CMfinder package 0.4.1.15 user manual

Software for inferring and scoring RNA secondary structure

Zasha Weinberg
(with lots of code by Zizhen Yao, Eric Nawrocki and others)

September 13, 2018

This package includes the following material from other sources (see COPYRIGHT files in the subdirectories for details; only the primary authors are given here):

- CMfinder version 0.3 [10] by Zizhen Yao (which included the cmfinder and pscore programs, and various related data files and Perl scripts)
- Infernal version 0.72 (http://infernal.janelia.org) by Sean Eddy and others
- Infernal version 1.1 (http://infernal.janelia.org) by Erik Nawrocki, Sean Eddy and others
- Vienna RNA versions 1.4 and 1.7 (http://www.tbi.univie.ac.at/RNA/) by Ivo Hofacker, Peter Stadler, Sebastian Bonhoeffer and others
- Pfold [1] by Bjarne Knudsen (Thanks to Zsuzsanna Sükösd for retrieving source code)
- CLUSTALW [3] version 1.83 (no author credits given)
- QRNA [2] by Elena Rivas and Sean Eddy
- RaveNnA by Zasha Weinberg [6, 7, 8] is the source of much of the code for `hmmpair`.

Chapter 1

Introduction

1.1 What does this software do?

This software provides three main programs:

- CMfinder. CMfinder is designed to predict an alignment and RNA secondary structure of multiple unaligned sequences, and is described in a paper [10].
- RNAphylo (also called "pscore"). RNAphylo is designed to score an alignment based on how much evidence it contains of the presence of a conserved RNA. It is described in Zizhen Yao's Ph.D. dissertation [9].
- hmmpair. Hmmpair is an alternate method to score alignments, which attempts to evaluate how much support there is in conserved sequences for aligning covarying base pairs. It uses profile HMMs to evaluate conservation. Hmmpair is described in a paper by Zasha Weinberg [5].

This software extends CMfinder version 0.3, which was written by Zizhen Yao.

1.1.1 How is this version 0.4 different from version 0.3?

The main changes are that CMfinder and RNAphylo are extended to allow for sequences in which an RNA is truncated. This situation frequently arises in metagenome sequences, and in the previous version of CMfinder creates problem in motif detection and scoring. Sequence truncation was not a significant issue when previous versions of CMfinder were developed, mainly because less metagenomic data was available.

The hmmpair method was not a part of version 0.3.

This new version of CMfinder does not implement some functionity in CMfinder 0.3, and therefore the old CMfinder 0.3 is provided (called `cmfinder`) as well as the updated version 0.4. Only one version of RNAphylo is provided.

CMfinder 0.4 is different from version 0.3 as follows:

- Alignments are robust to input sequences in which the RNA structure has been truncated, with appropriate command-line flags. CMfinder 0.4 deals with truncation case by using the truncation-aware alignment functions in Infernal 1.1, and by adjusting the mutual information and other calculations within CMfinder. Only truncations caused by limited contig assembly is supported. By contrast, biological causes of truncations (e.g., transposon insertion) are not supported. Basically, truncations can only occur at the end of a contig, never in the middle.

- CM functions are based on Infernal 1.1, rather than the pre-1.0 version of CMfinder 0.3.
- Degenerate nucleotides are dealt with better, by marginalizing over the possibilities in some contexts and ignoring the degenerate nucleotides (so they don't impact the calculations) where possible.
- Other experimental or minor features.
- CMfinder 0.4 does *not* implement functionality to deal with long input sequences (e.g. anchors) that CMfinder 0.3 implemented to help with eukaryotic sequences.

The updated RNAphylo provides the following new features compared to the CMfinder 0.3 package:

- Scoring is robust to input sequences in which the RNA structure has been truncated. The nucleotides missing due to trunctation are treated as not being a part of the alignment, so they do not contribute to the score. (The obvious solution would be marginalize over the possible nucleotides, but this can bias the score by unfairly considering non-pairing nucleotides as likely—it's safer to just ignore the missing nucleotides.) For subtle reasons, enabling this functionality requires restricting the scoring to a specific consensus secondary structure provided by the user, rather than considering all possible secondary structures.
- Degenerate nucleotides are also treated as being not part of the alignment, except for invocation of RNAfold as part of partition function calculations.
- The program is faster and takes less memory because it caches expensive phylogenetic tree calculations. However, the speed-up is only significant in practice for relatively large alignments with the truncation-aware scoring; typical calculations that were performed with the previous version of RNAphylo were already fast enough.
- Other experimental or minor features.

Additionally there are some changes to Infernal 1.1 to allow for CMs that are only calibrated (with `cmcalibrate`) for certain search modes. This modification can save time when the calibration time is significant compared to the search time. The implemention of this feature adds the flag `--partial` *code* to the `cmcalibrate` command. *code* = `lc` means local CYK only, `gc` means glocal CYK only, `li` means local inside only, `gi` means glocal inside only, `hmm` means hmmonly search. Searches will abort with an error if the search requires an un-calibrated mode.

1.1.2 Changes since version 0.4

- Version 0.4.1.15:
  - Minor bug fixes.
  - fixed problem in `candf` program that causes it to crash with large input sequences.
  - Added citation of paper describing this package in comparison to CMfinder version 0.3 [5].
- Version 0.4.1.9
  - Added `hmmpair` program, which implements an alternate alignment scoring method.

