new consensus structure is derived as described above. Any sequence with more than 15 basepair conflicts between its individual structure and the new consensus structure are removed from the seed. This procedure continues until 0 sequences are removed in the final step. The alignment generated during the final iteration became the seed alignment I used for SSU-ALIGN.

Table 1 lists the number of sequences removed at each stage of the procedure and statistics on basepair conflicts. The three final seed alignments used to create the SSU-ALIGN models are summarized in Table 2.

			cons	#	avg #	avg #				
		#	struct	cons	indiv.	conflict	initial sequence removal			
model	stage	seqs	x	bps	bps	bps	total	ambig	short	long
archaea	crw matches	23	0.170	472	456.74	1.30	0	0	0	0
archaea	post-initial removal	23	0.170	472	456.74	1.30				
archaea	1st refinement	23	0.130	474	456.74	0.52				
bacteria	crw matches	95	0.210	480	460.91	1.06	2	2	0	0
bacteria	post-initial removal	93	0.200	480	460.82	1.05				
bacteria	1st refinement	93	0.210	480	460.82	1.28				
eukarya	crw matches	148	0.420	422	466.25	12.94	42	11	33	7
eukarya	post-initial removal	106	0.440	442	487.50	16.04				
eukarya	1st refinement	106	0.410	448	487.50	8.71				
eukarya	2nd refinement	89	0.440	448	487.73	5.49				

Table 1: **Statistics on the conversion of CRW data to seed alignments for SSU-align** For each of the three models, statistics for the alignments at each stage of the crw conversion process are shown. "crw matches" alignments are subsets of the crw alignments for matching sequences and individual structures. "post-initial removal" alignments have had sequences more than 120% the median length ("long" column), less than 80% the median length ("short" column), or with more than 5 ambiguous bases ("ambig" column) removed. The remaining rows are for alignments following each round of the iterative refinement process using INFERNAL. More details on the crw conversion are in the text.

model name	number of sequences	consensus length	alignment length	number of basepairs	average sequence length	average pairwise identity
archaea	23	1508	1563	471	1485	81%
bacteria	93	1582	1689	480	1527	80%
eukarya	89	1881	2652	448	1800	79%

Table 2: **Statistics of the three seed alignments used by SSU-align.** These are the three alignments resulting from the CRW conversion depicted in Figure 5 and described in the text.

The final seed alignments are included in SSU-ALIGN package in the **`models/`** subdirectory as: **`archaea-0p1.stk`**, **`bacteria-0p1.stk`**, and **`eukarya-0p1.stk`**. Corresponding unaligned FASTA files are also included, with a **`.fa`** suffix replacing **`.stk`**.

With the exception of the next page, the remainder of this section includes 18 secondary structure diagrams displaying various statistics per consensus column of the three seed alignments, with figure numbers as indicated below. These structure diagrams can be created using **`ssu-draw`** with the following commands from the top-level SSU-ALIGN directory:

```
> ssu-draw -a models/archaea-0p1.stk
> ssu-draw -a models/bacteria-0p1.stk
> ssu-draw -a models/eukarya-0p1.stk
```

statistic	archaea	bacteria	eukarya
consensus sequence	Figure 12	Figure 18	Figure 24
information content	Figure 13	Figure 19	Figure 25
mutual information	Figure 14	Figure 20	Figure 26
frequency of deletions	Figure 15	Figure 21	Figure 27
frequency of insertions	Figure 16	Figure 22	Figure 28
average insertion length	Figure 17	Figure 23	Figure 29

Secondary structure diagrams summarizing the three models

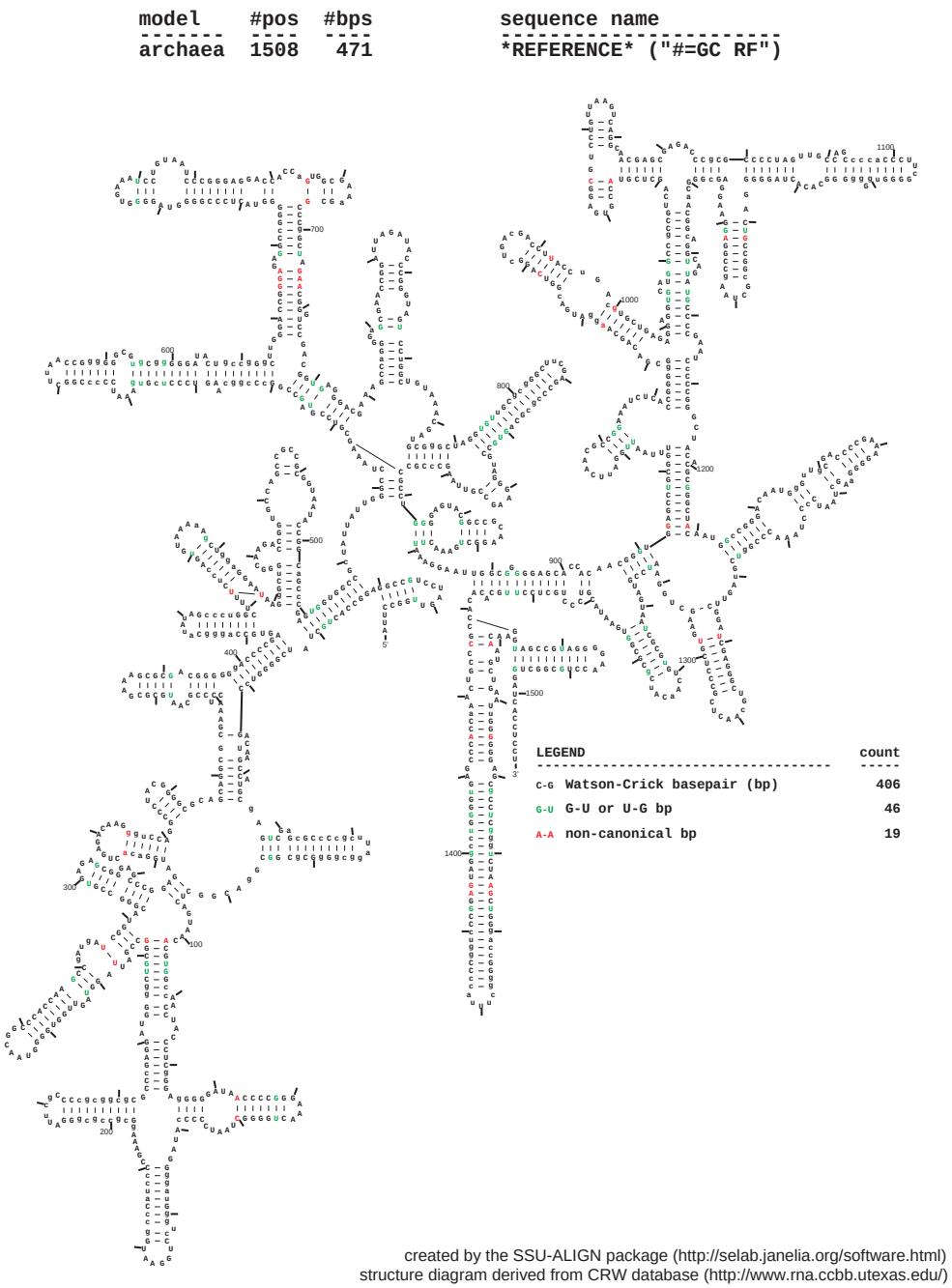

Figure 12: **Secondary structure diagram displaying the consensus sequence of the archaeal SSU model.** This is the single sequence that the model most closely represents and is the highest scoring possible sequence to the model. Uppercase nucleotides indicate highly conserved positions, and lowercase nucleotides indicate less well-conserved positions. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

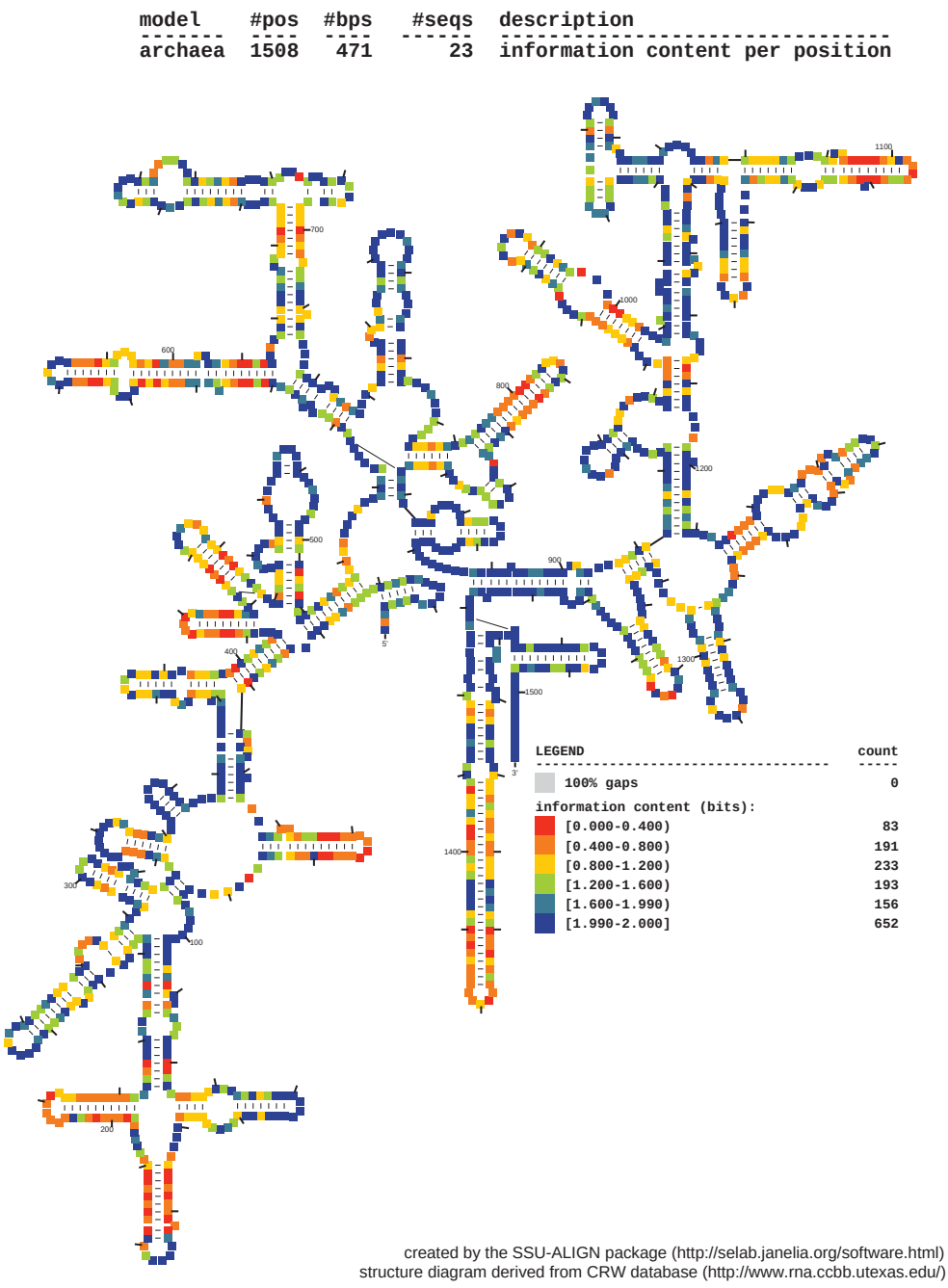

Figure 13: **Secondary structure diagram displaying primary sequence information content per consensus position of the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

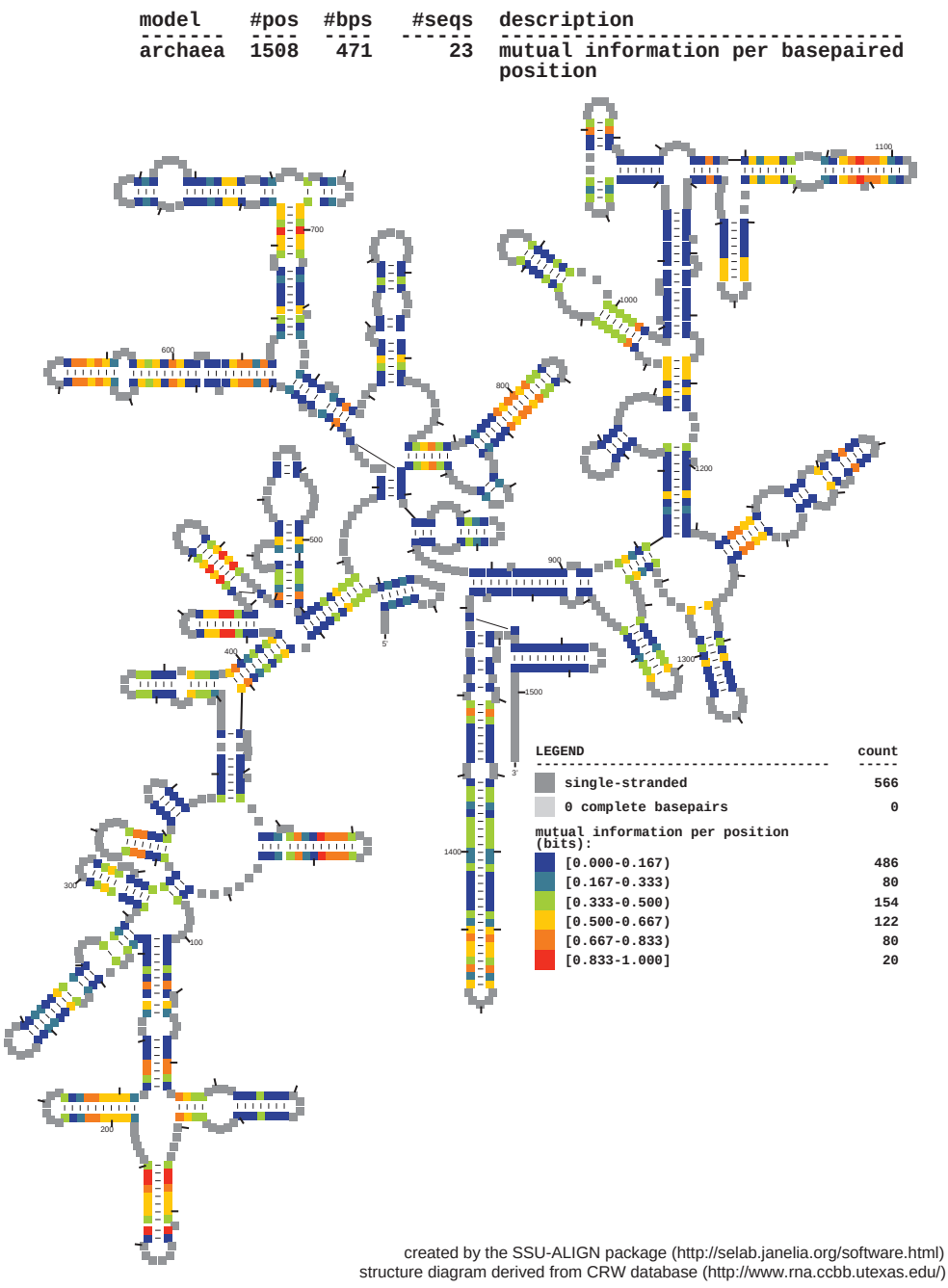

Figure 14: **Secondary structure diagram displaying extra information from conserved structure per consensus position of the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

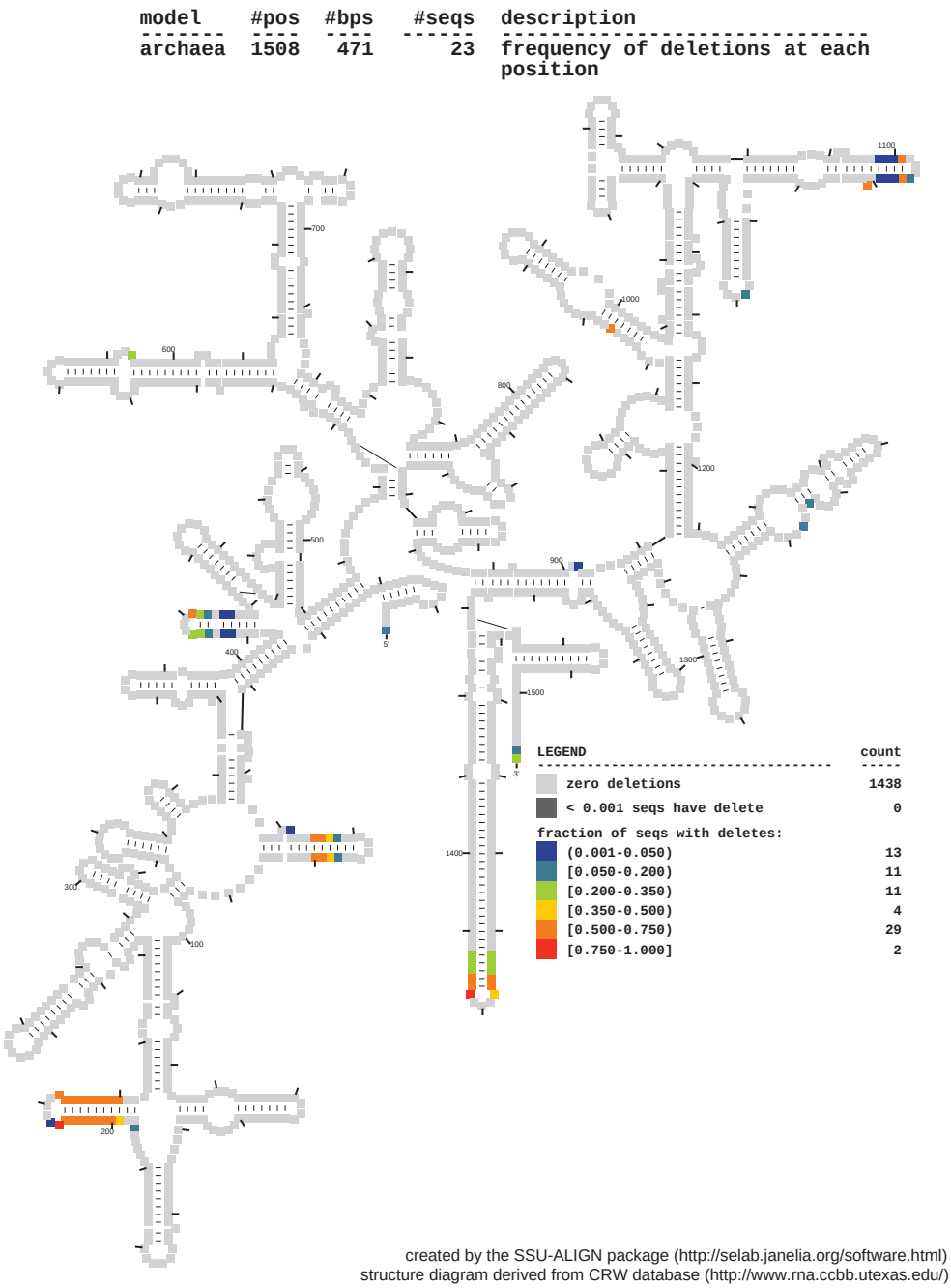

Figure 15: **Secondary structure diagram displaying frequency of deletions per consensus position of the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

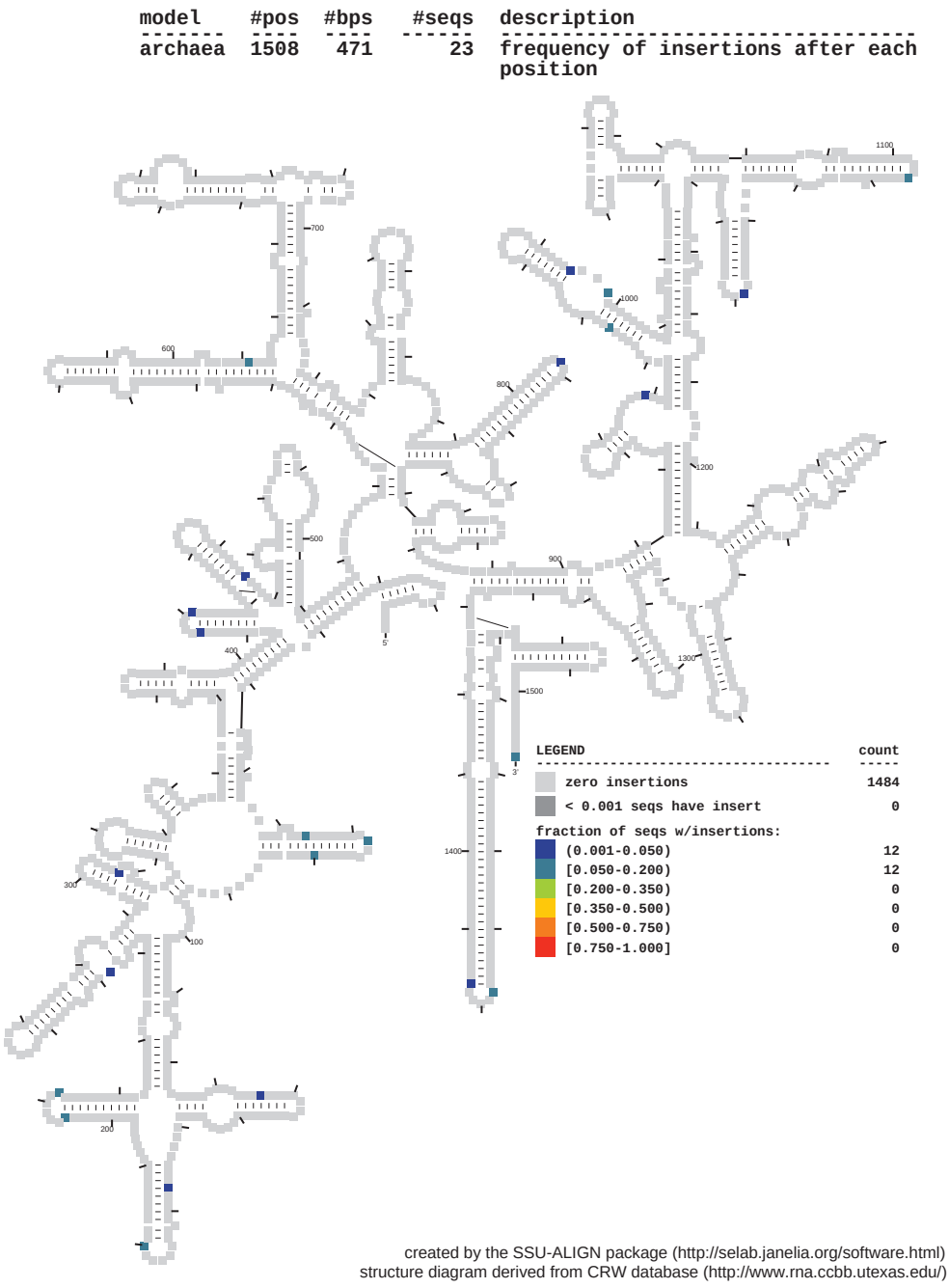

Figure 16: **Secondary structure diagram displaying frequency of insertions after each consensus position in the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

