A Practical Guide to Using TBA

Elliott H. Margulies Minmei Hou Webb Miller

April 28, 2005

This document is meant to help users understand the practical aspects of running TBA. Please see the README file provided with the source code for additional details.

Contents

1 Introduction

TBA, or the Threaded-Blockset Aligner, is a *local* multiple sequence alignment tool that differs somewhat from a global sequence aligner. TBA is in fact a suite of tools, not just one single executable file. While TBA can be used to align "vanilla" sequences from different species (i.e. one FASTA sequence entry per species), it can also handle more complex scenarios such as 1) aligning sub-sequences that originate from particular chromosomes of genome assemblies, or 2) situations where any species' sequence might be provided in more than one FASTA sequence entry. The latter might occur, for example, if there is a syntenic break in one of the species being analyzed or your sequence is a series of unassembled contigs. With TBA, there is no need to artificially fuse these multiple sequences together (i.e. with a rung of N's between the contigs).

2 The input to TBA

The starting material for running TBA includes:

1. A set(s) of sequences from different species
2. An evolutionary tree describing the relationship of these species
3. A parameter file describing how to run `blastz` for different species. For example, you might want to treat alignments involving non-placental mammals differently from placental mammals. This file is optional.

2.1 The sequence data

2.1.1 File naming convention

You should have one file per species. The name of this file should correspond to the species name that identifies the source of the sequences contained within. This in turn should correspond to the name used in the evolutionary tree. For example, if you are aligning sequences from human, mouse, and rat, name the files (or create links to your source files) human, mouse, and rat respectively. See below for tree format.

2.1.2 FASTA Header Formats

- In the simplest form of running TBA, you should supply one FASTA sequence entry per file. With this scenario, the FASTA header does not need to (and probably should not) conform to any particular format described below.

- If you will be providing TBA with multiple sequence entries per species, or want to denote that your sequences are part of some larger genome assembly, you can format the header of your FASTA entries as follows:

  ```
  >string1:string2:int1:char:int2
  ```

  string1 is the source organism identifier, usually the species name. This must be identical to the name of the sequence file.

  string2 is the contig name. Every FASTA entry in a species' sequence file must be uniquely identified. This is handled here, and can be as simple as a single number or letter, or more frequently the contig/chromosome name of origin.

  int1 is a 1-based offset value. Typically this value is set to 1. But, if your sequence is a sub-sequence of some larger contig/chromosome, you can specify its starting offset here. In this fashion, the coordinates of your resulting multiple alignment will be in the context of other genome-wide annotations and is thus very useful (despite being a bit confusing to the novice TBA user).

  char simply distinguishes whether this sequence comes from the top or bottom strand (+ or - respectively, and usually set to +).

  int2 is the size of the entire contig specified in **string2**. For example this would be the length of the entire chromosome from which your sub-sequence originates.

  If you never deal with sub-sequences from genome assemblies, then **int1** and **int2** are always 1 and the sequence length respectively.

- An alternative header format accepted by TBA is that adopted by the ENCODE MSA group. See http://hgwdev.cse.ucsc.edu/encode/downloads/msa/APR-2005/ for details of this header format, which has all the above necessary information (and more) embedded within its structure.

2.2 The evolutionary tree

This must be a binary tree described in a modified Newick format (see
http://evolution.genetics.washington.edu/phylip/newicktree.html
for details) where the branch lengths are removed, spaces are substituted for commas, and the ending semi-colon is removed. An example of such a tree is:

```
(((((((human chimp) gorilla) baboon) (rat mouse)) (cow pig)) chicken) fugu)
```

The latest version of a tree we use to analyze NISC-based comparative sequence data is available at ftp://kronos.nhgri.nih.gov/pub/outgoing/elliott/tba/aux/