  - Added `ScoreMotif.pl` script, which produces a score combining `RNAphylo` and `hmmpair`.
  - Other minor changes.

- Version 0.4.0.35:
  - First version of version 0.4 to be publically released.

1.1.3 Components of this software

The main components of this package are:

- `cmfinder.pl` : Perl script from version 0.3 of the package to predict a set of possible multiple-sequence alignments from unaligned sequences.
- `cmfinder04.pl` : Analogous to cmfinder.pl, but uses cmfinder04, the updated version of the program.
- `cmfinder` : C program that optimizes an alignment. This is the major component used by the `cmfinder.pl` script, and is specific to CMfinder version 0.3.
- `cmfinder04` : Same purpose as `cmfinder`, but with some new features.
- `RNAphylo` (previously called `pscore`) : a program to score an alignment based on evidence of conservation of a given secondary structure. A phylogenetic model is used.
- `hmmpair` : a program implementing a different approach to scoring alignments for conservation of secondary structure.
- `ScoreMotif.pl` : scores alignments for evidence of RNA structure using a combination of `RNAphylo` and `hmmpair`.

1.2 Which version of CMfinder should I use?

In general, `cmfinder04.pl/cmfinder04` is better for bacterial or archaeal sequences, whose non-coding sequences are typically not long, and which sometimes have truncations within incomplete genomes. `cmfinder04` is especially important for metagenome sequences, where truncations are frequent.

For eukaryotic sequences, which will often be quite long, `cmfinder.pl/cmfinder` is likely to be better.

1.3 Credit

If you use CMfinder or RNAphylo, please cite Zizhen Yao's paper in *Bioinformatics* [10], her Ph.D. dissertation [9] and/or this manual.

If you use hmmpair or specifically use the truncation-aware functionality, please cite the relevant paper by Zasha Weinberg [5] and/or this manual.

CMfinder 0.3 was conceived and written by Zizhen Yao while working on her Ph.D. with Larry Ruzzo. CMfinder version 0.4 is derived from this code base, and the modifications were made by Zasha Weinberg. This software package also makes extensive use of other software packages that are listed on the title page of this manual.

1.4 Licensing

The entire package is distributed freely but without any warranty whatsoever under the GNU General Public License. For details, see http://www.gnu.org/licenses.

Details on copyrights covering parts of this software are given in the `COPYRIGHT` files in the various subdirectories.

Chapter 2

Installation

Installation means compiling & linking executable files. There is no actual installation (like to `/usr/local`).

2.1 Building the package

Summary: Unpack the `.tgz` file and get into its root directory. Then run:

```
./configure
make
```

(There is no `make install`, you just run programs from the build locations. Perl scripts are in the `bin` subdirectory, and symbolic links to executables are also in this directory.)

2.1.1 options to configure

The `configure` script is an adaptation of the configure script for Infernal 1.1rc2. The `--enable-mpi` flag has been removed, and some flags might not work properly in the CMfinder package. Also, since there's no `make install` procedure, none of the flags relating to the installation directories will have any effect.

Flags that have been tested are:

- `--enable-debugging` / `--disable-debugging` (Will pass the flags `-O0 -g -Wall` to gcc)
- `--enable-threads / --disable-threads` (But note that only the `cmfinder04` executable has any multithreading capabilities; the other programs will completely ignore threading.)
- The environment variables listed at the end of the output of `./configure --help` should work.

2.2 Environment variables

If you wish to run `cmfinder` or `cmfinder.pl`, you will most likely need to do the following:

- Set environment variable `CMfinder` to the root directory of the un-packed files.
- Make sure that the command `clustalw` is in `PATH`

- Add the `bin` subdirectory to `PATH`

If you want to use ClustalW with `cmfinder04.pl` (which is not necessary), you'll need to make sure `clustalw` is in `PATH`.

2.3 Platforms on which the CMfinder package is known to work

Building has only been tested with the gcc compiler suite (`http://gcc.gnu.org`). To ensure compatibility, you will need the GNU C compiler (which provides the `gcc` command). The software is unlikely to work on a non-gcc compiler without modification.

The software has been tested on the following platforms:

- gcc version 4.5.3 under Cygwin 1.7.17 (32-bit) on Windows 7.
- gcc version 4.4.6 under Red Hat Linux 6.3 running Linux kernel 2.6.32 (64-bit).

I presume that this software will work on gcc version 4 or higher on any UNIX system. For Windows, you'll probably want to install Cygwin (`www.cygwin.com`), and be sure to install development tools like gcc and make (which are not installed by default). For MacOSX, you can use Darwin, but you'll need to install Xcode to get gcc and make.

2.4 Known issues with installation

If you get an error in `impl_ssl.h` like "`unknown type name 'FM_METADATA'`", you might have include files from Infernal or HMMER already in the include path. Check what was in your CFLAGS or CPPFLAGS environment variables when you ran `./configure`.

Chapter 3

Programs from CMfinder version 0.3

The following is derived from the `README` file from the CMfinder version 0.3 distribution. It covers only the `cmfinder.pl` Perl script, the alignment optimizing program `cmfinder` and some minor helper programs.

The perl script mentioned (`cmfinder.pl`) is found in the `bin` subdirectory. The executable files are found in the `cmfinder03` directory (once they are built in the installation process), and symbolic links to them are in the `bin` directory.