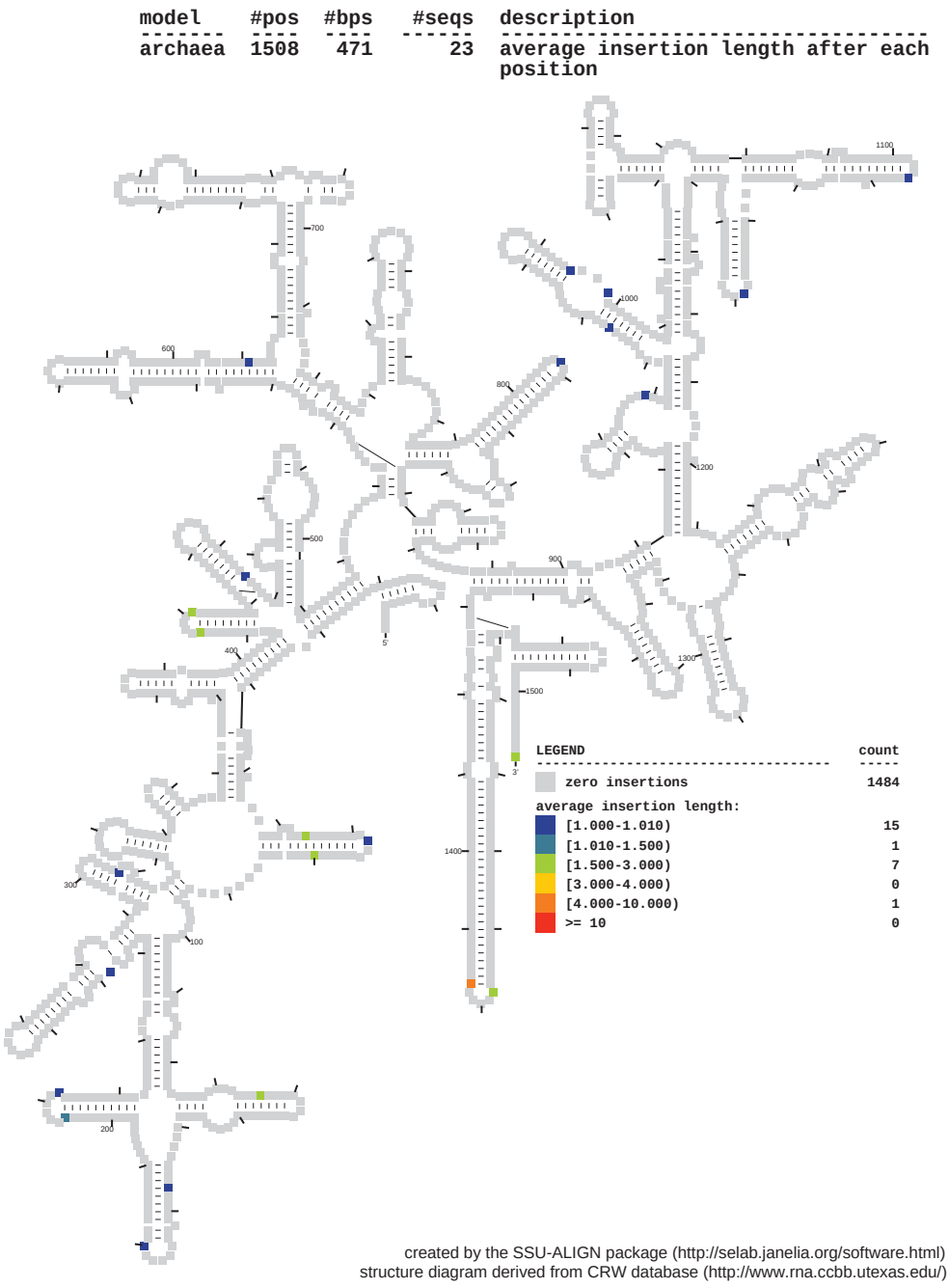

Figure 17: **Secondary structure diagram displaying average length of insertions after each consensus position in the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

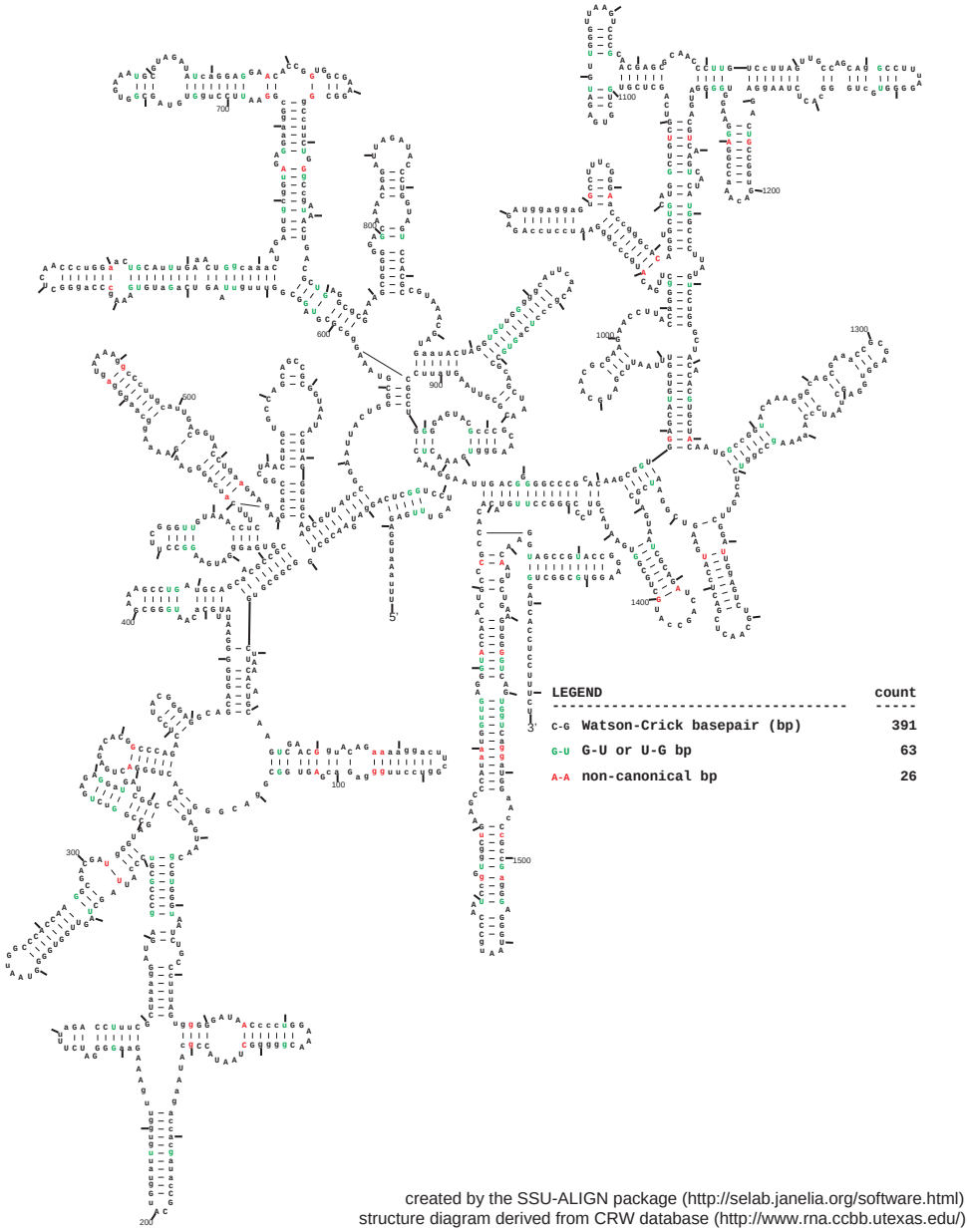

Figure 18: **Secondary structure diagram displaying the consensus sequence of the bacterial SSU model.** This is the single sequence that the model most closely represents, and is the highest scoring possible sequence to the model. Uppercase nucleotides indicate highly conserved positions, and lowercase nucleotides indicate less well-conserved positions. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

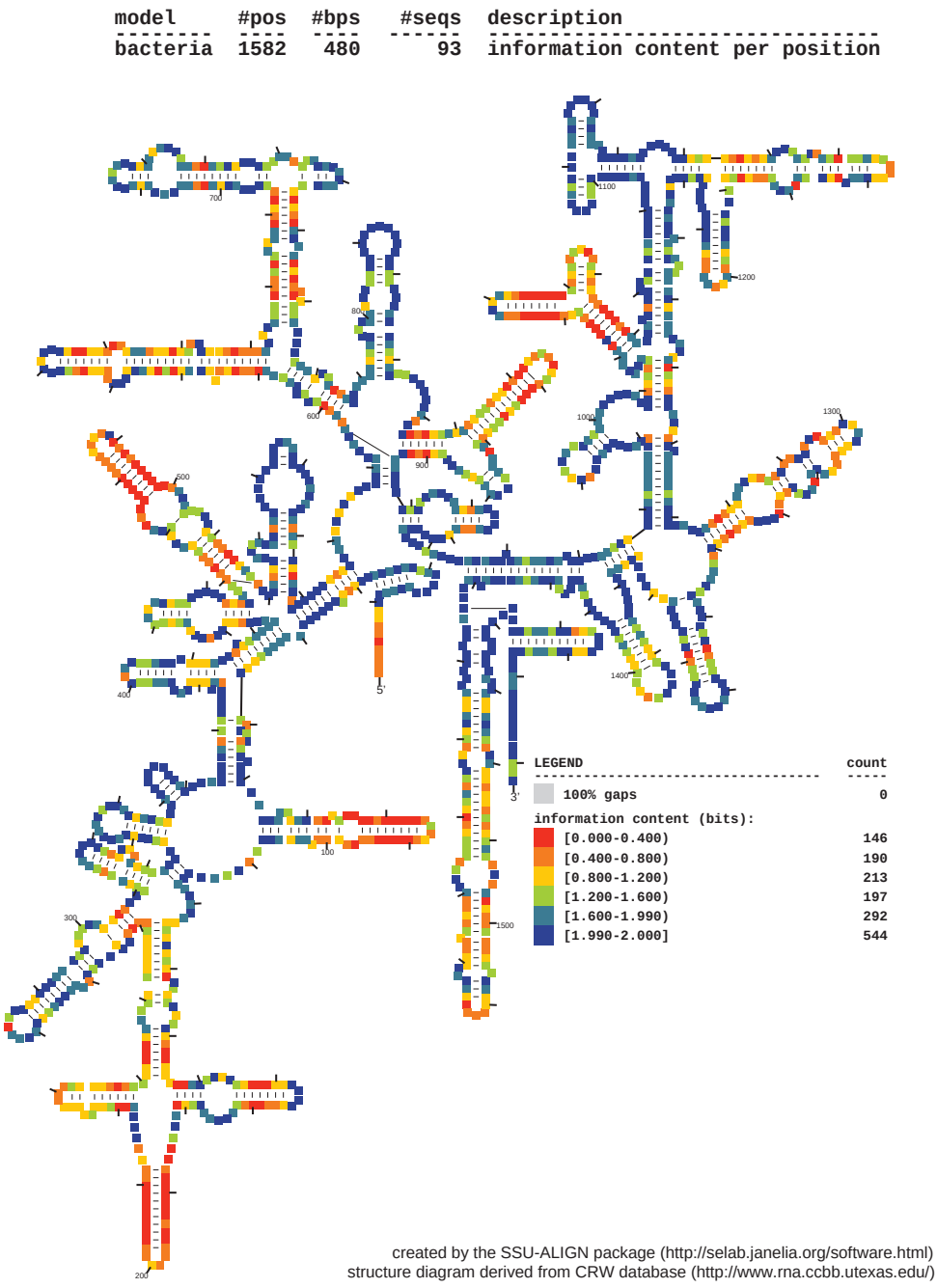

Figure 19: **Secondary structure diagram displaying primary sequence information content per consensus position of the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

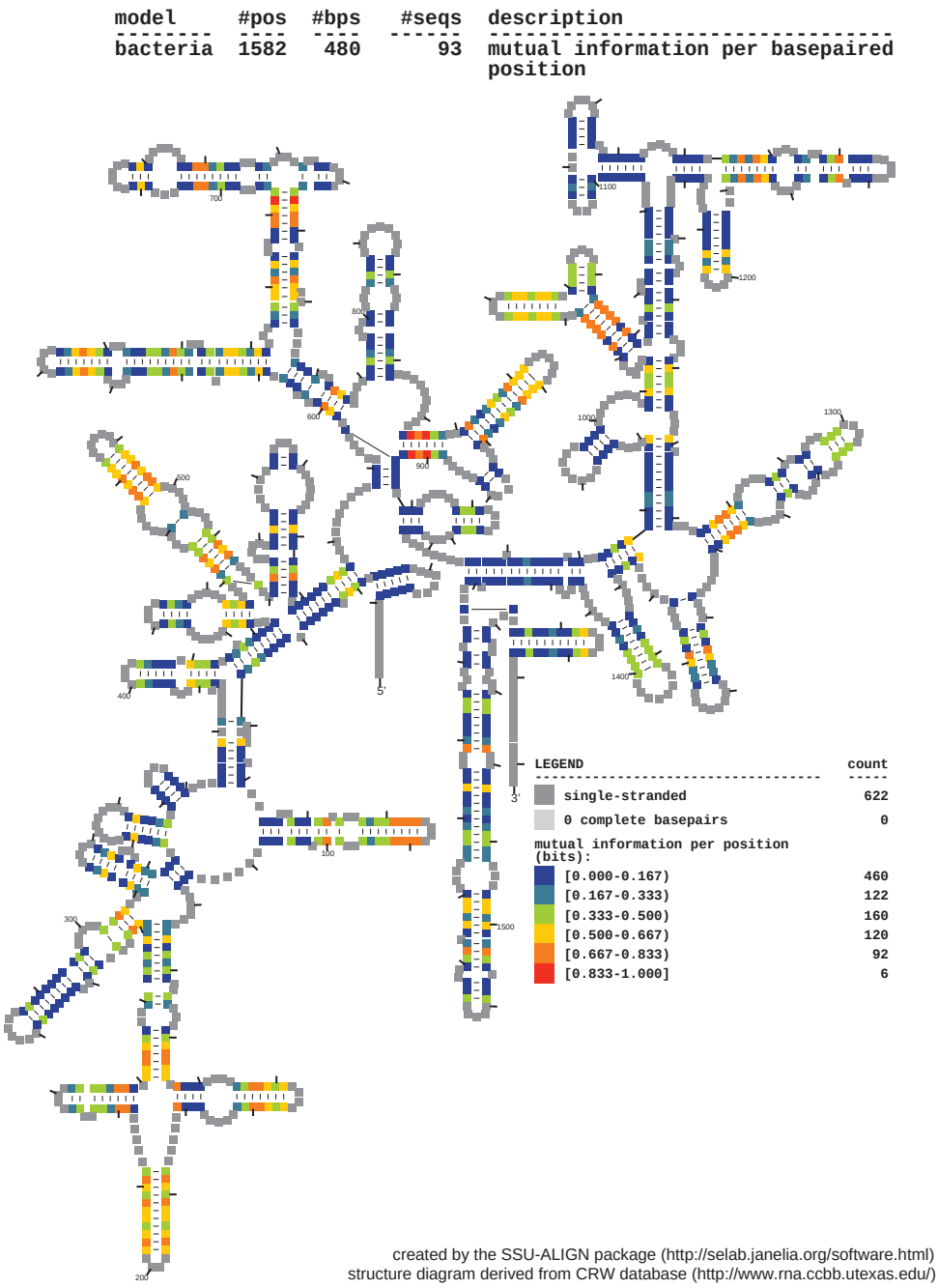

Figure 20: **Secondary structure diagram displaying extra information from conserved structure per consensus position of the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

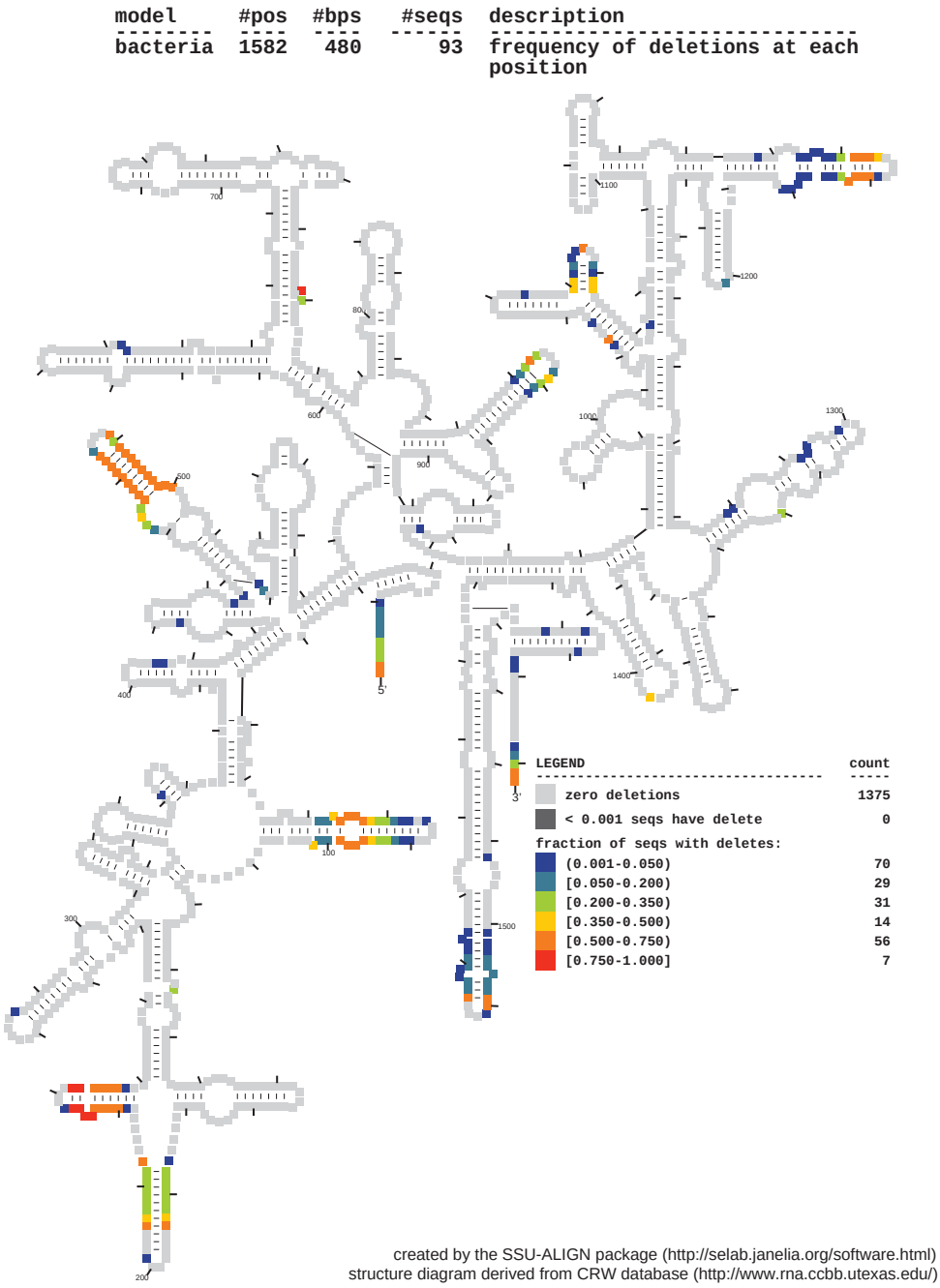

Figure 21: **Secondary structure diagram displaying frequency of deletions per consensus position of the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

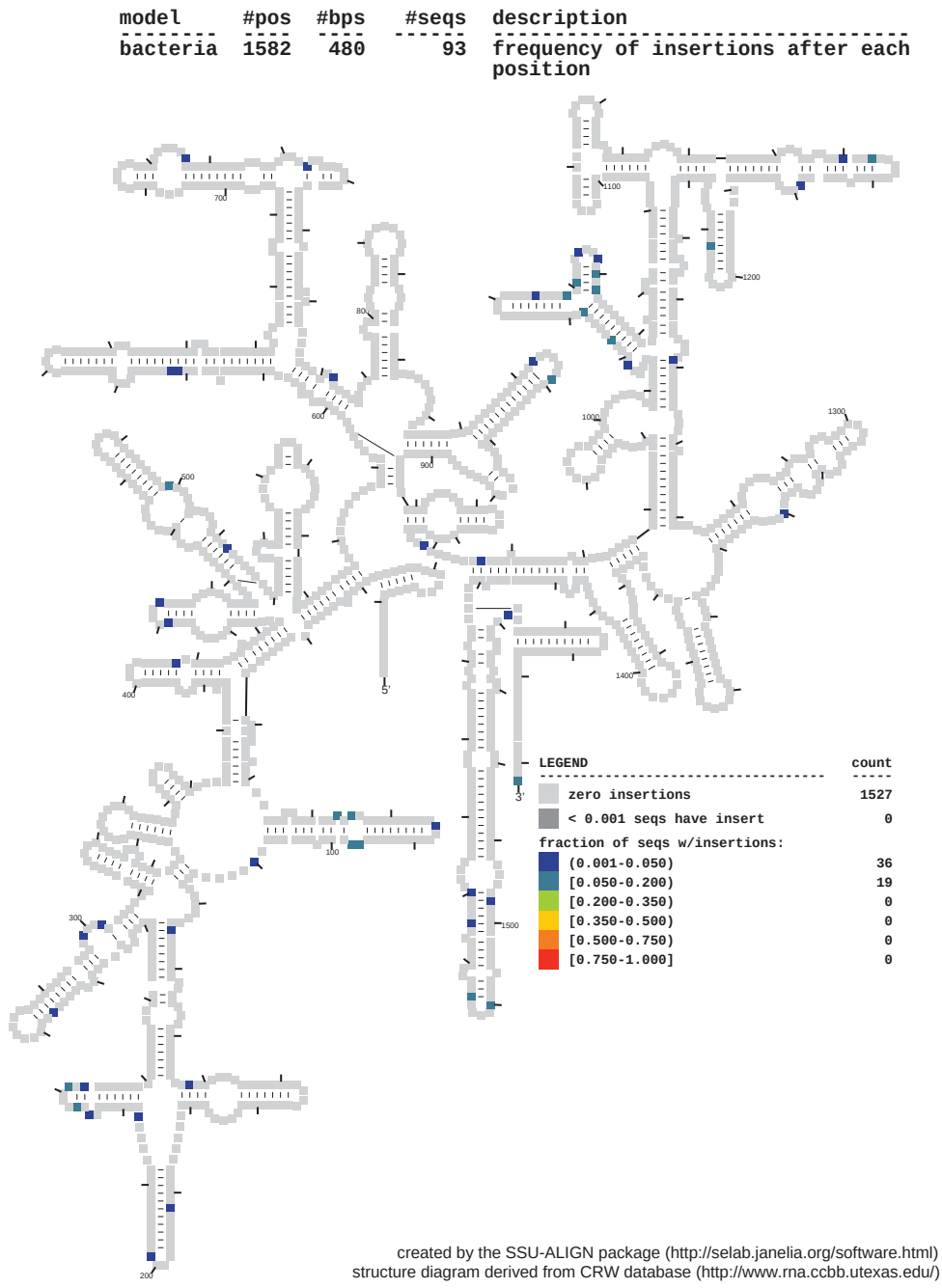

Figure 22: **Secondary structure diagram displaying frequency of insertions after each consensus position in the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

