# RAF 1.00
## User Manual

(Last modified: August 15, 2008)

# Contents

# 1 Description

RAF (RNA Alignment and Folding) is a program for multiple RNA alignment and folding. Given a set of homologous, unaligned RNA sequences, RAF compute a multiple alignment and consensus structure prediction using a progressive simultaneous alignment and folding algorithm.

The RAF program was developed by Chuong Do and Chuan-Sheng Foo at Stanford University. The source code for RAF is available for download from

*http://contra.stanford.edu/contrafold/*

under the BSD license.

Any comments or suggestions regarding the program should be sent to Chuong Do (*chuongdo@cs.stanford.edu*).

## 2 License (BSD)

# 3 Installation

At the moment, RAF is only available for Unix-based systems (e.g., Linux).

## 3.1 *nix installation

To compile RAF from the source code (for a *nix machine):

1. Download the latest version of the RAF source code from

   *http://contra.stanford.edu/contrafold/download.html*

2. Decompress the archive:

   ```
   $ tar zxvf raf_v#_##.tar.gz
   ```

   where the #'s are replaced with the appropriate version numbers for the tar.gz you want to install. This will create a subdirectory called `raf` inside of the current directory.

3. Change to the `raf/src` subdirectory, and compile the program.

   ```
   $ cd raf/src
   $ make clean
   $ make
   ```

4. Install the CONTRAfold ($\geq$ v2.02) and CONTRAlign ($\geq$ v2.01) programs, available from

   *http://contra.stanford.edu/contrafold/download.html*
   *http://contra.stanford.edu/contralign/download.html*

   Set the `CONTRAFOLD_DIR` environment variable to point to the directory containing the `contrafold` executable, and set the `CONTRALIGN_DIR` environment variable to point to the directory containing the `contralign` executable.

Now, your installation is complete!

# 4 Supported file formats

In this section, we describe the input and output file formats supported by the RAF program.

## 4.1 Input file formats

RAF has different input file formats for prediction and for training.

- For prediction, RAF accepts input files in MFA format. The sequences in the MFA format should be unaligned (containing no gaps).
- For training, RAF accepts input files in MFA format, augmented with a consensus secondary structure. The sequences in the MFA format must be pre-aligned (i.e., gap characters already inserted). The consensus secondary structure must not contain pseudoknots.

In the following subsections, we describe the prediction and training input MFA formats.

### 4.1.1 MFA format (for prediction)

An MFA (Multi-FASTA) format file for prediction consists of a collection of two or more FASTA-formatted sequences. Each sequence consists of:

1. A **single header line** beginning with the character '>' followed by a text description of the sequence. Note that the description must fit on the same line as the '>' character.
2. One or more lines containing **RNA sequence data**. Each of these lines may contain the letters 'A', 'C', 'G', 'T', 'U' or 'N' in either upper or lower case, any T's are automatically converted to U's, and all other letters are automatically converted to N's.

   The output of the program will retain the case of the input. All whitespace (space, tab, newline) is ignored. N's are treated as masked sequence positions which are ignored during all calculations (i.e., any scoring terms involving an N will be skipped). Other non-whitespace characters are not permitted.

For example, the following is a valid MFA file for prediction

```
>seq1
acguuggcu
>seq2
gCGUCu
```

But the following is not a valid MFA file (starts with the wrong header character):

```
# seqA
ATGACGGT
# seqB
GATCCAGGATACAG
```

Similarly, the following is not a valid MFA file (contains too few sequences):

```
>sequence
acggagaGUGUUGAU
CUGUGUGUUACUACU
caucuguaguucuag
uugua
```

#### 4.1.2 MFA format (for training)

An MFA (Multi-FASTA) format file for training consists of a collection of two or more pre-aligned FASTA-formatted sequences plus an annotated consensus structure. Each of the sequences and the secondary structure are specified as follows:

1. A **single header line** beginning with the character '>' followed by a text description of the sequence. Note that the description must fit on the same line as the '>' character. In the case of the consensus secondary structure, any description is acceptable; RAF will automatically detect the presence of the secondary structure based on the contents of the provided sequence (in particular, the secondary structure is taken to be the only sequence whose character data does not contain any letters).

2. One or more lines containing either **RNA sequence data** or a **consensus secondary structure**.

   - In the case of RNA sequence data, each of the lines may contain gap characters (see below) or the letters 'A', 'C', 'G', 'T', 'U' or 'N' in either upper or lower case, any T's are automatically converted to U's, and all other letters are automatically converted to N's.

     The output of the program will retain the case of the input. All whitespace (space, tab, newline) is ignored. N's are treated as masked sequence positions which are ignored during all calculations (i.e., any scoring terms involving an N will be skipped). Other non-whitespace characters are not permitted.
   - In the case of a consensus secondary structure, each of the lines may contain the characters '(', ')', or '.'. A nucleotide annotated with '(' pairs with the nucleotide annotated with the matching ')'. A '.' character indicates that the corresponding nucleotide is unpaired. Observe that the parentheses in the input file must be well-balanced, i.e., for each left parenthesis, the corresponding pairing position must be marked with a right parenthesis, and vice versa.

Since RAF deals with only non-pseudoknotted structures, the proper pairing will always be unambiguous.

3. In order to specify a pre-existing alignment, **gap characters** may be included in the sequence data above. In particular, valid gap characters include '.' and '-'. An MFA file specifying a gapped multiple sequence alignment should have the same total number of characters of sequence data (i.e., letters plus gaps) for each sequence.