3.1 Usage

3.1.1 Typical usage: the Perl script

A Perl script is provided that predicts a set of alignments with a secondary structure, given a set of unaligned sequences. This Perl script is called `cmfinder.pl` (in the `bin` subdirectory). The command line is:

`cmfinder.pl` *[options] seq-file*

The file *seq-file* must contain a list of sequences in FASTA format.

Zero or more output files are created that correspond to different possible alignments from the input sequences. The output motifs are named like *seq-file*`.motif.`*suffix* where the *suffix* indicates the number of stem-loops used in the motif, and the motif order. For example, *seq-file*`.motif.h2_1` is the first double stem-loop motif. The corresponding covariance model file is named in the the same way, such as `seq-file.cm.h2_1`.

Motifs can be combined (when the `-combine` flag is given), and the name of a combined motif is based on the names of the elementary motifs used. For example, *seq-file*`.motif.h2_1.h1_4` corresponds to the combination of the first double stem-loop motif with the fourth single stem-loop motif.

For most applications, you can simply use the default configuration, which includes combining simple motifs:

`cmfinder.pl --def` *seq-file*

More detailed options and arguments are explained below.

3.1.1.1 Input sequence file

The input sequence file (*seq-file*) must be in FASTA format.

3.1.1.2 Number of candidates

`-s1` *number-of-candidates-with-1-stem-loop*

By default 5 candidate alignments are predicted with one stem loop. Larger input sequences might benefit from more predictions.

`-s2` *number-of-candidates-with-2-stem-loops*

Same thing, for candidate alignments with two stem loops.

3.1.1.3 Minimum/maximum motif length

`-m1` *min-motif-length-with-1-stem-loop*

`-M1` *max-motif-legth-with-1-stem-loop*

`-m2` *min-motif-length-with-2-stem-loops*

`-M2` *max-motif-legth-with-2-stem-loops*

The lower bound of motif length is 15 bp, and the upper bound is 250 bp. The actual output motifs can be slightly outside the specified range, because we do not enforce the range during refinement iteration. Different values are used for motifs with one or two stem loops.

3.1.1.4 Fraction of sequences containing the motif

`-f` *fraction*

If it is expected that only a small fraction of sequences actually contain the motif, you can set the fraction to a small value (say 0.3) to affect its behavior. Usually, this parameter does not affect the performance much, if the embedded RNA structure is significant.

3.1.2 Components of CMfinder

CMfinder version 0.3 has 4 main components written as C programs: `candf` for searching candidates, `cands` for candidates comparison, `canda` for aligning similar candidates and `cmfinder` for EM refinement.

3.1.2.1 candf

This program takes a FASTA format sequence file as input, and outputs a list of candidate motif instances for each sequence. A user can specify the number of candidates, the minimum and maximum length of the motif, and the number of stem-loops in the motif. The energy function computation is based on the Vienna software package.

3.1.2.2 cands

This program performs pairwise candidate comparison using a tree-edit algorithm, i.e., using the `treedist` function in the Vienna software package. Then it selects at most one candidate from each sequence such that the sum of their pairwise distances are relatively small. These candidates are then aligned by `canda` to create a seed alignment for EM refinement. A user can use `-n` option to specify the multiple sets of candidates as seeds for EM. While we allow overlap between different seeds, we want them to be as different as possible. The `-f` fraction option helps to eliminate unlikely candidates from seed alignment. A user can also provide a *match-constraint-file* to specify the anchor points for alignment, that only the candidates that are consistent with the anchors are compared. These anchors are computed using BLAST.

3.1.2.3 canda

`canda` aligns all the candidates in the seed to a consensus candidate to create a seed alignment.

3.1.2.4 cmfinder

`cmfinder` performs EM refinement of a given alignment. The basic CM operations, i.e. CM alignment, model construction, parameterization are based on Infernal 0.7. The usage of `cmfinder` is:

`cmfinder` *[options] seq-file output-cmfile*

where

- *seq-file* are sequences in FASTA format, and
- *output-cmfile* will contain the output covariance model.

The main options are:

- `-a` *align-file* : the initial (seed) motif alignment. The alignment must be in Stockholm format by default. You can also use the `--format` option to use an alignment in another format (e.g. ClustalW format). In typical usage, this alignment is produced by `canda` (see above), but it can be obtained from other sources, such as ClustalW alignment. *cmfinder* does not use the secondary structure of the input alignment, so it is not necessary.

- `-i` *cmfile* : the initial covariance model. You must specify either an alignment (with `-a`), or a covariance model (with `-i`) as input.

- `-o` *align-file* : the output motif structural alignment in Stockholm format. If no output alignment filename is provided, the alignment will be printed to standard output.

- `-c` *candidate-file* : the candidate file.

  CMfinder was originally designed as a two-phase process: first we only iterate among candidates found by `candf`, then in the second phase, the CM model is used to scan the whole sequence to identify new candidates. In practice, however, the seed alignment is good enough to scan the sequence directly. So, by default, the first phase is skipped, unless the user explicitly provides the *candidate-file*.

3.1.3 Postprocessing of motifs

3.1.3.1 Combining motifs

Initially, `cmfinder.pl` predicts simple motifs with only one or two stem loops. For later processing, there is a heuristic process to combine multiple simple motifs into large structures. These heuristics are implemented in the `comb_motif.pl` script. The `cmfinder.pl` script invokes this script itself when run with the `-combine` flag; there is normally no need to run `comb_motif.pl` directly.