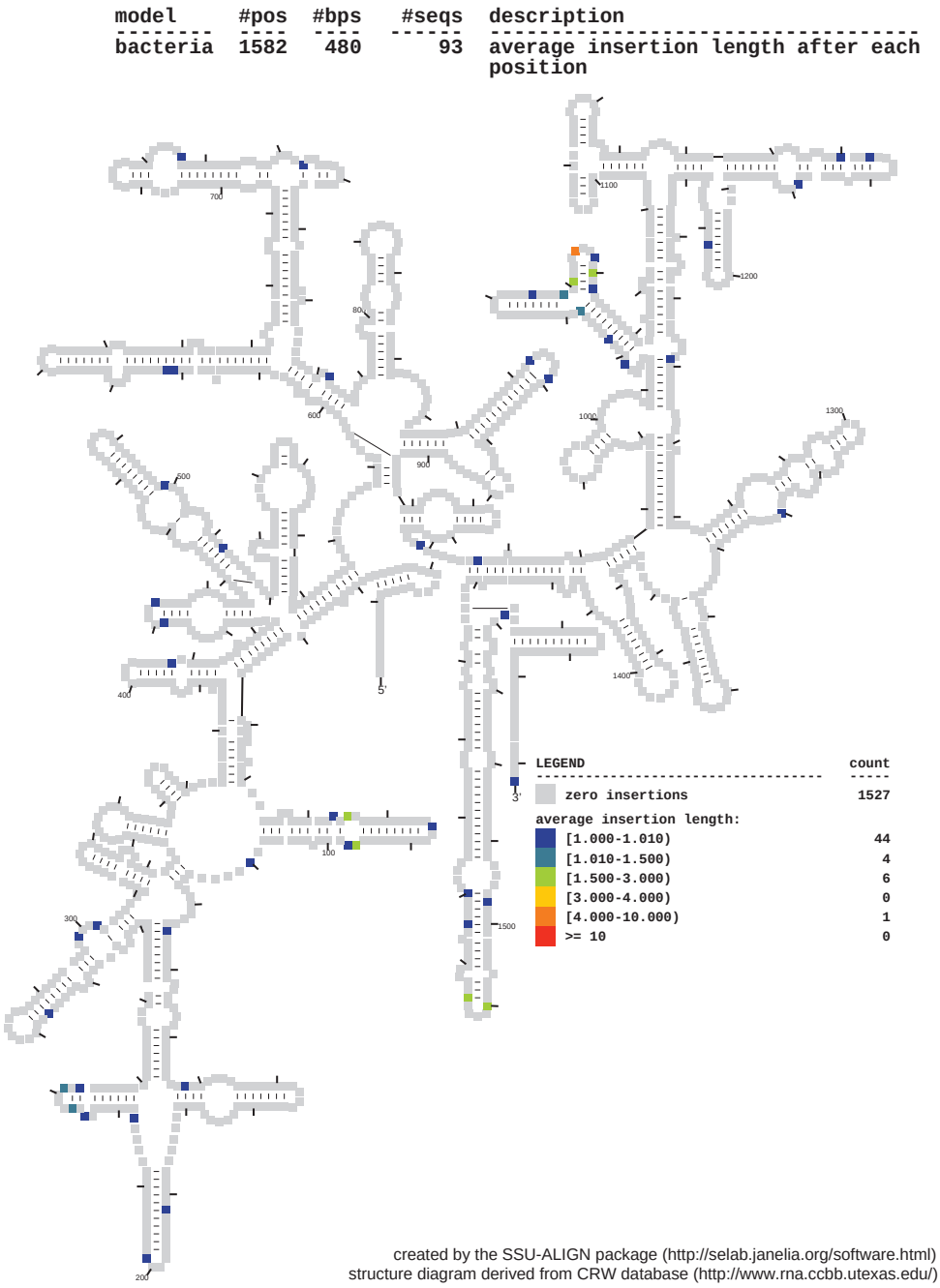

Figure 23: **Secondary structure diagram displaying average length of insertions after each consensus position in the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

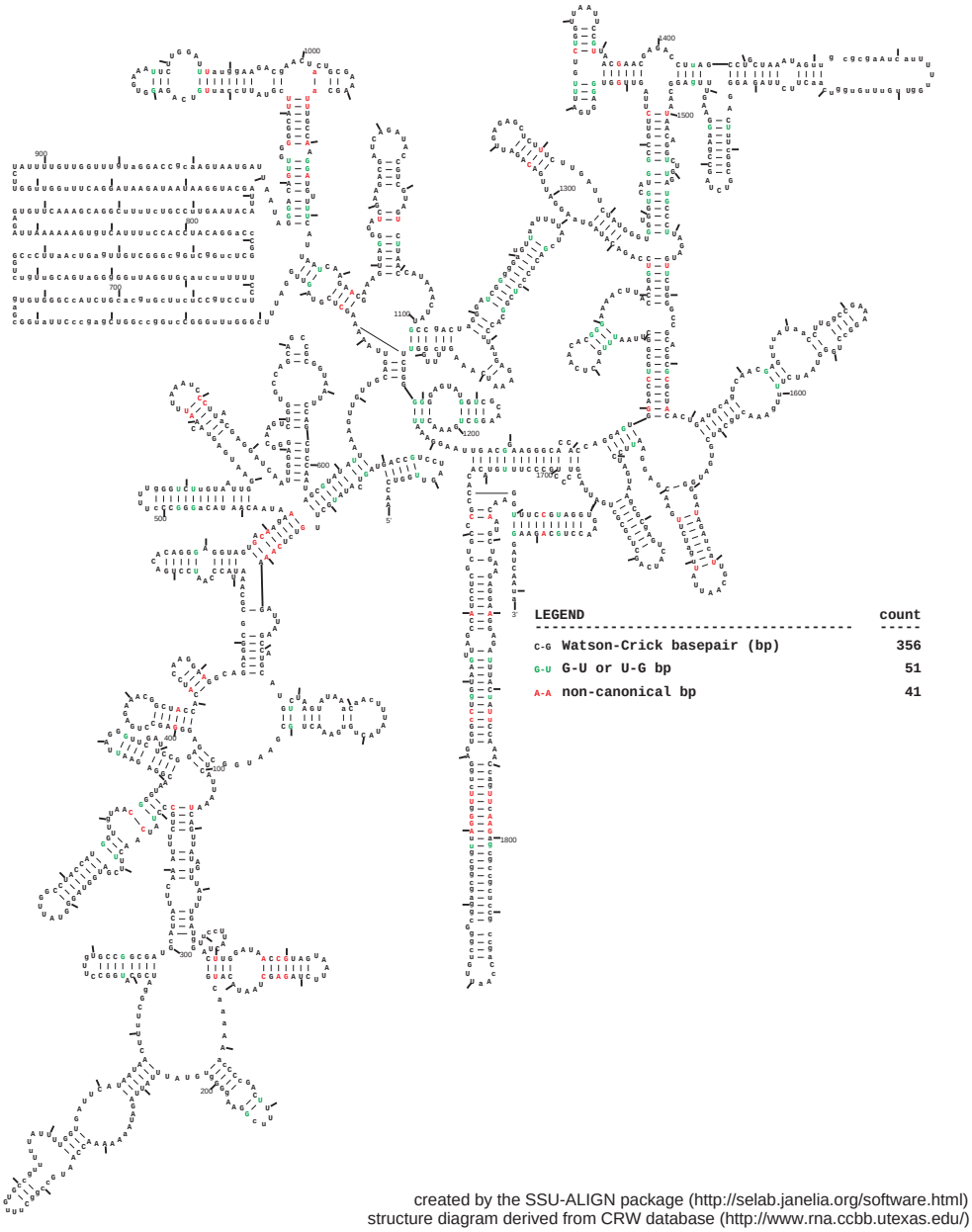

Figure 24: **Secondary structure diagram displaying the consensus sequence of the eukaryotic SSU model.** This is the single sequence that the model most closely represents, and is the highest scoring possible sequence to the model. Uppercase nucleotides indicate highly conserved positions, and lowercase nucleotides indicate less well-conserved positions. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

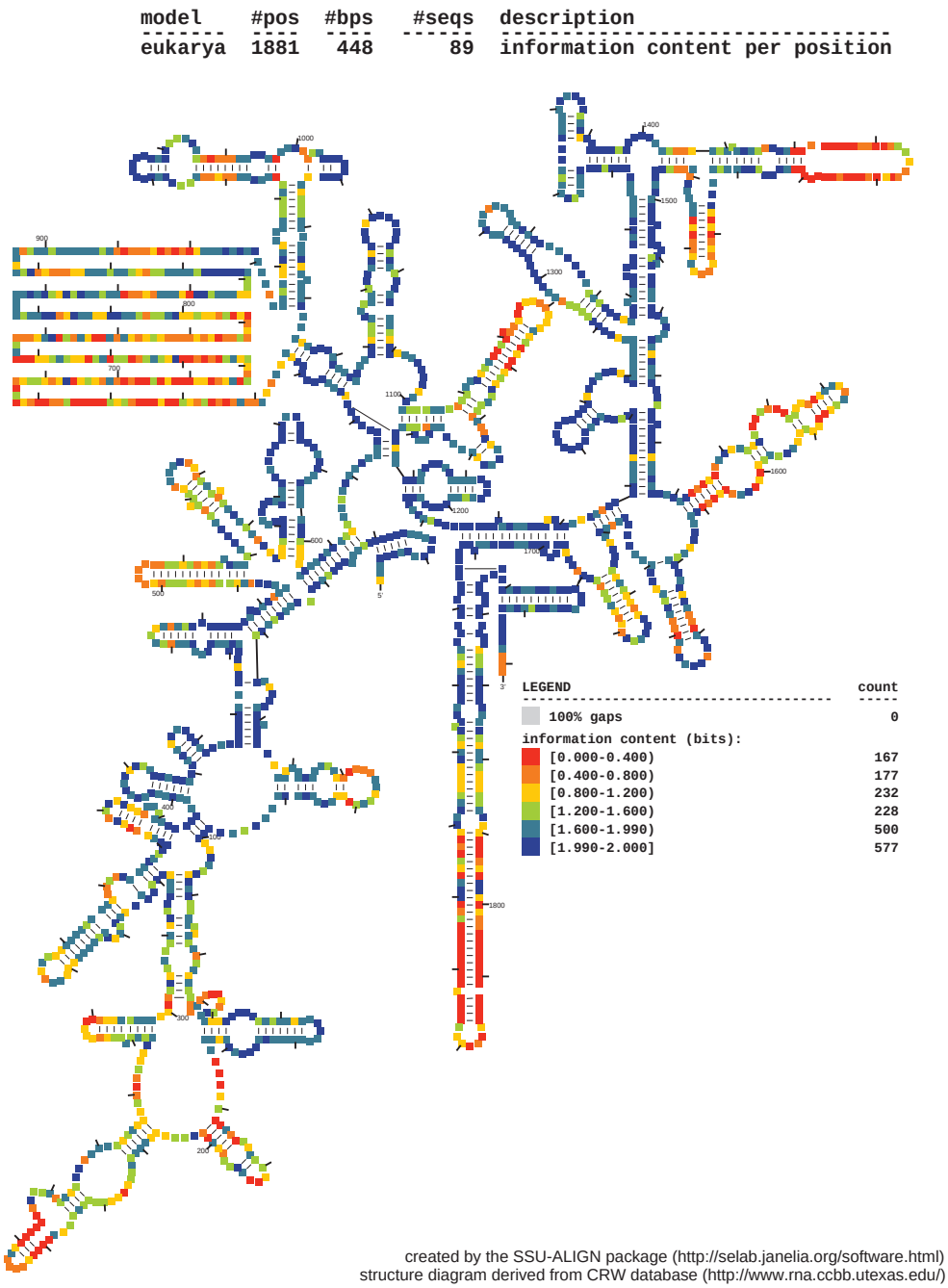

Figure 25: **Secondary structure diagram displaying primary sequence information content per consensus position of the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

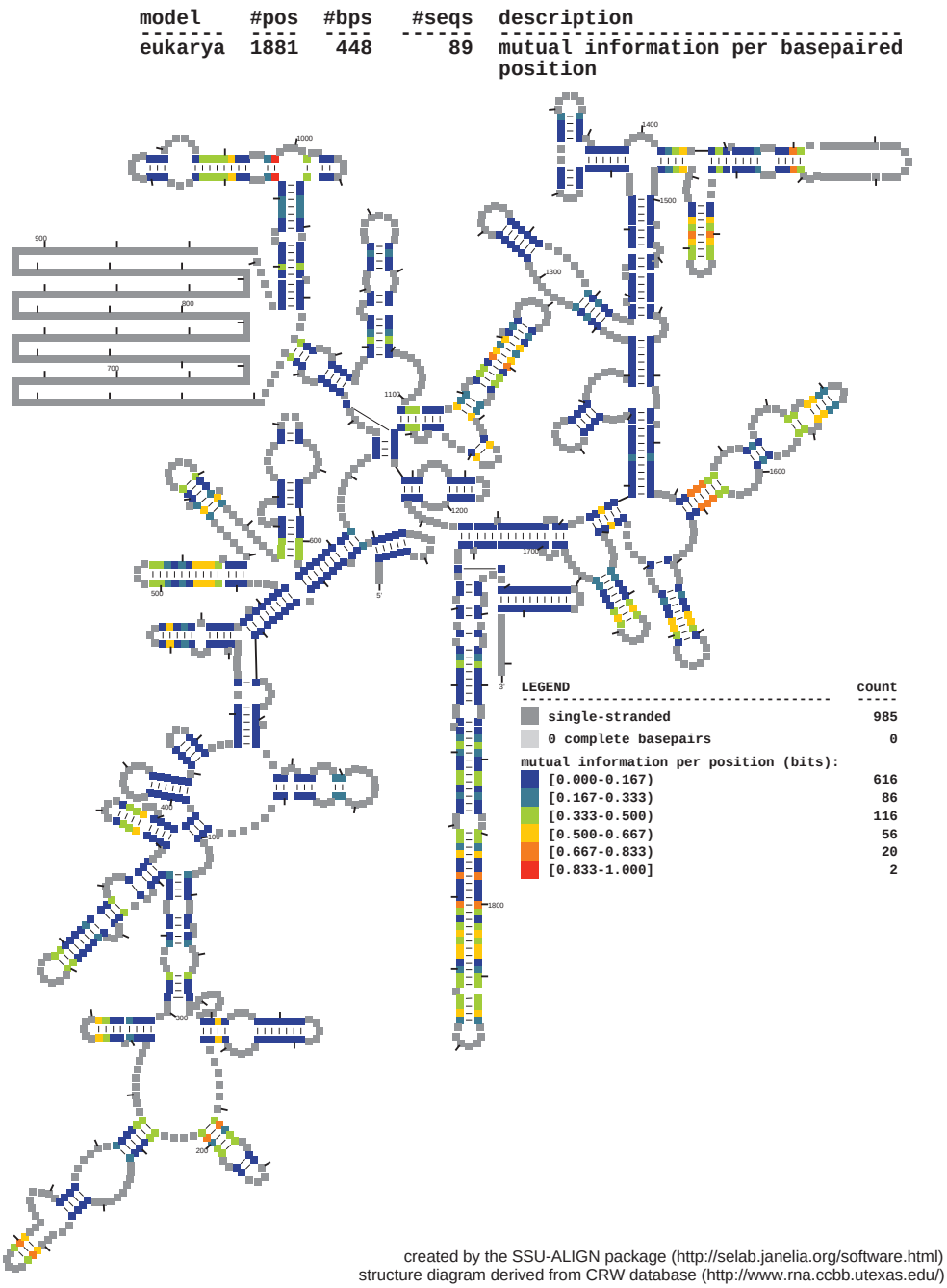

Figure 26: **Secondary structure diagram displaying extra information from conserved structure per consensus position of the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

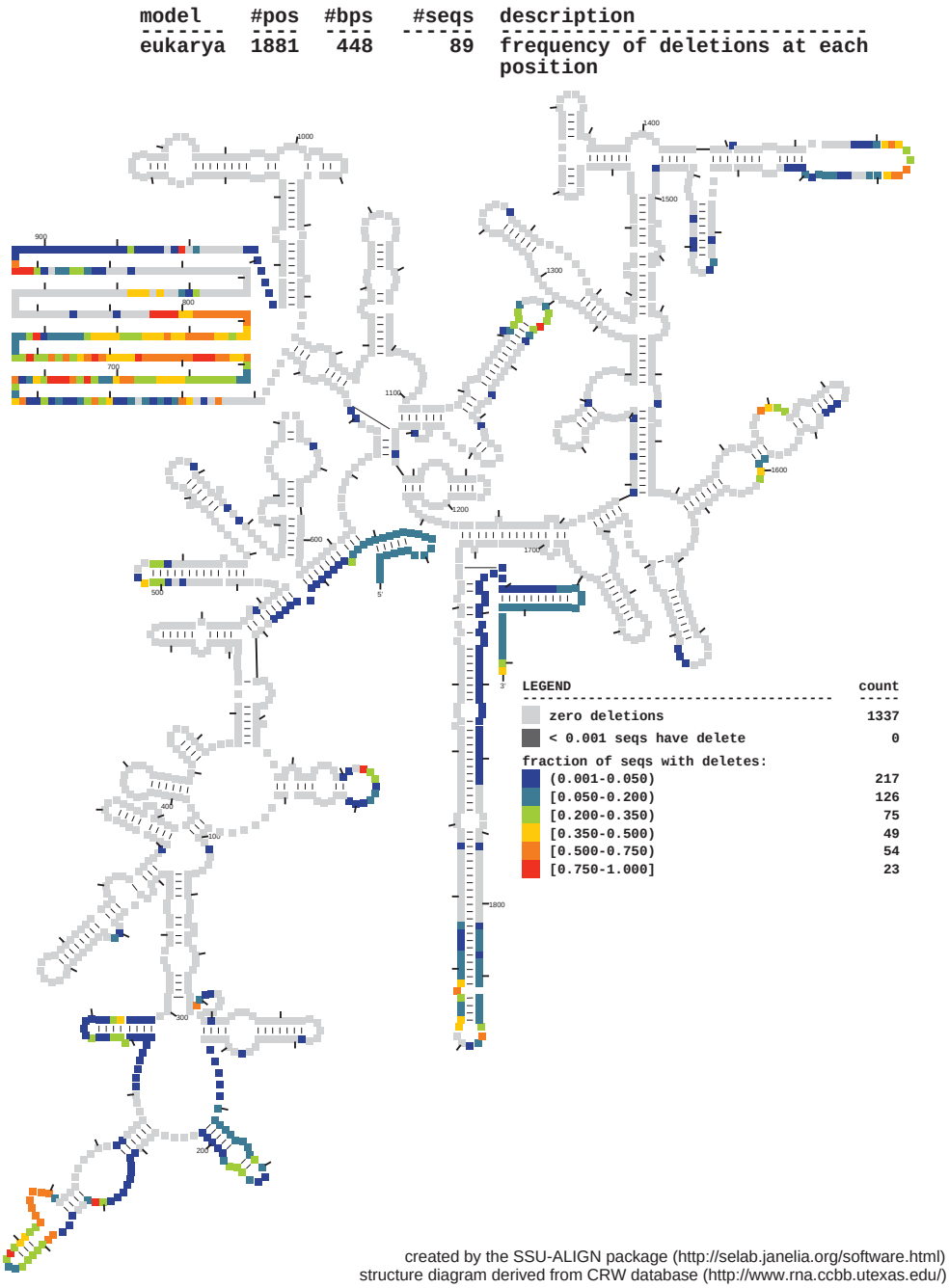

Figure 27: **Secondary structure diagram displaying frequency of deletions per consensus position of the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