For example, the following is a valid MFA file for training:

```
>seqA
aggcacgu
>seqB
agg--aug
>seqC
gaccgu.g
>structure
(((...))
```

However, the following is not (sequences of different length):

```
>seqA
aggcacgu
>seqB
agg--augg
>seqC
gaccgu.g
>structure
(((...))
```

Also, the following is invalid (too many structures provided)

```
>seqA
aggcacgu
>structureA
(((...))
>seqB
agg--aug
>structureB
(((...))
>seqC
gaccgu.g
>structureC
(((...))
```

## 4.2 Output formats

The results of a RAF alignment prediction are identical to the MFA for training format described in section 4.1.2.

# 5 Usage

RAF has two modes of operation: prediction mode and training mode.

- In "prediction" mode, RAF aligns a set of sequences using the default parameters or parameters specified on the commmand-line.
- In "training" mode, RAF learns new parameters from training data consisting of pre-aligned sequences.

Most users of this software will likely only ever need to use RAF's prediction functionality.

## 5.1 Prediction mode

In prediction mode, RAF computes multiple alignments for a given input sequence set, and prints the result to the console (i.e., stdout). The basic syntax for running RAF in prediction mode is

```
$ ./raf predict [OPTIONS] INFILE
```

For example, suppose the file "seq.mfa" contains a set of sequences to be folded. Then the command

```
$ ./raf predict seq.mfa
```

will fold and align the sequence and display the results to the console in MFA format.

### 5.1.1 Optional arguments

RAF accepts a number of optional arguments, which alter the default behavior of the program. To use any of these options, simply pass the option to the RAF program on the command line. For example,

```
$ ./raf predict --noncomplementary seq.mfa
```

The optional arguments include:

`--noshell` $\gamma$

This options turns off the use of alignment constraints during folding.

`--noncomplementary`

This option permits the use of non-{AU,CG,GU} base-pairings from being considered by CONTRAfold.

`--align_cutoff` $x$

This option sets the minimum posterior probablity threshold for aligned match positions to be $x$.

`--bp_cutoff` $x$

This option sets the minimum posterior probability threshold for base-pairing positions to be $x$.

`--vienna`

This option tells RAF to use ViennaRNA instead of CONTRAfold as the secondary structure folding algorithm for computing pairing posterior probabilities. The environment variable `VIENNA_RNA_DIR` must point to the directory where the `RNAfold` executable can be found. Note that in this case, it will be necessary to supply different program parameters in order to achieve good performance; the RAF program will not produce accurate results if run as is. Generally, this option gives lower accuracy even with a fully trained RAF model.

`--min_hairpin_len` $k$

This option sets the minimum length of a valid hairpin as considered by CONTRAfold.

`--num_ir` $k$

This option enables the use of $k$ steps of iterative refinement following the progressive alignment pass.

`--version`

Display the program version number.

`--verbose`

Show detailed console output.

With the exception of `--num_ir`, each of the above options is also applicable in training mode.

## 5.2 Training mode

In training mode, RAF infers a parameter set using multiple sets of aligned RNA sequences with known consensus secondary structures. By default, RAF uses a projected subgradient algorithm for optimization.

For example, suppose `input/*.mfa` refers to a collection of 100 files which represent aligned sequence sets with known structures. Calling

```
$ ./raf train input/*.mfa
```

instructs RAF to learn parameters for predicting all alignments and structures in

```
input/*.mfa
```

without using any regularization. The learned parameters after each iteration of the optimization algorithm are displayed on the console. *In general, running RAF without regularization is almost always a bad idea because of overfitting*.

### 5.2.1 Optional arguments

RAF training mode supports a number of optional arguments:

`--regularize` $C_1$ $C_2$

This option instructs RAF to learn using a penalty function of the form $\ell(\boldsymbol{w}) + C_1||\boldsymbol{w}||_1 + \frac{1}{2}C_2||\boldsymbol{w}||^2$, where $\ell(\boldsymbol{w})$ is the loss on the training set and the remaining terms are $L_1$ and $L_2$ regularization penalties.

`--num_epochs` $k$

This option sets the number of epochs over which optimization will be run to $k$.

`--subset_size` $k$

This option determines the size of the subset of sequences to be sampled for each step of the stochastic subgradient algorithm. By default, all sequences are used (i.e., batch subgradient optimization).

`--weights` $w_1$ $w_2$ `...`

This option sets the weights of the model to $w_1, w_2, \ldots$

`--eta` $\eta$

This option sets the base learning rate of the algorithm to $\eta$.

In general, we recommend that you do not perform training yourself unless you know what you are doing; also do not hesitate to ask us.

# 6 Citing RAF

If you use RAF in your work, please cite:

> Do, C.B., Foo, C.-S., and Batzoglou, S. (2008) A max-margin model for efficient simultaneous alignment and folding of RNA sequences. *Bioinformatics* 24: i68-i76.

Other relevant references include:

> Do, C.B., Woods, D.A., and Batzoglou, S. (2006) CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14): e90-e98.

> Do, C.B., Gross, S.S., and Batzoglou, S. (2006) CONTRAlign: Discriminative training for protein sequence alignment. In *Proceedings of the 10th Annual International Conference on Computational Molecular Biology (RECOMB)*, 160-174.