The usage is the following:

`comb_motif.pl` *seq-file motif1 motif2 ...*

If all the motif files share the common prefix, you only need to use the prefix.

3.1.3.2 Summarize motif features and statistics

CMfinder can output multiple motifs, and it is not always clear how to select the best one. The `summarize` program calculates a variety of statistics that can be useful to ranking motifs, according to a formula proposed by Zizhen Yao [9]. A method was also developed to select the best up-to-four motifs (`http://bio.cs.washington.edu/supplements/yzizhen/pipeline/`). These methods are useful for automated prediction of novel RNAs. When the goal is to predict possible structures of a given set of sequences, it is probably best to look at all or many of the predicted alignments.

Chapter 4

CMfinder version 0.4

This chapter explains CMfinder programs specific to version 0.4. In comparison to CMfinder version 0.3 (see previous chapter), the only differences are modifications in the `cmfinder04` executable and the `cmfinder04.pl` Perl script. For all other programs, the version 0.3 program is used.

The `cmfinder04.pl` Perl script is found in the `bin` subdirectory. The `cmfinder04` executable is found in the `cmfinder04` subdirectory. A symbolic link to `cmfinder04` is in the `bin` subdirectory. The modified version of the `summarize` program is accessed by running `cmfinder04` with the `--summarize` flag.

4.1 The cmfinder04.pl Perl script

The Perl script `cmfinder04.pl` has an identical purpose to the `cmfinder.pl` script from CMfinder version 0.3 (see section 3.1.1). It predicts a set of alignments, each with a consensus secondary structure, given unaligned sequences as input. The main difference is that it uses the updated `cmfinder04` executable (see below).

The command line is:

`cmfinder04.pl` *[options] seq-file*

where *seq-file* is a list of sequences in FASTA format.

A brief description of the various options can be obtained by running

```
cmfinder04.pl -h
```

The following command line is recommended in general for finding new motifs. It is not, however, based on any careful study:

`cmfinder04 -skipClustalW -minCandScoreInFinal 10 -combine -fragmentary`
`-commaSepEmFlags x--filter-non-frag,--max-degen-per-hit,2,--max-degen-flanking-nucs,`
`7,--degen-keep,--amaa` *seq-file*

(There should be a space before `-commaSepEmFlags`, but not before `7`.)

The most significant options are:

- `-fragmentary` : enables handling of fragmentary RNAs within the alignments. Without this option, `cmfinder04` will behave poorly (like version 0.3) with fragmented RNAs.
- `-combine` : attempt to combine simple motifs in order to possibly make a prediction with a more complex structure. `cmfinder04.pl` should generally be run with this option.

- `-s1` *number* : maximum number of single-hairpin motifs to generate. This number can be increased if the input sequences are large, or if a structure with many stems is expected.
- `-s2` *number* : maximum number of two-hairpin motifs to generate. This number can be increased if the input sequences are large, or if a structure with many stems is expected.
- `-minspan1` *number* : set the minimum span for single-hairpin motifs.
- `-maxspan1` *number* : set the maximum...
- `-minspan2` *number* : as above, for double-hairpin motifs
- `-maxspan2` *number* : as above, for double-hairpin motifs
- `-motifList` *file* : create the file *file* that lists, one per line, each motif file that was generated. Without this option, it is necessary to interrogate the file system to learn what motifs were predicted.

4.2 The cmfinder04 executable

The `cmfinder04` executable is similar to the `cmfinder` program from CMfinder version 0.3 (see section 3.1.2.4).

The general command line is:

`cmfinder04` *[options] seq-file*

where *seq-file* is a list of sequences in FASTA format. (Note that there is no longer an output CM file within the command line, in contrast to the version 0.3 program.)

The main options are:

- `-h` : a more comprehensive list of options
- `-a` *file* : input alignment file, in Stockholm format. This flag is required.
- `-o` *file* : output alignment file, in Stockholm format, of the final prediction.
- `--degen-keep` : keeps degenerate nucleotides as degenerate, and marginalize where necessary.
- `--fragmentary` : model fragmentary RNAs that can arise due to truncated sequences. Passes appropriate flags to the internal `cmsearch` program, and modifies the calculations of secondary structure appropriately.
- `--wgsc` : weight sequences using the GSC algorithm (implemented in Infernal 1.1). (Can be slow.)
- `--column-only-base-pair-probs` : Uses a simpler model in predicting a new secondary structure that treats columns independently (e.g., doesn't use the partition function). This can sometimes be helpful by usefully complementing predictions done without this flag.
- `--filter-non-frag` : before searching for new instances using the internal `cmsearch` program, first run an internal `cmsearch` with Infernal's `--notrunc` flag, and run the main `cmsearch` only on the hits from this pre-search. Thus, don't consider motif instances that are severely truncated.

- `--local` : run the internal `cmsearch` program in local mode.
- `--amaa` : run the internal `cmsearch` program with maximal-alignment accuracy.

The `summarize` program from version 0.3 is also implemented within `cmfinder04`. The main change is that it can use the `--fragmentary` flag, to adjust statistics with fragmented RNAs. The usage is:

`cmfinder04 --summarize` *input-sto-file*

Main flags are:

- `--fragmentary` : take into account fragmentary RNAs.
- `--summarize-gsc` : use the GSC algorithm (implemented in Infernal 1.1) to weight sequences.