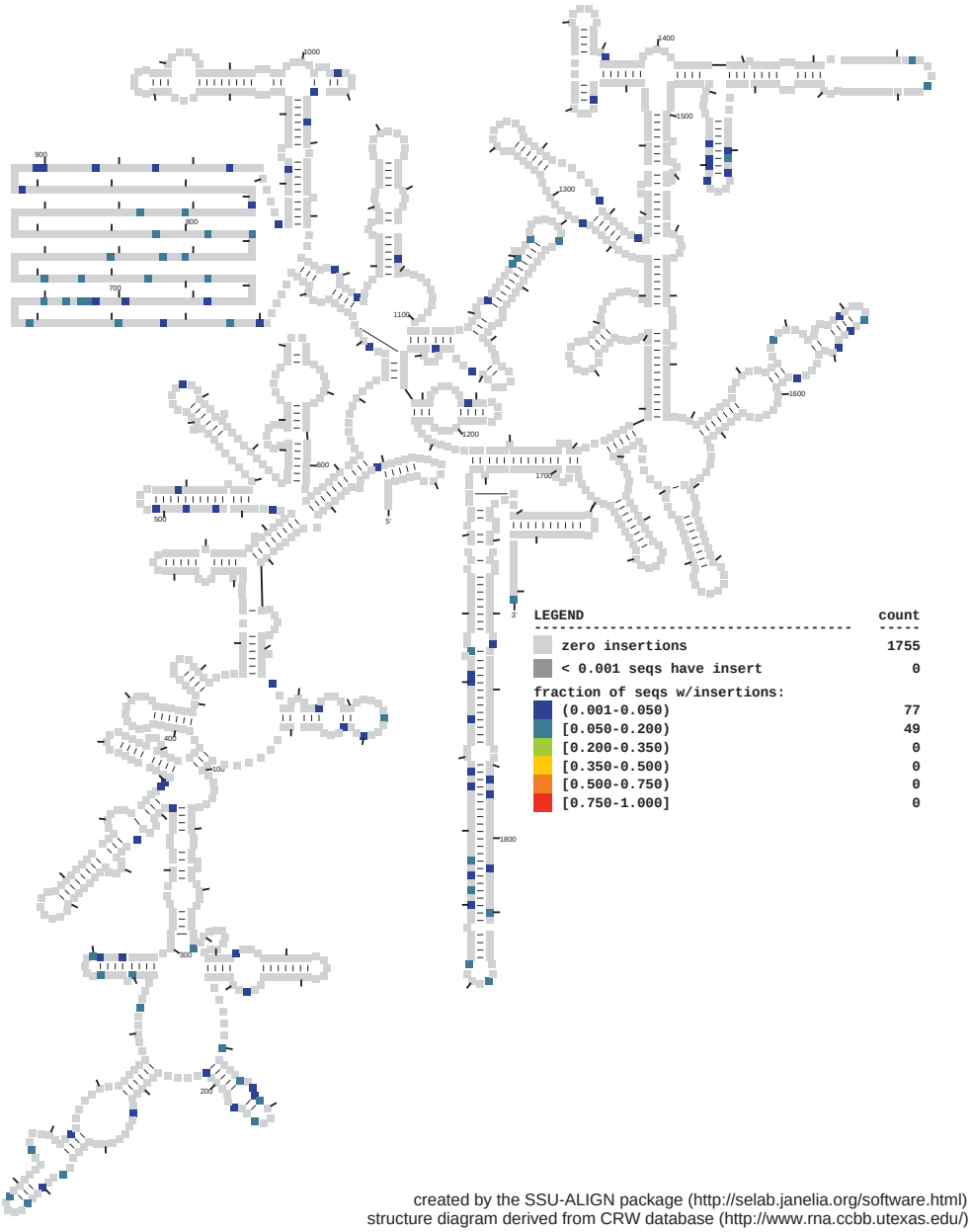

Figure 28: **Secondary structure diagram displaying frequency of insertions after each consensus position in the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

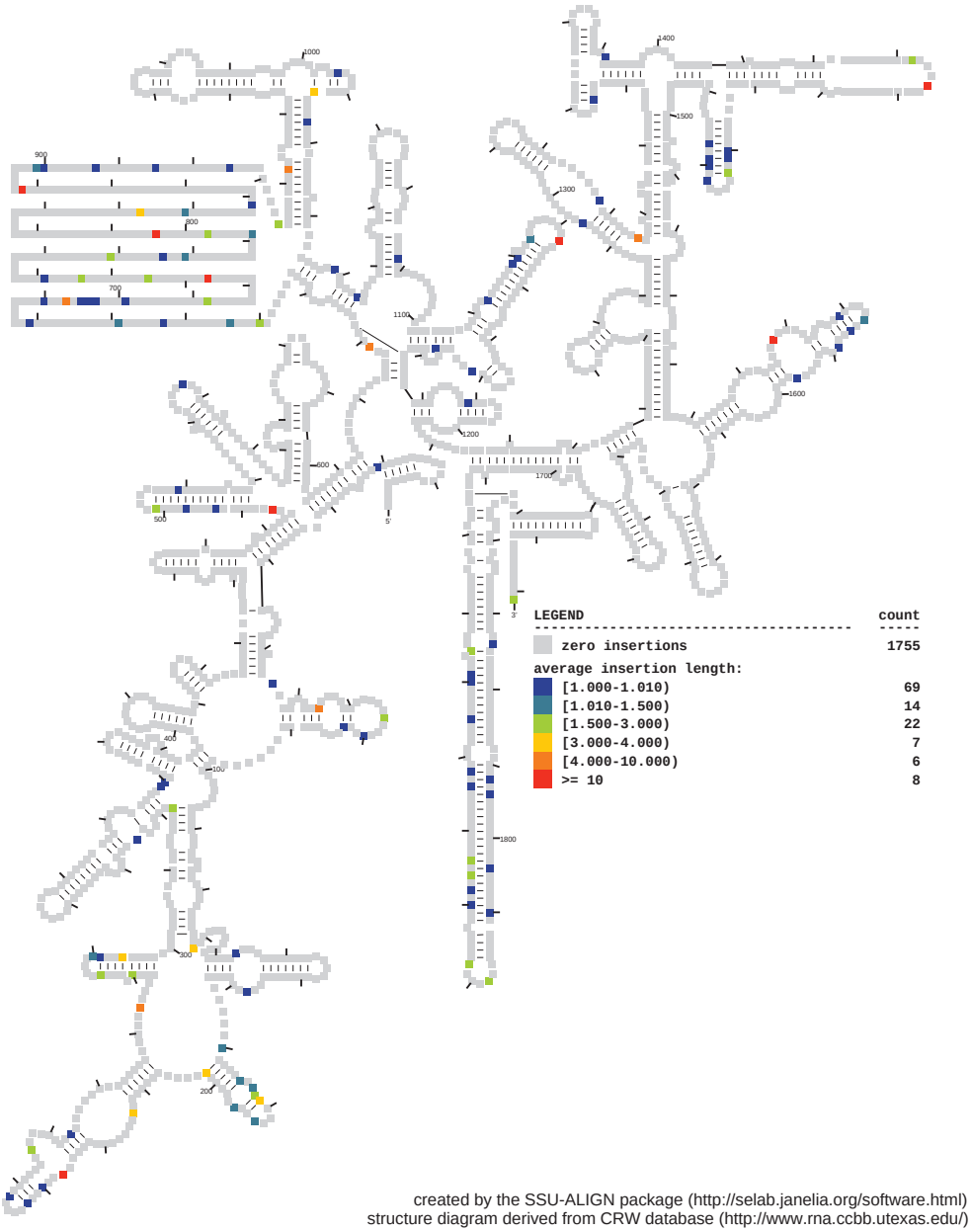

Figure 29: **Secondary structure diagram displaying average length of insertions after each consensus position in the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the crw database (Cannone et al., 2002) as described in the text. This diagram was generated by the `ssu-draw` program included in SSU-ALIGN

.

6 How SSU-ALIGN's default alignment masks were determined

This section describes how the default alignment masks used by SSU-ALIGN were determined. There are three default masks, one for each of the three default models: archaea, bacteria and eukarya.

The masks were determined based on large datasets of SSU sequences. Specifically, subsets of all the SSU sequences in the January 24, 2010 release of the GREENGENES database (DeSantis et al., 2006a) and the *Ref* set of SSU rRNA sequences from release 100 of the SILVA database (Pruesse et al., 2007) were used. Table 3 includes statistics on these alignments.

The first step towards determining the masks was to run **ssu-align** with the **--no-align** option on the combined SILVA *Ref* and GREENGENES unaligned datasets, in FASTA format. If the combined dataset is in the file **ggsil.fa**, this could be reproduced using the command:

```
> ssu-align --no-align ggsil.fa ggsil
```

From this, three new FASTA files are created in the new directory **ggsil/**: **ggsil.archaea.fa**, **ggsil.bacteria.fa**, and **ggsil.eukarya.fa**, the predicted set of archaeal, bacterial and eukaryotic SSU sequences (possibly with some of the original GREENGENES or SILVA sequence trimmed away from the ends).

Next, I used Robert Edgar's fast UCLUST tool (version 1.0.50) to remove redundancy from each of these sets of sequences. Specifically, for archaea, I executed the following three **uclust** commands:

```
> uclust1.0.50_linuxi86_64 --sort ggsil.archaea.fa --output ggsil.sorted.archaea.fa
> uclust1.0.50_linuxi86_64 --input ggsil.sorted.archaea.fa --uc
ggsil.f97.archaea.uc --id 0.97
> uclust1.0.50_linuxi86_64 --input ggsil.sorted.archaea.fa --uc2fasta
ggsil.f97.archaea.uc --types S --output ggsil.f97.archaea.fa
```

The final output file is **ggsil.f97.archaea.fa**. This file has been filtered to 97% identity, i.e. each sequence should be at least 3% different from all other sequences in this dataset. I repeated the same procedure for the bacterial an eukaryotic datasets that were output from the initial **ssu-align** step as well.

I then created alignments of these datasets using **ssu-align**. For example, for the archaeal dataset, I executed:

```
> ssu-align --no-search -n archaea ggsil.f97.archaea.fa arc4mask
```

This generates the alignment **arc4mask.archaea.stk** in the directory **arc4mask/**. The final step was to execute **ssu-mask** on these alignments. By default, **ssu-mask** will examine the posterior probabilities in the alignment file to determine a mask which defines which consensus columns to keep and which to remove. Any consensus column for which 95% of the sequences that do not have a gap in the column have a posterior probability of at least 0.95 will be kept by **ssu-mask**, all others will be removed.

	num seqs	num seqs	mask	
domain	unfiltered	filtered	included	excluded
archaea	23197	3853	1376	132
bacteria	739075	118742	1376	212
eukarya	47442	12475	1343	538

Table 3: Statistics on the alignments used to determine the default SSU-ALIGN 0.1 alignment masks. The alignments were derived from the GREENGENES (DeSantis et al., 2006a) and SILVA (Pruesse et al., 2007) databases as described in the text.

The default archaeal, bacterial and eukaryotic masks are shown in figures 30, 31, 32, respectively. Each of theses figures includes two diagrams of the mask displayed on the consensus secondary structure of the corresponding model. The diagrams on the left of each figure show excluded positions in pink, and included positions as black. The diagrams on the right show excluded positions as open circles, and included positions as solid squares, with positions colored by deletion (gap) frequency in the filtered alignments used for determining the masks. Gray and dark blue positions are never or rarely gaps; orange and red positions

are often gaps. Notice that nearly all of the positions that are excluded by the mask are themselves or are nearby positions that are often gaps. This is intuitive: alignment ambiguity increases (alignment confidence decreases) in parts of SSU that tolerate insertions and deletions because it becomes more difficult to determine which exact nucleotides are homologous between different sequences in these regions. Section 3, page 15 includes further discussion on alignment ambiguity.

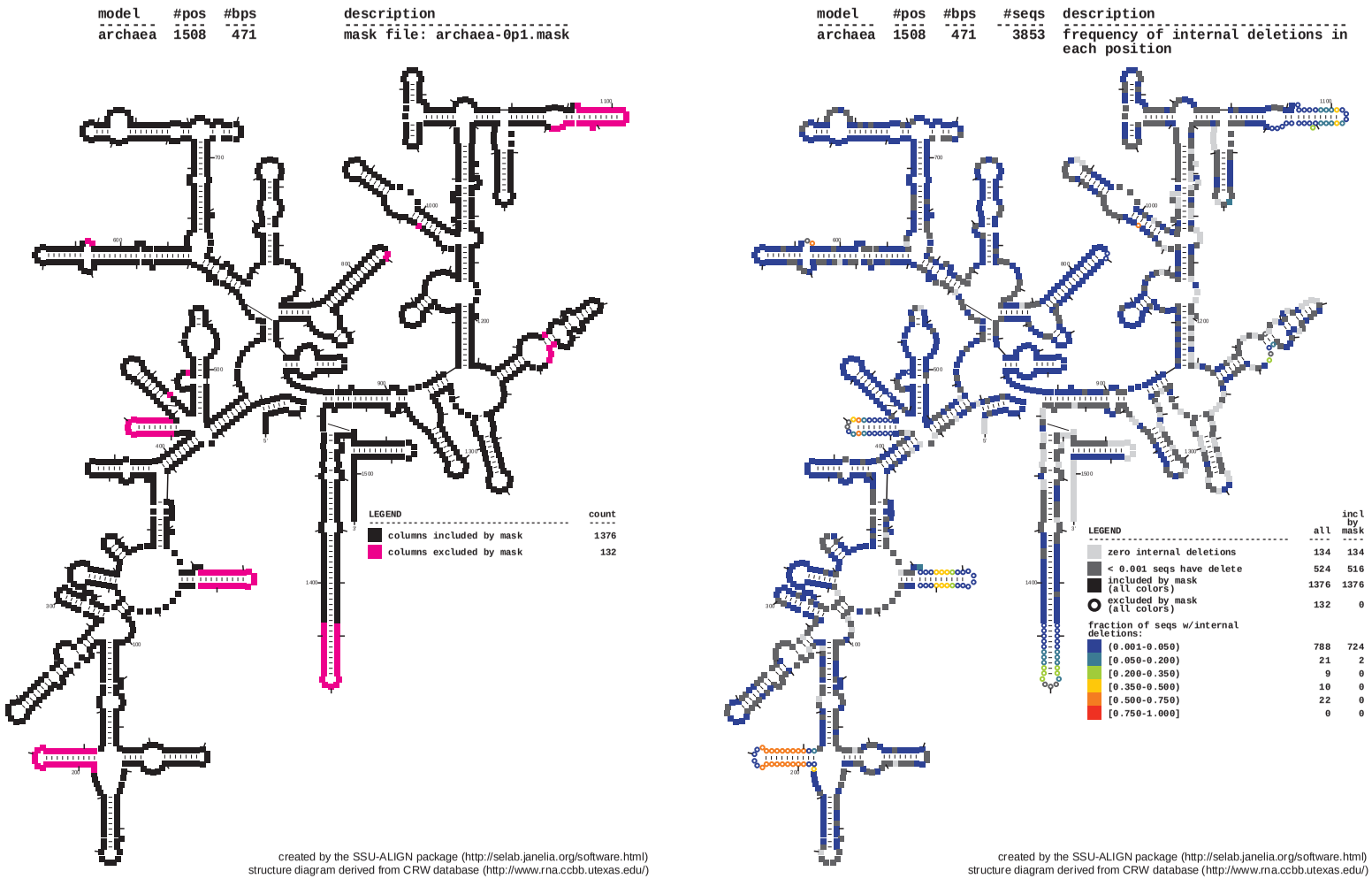

Figure 30: **Two diagrams of the default archaeal mask.** Left: pink positions are excluded, black positions are included. Right: Open circles are excluded, filled squares are included. Positions are colored based on frequency of deletions (gaps) in the filtered alignment of archaea from GREENGENES and SILVA *Ref* (see text) as indicated in the legend.

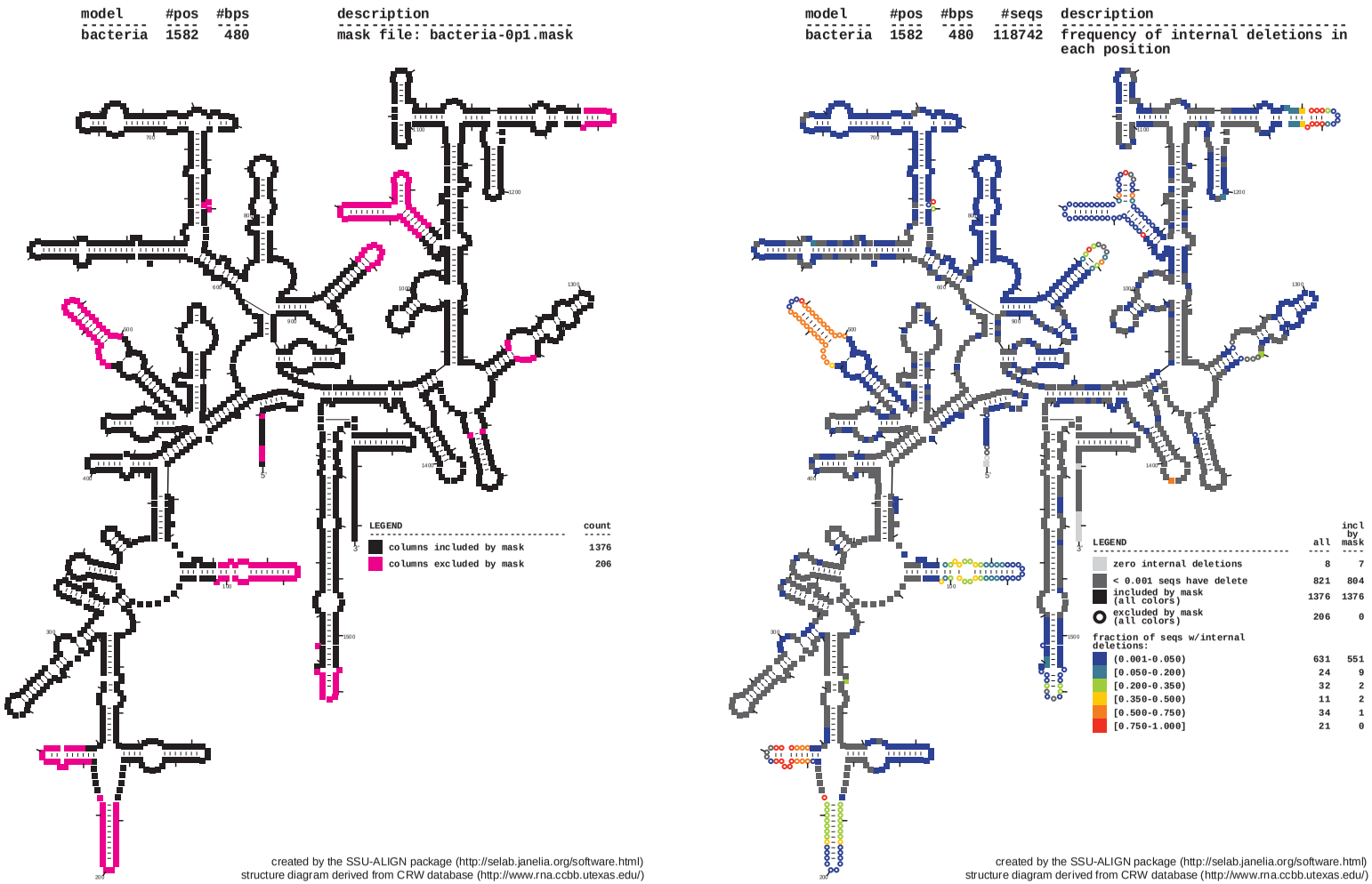

Figure 31: **Two diagrams of the default bacterial mask.** Left: pink positions are excluded, black positions are included. Right: Open circles are excluded, filled squares are included. Positions are colored based on frequency of deletions (gaps) in the filtered alignment of bacteria from GREENGENES and SILVA *Ref* (see text) as indicated in the legend.

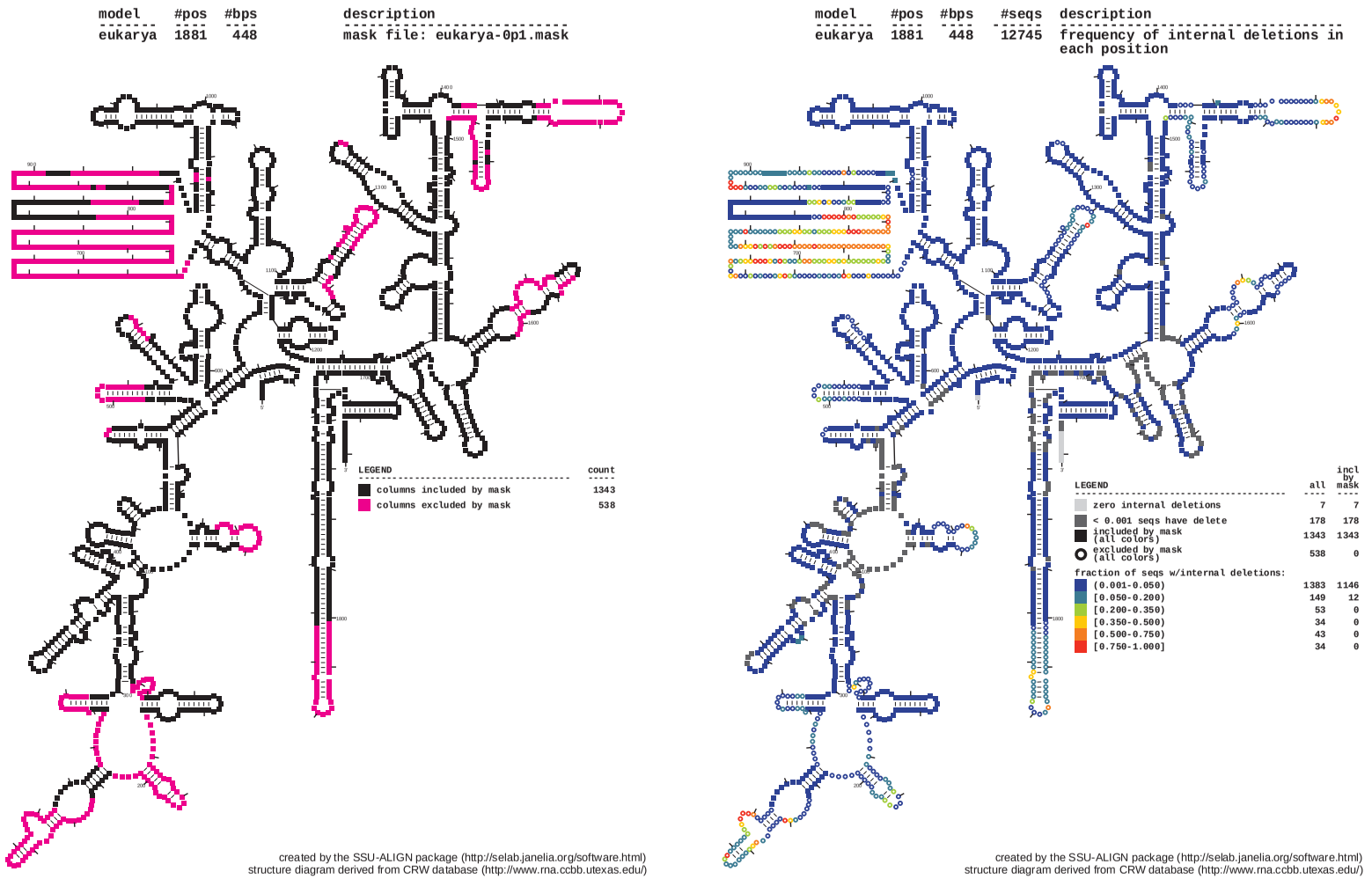

Figure 32: **Two diagrams of the default eukaryotic mask.** Left: pink positions are excluded, black positions are included. Right: Open circles are excluded, filled squares are included. Positions are colored based on frequency of deletions (gaps) in the filtered alignment of eukarya from GREENGENES and SILVA *Ref* (see text) as indicated in the legend.