Chapter 5

RNAphylo: scoring alignments

RNAphylo (also previously called "pscore") is a program that scores an alignment for evidence of a conserved RNA secondary structure. It is based on a phylogenetic model using an explicit phylogenetic tree, and its score is a LOD score. Zizhen Yao's dissertation explains the theory behind RNAphylo as of version 0.3 of this software package [9].

Within version 0.4, the main change is to extend RNAphylo to gracefully accept truncated RNAs within the input.

The basic invocation is:

`RNAphylo` *[options] input-sto-file*

where *input-sto-file* is an input alignment in Stockholm format.

Flags that seem to work well, anecdotally:

`RNAphylo --fragmentary --partition --partition-close-to-fuzzy-limit 3 --suspicious-degen-nucs 2` *input-sto-file*

(There should be a space before the `3`.)

The main options are:

- `-h` : a more comprehensive list of options
- `--fragmentary` : adjust the model to gracefully handle RNAs that are truncated on the $5'$ or $3'$ ends. Consecutive gaps in the $5'$ or $3'$ ends of a sequence within the alignment are assumed to be truncated. Implies `--only-ss-cons` and `--degen-is-absent`
- `-t` *file* : The phylogenetic tree relating the input sequences, in Newick format. See section 5.1 for more information on where this tree could come from.
- `-s` *file* : The model for single bases. The format of this file is not documented. An example is the file `data/single_model`
- `-p` *file* : The model for paired bases. An example is the file `data/pair_model`
- `-g` *file* : The input SCFG grammar file. The format is not documented. An example is `data/scfg`
- `--partition` : use the partition function in the scoring.
- `--degen-is-absent` : model degenerate nucleotide as being absent (holes in the alignment), and don't marginalize.

- `--only-ss-cons` : only allow emission of base pairs within the `SS_cons` line of the input alignment. This is necessary for `--fragmentary`; without it, we have to remove any pair that has one side in a truncated area, because this is considered absent from the alignment, and then we can't consider the remaining paired nucleotide in the singleton (unpaired) model, because that would be inconsistent, and so the alignment essentially disappears.

5.1 The phylogenetic tree input

`RNAphylo` requires a phylogenetic tree as input. The names of the terminal nodes in the phylogenetic tree must be the same as the sequence names within the input alignment.

The file `data/tree.newick` is a phylogenetic tree of vertebrates used to score predicted RNAs in those organisms. The creation and use of this tree is described in Zizhen Yao's Ph.D. dissertation[9].

For use with bacteria, a phylogenetic tree is generally unavailable. Programs from a version of Pfold [1] can be used to infer a phylogenetic tree from the alignment. To do this:

- The input alignment in Stockholm format must be converted to the "col" format, e.g., using the program `pfold-extras/fasta2col`.
- The alignment is then processed using the program `pfold-extras/findphyl` with the file `pfold-extras/pscore.rate` as the sole command-line parameter. See below for more information on this file.
- The output of `findphyl` is processed using the program `pfold-extras/mltree` again with `pfold-extras/pscore.rate` as the sole command-line parameter.
- This output must then be converted to Newick format, for `RNAphylo`.

The file `data/pscore.rate` is a rate file that can be interpreted by the `findphyl` and `mltree` programs that are part of `pfold-extras`. The file `data/pscore.rate` is a conversion of RNAphylo's `data/single_model` file into the format used by programs in `pfold-extras`.

In `data/pscore.rate`, the line of 4 numbers in the section "Single group" are the background frequencies of the 4 nucleotides (corresponding to the "Frequency" section in `data/single_model`). Following those numbers in `data/pscore.rate` are three 4×4 matrices. If these matrices are A, B and C and the "Rate Matrix" in the `data/single_model` file is M, then A is the eigenvectors of M, B is the diagonal matrix of eigenvalues of M and C is the inverse of A.

Chapter 6

hmmpair: scoring alignments using sequence-only realignment

6.1 Explanation of hmmpair method

This description of the hmmpair method appears almost verbatim in a paper in preparation as of 2016.

The basic idea of hmmpair is to evaluate covariation in an alignment to see whether the alignment of covarying base pairs is supported by adjacent sequence conservation. To evaluate alignment support, a profile HMM is trained on the alignment. In an alignment with short sequences, the profile HMM might give a high probability of aligning given nucleotides, simply because there are not many other possible alignments. Therefore, we add 200 N nucleotides on both $5'$ and $3'$ sides to each sequence in the alignment. These N nucleotides represent the 4 nucleotides with equal probability.

For each pair of aligned sequences, S_1 and S_2, and for each consensus base pair, B, we determine if the pairs corresponding to B in S_1 and S_2 are covarying. To covary, the $5'$ nucleotides in pair B in the two sequences must differ, as must the $3'$ nucleotides in pair B. Also, the pairs in both sequences must be Watson-Crick or G-U, and degenerate nucleotides (nucleotides other than A, C, G or T/U) are not allowed.

For covarying pairs, we compute a "support probability", which measures the extent to which the alignment of the covarying pairs is supported. For both sequences (S_1 and S_2), we compute the probability of emitting an alignment of that sequence to the profile HMM in which both of its paired nucleotides are placed in their original columns. The overall support probability is the product of the probabilities for each sequence, S_1 and S_2. The probabilities can be calculated using the HMM forward algorithm. The partition function is also calculated for all sequences in the alignment using code from the ViennaRNA package. The overall support probabilities are multiplied by the probabilities of the two base pairs forming. (This use of the partition function seems to work slightly better than not using it.)