7 Description of output files

This section describes the content of the different output file formats created by the various programs in the SSU-ALIGN package. For demonstration, the descriptions below refer to the specific files created during the tutorial in section 4, in the directory **myseqs/**. Specifically, we'll go over the files created by the following four commands from the tutorial:

```
> ssu-align seed-15.fa myseqs
> ssu-mask myseqs
> ssu-draw myseqs
```

ssu-align output files

As shown in the tutorial, when executed with the command above, **ssu-align** prints information about the output files it creates to the screen:

```
# Stage 1: Determining SSU start/end positions and best-matching models...
#
# output file name         description
# -----------------------  ------------------------------------------
  myseqs.tab               locations/scores of hits defined by HMM(s)
  myseqs.archaea.hitlist   list of sequences to align with archaea CM
  myseqs.archaea.fa              5 sequences to align with archaea CM
  myseqs.bacteria.hitlist  list of sequences to align with bacteria CM
  myseqs.bacteria.fa             5 sequences to align with bacteria CM
  myseqs.eukarya.hitlist   list of sequences to align with eukarya CM
  myseqs.eukarya.fa              5 sequences to align with eukarya CM
#
# Stage 2: Aligning each sequence to its best-matching model...
#
# output file name         description
# -----------------------  --------------------------------------
  myseqs.archaea.stk       archaea alignment
  myseqs.archaea.cmalign   archaea cmalign output
  myseqs.archaea.ifile     archaea insert info
  myseqs.bacteria.stk      bacteria alignment
  myseqs.bacteria.cmalign  bacteria cmalign output
  myseqs.bacteria.ifile    bacteria insert info
  myseqs.eukarya.stk       eukarya alignment
  myseqs.eukarya.cmalign   eukarya cmalign output
  myseqs.eukarya.ifile     eukarya insert info
  myseqs.scores            list of CM/HMM scores for each sequence
```

I'll walk through the format of these output files:

.tab suffixed files

The **.tab** files are created by INFERNAL's **cmsearch** program which is called internally by **ssu-align**. **cmsearch** builds profile HMMs for each model in the input CM file and scans each input sequence for high scoring alignments to the HMM. The **.tab** file lists the locations and scores of each of these alignments. Let's look at the header of the **myseqs.tab** file created here:

```
# command:    ssu-cmsearch --hmm-cW 1.5 --no-null3 --noalign -T -1 --tab myseqs/myseqs.tab --viterbi /home/nawrocke/share/
# date:       Mon Feb 22 09:35:05 2016
# num seqs:   15
# dbsize(Mb): 0.040518
#
# Pre-search info for CM 1: archaea
#
# rnd  mod  alg  cfg   beta  bit sc cut
# ---  ---  ---  ---  -----  ----------
#   1  hmm  vit  loc      -       -1.00
#
```

The first four lines list the **cmsearch** command executed from within **ssu-align**, the date it was executed, the number of sequences in the sequence file being searched, and the number of millions of nucleotides

(Mb) searched in that sequence file (double the actual size of the sequences because both strands are searched).

Next, the pre-search information describing how the search will be conducted is printed for the first model in the CM file: **archaea**. The **rnd**, **mod**, **alg** and **cfg** columns report that the first and only round of searching will be conducted with a profile **hmm** using the **vit**erbi algorithm for scoring **loc**al alignments to the model. The **beta** column is irrelevant when searching with an HMM. The **bit sc cut** column shows the minimum score threshold, alignments scoring above **-1.00** bits will be reported.

After the pre-search information comes four lines of column headings followed by data lines reporting high scoring local alignments. One line is printed per alignment, or *hit* (it is possible to see more than one hit per sequence). Remember these are scores against the archaea model.

```
# CM: archaea
#                                                                  target coord   query coord
#                                                                ---------------  -----------
# model name  target name                                          start      stop  start   stop    bit sc   E-value  GC%
# ----------  ----------------------------------------------  ----------  ----------  -----  -----  --------  --------  ---
  archaea     00052::Halobacterium_sp.::AE005128                        1        1473      1   1508   2080.08         -   58
  archaea     00052::Halobacterium_sp.::AE005128                     1229        1217      1   1508      1.41         -   54
  archaea     00013::Methanobacterium_formicicum::M36508                1        1476      1   1508   2108.16         -   56
  archaea     00013::Methanobacterium_formicicum::M36508             1227         984      1   1508      1.80         -   59
  archaea     00004::Nanoarchaeum_equitans::AJ318041                    1         865      1   1508   1112.07         -   67
  archaea     00004::Nanoarchaeum_equitans::AJ318041                  619         570      1   1508      1.00         -   64
  archaea     00121::Thermococcus_celer::M21529                       202        1687      1   1508   2223.71         -   66
  archaea     00121::Thermococcus_celer::M21529                      1237        1188      1   1508      1.00         -   62
  archaea     00115::Pyrococcus_furiosus::U20163|g643670              260         309      1   1508      1.00         -   62
  archaea     00115::Pyrococcus_furiosus::U20163|g643670              922           1      1   1508   1354.25         -   66
  archaea     00035::Bacteroides_fragilis::M61006|g143965              60        1533      1   1508    576.22         -   51
  archaea     01106::Bacillus_subtilis::K00637                         55        1548      1   1508    768.61         -   55
  archaea     01106::Bacillus_subtilis::K00637                        684         672      1   1508     -0.63         -   46
  archaea     00072::Chlamydia_trachomatis.::AE001345                 123         853      1   1508    212.41         -   51
  archaea     01351::Mycoplasma_gallisepticum::M22441                 110         120      1   1508      1.43         -   36
  archaea     01351::Mycoplasma_gallisepticum::M22441                 872          69      1   1508     95.30         -   45
  archaea     00224::Rickettsia_prowazekii.::AJ235272                 104        1590      1   1508    613.67         -   51
  archaea     00224::Rickettsia_prowazekii.::AJ235272                 748         739      1   1508      0.79         -   40
  archaea     01223::Audouinella_hermannii.::AF026040                1079        1767      1   1508    222.16         -   54
  archaea     01223::Audouinella_hermannii.::AF026040                 880         871      1   1508      0.79         -   40
  archaea     01240::Batrachospermum_gelatinosum.::AF026045          1077        1761      1   1508    219.14         -   54
  archaea     01240::Batrachospermum_gelatinosum.::AF026045           878         869      1   1508      0.79         -   40
  archaea     00220::Euplotes_aediculatus.::M14590                    710        1081      1   1508     96.64         -   48
  archaea     00220::Euplotes_aediculatus.::M14590                    508         499      1   1508      0.79         -   40
  archaea     00229::Oxytricha_granulifera.::AF164122                 335         344      1   1508      0.79         -   40
  archaea     00229::Oxytricha_granulifera.::AF164122                 138          31      1   1508     86.31         -   53
  archaea     01710::Oryza_sativa.::X00755                           1190        1883      1   1508    202.90         -   53
  archaea     01710::Oryza_sativa.::X00755                            989         980      1   1508      0.79         -   40
```

Here's a brief description of each column:

model name the name of the model used for the search.

target name the name of the target sequence.

start (target coord) first sequence nucleotide in the alignment

end (target coord) final sequence nucleotide in the alignment

Note that under **target coord** some of the sequences' **start** position is greater than their **stop** positions. This occurs if the program has determined the sequence in the target sequence file is a reverse complemented SSU sequence.

The **start** and **stop** columns under **query coord** are uninformative for HMM searches like these. (**cmsearch** will print informative numbers in these columns when the CM is used for a search.) With HMM searches the **begin** column will always be **1** and the **end** column will always be the final position of the model.

bit sc the bit score of the HMM alignment of target sequence nucleotides **begin** to **end**.

E-value this will always read **-**. It would include an E-value of the bit score if the model had been calibrated with **cmcalibrate**. Currently, **ssu-align** does not use calibrated models, mainly because they are most useful for identifying remotely homologous structural RNAs using models of much smaller RNAs, such as transfer RNA. See the INFERNAL user's guide (Nawrocki et al., 2009a) for more information.

GC% the percent of the nucleotides in the aligned sequence that are either **G** or **C**. This is largely irrelevant for SSU rRNA, but is sometimes useful when using **cmsearch** with smaller models.

.hitlist suffixed files

The **.hitlist** files are simple files created for each model that list the sequences that were best-matches to that model, and as a result were aligned to that model in stage 2. If a model has zero sequences for which it is the best-matching model, this file will not be created for that model. The file contains no new information that is not in the **.tab** file and is only created as a convenience, because it is cumbersome to determine which model gave the highest score to a particular sequence in the **.tab** file. For model *x*, the **hitlist** file contains four columns providing four pieces of data for each sequence whose best-matching model was *x*. For example, take a look at the **myseq.archaea.hitlist** file:

```
# List of 5 subsequences to align to CM: archaea
# Created by ssu-align.
#
# target name                                start    stop     score
# ----------------------------------------  ------  ------  --------
  00052::Halobacterium_sp.::AE005128              1    1473   2080.08
  00013::Methanobacterium_formicicum::M36508      1    1476   2108.16
  00004::Nanoarchaeum_equitans::AJ318041          1     865   1112.07
  00121::Thermococcus_celer::M21529             202    1687   2223.71
  00115::Pyrococcus_furiosus::U20163|g643670    922       1   1354.25
```

The four columns correspond to those of the same name in the **.tab** suffixed files as explained above: the **target name**, **start** and **stop** (which correspond to the target sequence coordinates), and **score**, which is the primary sequence HMM score assigned to this subsequence by model *x*: the scores of all other hits from this and other other models are less than this score.

.fa suffixed files

This is a FASTA-formatted sequence file containing the sequences listed in the corresponding **hitlist** file. For example, the file **myseqs/myseqs.archaea.fa** contains the five sequences listed in **myseqs/myseqs.archaea.hitlist**. These sequences were copied from the original sequence file **seed-15.fa** that was used as input to **ssu-align**. Only the nucleotides from positions **start** to **stop** as listed in the **hitlist** file were copied, so sometimes the sequences in the **.fa** file will be subsequences of those from the original file. **ssu-align** uses the **.fa** files it creates as input to the **cmalign** program, which it calls internally to generate its structurally annotated alignments.

.cmalign suffixed files

The **.cmalign** files are the standard output created by the INFERNAL program **cmalign** which is called internally during the alignment stage of **ssu-align**. There is one such file created for each model that was the best-matching model to at least one sequence in **ssu-align**'s search stage. Take a look at the beginning of the file **myseqs.archaea.cmalign**:

```
# command: ssu-cmalign --cm-name archaea --mxsize 4096 --no-null3 --sub --ifile myseqs/myseqs.archaea.ifile
-o myseqs/myseqs.archaea.stk /home/nawrocke/share/ssu-align-0.1.1/ssu-align-0p1.cm myseqs/myseqs.archaea.fa
# date:    Mon Feb 22 09:35:27 2016
#
# cm name                   algorithm  config  sub  bands     tau
# ------------------------  ---------  ------  ---  -----  ------
# archaea                     opt acc  global  yes    hmm   1e-07
```

This section includes the command used to execute **cmalign**, the date of execution, and information on the alignment parameters used by the program.

The **cm name** column reports the name of the model used for alignment. **algorithm** gives the name of the algorithm, in this case **opt acc** stands for *optimal accuracy* (Holmes, 1998) (also known as maximum expected accuracy). This algorithm is similar to the CYK algorithm described in (Nawrocki, 2009), but returns the alignment that maximizes the sum of posterior probability labels on aligned nucleotides instead of the maximally scoring alignment. In practice, for SSU the CYK and optimally accurate alignment are very often identical, and if not, they are nearly identical. The next two columns, **config** and **sub**, read **global** and **yes** respectively, which tells us the program will first predict the start and end points of the alignment to the model using an HMM (the **sub yes** part) and then align the region of the model that spans from start to end *globally* to the sequence (the **global** part). In this case, *global* alignment means that the program is forced to align the full model region from start to end to the sequence (it is *not* allowed to skip large parts of the model without large score penalties as it would if *local* alignment was being performed). The **bands** column tells us that bands (constraints) from an HMM alignment will be used to accelerate alignment to the CM. This is explained more in Chapter 8 of (Nawrocki, 2009). The **tau** column reports the probability loss allowed when computing the HMM bands. In this case, **1e-07** probability mass is allowed outside each band.

The next section includes per-sequence information on the alignment that was created:

```
#                                                          bit scores
#                                                  ------------------
#   seq idx  seq name                                len     total    struct  avg prob      elapsed
# ---------  ----------------------------------------  -----  --------  --------  --------  -----------
          1  00052::Halobacterium_sp.::AE005128          1473   2335.05    290.70     1.000  00:00:01.00
          2  00013::Methanobacterium_formicicum::M36508  1476   2331.84    262.88     0.999  00:00:01.01
          3  00004::Nanoarchaeum_equitans::AJ318041       865   1294.71    170.40     0.999  00:00:00.46
          4  00121::Thermococcus_celer::M21529           1486   2430.39    249.92     0.998  00:00:01.02
          5  00115::Pyrococcus_furiosus::U20163|g643670   922   1496.81    138.96     0.997  00:00:00.51
```

We'll go through each of these columns:

seq idx the index of the sequence in the file.

seq name the name of the sequence.

len length of the sequence; the full sequence hit from the search stage is aligned, no trimming of ends is permitted, as it was in the search stage with **cmsearch**.

total the bit score of the CM alignment. For more information, see section 5 of the INFERNAL User's Guide (Nawrocki et al., 2009a).

struct the secondary structure score component of the **total** bit score. These are the added bits that are due solely to the modeling of the consensus secondary structure of the molecule by the CM.

avg prob the average posterior labeling, or confidence estimate, of the aligned nucleotides. The higher this value is the less ambiguous and more well-defined the alignment is. The highest this can possibly be is **1.000**, which means very nearly 100% of the probability mass of the alignment to the model is contained in the single, optimally accurate alignment that was reported by the program. In other words, the reported alignment receives a significantly higher score than any other alternative alignment. The program derives this value by evaluating the score of every possible alignment (consistent with the HMM bands) of the sequence to the model, and comparing the best, optimal score versus all of the rest.

elapsed the amount of actual time (wall time) it took the program to align this sequence. In general, less well-defined alignments with lower **avg prob** will take longer than more well-defined ones. This is because the HMM bands are usually tighter and act as stricter constraints to the CM alignment when the alignment is well-defined. Tighter bands lead to quicker alignments because fewer possible alignments to the CM must be considered.

.stk suffixed files

The **`.stk`** suffixed files are Stockholm-formatted alignment files. These are the alignments generated by **`cmalign`**. The statistics in the **`.cmalign`** suffixed files correspond to these alignments. One alignment is created for each model that was the best-matching model to at least one sequence in **`ssu-align`**'s search stage. An explanation of Stockholm alignments can be found at the beginning of the tutorial, on page 19.

.scores suffixed files

The **`.scores`** file are meant to be a useful summary file. They contain various statistics from each of the other output files for every sequence in the original input sequence file.

Take a look at the file **`myseqs.scores`**:

```
#                                                                   best-matching model                    second-best model
#                                                   ------------------------------------------------   -------------------
#     idx  sequence name                            model name  beg   end    CM sc   struct   HMM sc   model name   HMM sc  HMMdiff
# -------  ---------------------------------------  ----------  ----  ----  -------  -------  -------  ----------  -------  -------
        1  00052::Halobacterium_sp.::AE005128           archaea       1  1473  2335.05   290.70  2080.08  bacteria     471.45  1608.63
        2  00013::Methanobacterium_formicicum::M36508   archaea       1  1476  2331.84   262.88  2108.16  bacteria     693.10  1415.06
        3  00004::Nanoarchaeum_equitans::AJ318041       archaea       1   865  1294.71   170.40  1112.07  bacteria     255.78   856.29
        4  00121::Thermococcus_celer::M21529            archaea     202  1687  2430.39   249.92  2223.71  bacteria     632.38  1591.33
        5  00115::Pyrococcus_furiosus::U20163|g643670   archaea     922     1  1496.81   138.96  1354.25  bacteria     378.49   975.76
        6  00035::Bacteroides_fragilis::M61006|g143965  bacteria      5  1537  2136.76   382.95  1797.77  archaea      576.22  1221.55
        7  01106::Bacillus_subtilis::K00637             bacteria      1  1552  2465.43   289.74  2212.50  archaea      768.61  1443.89
        8  00072::Chlamydia_trachomatis.::AE001345      bacteria      1   879  1215.64   183.53  1030.27  archaea      212.41   817.86
        9  01351::Mycoplasma_gallisepticum::M22441      bacteria    881     5  1165.86   217.01   940.15  archaea       95.30   844.85
       10  00224::Rickettsia_prowazekii.::AJ235272      bacteria     93  1594  2278.46   281.43  2037.49  archaea      613.67  1423.82
       11  01223::Audouinella_hermannii.::AF026040      eukarya       1  1770  2682.07   221.33  2481.69  archaea      222.16  2259.53
       12  01240::Batrachospermum_gelatinosum.::AF026045 eukarya      1  1764  2665.16   217.20  2463.49  archaea      219.14  2244.35
       13  00220::Euplotes_aediculatus.::M14590         eukarya       1  1082  1213.07    78.14  1103.29  archaea       96.64  1006.65
       14  00229::Oxytricha_granulifera.::AF164122      eukarya     600     1   851.30    23.53   788.12  archaea       86.31   701.81
       15  01710::Oryza_sativa.::X00755                 eukarya      75  1886  2687.06   142.17  2570.56  archaea      202.90  2367.66
```

There are four rows containing column headings prefixed with **`#`**. Then there are 15 data rows, one for each sequence in the input sequence file **`seed-15.fa`**. Data rows are separated into 12 columns:

`idx` the index of the sequence in the file.

`sequence name` the name of the sequence.

The next 6 columns all describe the *best-matching* model for the sequence. This is the model that assigned the highest primary sequence-based local profile HMM alignment score to the sequence. If no model aligned the sequence with a score higher than the minimum threshold of 50 bits then the sequence was skipped and not aligned, and all these columns will read **`-`**. There are no such sequences in the **`seed-15.fa`** file, but if there were, an additional **`.nomatch`** file would have been created, as described below. (Note: the minimum bit score threshold value can be changed to **`<x>`** using the **`ssu-align`** command-line option **`-b <x>`**, as explained in the **`ssu-align`** anual page at the end of this guide).

`model name` name of best-matching model.

`beg` first nucleotide in the maximal scoring local HMM alignment to the best-matching model.

`end` final nucleotide in the maximal scoring local HMM alignment to the best-matching model.

`CM sc` the CM bit score for the best-matching model assigned to the sequence from positions **`beg`** to **`end`**.

`struct` the number of extra bits included in the CM bit score that are dervied from the secondary structure component of the model.

`HMM sc` the HMM bit score for the local alignment of the best-matching model to the sequence from positions **`beg`** to **`end`**.

For more information on bit scores, see section 5 of the INFERNAL User's Guide (Nawrocki et al., 2009a).

The next 2 columns describe the **second-best model**. This is the model that assigned the second-highest primary sequence-based local profile HMM alignment score to the sequence. If only one models score exceeded the minimum of 50 bits then these columns will each read "**-**".

model name name of second-best-matching model.

HMM sc the HMM bit score for the local alignment of the second-best-matching model to the sequence. This alignment was not necessarily from **beg** to **end** (those were the coordinates of the alignment to the best-matching model). The sequence coordinates of the second-best model's alignment can be found in the file **myseqs.tab**.

The final column, **HMMdiff**, reports the score difference between the best-matching model HMM alignment and the second-best matching model HMM alignment. This is included because it is an indication of how clearly "homologous" the sequence is to the best-matching model rather than to the second-best-matching model. The higher this score difference is the more obvious it is that the sequence falls within the sequence diversity represented by the best-matching model. Sequences that are phylogenetically novel and do not obviously match any single model much better than any other one should have relatively small score differences in this column.

.nomatch suffixed files

A **.nomatch** file simply lists the sequences in the input sequence file that do not match to any model, one sequence name per line. For a sequence to match to a model it must score above the minimum profile HMM bit score threshold to at least one model. By default, this threshold is 50 bits, but it can be changed to **<x>** bits with the **-b <x>** option to **ssu-align**. All 15 sequences in the tutorial file **seed-15.fa** score above this threshold to at least one model, so no **.nomatch** file is created.

.sum suffixed files

The **.sum** files include the text reported to the screen (standard output) by **ssu-align**. These files serve as a reference to remind the user how **ssu-align** was run (parameters, input file names, etc.). All six programs in the SSU-ALIGN package generate specifically named **.sum** suffixed files. For example, **ssu-mask** generates a file ending with the suffix **.ssu-mask.sum**.

As a special case, when **ssu-prep** is used to create parallel jobs of **ssu-align**, a **ssu-prep.sum** file is created. Then, when the **ssu-prep**-generated shell script that executes the parallel alignment jobs is run, a **ssu-align.sum** file will be created. This **ssu-align.sum** file will contain the **ssu-align.sum** files from all parallel alignment jobs concatenated together, followed by a section describing the merge performed automatically by the final job and alignment statistics summarizing all parallel jobs.

ssu-align.sum files include more information than the other program's **.sum** files. Specifically, they include two sections labelled **Summary statistics** and **Speed statistics** which summarizes the number of sequences in the input target sequence file that match each model, and how fast the program completed the search and alignment stages, respectively. Below is a short description of each of the fields in the **Summary statistics** section of the **myseqs.ssu-align.sum** file we just created:

```
# Summary statistics:
#
# model or       number  fraction        average   average
# category      of seqs  of total         length  coverage    nucleotides
# ------------  -------  --------  -------------  --------  -------------
  *input*            15    1.0000        1350.60    1.0000          20259
#
  archaea             5    0.3333        1244.40    0.9546           6222
  bacteria            5    0.3333        1268.60    0.9793           6343
  eukarya             5    0.3333        1405.60    0.9675           7028
```

```
#
  *all-models*       15    1.0000        1306.20    0.9671          19593
  *no-models*         0    0.0000              -         -              0
```

model or category the name of the model or category the row pertains to. Categories include ***input***, the set of all full length target sequence file used as input; ***all-models***, the set of sequences that match any model; and ***no-models***, the set of sequences that do not match any model above threshold.

number of seqs number of seqs that belong in a category, or were best-matches to a model.

fraction of total the number of sequences in this model/category divided by the total number of sequences listed in the ***input*** category.

average length the average sequence length of the model/category. For models, this is the average length of the subsequences that survive the search stage, which are not necessarily the full length sequences in the input sequence file.

average coverage length of surviving sequences divided by the full length of those sequences in the input sequence file.

nucleotides summed length of surviving sequences in model/category. This is equal to the product of the values in **number of seqs** and **average length** columns.

Now, take a look at the **Speed statistics** section. Each field in this table is described below:

```
# Speed statistics:
#
# stage      num seqs  seq/sec  seq/sec/model    nucleotides    nt/sec
# ---------  --------  -------  -------------  -------------  --------
  search           15    1.159          0.386          20259    1565.5
  alignment        15    0.972          0.972          19593    1270.0
```

stage the stage of the program this row pertains to, either **search** or **alignment**.

num seqs the number of sequences processed by this stage.

seq/sec the number of sequences processed by this stage per second.

seq/sec/model the number of sequences processed by this stage per second per model. This is the value in the **seq/sec** column divided by the number of models used to search for or align the sequences. Remember that in the **search** stage, all 3 models are used by default, whereas only 1 model is used in the **alignment** stage (figure 1).

nucleotides summed length of all sequences processed by this stage.

nt/sec number of nucleotides processed by this stage per second.

.log suffixed files

The **.log** files include information on all of the system commands run by **ssu-align**. As with **.sum** suffixed files, program specific **.log** suffixed files are created by each of the six SSU-ALIGN programs. For example, **ssu-mask** generates a log file ending with the suffix **.ssu-mask.log**.

As a special case, when **ssu-prep** is used to create parallel jobs of **ssu-align**, a **ssu-prep.log** file is created. Then, when the **ssu-prep**-generated shell script that executes the parallel alignment jobs is run, a **ssu-align.log** file will be created. This **ssu-align.log** file will contain the **ssu-align.log** files from all parallel alignment jobs concatenated together.

The **.log** files list all of the commands that were executed by the program along with their STDERR and sometimes their STDOUT output (for example, the **cmsearch** and **cmalign** commands), as well as the

"globals hash", a set of variables defined in the SSU-ALIGN PERL module **ssu.pm** which is installed in the $SSUALIGNDIR. The **.log** files should mainly be useful to expert users or developers who are trying to figure out why a specific run of a program failed, or how to reproduce a specific step executed by the program (e.g. the **cmsearch** step of **ssu-align**).

ssu-mask output files

The **ssu-mask** example from the tutorial is executed with:

```
> ssu-mask myseqs
```

This command generates three output files per-domain, each of which is briefly explained below. These files are also discussed in the Tutorial section.

.mask suffixed files

Mask files are simple one-line text files that correspond to a specific domain model/alignment file and include only **0** and **1** characters. Each character corresponds to a consensus column of the model/alignment. For example, the **myseqs.archaea.mask** file contains 1508 characters, one per consensus column of the default SSU-ALIGN archaeal model described in section 5. A **0** at position *x* indicates that consensus position *x* is excluded from the mask and will be removed from the alignment when the mask is applied. A **1** at position *x* indicates that it is included by the mask and will be kept when the mask is applied. A consensus position of an alignment is one where the **#=GC RF** annotation is a nongap (see section 3 for more detail).

.mask.pdf and .mask.ps suffixed files

These are structure diagram files that display a mask on the consensus secondary structure of a domain. Pink positions are excluded by the mask. Black positions are included by the mask. Either **.pdf** or postscript **.ps** files will be created, but not both. If you have the program **ps2pdf** in your path, **.pdf** files will be created, otherwise postscript will be. One difference between these file types is their sizes, the PDF files created by **ssu-mask** are only about 10% the size of the postscript files.

.mask.stk suffixed files

The **.mask.stk** files are Stockholm-formatted alignment files that include a subset of the columns of the alignments created by **ssu-align**. The consensus columns excluded by the mask have been removed. Additionally, all insert columns have been removed. An insert column is one which is a gap in the reference (**#=GC RF**) annotation of the alignment. For more information on inserts, see section 3.

.list suffixed files

.list suffixed files are only generated if **ssu-mask** is executed using the **--list** option. A list file is generated for a specific alignment file, and it simply lists the name of each sequence in the alignment, one per line.

.afa suffixed files

.afa suffixed files are FASTA-formatted alignment files that are created by **ssu-mask** only if the **--stk2afa** option is used. These files differ from unaligned FASTA files in that each sequence may contain gaps, and all sequences are guaranteed to be the same length. **.afa** files created by **ssu-mask** contain two types of gap characters. A **-** character is a gap in a consensus column of an alignment. A **.** character is a gap in an insert column of an alignment. See section 3 for more information on the distinction between consensus

and insert columns. Note that **.afa** alignment files do not include structure information that is included in Stockholm alignment files.

ssu-draw output files

The **ssu-draw** example from the tutorial is executed with:

```
> ssu-draw myseqs
```

This command generates two output files per-domain, each of which is briefly explained below.

.pdf or .ps suffixed files

These are structure diagram files that display either alignment summary statistics (such as information content or frequency of gaps) or individual aligned sequences on the consensus secondary structure for a domain. They may be multiple pages. A brief description is included at the top of each page. Along with each of these files, a corresponding **.drawtab** file is generated, as described below. Either **.pdf** or postscript **.ps** files will be created, but not both. If you have the program **ps2pdf** in your path, **.pdf** files will be created, otherwise postscript will be. One difference between these file types is their sizes, the PDF files created by **ssu-draw** are only about 10% the size of the postscript files.

.drawtab suffixed files

.drawtab suffixed files are tab-delimited text files meant to be easily parseable that contain the information displayed in a corresponding **.pdf** or **.ps** structure diagram file. In these files, lines beginning with **#** are comment lines. Comments at the beginning of **.drawtab** files explain the meaning of each of the tab-delimited fields in the non-comment lines.

8 Running times and output file sizes for example datasets

This section contains timing and output file size statistics for aligning, masking and drawing SSU sequences with the SSU-ALIGN package to give you an idea of how much time and disk space you'll need for processing various sized datasets. The statistics are displayed in tables 4, 5, and 6 on the next two pages. The timing statistics reported are for single execution threads running on 3.0 Intel Xeon processors. All programs were run with default parameters (no options), with the exception that some of the *synthetic-v4* datasets were processed with a truncated set of models as explained further below.

Statistics are reported for real and synthetic datasets. The real datasets were obtained exclusively from existing SSU databases: GREENGENES (DeSantis et al., 2006b), SILVA (Pruesse et al., 2007) and RDP (Cole et al., 2009). For GREENGENES, two separate datasets were processed: the "core set", which is the reference alignment used by that database (*Ggenes-core* dataset), and all the SSU sequences in the database (*Ggenes-full* dataset) after the March 23, 2010 update. Two datasets from SILVA, *Silva-Ref* and *Silva-Parc*, were processed. dataset The Ref set is a subset of the Parc set that includes sequences that are generally longer and of higher quality than the other sequences in Parc. Both sets of sequences are from release 102 of the database. Finally, a single RDP dataset was analyzed: the full set of sequences in release 10, update 20. The *Silva-Parc* and the *RDP* datasets contain over 1 million sequences.

The synthetic datasets were fabricated using the INFERNAL package. Specifically, the **cmemit** program was used with default settings and the **-n <N>** option to generate **<N>** sequences, where **<N>** is reported in the *number of seqs* columns of the tables. For the *synthetic-full* datasets, the default three SSU-ALIGN SSU CMs were used to generate the sequences, each contributing one third of the total sequences. For the *synthetic-v4* datasets, smaller, truncated versions of the default archaeal and bacterial models were used to generate the sequences, half each. These CMs only model the V4 hypervariable region of SSU. They were constructed using the same **ssu-build** commands from the tutorial section (section 4, page 33).

For all datasets the default models were used to align, mask and draw sequences using **ssu-align**, **ssu-mask** and **ssu-draw**. These runs with default settings are summarized in the rows of the tables that include *default* in the *models* column. Additionally, for the *synthetic-v4* dataset, the smaller V4 models were used for aligning and masking. Drawing was impossible for these runs because non-default models were used. These runs correspond to the rows in the tables marked *v4* under *model*.

For all datasets with more than 10,000 sequences **ssu-prep** was used to parallelize alignment. These datasets were split up into 100 separate jobs and run in parallel on a cluster. This is indicated by the *# of procs* column in table 5. For these datasets, additional columns in table 5 report the time required to run **ssu-prep** and **ssu-merge** (which is automatically executed by the final **ssu-align** job), and the elapsed or *wall* time required to perform all parallel alignment jobs plus the final merge.

As an example of a non-parallel run, the actual commands used to process the GREENGENES core set dataset were as follows. The FASTA file included the unaligned core set sequences is named **ggcs-unaln.fa**. The following three commands would align, mask and draw the core set, respectively. All output files will be created in a directory called **ggcs**. Importantly, the three commands must be run in succession (masking cannot begin until alignment is complete).

```
> ssu-align ggcs-unaln.fa ggcs
> ssu-mask ggcs
> ssu-draw ggcs
```

As an example of a parallel run, the actual commands used to process the GREENGENES full dataset were as follows. The FASTA file included the unaligned core set sequences is named **gg-unaln.fa**. The following four commands would partition, align, mask and draw the full dataset, respectively. All output files will be created in a directory called **gg**. Importantly, the four commands must be run in succession (masking cannot begin until alignment is complete).

```
> ssu-prep gg-unaln.fa gg 100 janelia-cluster-presuf.txt
> sh gg.ssu-align.sh
> ssu-mask gg
> ssu-draw gg
```

The **ssu-prep** command above creates the **gg.ssu-align.sh** script that will submit 100 **ssu-align** jobs to the cluster here at Janelia. The Janelia-specific prefixes and suffixes for the commands are in the file **janelia-cluster-presuf.txt** which is in the **tutorial/** subdirectory of SSU-ALIGN. See the tutorial section 4, page 29 for more details on **ssu-prep**.

The statistics in tables 4, 5 and 6 highlight some key points:

- Deep alignments are mostly composed of insert columns which dramatically inflate the size of the alignment files (table 4). All insert columns will be removed automatically by **ssu-mask** (see section 3 for more discussion of insert versus consensus columns). The full *RDP* bacterial alignment of about 1.2 million sequences is 70 Gb and contains 29,736 total columns, 28,154 of which are inserts (about 95%). The masked alignment of 1397 columns is about 3.5 Gb. Note that insert columns will not be included in output alignments of **ssu-align** when the **--rfonly** option is used.
- **ssu-align** creates alignments of SSU sequences at a rate of roughly 1 sequence/second (table 6).
- Short ($\sim$240 nt) V4 subsequences are aligned at about 10 sequences per second with the default, full-length models (table 6).
- Aligning short sequences with truncated models is significantly faster than with full models. V4 subsequences are aligned at about 100 sequences per second with V4-specific models (table 6).
- Aligning 1 million typical SSU sequences requires about 300 CPU hours. This can be parallelized on a cluster of 100 processors in about 3.5 hours. Note that for about 25 of the final 30 minutes only one processor is still running, this is the processor that is performing the merge of the 100 per-job alignments. (table 5).

dataset	models	domain	# of sequences	average sequence length	total # of columns	# of insert columns	# of consensus columns	# columns included by mask	alignment file size (Mb)	masked alignment file size (Mb)
Ggenes-core	default	archaea	186	1407.5	1757	249	1508	1380	0.7	0.5
	default	bacteria	4752	1438.1	5319	3737	1582	1362	50.7	13.1
Ggenes-full	default	archaea	8407	1365.0	3151	1643	1508	1382	53.3	23.5
	default	bacteria	499787	1400.3	11900	10318	1582	1404	11910.9	1419.4
Silva-Ref	default	archaea	19260	1169.0	12107	10599	1508	1383	469.2	56.1
	default	bacteria	391158	1413.0	24540	22958	1582	1396	19254.6	1148.6
	default	eukarya	50350	1727.3	20950	19069	1881	1418	2117.0	150.1
Silva-Parc	default	archaea	56113	862.1	14299	12791	1508	1383	1613.0	163.0
	default	bacteria	1062938	1016.4	27918	26336	1582	1391	59498.1	3105.0
	default	eukarya	123454	1092.6	24247	22366	1881	1424	6003.9	368.6
RDP	default	archaea	55249	830.9	12724	11216	1508	1382	1411.7	158.4
	default	bacteria	1182624	975.1	29736	28154	1582	1397	70455.3	3426.5

Table 4: **Statistics and resulting output file sizes for various SSU datasets.** For each dataset, the number of and average length of sequences for each domain, as classified by **ssu-align**, is reported. For each dataset and domain an alignment is created. The four columns in the middle block report the total number of columns, the number of insert and consensus columns (consensus column counts are invariant per domain across all datasets), and the number of consensus columns remaining (included) after masking with **ssu-mask**. The column second from the right reports the size in megabytes (Mb) of each full alignment file that is output by **ssu-align**. These alignments include insert and consensus columns for all sequences. The rightmost column reports the size in Mb of the masked alignment output from **ssu-mask**. These alignments include a subset of the consensus columns and zero insert columns. See section 3 for more information on consensus versus insert columns.

dataset	models	# of seqs	average seq length	running time (hh:mm:ss) ssu-prep	ssu-align	ssu-mask	ssu-draw	ssu-merge	parallel ssu-align statistics # of procs	wall time
Ggenes-core	default	4938	1437.0	-	1:47:48	0:00:04	0:00:05	-	1	-
Ggenes-full	default	508194	1399.7	0:01:50	183:04:18	0:07:51	0:08:49	0:06:12	100	2:01:08
Silva-Ref	default	460783	1437.1	0:01:18	175:07:23	0:13:13	0:15:25	0:09:03	100	1:58:50
Silva-Parc	default	1246462	1017.0	0:02:28	315:20:54	1:07:07	1:05:40	0:24:27	100	3:47:06
RDP	default	1237963	968.7	0:02:27	293:15:32	1:24:13	1:13:49	0:25:05	100	3:38:54
synthetic-full	default	100	1603.7	-	0:02:39	0:00:01	0:00:02	-	1	-
synthetic-full	default	1000	1606.9	-	0:26:12	0:00:01	0:00:02	-	1	-
synthetic-full	default	10000	1607.0	-	4:17:36	0:00:05	0:00:06	-	1	-
synthetic-full	default	100000	1606.9	0:00:51	44:56:37	0:00:57	0:01:14	0:01:08	100	0:30:22
synthetic-full	default	1000000	1607.1	0:02:00	450:15:46	0:12:06	0:13:35	0:12:12	100	5:06:52
synthetic-v4	default	100	239.2	-	0:00:37	0:00:01	0:00:02	-	1	-
synthetic-v4	default	1000	239.2	-	0:02:44	0:00:01	0:00:02	-	1	-
synthetic-v4	default	10000	239.2	-	0:26:36	0:00:03	0:00:04	-	1	-
synthetic-v4	default	100000	239.2	0:01:02	4:37:51	0:00:07	0:00:23	0:00:27	100	0:03:17
synthetic-v4	default	1000000	239.2	0:01:37	45:14:17	0:03:28	0:04:27	0:03:57	100	0:31:46
synthetic-v4	v4	100	239.5	-	0:00:06	0:00:01	-	-	1	-
synthetic-v4	v4	1000	239.6	-	0:00:34	0:00:01	-	-	1	-
synthetic-v4	v4	10000	239.6	-	0:05:31	0:00:01	-	-	1	-
synthetic-v4	v4	100000	239.6	0:00:08	1:03:58	0:00:15	-	0:00:14	100	0:00:55
synthetic-v4	v4	1000000	239.6	0:00:23	9:42:32	0:02:60	-	0:01:57	100	0:07:60

Table 5: **Statistics and running times for various SSU datasets.** For each dataset, the number of and average length of all sequences is reported. The running time of each program: **`ssu-prep`**, **`ssu-align`**, **`ssu-mask`**, **`ssu-draw`**, and **`ssu-merge`** is reported in the middle block. For datasets with more than 10,000 sequences, **`ssu-prep`** was used to create 100 parallel **`ssu-align`** jobs. For these datasets: the *ssu-align* column under *running time* reports the summed time for all 100 jobs minus the time required to merge the results; *wall time* reports the actual wall time required for the 100 jobs to run and be merged together. All running times are given in *hh:mm:ss* format; *hh*=hours, *mm*=minutes, *ss*=seconds. All programs were run as single execution threads on 3.0 GHz Intel Xeon processors.

dataset	models	# of sequences	average sequence length	ssu-align running time (hh:mm:ss) total	stage 1 (search) per seq	stage 2 (alignment) per seq	both stages per seq
Ggenes-core	default	4938	1437.0	1:47:48	0:00:00.64	0:00:00.67	0:00:01.31
Ggenes-full	default	508194	1399.7	183:04:18	0:00:00.63	0:00:00.67	0:00:01.30
Silva-Ref	default	460783	1437.1	175:07:23	0:00:00.66	0:00:00.71	0:00:01.37
Silva-Parc	default	1246462	1017.0	315:20:54	0:00:00.46	0:00:00.45	0:00:00.91
RDP	default	1237963	968.7	293:15:32	0:00:00.43	0:00:00.42	0:00:00.85
synthetic-full	default	1000000	1607.1	450:15:46	0:00:00.78	0:00:00.84	0:00:01.62
synthetic-v4	default	1000000	239.2	45:14:17	0:00:00.09	0:00:00.07	0:00:00.16
synthetic-v4	v4	1000000	239.6	9:42:32	0:00:00.01	0:00:00.02	0:00:00.03

Table 6: **Statistics and `ssu-align` per-sequence running times for various SSU datasets.** For each dataset, the number of and average length of all sequences is reported. The running time of **`ssu-align`** is reported in the *total* column. **`ssu-align`** proceeds through two stages, the search stage first determines the domain of each sequence and the alignment stage structurally aligns each sequence to a domain specific model. The *per seq* columns report the average time per sequence for the search stage, alignment stage, and for the entire program execution (*both stages per seq* column). All running times are given in *hh:mm:ss* format; *hh*=hours, *mm*=minutes, *ss*=seconds. All programs were run as single execution threads on 3.0 GHz Intel Xeon processors.

9 Manual pages

SSU-ALIGN(package) - alignment, masking and visualization of SSU rRNA

Synopsis

ssu-align Align small subunit ribosomal RNA (16S/18S SSU rRNA) sequences

ssu-build Build SSU rRNA covariance models from multiple sequence alignment(s)

ssu-draw Draw secondary structure diagrams of SSU rRNA

ssu-mask Mask (remove columns from) SSU rRNA multiple sequence alignments

ssu-merge Merge SSU rRNA alignments created by parallel ssu-align jobs

ssu-prep Prepare SSU rRNA sequences for parallel ssu-align jobs

Description

SSU-ALIGN is a suite of several programs for creating, masking and visualizing small subunit ribosomal RNA (SSU rRNA) multiple sequence alignments. It uses profile probabilistic models called ”profile hidden Markov models” (profile HMMs) and ”covariance models” (CMs) that represent archaeal 16S SSU rRNA, bacterial 16S SSU rRNA and eukaryotic 18S SSU rRNA. Each of these profile models was built from a reference alignment derived from Robin Gutell's Comparative RNA Website database (CRW) (Cannone et al., BMC Bionformatics. 2002; 3:2.) as described in the user's guide. The alignments created by SSU-ALIGN explicitly take into account the conserved secondary structure of SSU as annotated by CRW. SSU-ALIGN output alignments are created at a rate of roughly one sequence per second for full length SSU sequences.

The main program, **ssu-align** takes a FASTA-formatted sequence file containing SSU sequences, classifies them by domain of life (i.e. archaea, bacteria, or eukarya) and creates structurally annotated domain-specific alignments.

ssu-mask removes columns from (masks) alignments created by **ssu-align** that are likely to contain alignment errors. The set of columns to remove is determined based on per-nucleotide confidence estimates (posterior probabilities, annotated in 'PP' lines) contained within the alignments as calculated by **ssu-align.** Alternatively, preset masks can be applied to remove a consistent set of columns to allow comparison across multiple datasets. **ssu-mask** can also be used to convert alignments to aligned FASTA and to remove a subset of sequences from an alignment.

The **ssu-draw** program generates secondary structure diagrams of SSU in postscript or PDF format from **ssu-align** generated alignments. (PDF is possible only if a postscript to pdf converter like 'ps2pdf' is in your PATH). Diagrams of individual aligned sequences or alignment summary statistics can be generated.

The **ssu-prep** program can be used for splitting up large sequence files into many smaller ones to be aligned separately in parallel on a cluster or multi-core machine. It outputs a shell script that will execute the set of parallel **ssu-align** jobs and then automatically merge their results. The final alignments will be identical to what would have been created by a single **ssu-align** job that processed the full original sequence file.

ssu-merge merges the results of a set of parallel **ssu-align** jobs created by **ssu-prep.** It is automatically called by the final **ssu-align** job of the set, and so should only be necessary to be called manually on the command-line in the event that one or more of the jobs fail. If this happens, only the failed jobs need be rerun, and then the full set can be merged by executing **ssu-merge.**

Finally, **ssu-build** constructs new profile models (CMs) from input structurally annotated alignments. These can either be full length or partial SSU models. For example, models that represent only hypervariable region V4 can be constructed for aligning datasets from a PCR-based study that targetted the V4 region.

SSU-ALIGN is designed to be able to create and manipulate very large alignments (up to millions of sequences). The main limitation for large alignments is the time required to create them (about 1 sequence/second/cpu). The memory requirements of SSU-ALIGN are effectively independent of the number of sequences in the alignments.

Each of the six programs has its own man page.

The SSU-ALIGN programs internally execute programs from the INFERNAL package and the Easel sequence analysis library. INFERNAL is developed by Eric Nawrocki, Diana Kolbe, and Sean Eddy. Easel is developed by Sean Eddy. The INFERNAL and Easel programs installed with SSU-ALIGN include a *ssu-* prefix to distinguish them from separate installations of INFERNAL and/or Easel that may be on your system. For example, the **cmsearch** INFERNAL program used by SSU-ALIGN is called **ssu-cmsearch.** Each of these programs has its own man page as well.

`ssu-align` - align small subunit ribosomal RNA (16s/18s SSU rRNA) sequences

Synopsis

ssu-align *[options] seqfile output-dir*

Description

ssu-align identifies archaeal, bacterial and eukaryotic SSU rRNA sequences in the target sequence file *seqfile* and aligns them using CMs to create domain-specific SSU multiple sequence alignments.

The *seqfile* must be in FASTA format. The sequences within the file should each contain 0 or 1 SSU or partial SSU sequences. Only the top-scoring SSU sequence within each sequence will be aligned, so the *seqfile* should not contain long sequences like a bacterial genome or eukaryotic chromosomes, because only one SSU sequence per target sequence will be identified and aligned. The SSU sequence can be on either strand of a sequence (be part of either the actual sequence in the file, or its reverse complement).