The `hmmpair` score for an alignment is the sum of all support probabilities. (We also considered selecting the maximum of support probabilities, but this method seemed to work slightly less well. We did not try to normalize the sum of support probabilities by the number of sequences.)

The `hmmpair` algorithm explicitly handles the possibility of truncated sequences. It assumes that gap columns within an aligned sequences that are $5'$ to any nucleotides represent a truncation, and not merely a gap (and same thing for $3'$). Base pairs that involve truncated nucleotides are not considered. However, these pairs could not contribute to the score anyway, because the gap character is not consistent with a Watson-Crick or G-U pair.

If paired columns have non-canonical base pairs (i.e., are not Watson-Crick or G-U) in more than 5% of the sequences, the pair is considered not well conserved. In this case, no support probabilities are calculated for the pair. The frequency of non-canonical pairs is a weighted frequency, where each sequence is weighted according to the GSC algorithm as implemented by the Infernal software package. (The code is taken from the R2R package [4].)

In many lineages of bacteria, Rho-independent transcription terminators are common. As expected, their RNA stems exhibit covariation. However, such terminator hairpins are not in themselves interesting in our search for novel RNAs. Any base pair in any sequence that is entirely contained within the stem of a predicted terminator hairpin is ignored, and its support probabilities are not calculated. In this manner, terminator hairpin pairs are ignored in the `hmmpair` score, but other pairs are counted. Note that pairs involving one nucleotide with the terminator hairpin and one outside of it are counted. Such partially overlapping pairs are common mechanisms for gene regulation in cis-regulatory RNAs. Note:

- It is not as easy to apply a similar trick to allow RNAphylo to ignore terminators, since RNAphylo operates on all sequences, and only some might have a terminator. Thus, more heuristics are needed to decide when to ignore them. By contrast, the `hmmpair` algorithm naturally operates on pairs of sequences, and it is easier to ignore terminators.

- The code in this software package (CMfinder version 0.4.1.15) does not implement the predicting of transcription terminators. If a terminator is on the $3'$ end of an alignment, then the `hmmpair` program (within the CMfinder 0.4 package) might not be able to see all of the nucleotides involved in the terminator, and therefore the terminator might not be predicted. In view of this issue, another tool is needed that sees full containing sequences and reports the location of terminator hairpins to the `hmmpair` program.

6.2 Using the hmmpair program

To score an alignment, it might be easiest to use the `ScoreMotif.pl` script (see chapter 7).

The `hmmpair` command is build in the `hmmpair` directory, and is available as a symbolic link in the `bin` directory. The command line is as follows:

> `hmmpair` *sto-file max-non-canonical-pair-frequency fragmentary-policy size-of-poly-N uniform-ends terminator-positions-file*

The parameters are:

- *sto-file* : the input alignment file in Stockholm format.

- *max-non-canonical-pair-frequency* : the maximum frequency of non-canonical base pairs to allow. Base paired columns that exceed this frequency are ignored in the calculation. The value `0.05` is recommended.

- *fragmentary-policy* : one of the following single-letter values: `d` (disable fragmentary/truncated sequences), `f` (use fragmentary calculations). The value `f` is recommended.

- *size-of-poly-N* : the number of “N” nucleotides to be added on the $5'$ and $3'$ ends of each sequence in the alignment before folding and determining the partition function. The value 200 is recommended.

- *uniform-ends* : sets start/end transitions in profile HMM in a way that didn't help. The value `0` (zero) is recommended, which disables this feature.
- *terminator-positions-file* : can be the text `NULL` to indicate that no transcription terminator predictions are available. Otherwise, it is a comma-separated file in which each line gives the coordinates of the stem of a terminator. The first field in the comma-separated line is the sequence accession. The second and third values are the start and end nucleotide coordinate of the terminator stem. It does not matter if the terminator is on the sense or antisense strand.

  To use this feature, the sequence names in the alignment must be in the Rfam format *seqId*/*start*-*end* , where *seqId* is the sequence accession, *start* is the $5'$ nucleotide coordinate of the sequence and *end* is the $3'$ end. Thus, if *start* is greater than *end*, the sequence is encoded on the reverse complement strand.

  An example of a terminator file is `example/848-8255-0-0.emblcsv` , and the associated alignment file is `example/848-8255-0-0.sto`.

`hmmpair` only considers the secondary structure indicated by brackets or parentheses in the #=GC SS_cons line. Motif predictions that include pseudoknots might benefit from an extended version of hmmpair that considers these additional base pairs.

Chapter 7

ScoreMotif.pl: scoring alignments by combining RNAphylo and hmmpair

7.1 Background on ScoreMotif.pl

The script `ScoreMotif.pl` provides additional automation for running the `RNAphylo` and `hmmpair` programs, and calculates a score combining these two programs. In all cases, higher scores represent more evidence of an RNA.

The method of combining the scores is as follows. First, the `RNAphylo` and `hmmpair` scores of almost 269,000 alignments were calculated. Given a new input alignment, `ScoreMotif.pl` calculates the rank index of each score within the almost 269,000 scores.

For example, the file `examples/848-4892-0-0.sto` gets an `hmmpair` score of 2.98024. This is the 38468th-best score out of the 269,000 alignments. The same example file gets an `RNAphylo` score of `2.43`, which is the 139706th-best `RNAphylo` score. Calculating the rank orders is one way of trying to normalize the different scales.