CMs are probabilistic profile models of RNA consensus sequence and secondary structure. CM files can include one or more models. The default CM file, **ssu-align-0p1.cm,** will be used by **ssu-align** unless the **-m** <*s*> option is set to specify that CM file <*s*> be used instead. The default CM file contains three CMs: an archaeal 16S SSU model, a bacterial 16S SSU model and a eukaryotic 18S SSU model. These models were built from structural alignments from Robin Gutell's Comparative RNA Website database (CRW) (Cannone et al., BMC Bionformatics. 2002; 3:2.) as described in the SSU-ALIGN user's guide. Other CM files can be created using **ssu-build.**

ssu-align proceeds through two stages to create alignments. In the first stage, the best-matching model and the begin and end positions of the SSU (sub)sequence are defined for each target sequence. This is achieved by searching each target with *n* HMMs, one for each of the *n* models in the CM file to find the highest scoring SSU hit: the subsequence defined by a begin and end position that has the highest bit score to each HMM. In this stage, only primary sequence conservation contributes to the score. The model that finds the highest scoring hit for each target sequence is that sequence's 'best-matching' model. If the best-matching model's hit has a score greater than a minimum bit score (by default 50 bits, but changeable to <*x*> with **-b** <*x*>) and a length greater than a minimum length (by default 1 nucleotide, but changeable to <*n*> with **-l** <*n*>), the target subsequence survives to stage 2.

In stage 2, each surviving target subsequence defined by the begin and end positions of the highest scoring hit is extracted (that is, suspected non-SSU nucleotides at the begin and ends may be trimmed off) and aligned to its best-matching model. At this stage, both primary sequence and well-nested secondary structure conservation of the SSU model are taken into account during alignment.

ssu-align is a PERL script that executes C programs from the INFERNAL package (bundled within the SSU-ALIGN package), such as **ssu-cmalign** and **ssu-cmsearch,** as well as C executable 'miniapps' from the EASEL sequence analysis library package, such as **ssu-esl-sfetch.** All necessary INFERNAL and EASEL programs are installed with the SSU-ALIGN package and are prefixed with 'ssu-' to differentiate them from programs from INFERNAL installations on your machine. For example, the executable program **cmsearch** is actually **ssu-cmsearch.**

ssu-align creates a directory called *output-dir* into which it places all of its output files. Several different types of output files are created including alignments and unaligned hits for each model that is the best-matching model for at least one sequence. As each file is created, its name and a very brief description is printed as part of **ssu-align's** standard output. The file types are described in more detail in the user guide. All output file names begin with *output-dir* followed by a '.'. The file <output-dir>.ssu-align.sum includes the standard output of the program. The file <output-dir>.ssu-align.log includes all of the commands (with options) executed by **ssu-align,** such as **ssu-cmsearch** and **ssu-cmalign** commands.

Options

-h Print brief help; includes version number and summary of all options.

-f Allow output directory *output-dir* to be deleted and overwritten. Without this option, if the *output-dir* directory already exists, **ssu-align** exits with an error.

-m <*f*> Use CM file <*f*> instead of the default three model (archaea, bacteria, eukarya) SSU-ALIGN 0.1 CM file.

-b <*x*> Set the minimum bit score threshold for stage 1 survival of an HMM hit as <*x*>. Only hits that meet or exceed this threshold will be aligned in stage 2. By default, <*x*> is 50 bits.

-l <*n*> Set the minimum length threshold for stage 1 survival of an HMM hit as <*x*>. Only hits that meet or exceed this threshold will be aligned in stage 2 so this is the minimum sequence length allowed in any of the output alignments. By default, <*n*> is 1. Note that each hit still must exceed the minimum bit score threshold.

-i Output alignments as interleaved Stockholm format in which each sequence is split up across several lines, instead of the default Pfam Stockholm format in which each sequence occurs on exactly one line. Importantly, if **-i** is set for **ssu-align,** subsequent calls of **ssu-mask** and/or **ssu-draw** on the same *output-dir* must also include **-i.**

-n <*s*> Only search with and align to a single CM named <*s*> from the CM file. Unless **-m** is used, the default CM file will be used and so valid strings for <*s*> are only 'archaea', 'bacteria' and 'eukarya'. A related option is **--aln-one** (see below).

Options for Controlling Output

--dna Output alignments and sequence files as DNA, not as RNA. By default, the output alignments will be RNA, even if the input sequences in *seqfile* were DNA.

--rfonly Discard insert columns and only include consensus columns in the output alignments. The alignments will be a fixed, predicted width (the number of consensus positions for each model), but will not include all target nucleotides. Note that insert columns are automatically removed by the **ssu-mask** program which is recommended prior to inputting alignments into phylogenetic inference programs and using **--rfonly** can greatly reduce the size of the alignments, especially for very deep alignments (hundreds of thousands of sequences).

Options for Skipping the Search or Alignment Stages

--no-align Only perform the search stage (stage 1). Determine the best-matching model, define the likely begin and end positions for each SSU (sub)sequence, and extract those subsequences from *seqfile.* Do not align each extracted sequence to its best matching model.

--no-search Skip the stage 1 search and only perform the alignment stage. This option should only be used if it is known that each full length target sequence in *seqfile* is a SSU sequence or subsequence that matches to the CM in the CM file. This is only allowed if the CM file contains exactly 1 CM or if the **-n** <*s*> option is used to specify a single CM to use from a multi-CM file.

Expert Options for Tuning the Search Stage:

--toponly Only search the top strand of each sequence in stage 1. This option should only be used if the user is confident no sequences in *seqfile* contain SSU sequences on the opposite strand.

--forward In the stage 1 search, use the HMM Forward algorithm instead of the HMM Viterbi algorithm. Forward is about three times slower than Viterbi, but might be more sensitive in some situations.

--global Require hits in the stage 1 search to globally match the model, i.e. be full length. By default, hits can locally match the model, which allows partial SSU sequences to be found and aligned.

--keep-int Keep some files that are normally removed by the program, including the **ssu-cmsearch** output from stage 1 and the input to the **ssu-esl-sfetch** program which is used to extract search hits prior to alignment.

Expert Options for Tuning the Alignment Stage:

--no-trunc Following the search stage, do not truncate sequences to the begin and end positions of the best-matching HMM hits. Instead, align the full target sequences in stage 2.

--aln-one <*s*> Only create alignments to the CM named <*s*>. All CMs will still be used in the search stage. If the default CM file is used (i.e. if **-m** is not set), <*s*> can be 'archaea', 'bacteria', or 'eukarya'. This option might be useful if the user wants to classify archaeal, bacterial, and eukaryotic sequences but only produce bacterial alignments, for example. If the related **-n** option is used for the same case, the user may find the bacterial alignments include archaeal and eukaryotic sequences, as well as bacterial ones.

--no-prob Do not include per-nucleotide posterior probabilities that estimate alignment confidence in the output alignments. These probabilities are included by default and are used by **ssu-mask** to determine unreliably aligned regions and remove them.

--mxsize <*x*> Set the maximum allowable dynamic programming matrix size used for alignment to <*x*> megabytes. By default this size is 4,096 Mb. This should be large enough for the vast majority of alignments, however if it is not, **ssu-align** will exit prematurely and report an error message that the matrix exceeded it's maximum allowable size. In this case, the **--mxsize** can be used to raise the limit.

`ssu-build` - build SSU rRNA covariance models

Synopsis

Build models using a user-created Stockholm alignment:

ssu-build *[options] alignment-file*

Build models using default SSU-ALIGN seed alignment:

ssu-build -d *[options] seed-name*

Description

ssu-build reads a multiple alignment of SSU rRNA sequences from *alignment-file,* constructs a covariance model and saves the CM to a file.

The alignment file must be in Stockholm format, and must contain consensus secondary structure annotation.

Alternatively, with the **-d** option, **ssu-build** uses the default SSU-ALIGN seed alignment for family *seed-name* as the input alignment. Possible choices for *seed-name* are *archaea, bacteria* and *eukarya.* The **-d** option is expected to mainly be useful in combination with the **--trunc** <*s*> option, which specifies that a subset of the input alignment be used to construct a non-full length model of a SSU rRNA, such as within positions spanned by a specific set of primers designed to target a variable region. (See the description of the **--trunc** <*s*> option below for details on its usage.)

When the **-d** option is used in combination with the **--trunc** <*s*> option, **ssu-build** will create secondary structure diagrams that highlight the specific SSU rRNA region that the newly constructed model represents.

The **--trunc** <*s*> option requires the user to know the beginning and ending *consensus* position of the desired region. Running **ssu-build** with the *--num* option will cause it to reformat the input alignment such that its consensus columns are numbered to make it easier to determine the appropriate column range. Additionally, the secondary structure diagrams of the default seed models included in the User's Guide have consensus positions of the default models numbered which may help determine column ranges when **-d** is being used in combination with **--trunc.**

ssu-build uses the program **cmbuild** from the INFERNAL package to construct CMs. Interested and motivated users can use **cmbuild** directly to create models. The version installed with SSU-ALIGN is actually called **ssu-cmbuild** to distinguish it from any versions of the program distributed with INFERNAL on your system.

Options

-h Print brief help; includes version number and summary of all options.

-d Specify that the command-line argument is the name of a SSU-ALIGN 0.1 seed alignment, either 'archaea', 'bacteria', or 'eukarya'.

-f Allow overwriting of the CM file.

-o <*s*> Specify that the CM file be named <*s*>. Without this option, **ssu-build** will automatically name the CM file based on the input alignment file name and command-line options that are used.

-n <*s*> Specify that the CM itself be named <*s*>. Without this option, **ssu-build** will automatically name the CM based on the input alignment file name and command-line options that are used. The name of the CM is important because it will be included as part of all output file names for files created by other SSU-ALIGN scripts that use this CM.

--append <*s*> Append this CM to CM file <*s*> that already exists. This allows creation of multi-CM files that can be used for discriminating between SSU sequences from different phylogenetic clades, such as the default SSU-ALIGN 0.1 CM file, which contains 3 CMs, one for each of the three domains of life.

--key-out <*s*> Include the string <*s*> as part of all output file names from **ssu-build,** immediately before the suffix. For example, foo.<s>.cm would be created instead of foo.cm.

Options for Building Models from a Truncated Version of the Alignment:

--trunc <*s*> Specify the range of alignment consensus columns to use for building the model. Without this option, the full alignment will be used. The <*s*> string must not contain any whitespace and must be formatted as <*start*>-<*end*> where <start> is the first consensus position to include and <end> is the final one, separated by a single '-' character. This option can be useful for creating a model that matches a specific region of the SSU molecule. For example, running **ssu-build -d --trunc 525-765 archaea** will build a CM that matches the so-called hypervariable V4 region of the archaeal SSU molecule. See the **--num** option for how to create alignments with consensus columns numbered to facilitate choosing start and end positions.

Options for Reformatting Input Alignments

--num Do not build a CM. Instead, create a new alignment that is identical to the input alignment (or default seed alignment if **-d)** but that has extra annotation numbering consensus positions. This option is meant to be helpful for determining start and end positions to use with the **--trunc** option.

-i Output alignments in interleaved Stockholm format of a fixed width that may be more convenient for reading. Without this option, output alignments will contain one line per sequence, and thus may contain very long lines.

Options for Defining Consensus Columns

--rf Use reference coordinate annotation (#=GC RF line in the input Stockholm alignment) to determine which columns in the input alignment are consensus, and which are not. Any non-gap character indicates a consensus column. (For example, mark consensus columns with "x", and insert columns with "."). This is true by default if the **-d** option is used, in which case **ssu-build** will use the #=GC RF annotation in the default seed model. However, if **-d** is not used, the default is to determine which columns are consensus automatically; if the frequency of gap characters in a column is greater than a threshold, gapthresh (0.8 by default, but settable with the **--gapthresh** option), the column is not consensus.

--gapthresh <*x*> Determine which columns in the input alignment are consensus based on the frequency of gap characters in each column. If less than or equal to <*x*> then the column is defined as a consensus one. As discussed in the description of the **--rf** option, this is default unless the **-d** and/or **--rf** options are enabled, with an <*x*> of 0.8.

Options Controlling Output of Structure Diagrams

--ps2pdf <*s*> Specify that an executable named <*s*> in your PATH can be used for converting postscript files to pdf files with the usage: <**s**> **foo.ps foo.pdf** This executable will be used for creating a pdf diagram of the showing the location of a truncated model on the SSU molecule if the **--trunc** and **-d** options are used. By default, **ssu-build** will use the executable program named **ps2pdf** if it is in your PATH to do this conversion. If **ps2pdf** does not exist, a postscript file will be generated.

--ps-only Specify that postscript output is preferable to pdf. No postscript to pdf conversion will be attempted.

Expert Options for Model Construction

--eent Use the entropy weighting strategy to determine the effective sequence number that gives a target mean match state relative entropy. The default target mean match state relative entropy is 0.59 bits but can be changed to <*x*> with **--ere** <*x*>**.**

--ere <*x*> Set the target mean match state relative entropy as <*x*>. By default the target relative entropy per match position is 0.59 bits.

`ssu-draw` - draw secondary structure diagrams of SSU rRNA

Synopsis

Draw structure diagrams for all alignments created by ssu-align in directory <dir>:

ssu-draw *[options] dir*

Draw structure diagrams for a single alignment file <aln>:

ssu-draw -a *[options] aln*

Description

ssu-draw reads alignments created by **ssu-align,** calculates per-consensus-column statistics on them and creates secondary structure diagrams that display those statistics using "heat map"-like color schemes. The statistics are also printed in tabular format to output text files for easy parsing. **ssu-draw** can also draw secondary structure diagrams of individual sequences in the alignment with the **--indi** option.

Alignment masks, which define a subset of consensus positions to exclude from an alignment for purposes such as phylogenetic inference, can be displayed on the structure diagrams. Positions excluded by the mask will be drawn as open circles. Consensus columns are those which have a non-gap (non-'.') character in the reference annotation for the alignment. The reference annotation appears in lines beginning with "#=GC RF". **ssu-draw** will automatically display masks created by **ssu-mask** if they exist in the same directory as the alignments.

An important caveat of **ssu-draw** is that only consensus positions of the SSU models models are drawn. Inserted residues (i.e. any target residue that does not align to a consensus alignment position) are not drawn. Another important limitation is that **ssu-draw** cannot draw diagrams for alignments created by **ssu-align** using models other than the default ones (with the **-m** option to **ssu-align**) unless the motivated user creates their own template postscript file for the models being used. For more information on this, see the **ssu-esl-ssdraw** manual page.

Output

The secondary structure diagrams created by **ssu-draw** are either pdf or postscript files. If the executable program **ps2pdf** exists in the user's path, the files will be pdf. If not, they will be postscript. The **--ps2pdf** and **--ps-only** options, described below, can be used to modify this default behavior.

By default, a single pdf or postscript that contains nine pages will be drawn for each alignment. The nine pages are alignment summary diagrams and display the alignment consensus sequence, information content, mutual information, insert frequency, average insert length, deletion frequency, internal deletion frequency, average posterior probability and fraction of sequences that span each position, respectively. The consensus sequence can be displayed on these alignment summary diagrams if the **--cnt** option is used. Additionally, the alignment statistics that are displayed in the structure diagrams are printed in tabular format to output text files ending with the ".drawtab" suffix to allow easy parsing.

Options

-h Print brief help; includes version number and summary of all options.

-a Specify that the single command-line argument is a Stockholm alignment file, not a directory.

-f Force; override the limitation that structure diagram files file, not a directory.

-d Display the pre-calculated, default SSU-ALIGN v0.1 masks on the secondary structure diagrams. Positions excluded by the mask will appear as open circles.

-s <*f*> Display the pre-calculated mask in file <*f*>. This option will only work on a single alignment, so either **-a** must be used, or <*dir*> must only contain a single alignment. The path and file specified by <*f*> must exist either with respect to the current working directory or SSUALIGNDIR (environment variable)..

-k <*s*> Display pre-calculate masks in files called <*modelname*>.<*s*>.*mask* on the secondary structure diagrams for alignments created by **ssu-align** with models named <*modelname*>. If **ssu-align** was used with the default CM file, <*modelname*> will be 'archaea', 'bacteria' or 'eukarya'. The mask files must exist in either the current working directory, the directory that includes the alignments being masked, or SSUALIGNDIR.

-m <*f*> Specify that the CM file <*f*> was used by **ssu-align** to create the alignment(s) being masked with a command like **ssu-align -m** <*s*> **foo.fa foo.** <*f*> must be a path that exists with respect to either the current working directory or SSUALIGNDIR.

-t <*f*> Specify that the template postscript file to use for drawing secondary structure diagrams is file <*f*>. If not used, the default v0.1 template file will be used. <*f*> must be a path that exists with respect to either the current working directory or SSUALIGNDIR.

-i Specify that the **-i** option was used by **ssu-align** to create the alignments being masked.

Miscellaneous Input/Output Options:

--ps2pdf <*s*> Specify that an executable named <*s*> in your PATH can be used for converting postscript files to pdf files with the usage: <**s**> *foo.ps foo.pdf*

--ps-only Specify that postscript output is preferable to pdf. No postscript to pdf conversion will be attempted.

--ifile <*f*> Insert information for the alignment, created by **ssu-align,** is in file <*f*>. This option only works in combination with **-a.**

--key-out <*s*> Include the string <*s*> as part of all output file names from **ssu-draw,** immediately before the suffix. For example, foo.archaea.<s>.pdf would be created instead of foo.archaea.pdf.

--no-mask Do not display masks on the structure diagrams. By default, **ssu-draw** will display masks created by **ssu-mask** if they exist in the same directory as the alignments.

--mask-key <*s*> Specify that the mask files created by **ssu-mask** include the string <*s*> before the ".mask" suffix. This will be true if the **--key-out** <*s*> option was used when **ssu-mask** was run.

Options for One Page Alignment Summary Diagrams:

In addition to the default nine page diagram displaying all available alignment statistics, each of the nine statistics can be drawn on single page diagrams using the following options. The output files will be named automatically based on the alignment file name.

--prob Draw a single page diagram displaying average posterior probability per consensus position for each alignment to a file.

--ifreq Draw a single page diagram displaying frequency of inserts after each consensus position for each alignment to a file.

--iavglen Draw a single page diagram displaying average insert length after each consensus position for each alignment to a file.

--dall Draw a single page diagram displaying the fraction of sequences that have a gap (deletion) at each consensus position for each alignment to a file.

--dint Draw a single page diagram displaying the fraction of sequences that have an internal gap (internal deletion) at each consensus position for each alignment to a file. An internal gap for a sequence is one that occurs after (3' of) the first aligned nongap residue in the sequence and before (5' of) the final aligned nongap residue in the sequence.

--span Draw a single page diagram displaying the fraction of sequences that span each consensus position for each alignment to a file. A sequence s spans consensus position 'x' that is actual alignment position 'a' if s has at least one non-gap residue aligned to a position 'b' <= 'a' and at least one non-gap residue aligned to a position 'c' >= 'a'

--info Draw a single page diagram displaying the information content of each consensus position for each alignment to a file.

--mutinfo Draw a single page diagram displaying the mutual information per basepaired position for each alignment to a file.

--cnt Draw consensus nucleotides on all alignment summary diagrams

--no-aln Do not draw the default nine page diagram displaying all available statistics.

Options for Drawing Structure Diagrams for Individual Sequences:

Structure diagrams for individual sequences can be drawn, one sequence per page. For each sequence, an additional page displaying the posterior probability for the sequence will be drawn. To draw all sequences in the alignment, use the **--indi** option. The resulting pdf or postscript files will be large for large alignments. For pdfs, the file size will be about 1 Mb for every 20 sequences. For postscript file, the size will be about 1 Mb for every 2 sequences. If you only want to draw individual diagrams for a subset of the sequences in the alignment, use **ssu-mask** with the **--seq-r** or **--seq-k** options to create an alignment of the subset of sequences you want to draw, and then run **ssu-draw** on that alignment with the **-a** option. See the **ssu-mask** manual page for more information.

--indi Draw sequence and posterior probability diagrams for all sequences in each alignment.

--rf Draw a single page diagram displaying the model's reference sequence/structure to a file. The sequence displayed will be the exact sequence from the #=GC RF annotation of the alignment file.

--cons Draw a single page diagram displaying the alignment consensus sequence to a file. The consensus sequence is defined as the most frequent nucleotide at each position. Nucleotides that occur in at least 75% of the sequences that do not have a gap at the position will be uppercase; others will be lowercase.

--no-pp With **--indi,** do not draw posterior probability diagrams, only draw individual sequence diagrams.

--no-bp With **--indi, --rf,** or **--cons,** do not color nucleotides based on their basepair type.

--no-ol With **--indi,** do not outline nucleotides that are not the most common nucleotide in the alignment at their respective consensus position.

Options for Omitting Sections of Structure Diagrams:

--no-leg Omit the legend from all structure diagrams.

--no-head Omit the header from all structure diagrams.

--no-foot Omit the footer from all structure diagrams.

`ssu-mask` - mask (remove columns from) SSU rRNA multiple sequence alignments

Synopsis

Mask all alignments created by ssu-align in directory <dir>:

ssu-mask *[options] dir*

Mask single alignment file <aln>:

ssu-mask -a *[options] aln*

List sequences in alignments in <dir>:

ssu-mask --list *[options]* <*dir*>

List sequences in alignment file <aln>:

ssu-mask -a --list *[options]* <*aln*>

Convert Stockholm alignments in <dir> to aligned FASTA:

ssu-mask --stk2afa *[options]* <*dir*>

Convert Stockholm alignment <aln> to aligned FASTA:

ssu-mask -a --stk2afa *[options]* <*aln*>

Extract a subset of sequences from an alignment:

ssu-mask -a --seq-k <*listfile*> *[options]* <*aln*>

Remove a subset of sequences from an alignment:

ssu-mask -a --seq-r <*listfile*> *[options]* <*aln*>

Description

ssu-mask reads multiple sequence alignments created by **ssu-align** in the directory *dir,* removes some columns from them (i.e. masks them), and then outputs the masked alignments into the same directory.

The **-a** option specifies that the command line argument is a single alignment file to mask, not a directory.

By default, the alignment masks are automatically computed by **ssu-mask** based on alignment confidence estimates in the alignment files (posterior probabilities) and applied to remove columns from the alignments, as described below. Alternatively, pre-computed masks can be used to enable comparison across multiple datasets. To use the default SSU-ALIGN v0.1 mask, see the **-d** option.

The main purpose of masking is to remove the columns from an alignment that are most likely to contain alignment errors prior to using it as input for a phylogenetic inference program.

An alignment mask defines which columns will be kept and which will be removed. **ssu-mask** can either calculate an appropriate mask for each alignment based on alignment posterior probability values (confidence levels in the alignment) and/or gap frequencies, or use a pre-calculated mask from an input file. In all cases, all non-consensus columns (called insert columns) are automatically removed. Residues in these columns are **not aligned.** In **ssu-align** created alignments, consensus columns are those which have a non-gap (non-'.') character in the reference annotation for the alignment. The reference annotation appears in lines beginning with "#=GC RF".

By default, **ssu-mask** determines a mask based on the posterior probability (PP) annotation for aligned residues. PP annotation appears in #=GR PP lines in the alignments. Characters in Stockholm alignment

PP annotation can have 12 possible values: the ten digits '0-9', '*', and '.'. If '.', the position corresponds to a gap in the sequence. A value of '0' indicates a posterior probability of between 0.0 and 0.05, '1' indicates between 0.05 and 0.15, '2' indicates between 0.15 and 0.25 and so on up to '9' which indicates between 0.85 and 0.95. A value of '*' indicates a posterior probability of between 0.95 and 1.0. Higher posterior probabilities correspond to greater confidence that the aligned residue belongs where it appears in the alignment.

By default, any consensus column in which greater than 95% of the sequences have a PP of 0.95 or better (i.e. have a PP value of '*') will be included by a PP-based masked. The 95% value can be changed to <*x*> with the **--pf** <*x*> option. And the 0.95 PP threshold can be changed to <*y*> with the **--pt** <*y*> option.

The frequency of gaps in each column will also be considered if the **--gapthresh** <*x*> option is used. Any consensus column in which more than <*x*> fraction of the sequences contain gaps will be removed. With **--gapthresh,** additional columns will also be removed based on PP as described above unless the **--gaponly** option is used, which specifies that gap frequency be the only determining factor for mask construction.

Alternatively, pre-existing mask files can be used to remove columns from the alignments by using the **-d, -k,** or **-s** options. See the description of those options below for details. A mask file defines which columns to keep/remove. The mask is a string that may only contain the characters '0' and '1'. A '0' at position x of the mask indicates that column x is excluded by the mask and should be removed during masking. A '1' at position x of the mask indicates that column x is included by the mask and should not be removed during masking. All lines in the *maskfile* that begin with '#' are considered comment lines and are ignored. All non-whitespace characters in non-comment lines are considered to be part of the mask. The length of the mask must equal the consensus length (number of consensus columns) of the alignment to be masked.

In addition to masking, **ssu-mask** can be also be used to list the names of sequences in alignments with the **--list** option, convert Stockholm formatted alignments to aligned FASTA format with the **--stk2afa** option, or remove a subset of sequences from an alignment with the **--seq-k** or **--seq-r** options, as described below in the **OPTIONS** section.

ssu-mask is a PERL script that uses the C programs **esl-alimask, esl-alistat, esl-reformat,** and **esl-alimanip** C programs from the EASEL package to mask alignments, list sequences, convert formats, and remove sequences, respectively. The versions of these programs installed with SSU-ALIGN are prefixed with 'ssu-' to distinguish them from any versions of the program distributed with INFERNAL or HMMER on your system. For example, the version of **esl-alimask** called by **ssu-mask** is actually called **ssu-esl-alimask.**

Options

-h Print brief help; includes version number and summary of all options.

-a Specify that the single command-line argument is a Stockholm alignment file, not a directory.

-d Use the pre-calculated, default SSU-ALIGN v0.1 masks. These masks were determined by applying the default 95%/0.95 PP-based masking strategy described above on very deep SSU alignments. See the User's Guide for details. The default masks are named *archaea-0p1.mask, bacteria-0p1.mask,* and *eukarya-0p1.mask* and are installed in SSUALIGNDIR (environment variable).

-s <*f*> Use the pre-calculated mask in file <*f*>. This option will only work on a single alignment so either **-a** must be used, or <*dir*> must only contain a single alignment. The path and file specified by <*f*> must exist either with respect to the current working directory or SSUALIGNDIR.

-k <*s*> Use pre-calculated masks in files called <*modelname*>.<*s*>.*mask* for masking alignments created by **ssu-align** with models named <*modelname*>. If **ssu-align** was used with the default CM file, <*modelname*> will be 'archaea', 'bacteria' or 'eukarya'. The mask files must exist in either the current working directory, the directory that includes the alignments being masked, or SSUALIGNDIR.

-m <*f*> Specify that the CM file <*f*> was used by **ssu-align** to create the alignment(s) being masked with a command like **ssu-align -m** <*s*> **foo.fa foo.** <*f*> must be a path that exists with respect to either the current working directory or SSUALIGNDIR.

-t <*f*> Specify that the template postscript file to use for drawing secondary structure diagrams is file <*f*>. If not used, the default v0.1 template file will be used. <*f*> must be a path that exists with respect to either the current working directory or SSUALIGNDIR.

-i Specify that the **-i** option was used by **ssu-align** to create the alignments being masked.

Options for Controlling Mask Construction:

--pf <*x*> Specify that a consensus column is kept (included by mask) if the fraction of sequences with a non-gap residue in that column with a posterior probability of at least <*y*> (from **--pt** <*y*>) is <*x*> or greater. All other consensus columns and all non-consensus (insert) columns are removed (excluded by mask). By default <*x*> is 0.95.

--pt <*y*> Specify that a column is kept (included by mask) if <*x*> (from **--pf** <*x*>) fraction of sequences with a non-gap residue in that column have a posterior probability of at least <*y*>. All other consensus columns and all non-consensus (insert) columns are removed (excluded by mask). By default <*y*> is 0.95. See the DESCRIPTION section for more on posterior probability (PP) masking. Due to the granularity of the PP annotation, different <*y*> values within a range covered by a single PP character will be have the same effect on masking. For example, using **--pt** *0.86* will have the same effect as using **--pt** *0.94***.**

--rfonly Keep all consensus columns and remove all non-consensus (insert) columns. Do not remove any consensus columns based on posterior probabilities or gap frequencies.

--gapthresh <*x*> Remove all consensus columns for which the fraction of sequences in the alignment that have a gap ('.', '-', or '_') at that position is greater than <*x*> and all non-consensus (insert) columns. Other consensus columns may be removed based on posterior probabilities as well unless the **--gaponly** option is used.

--gaponly With **--gapthresh** <*x*>**,***only* remove consensus columns based on gap frequencies. Do not remove any columns based on PPs.

Miscellaneous Output Options:

--afa Output alignments in aligned FASTA (afa) format instead of Stockholm. Note that the output alignments will not be valid input to the **ssu-draw** or **ssu-build** programs.

--dna Output DNA alignments, not RNA ones. By default, RNA alignments are output, even if the input is DNA.

--key-out <*s*> Include the string <*s*> as part of all output file names from **ssu-mask,** immediately before the suffix. For example, foo.archaea.<s>.mask would be created instead of foo.archaea.mask.

Options for Creating Secondary Structure Diagrams Displaying Masks:

If **-d** is used, **ssu-mask** will attempt to draw secondary structure diagrams displaying which consensus columns are kept and which are removed by the mask(s). The diagrams will initially be created as postscript files, but will be converted to pdf files if the program **ps2pdf** (or another program <s> specified by **--ps2pdf** <*s*>) is installed and is in the user's PATH. Otherwise, the output diagrams will be postscript files.

--ps2pdf <*s*> Specify that an executable named <*s*> in your PATH can be used for converting postscript files to pdf files with the usage: <**s**> *foo.ps foo.pdf*

--ps-only Specify that postscript output is preferable to pdf. No postscript to pdf conversion will be attempted.

--no-draw Do not draw any mask diagrams.

Options for Alternatives to Masking (listing, Converting, or Removing Sequences):

--list For each alignment, create a file that simply lists each sequence in the alignment on a separate line. Masking is not performed.

--stk2afa Convert each Stockholm alignment to aligned FASTA format. Masking is not performed.

--seq-k <*f*> Remove all sequences **except** those listed in file <*f*>. The file must contain each sequence name on a separate line. All names in the file must exist in the alignment. This file must be used in combination with **-a** because it will only work on a single alignment. Masking is not performed.

--seq-r <*f*> Remove all sequences listed in file <*f*>. The file must contain each sequence name on a separate line. All names in the file must exist in the alignment. This file must be used in combination with **-a** because it will only work on a single alignment. Masking is not performed.

`ssu-merge` - merge SSU rRNA alignments created by parallel ssu-align jobs

Synopsis

Merge results from parallel ssu-align jobs:

ssu-merge *[options] output-dir-created-by-ssu-prep*

Merge a list of alignments, all created with the same CM:

ssu-merge --list *[options] list-file output-alignment*

Description

ssu-merge merges results from parallel **ssu-align** jobs. The **ssu-align** jobs must have been executed by a shell script created by **ssu-prep.** Alternatively if **--list** is used, **ssu-merge** merges all alignments in *list-file* into a single alignment, outputs it as *output-alignment* and exits.

When **--list** is not used, **ssu-merge** first examines the directory *output-dir-created-by-ssu-prep* to make sure that (a) it was created by **ssu-prep** and (b) all of the parallel **ssu-align** jobs created by **ssu-prep** have successfully finished running. If both (a) and (b) are true, **ssu-merge** merges the output files from all of the jobs together, removing the individual job output files once it has successfully finished.

Users should rarely need to call **ssu-merge** directly. This is because it is called internally by the final **ssu-align** job in a set of parallel jobs created by **ssu-prep** to automatically merge the output of all the jobs once they are all complete. However, if the **--no-merge** option was specified to **ssu-prep,** or one of the jobs failed to finish successfully, it will be necessary for the user to directly execute **ssu-merge.**

For example, given the original **ssu-prep** command: **ssu-prep -x my.fa foo 8**

Followed by the execution of the resulting output shell script: **sh foo.ssu-align.sh**

The 8th and final job submitted by this shell script will wait for all jobs to finish and automatically merge the results of all the jobs by calling **ssu-merge.** However, if an error occurs preventing one or more of the jobs from successfully finishing, the final job will output an error message informing the user what jobs have failed and how to rerun them.

Once they have been rerun, the user will need to manually merge the output of all the jobs with the command: **ssu-merge foo**

If the **--list** option is used, **ssu-merge** expects two command-line arguments. The first (*list-file*) is the name of a file that lists all input alignment files to merge. The second (*output-alignment*) is the desired name for the merged output alignment. All input alignments must be in Stockholm format, include #=GC RF 'reference' annotation, and have the same number of nongap RF positions. In this mode, **ssu-merge** will simply merge all the input alignments in all the files into a single merged alignment, save it to *output-alignment* and exit.

Options

-h Print brief help; includes version number and summary of all options.

-f Allow output files that already exist to be overwritten. This option should only be needed if **ssu-merge** has already been executed on *<output-dir-created-by-ssu-prep>.*

--rfonly When merging alignments, discard insert columns and only include consensus columns in the alignment. The alignments will be a fixed, predicted width (the number of consensus positions for each model), but will not include all target nucleotides. Note that insert columns are automatically removed by the **ssu-mask** program which is recommended prior to inputting alignments into phylogenetic inference programs and using **--rfonly** can greatly reduce the size of the final merged alignments, especially for very deep alignments (hundreds of thousands of sequences). The **--rfonly** option can also be specified to **ssu-prep,** in which case it will be set by the internal merge performed by the resulting final **ssu-align** job.

Options for List Mode

-i Input alignments are interleaved Stockholm format in which each sequence is split up across several lines, as opposed to so-called Pfam Stockholm format in which each sequence occurs on exactly one line. Alignments created by **ssu-align** are by default in Pfam Stockholm format, but will be interleaved Stockholm if the **-i** option was supplied to **ssu-align.** This option only works in combination with **--list.**

--dna Output the merged alignment as DNA, not as RNA. By default, the output alignment will be RNA, even if the input alignments were DNA. This option only works in combination with **--list.**

`ssu-prep` - prepare SSU rRNA sequences for parallel ssu-align jobs

Synopsis

Add a prefix and suffix to ssu-align job commands as defined in <prefix-suffix-file>:

ssu-prep *[options] seqfile output-dir n prefix-suffix-file*

Run all jobs in parallel on one machine with <n> cores:

ssu-prep -x *[options] seqfile output-dir n*

Omit prefixes and suffixes from ssu-align job commands, to be manually added later:

ssu-prep -y *[options] seqfile output-dir n*

Description

ssu-prep reads the unaligned FASTA-formatted *seqfile,* splits it up into *n* smaller sequence files and creates a shell script that will execute *n* **ssu-align** jobs in parallel, each processing one of the small sequence files. The results of all jobs will automatically be merged together by the final job, giving the same results as if a single **ssu-align** job was run for the complete *seqfile.* The advantage of using **ssu-prep** is that it allows the *n* smaller jobs to be run simultaneously, in parallel, either on a multi-core machine or on a compute cluster, so the final result is obtained up to (nearly) *n* times faster. The merging is performed by an internal call to the **ssu-merge** script. See the **ssu-merge** manual page for more information.

All of the **ssu-align** options (see the **ssu-align** manual page) are available to **ssu-prep.** Any of these options that are set on the **ssu-prep** command line will identically set in the **ssu-align** job commands in the output shell script. The output shell script will be named <*output-dir*>*.ssu-align.sh* and can be executed with the command **sh** <**output-dir**>**.ssu-align.sh.**

The smaller sequence files created by **ssu-prep** will be named as *seqfile.*<*i*> where <*i*> is a number ranging from 1 to *n.* By default, **ssu-prep** will try to divide the sequences such that, as nearly as possible, all jobs process the same number of nucleotides, with the restriction that sequences are never reordered (by concatenating the *n* smaller sequence files in numerical order, the original **seqfile** would be reproduced).

ssu-prep will create an output directory called *output-dir* into which all output files will be placed, including the small sequence files. When the **ssu-align** jobs are run, each one will create a subdirectory within *output-dir* called *output-dir.*<*i*> and place its output files there. When the *n* jobs are merged, these directories will be emptied and removed (unless the **--keep-merge** option is set).

Customizing Parallelization for the Ssu-align Jobs Using Prefixes and Suffixes

The main purpose of **ssu-prep** is to allow simple parallelization of large **ssu-align** jobs by breaking large jobs into many smaller ones, each of which will run separately. Currently, the **SSU-ALIGN** package does not support more respectable paralellization strategies such as MPI, nor does it use threads.

In order to allow jobs to run in parallel, **ssu-prep** allows the user to set a fixed prefix and suffix string to be added before and after each **ssu-align** job command. The three different usages listed in the SYNOPSIS above control how this prefix and suffix are defined.

In the first case, if neither the **-x** nor the **-y** options are used, the prefix and suffix are defined in the <*prefix-suffix-file*> as discussed in FORMAT OF THE PREFIX-SUFFIX-FILE below.

Secondly, if **-x** is used the job commands will be run all at once on a single multi-core machine, and so no prefix or suffix is necessary. (Actually, in this case the suffix used will be ' > /dev/null & ' for all but the final job, which makes the jobs silently and simultaneously run in the background.)

Finally, if the user specifies **-y,** no prefix or suffix will be added to the **ssu-align** job commands, but the user should manually add their own later. The **-y** option is meant to be useful for users who want the jobs to have prefixes and suffixes that are not constant, or that are more complex than those possible by the simplistic use of a two-line **prefix-suffix-file.** For example, the **ssu-align** commands might need to be preceded by a 'ssh <host>;' command, but where <host> is different for each command.

Format of the Prefix-suffix-file

The *prefix-suffix-file* must contain exactly two non-comment lines (comment lines begin with '#'). The first line will become a prefix for all **ssu-align** commands and the second line will become a suffix. For example, if the first line is:

qsub -N ssu-align -o /dev/null -b y -j y -cwd -V '

And the second line is:

'

Then a resulting ssu-align job command in the output shell script might be:

qsub -N ssu-align -o /dev/null -b y -j y -cwd -V ' ssu-align foo/foo.1/my.fa.1 foo/foo.1 '

In contrast, if the **-y** option is used, specifying no prefix or suffix, the same job command would appear as:

ssu-align foo/foo.1/my.fa.1 foo/foo.1

in the output shell script.

Specifying Options for the Parallel Alignment Jobs

All **ssu-align** options can be set on the **ssu-prep** command line, and they will be added to the **ssu-align** command-line of the resulting parallel jobs in the output shell script. See the **ssu-align** manual page for a list of available options. Additionally, **ssu-prep** specific options are listed below.

Options

-h Print brief help; includes version number and summary of all options.

-x Specify that all *n* jobs in should be run in parallel on a single machine that has *n* processors (cores). See the CUSTOMIZING PARALLELIZATION section above for more details. Note that **ssu-prep** does not check to make sure your computer actually has *n* processors.

-y Specify that the **ssu-align** jobs should have no prefix or suffix. The user should manually add them to the output shell script later in order to ensure the jobs actually run in parallel. See the CUSTOMIZING PARALLELIZATION section above for more details.

-f Allow output directory *output-dir* to be deleted and overwritten. Without this option, if the *output-dir* directory already exists, **ssu-prep** exits with an error.

-q Specify that the third **ssu-prep** command-line argument, *n,* specifies the desired number of sequences per job, not the number of total jobs. The number of total jobs will be dependent on the number of sequences in *seqfile.*

-e Specify that each job should process the same number of sequences, instead of the same number of total nucleotides, which is the default behavior.

--no-bash Specify that the output shell script that will submit/run all *n* jobs not be written in BASH shell syntax, which it is by default.

--no-merge Specify that the final job should not automatically merge the results of all jobs. The user can still manually merge the jobs using the **ssu-merge** program after all jobs are finished running.

--keep-merge Do not automatically remove the smaller per-job output files once they have successfully been merged together. If this option is selected, a shell script called *output-dir.cleanup.sh* will be created that can be used to remove all the smaller per-job output files at a later time.

References

Brown, M. P. (2000). Small subunit ribosomal RNA modeling using stochastic context-free grammars. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 8:57–66.

Cannone, J. J., Subramanian, S., Schnare, M. N., Collett, J. R., D'Souza, L. M., Du, Y., Feng, B., Lin, N., Madabusi, L. V., Müller, K. M., Pande, N., Shang, Z., Yu, N., and Gutell, R. R. (2002). The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3:2.

Caporaso, J. G., Bittinger, K., Bushman, F. D., DeSantis, T. Z., Andersen, G. L., and Knight, R. (2010). PyNAST: a flexible tool for aligning sequences to a template alignment. *Bioinformatics*, 26:266–267.

Claesson, M. J., O'Sullivan, O., Wang, Q., Nikkil, J., Marchesi, J. R., Smidt, H., de Vos, W. M., Ross, R. P., and O'Toole, P. W. (2009). Comparative analysis of pyrosequencing and a phylogenetic microarray for exploring microbial community structures in the human distal intestine. *PLoS One.*, 4:e6669.

Cole, J. R., Wang, Q., Cardenas, E., Fish, J., Chai, B., Farris, R. J., Kulam-Syed-Mohideen, A. S., McGarrell, D. M., Marsh, T., Garrity, G. M., and Tiedje, J. M. (2009). The Ribosomal Database Project: Improved alignments and new tools for rRNA analysis. *Nucleic Acids Res.*, 37:D141–D145.

DeSantis, T. Z., Hugenholtz, P., Keller, K., Brodie, E. L., Larsen, N., Piceno, Y. M., Phan, R., and Andersen, G. L. (2006a). NAST: A multiple sequence alignment server for comparative analysis of 16S rRNA genes. *Nucleic Acid Res.*, 34:W394–399.

DeSantis, T. Z., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, E. L., Keller, K., Huber, T., Dalevi, D., Hu, P., and Andersen, G. L. (2006b). Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Appl Environ Microbiol.*, 72:5069–5072.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. J. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK.

Eddy, S. R. (2002). A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*, 3:18.

Eddy, S. R. and Durbin, R. (1994). RNA sequence analysis using covariance models. *Nucleic Acids Res.*, 22:2079–2088.

Finn, R. D., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J. E., Gavin, O. L., Ceric, G., Forslund, K., Holm, L., Sonnhammer, E. L. L., Eddy, S. R., and Bateman, A. (2010). The Pfam protein families database. *Nucleic Acids Res.*, 38:D211–D222.

Gardner, P. P., Daub, J., Tate, J. G., Nawrocki, E. P., Kolbe, D. L., Lindgreen, S., Wilkinson, A. C., Finn, R. D., Griffiths-Jones, S., Eddy, S. R., and Bateman, A. (2009). Rfam: Updates to the RNA families database. *Nucleic Acids Res.*, 37:D136–D140.

Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., and Eddy, S. R. (2003). Rfam: an RNA family database. *Nucleic Acids Res.*, 31:439–441.

Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. R., and Bateman, A. (2005). Rfam: Annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.*, 33:D121–D141.

Holmes, I. (1998). *Studies in Probabilistic Sequence Alignment and Evolution*. PhD thesis, University of Cambridge.

Kolbe, D. L. and Eddy, S. R. (2009). Local RNA structure alignment with incomplete sequence. *Bioinformatics*, 25:1236–1243.

Nawrocki, E. P. (2009). *Structural RNA Homology Search and Alignment Using Covariance Models*. PhD thesis, Washington University School of Medicine.

Nawrocki, E. P. and Eddy, S. R. (2007). Query-dependent banding (QDB) for faster RNA similarity searches. *PLoS Comput. Biol.*, 3:e56.

Nawrocki, E. P., Kolbe, D. L., and Eddy, S. R. (2009a). Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, 25:1335–1337.

Nawrocki, E. P., Kolbe, D. L., and Eddy, S. R. (2009b). The Infernal user's guide. [http://infernal.janelia.org/].

Noller, H. F. and Woese, C. R. (1981). Secondary structure of 16S ribosomal RNA. *Science*, 212:403–411.

Pruesse, E., Quast, C., Knittel, K., Fuchs, B. M., Ludwig, W., Peplies, J., and Glockner, F. O. (2007). SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. *Nucleic Acids Res.*, 35:7188–7196.

Schloss, P. D. (2009). A high-throughput DNA sequence aligner for microbial ecology studies. *PLoS One*, 4:e8230.

Smit, S., Rother, K., Heringa, J., and Knight, R. (2008). From knotted to nested RNA structures: a variety of computational methods for pseudoknot removal. *RNA*, 14:410–416.

Wang, Q., Garrity, G. M., Tiedje, J. M., and Cole, J. R. (2007). Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl Environ Microbiol.*, 73:5261–5267.

Wimberly, B. T., Brodersen, D. E., , Morgan-Warren, R. J., Carter, A. P., Vonrhein, C., Hartsch, T., and Ramakrishnan, V. (2000). Structure of the 30s ribosomal subunit. *Nature.*, 407:327–339.

Woese, C. R., Gutell, R., Gupta, R., and Noller, H. F. (1983). Detailed analysis of the higher-order structure of 16S-like ribosomal ribonucleic acids. *Microbiol Rev.*, 47:621–669.

Woese, C. R., Magrum, L. J., Gupta, R., Siegel, R. B., Stahl, D. A., Kop, J., Crawford, N., Brosius, J., Gutell, R., Hogan, J. J., and Noller, H. F. (1980). Secondary structure model for bacterial 16S ribosomal RNA: phylogenetic, enzymatic and chemical evidence. *Nucleic Acids Res.*, 10:2275–93.