To calculate the combined score, `ScoreMotif.pl` just takes the minimum (better) rank, and then reports the negative of that number. Thus, in this case, $38468 < 139706$, so the final combined score is -38468. Taking the negative means that higher scores are better.

7.2 Example commands with ScoreMotif.pl

Running

```
perl bin/ScoreMotif.pl examples/848-4892-0-0.sto
```

gives `hmmpair` score 2.98024, `RNAphylo` score `2.43` and combined score -38468.

Running

```
perl bin/ScoreMotif.pl examples/848-8255-0-0.sto
```

gives `hmmpair` score 16.372, `RNAphylo` score `354.51` and combined score -105. Note: this command can take about a minute to run.

However, this latter alignment has predicted terminators within the $3'$ hairpin. Running

```
perl bin/ScoreMotif.pl -terminators examples/848-8255-0-0.emblcsv
examples/848-8255-0-0.sto
```

gives `hmmpair` score 13.4182, `RNAphylo` score `354.51` (again) and combined score -394.

Adding the `-terminators` flag leads to a poorer `hmmpair` score, because `hmmpair` will ignore the base pairs in the predicted terminators (which is the $3'$ hairpin). However, even with this reduction, the motif is very good, and indeed corresponds to a MoCo riboswitch. The use of the `-terminators` flag does not affect the `RNAphylo` score, because it is not as easy to ignore terminators for this program.

7.3 Command line for ScoreMotif.pl

The basic command is:

`perl bin/ScoreMotif.pl` *[flags] sto-file*

where *sto-file* is the input alignment in Stockholm format, and the following flags are recognized:

- `-dataDir` *dir* : directory for data files
- `-rnaphyloDataDir` *dir* : directory for RNAphylo data files, which is just `pscore.rate`
- `-hmmpairDb` *file* : the file containing a large number of hmmpair scores in which ScoreMotif.pl should calculate the rank. The file is made in Perl using the DB_FILE module with the $DB_TREE method, with $DB_TREE->{'compare'} set to the <=> function.
- `-rnaphyloDb` *file* : as for `-hmmpairDb`, the file containing scores for RNAphylo
- `-terminators` *file* : comma-separated file containing positions of predicted transcription terminators, whose pairs hmmpair should ignore. The format of this file is explained in section 6.2 under the description of the *terminator-positions-file* command line parameter.
- `-pfoldRateFile` *file* : the rate file for inferring phylogenetic code.
- `-tempBase` *path* : a prefix of paths to use to creating temporary files. By default it is /tmp/PID , where PID is the process ID of the `ScoreMotif.pl` script.
- `-pfoldPath` *dir* : directory containing executables that are part of a version of Pfold that inferred phylogenetic trees.
- `-hmmpairExe` *file* : the executable hmmpair file.
- `-rnaphyloExe` *file* : the executable RNAPhylo file

Chapter 8

Guide to the code

8.1 Guide to the code in cmfinder04

8.1.1 Interface with Infernal 1.1

A part of the EM-based algorithm of CMfinder requires performing CM searches and creating/using alignments. Some of these operations essentially translate to running programs within the Infernal 1.1 software package. For speed and convenience, the relevant programs have been converted so that their "`main`" functions can be executed as subroutines within the `cmfinder04` executable.

The source code files `from_cmbuild.c`, `from_cmcalibrate.c` and `from_cmsearch.c` implement this transformation of the Infernal programs `cmbuild`, `cmcalibrate` and `cmsearch`, respectively.

The interface to `cmbuild` is implemented by the function `cmbuild` in `from_cmbuild.c`. The interface to `cmsearch` is implemented by the function `cmsearch_wrapper` in `global.c`. A partial interface to `cmcalibrate` is implemented by the function `cmcalibrate_viterbi` in `global.c`. The use of `cmcalibrate` was an experiment, and is not recommended in general, because it makes `cmfinder04` run much more slowly.

8.1.2 Source files

`cmfinder04` is built from the following source files:

- `cmfinder.c` : has `main` function and functions to process the command line. After processing the command lines, the main Expectation Maximization loop starts with the first call to `M_step`.
- `cmfinder.h` : `#include`s most files.
- `em.c` : most functions implementing aspects of Expectation Maximization.
- `cand.c` and `cand.h` : Various operations on candidates, i.e., predicted instances of a motif within a specific sequence.
- `base_pair_prediction.c` : Functions related to predicting base pairs, including partition function-based probabilities and mutual information used in the EM algorithm, and statistics computed by the `--summarize` functionality.
- `from_cmbuild.c` : converts the file `cmbuild.c` (from Infernal) to something that can be run as a subroutine. (See previous section.)

- `from_cmcalibrate.c` : converts the file `cmcalibrate.c` (from Infernal) to something that can be run as a subroutine. (See previous section.)
- `from_cmsearch.c` : converts the file `cmsearch.c` (from Infernal) to something that can be run as a subroutine. (See previous section.)
- `global.c` : miscellaneous functions, including some functions related to allowing the execution of Infernal programs as subroutines. The function `save_and_load_CM_via_memory_copying` implements a hack that simulates allows a CM to be re-used for multiple searches, circumventing a sanity check that makes sense in the context of how Infernal's code is supposed to be used.

8.2 Guide to the code in RNAphylo

Note that the term "absent" refers to nucleotides that are unknown because they are degenerate or because they are truncated. In most cases, "absent" nucleotides are treated as holes in the alignment. (In other words, they are considered as never having been present. In contrast, a more typical formulation is to consider them as missing information that must be inferred in a Bayesian manner.)

RNAphylo is made up of the following source files:

- `RNAphylo.c` : The `main` function is here, and this function drives most of the calculation, after processing the command line and initializing variables. The calculation proper starts after the line

```
//Conserved singlet model
```

- `phytree.c` and `phytree.h` : Functions for actually processing the phylogenetic calculations, including calls to QRNA functions to manipulate rate matrices.
- `my_matrix.c` and `my_matric.h` : generic matrix calculations (i.e., not specific to rate matrices).
- `grammar.c` and `grammar.h` : processing of the SCFG, including loading/saving, and dynamic-programming calculations.
- `global.c` and `global.h` : miscellaneous functions.
- `get_tm_freq_cache.cpp` : implements caching of rate matrix calculations, which are frequently repeated. This file is implemented in C++, because of the convenience of STL classes like `map` for implementing the cache. Note: although it's referred to as a cache, cached information is never evicted.

8.3 Guide to the code in hmmpair

For historical reasons, hmmpair is linked against code from Vienna RNA version 1.7.1, while the CMfinder and RNAphylo programs are linked against Vienna RNA version 1.4.

The source files are as follows:

- `AlignmentConsensusAndScoring.cpp` : This is the most important code for implementing the hmmpair method. The function `HmmForwardScore_SS` is the main entry point. The struct `DegappedSeqFromMsa` deals with de-gapping the input aligned sequences, adding flanking poly-N sequences, and maintains maps between the positions in the input alignment and positions in the de-gapped sequences. The class `PartitionFunctionLookup` creates a lookup table to store base-pairing probabilities using the partition function implemented by the Vienna RNA library. The class `PositionsToIgnore` implements the logic to ignore base pairs within predicted Rho-independent transcription terminator hairpins. The function `DetermineSeqBoundsForFragmentary` handles aspects of truncated sequences. The function `HmmForwardScore_SS_MakeHmm` makes a profile HMM that will be used to analyze the alignment.

- `GSCConsensus.cpp` : this source file calculates consensus information from the alignment after weighting sequences according to the GSC algorithm. The method was previously used by the R2R program [4], and the source code is largely identical to a source file from R2R.

- `hmmpair.cpp` : contains the `main` function, parses command line, and has a few utility functions.

- `Cm2HMM.cpp` : code for converting CMs into profile HMMs. Only some of the code in this file and the files below is actually used. The code is from the RaveNnA package [6, 7, 8].

- `Cm2HmmOptimize.cpp` : code for optimization of converted profile HMMs.

- `CommaSepFileReader.cpp` : code to read comma-separated files.

- `CovarianceModel.cpp` : a class that wraps the CM structure implemented by Infernal.

- `InfernalHmm.cpp` : a class that partly implements profile HMMs based on the Infernal code for CMs.

- `HmmType1.cpp` : a profile HMM class that is faster to use for scanning.

- `MarkovModelStats.cpp` : implements Markov models.

- `MiscExceptions.cpp` : utilities related to C++ exceptions.

- `MLHeuristic.cpp` : code related to the Maximum-Likelihood (ML) heuristic for setting profile HMM transition scores based on a CM [6].

- `NaryCounter.cpp` : counts in an arbitrary numeric base.

- `NoUnderflowDouble.h` : A numeric class that uses extra bits for exponents to avoid underflow errors.

- `Optimize.cpp` : code for numeric optimization.

- `ScanHmm.cpp` : implements Viterbi and forward algorithms for profile HMMs of various flavors.

- `SymbolicMath.cpp` : implements a numerical type that is a symbolic expression.

Bibliography

[1] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammar. *Nucleic Acids Res*, 31:3423–3428, 2003.

[2] E. Rivas and S. R. Eddy. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(1):8, 2001.

[3] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, Nov. 1994.

[4] Z. Weinberg and R. R. Breaker. R2R—software to speed the depiction of aesthetic consensus RNA secondary structures. *BMC Bioinformatics*, 21:3, 2011.

[5] Z. Weinberg, C. E. Lünse, K. A. Corbino, T. D. Ames, J. W. Nelson, A. Roth, K. R. Perkins, M. E. Sherlock, and R. R. Breaker. Detection of 224 candidate structured RNAs by comparative analysis of specific subsets of intergenic regions. *Nucleic Acids Res*, 45(18):10811–10823, 2017.

[6] Z. Weinberg and W. L. Ruzzo. Faster genome annotation of non-coding RNA families without loss of accuracy. *RECOMB 2004*, pages 243–251, 2004.

[7] Z. Weinberg and W. L. Ruzzo. Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics*, 20(suppl_1):i334–i341, 2004. ISMB 2004.

[8] Z. Weinberg and W. L. Ruzzo. Sequence-based heuristics for faster annotation of non-coding RNA families. *Bioinformatics*, 22(1):35–39, 2006.

[9] Z. Yao. *Genome scale search of noncoding RNAs Bacteria to Vertebrates.* PhD thesis, University of Washington, 2008.

[10] Z. Yao, Z. Weinberg, and W. L. Ruzzo. CMfinder–a covariance model based RNA motif finding algorithm. *Bioinformatics*, 22(4):445–452, Feb. 2006.