2.3 The optional `blastz` parameter file

Known as "the blastz specs file", it contains optional command line parameters for `blastz` when different species' sequences are being aligned. Here is an example of such a file:

```
# This is a sample "blastz.specs" file
#define PLACENTAL human chimp gorilla baboon rat mouse cow pig
#define NON_PLACENTAL chicken fugu
PLACENTAL : PLACENTAL
        B=2 C=0
NON_PLACENTAL : *
        Q=HoxD55.q
```

This file states that when both species being aligned are "PLACENTAL", the `B=0 C=2` options will be used for `blastz`. When one of the species is "NON_PLACENTAL", the `Q=HoxD55.q` option will be used. The latest version of this file we use to analyze NISC-based comparative sequence data is available at ftp://kronos.nhgri.nih.gov/pub/outgoing/elliott/tba
Please look at this file for formatting details as the program is particular about tabs vs. spaces and does not let you know.

3 Overview of running TBA

There are typically three steps involved with generating a multiple alignment:

1. generating a series of pair-wise alignments to "seed" the multiple alignment process
2. generating the multiple alignment
3. "projecting" the alignment onto a reference sequence

These are described in detail below:

3.1 Generating pair-wise alignments

This process is performed with the program `all_bz`, which is essentially a program that executes a series of `blastz` commands.

- The exact pair-wise combinations of species needed to seed the multiple alignment are determined from the evolutionary tree.
- And "the blastz specs file" is also optionally used to determine which parameters should be used for each pair-wise `blastz`-generated alignment.
- Note again that the names of the species used in the tree and "the blastz specs file" **must** correspond to the filenames of your sequence data (and **string1** if you choose to use a formatted header.

Given the above information, `all_bz` will write out a series of `blastz` commands to be executed. By default, `all_bz` will execute each command directly in a serial fashion (use the + option to see what is actually being executed). Alternatively, if you have a cluster of CPUs available to you, and the number of species being aligned is large, you can use the - option to simply create a list of `blastz` commands to be executed. This list can then be processed in a suitable fashion to be parallelized on your particular system. Here is a typical example of how I would run `all_bz`:

```
all_bz - "(((((((human chimp) gorilla) baboon) (rat mouse)) (cow pig)) \
    chicken) fugu)" blastz.specs >&all_bz.log
```

3.1.1 Anatomy of an Alignment Command from `all_bz`

Read this subsection only if you are interested in what `all_bz` is doing.

Here is the first pair-wise alignment command from the above `all_bz` execution:

```
Blastz human chimp Y=3400 H=2000 B=2 C=0 | lav2maf /dev/stdin human chimp \
    | single_cov2 /dev/stdin > human.chimp.maf
```

A note on Big "B" `Blastz`: You will notice that the commands being executed by `all_bz` are in fact running big "B" `Blastz`, which *is* different from little "b" `blastz`. The big "B" version is simply another script that executes a series of little "b" `blastz` commands. This gets around a current limitation of little "b" `blastz` in that it only aligns the first FASTA sequence entry of sequence 1. With big "B" `Blastz`, all FASTA entries in sequence 1 are aligned to all FASTA entries in sequence 2 by running little "b" blastz several times.

Post-processing of the Pair-wise Alignment: The default alignment format for `blastz` is "lav" format. This is first converted to a MAF with `lav2maf` and then processed with `single_cov2`. The latter removes lower-scoring alignable in regions with more than one resulting alignment such that each position in sequence 1 aligns to only one position in sequence 2. The “2” in `single_cov2` simply denotes the version that runs on MAF files instead of lav files.

The resulting pair-wise alignment MAF file is named species1.species2.maf. This naming convention is understood by downstream programs. Again, note the importance of naming your sequence input files something short and sweet. While longer names will work, its much more “elegant” to use some short species name.

3.2 Creating the multiple alignment

This is done with `tba`. Like the other programs in this package, `tba` runs an iterated series of other programs to create the resulting multiple alignment. These include `get_covered`, `multiz`, `pair2tb`, `maf_project`, and several other standard unix shell commands. It requires you to specify the evolutionary tree, all the pair-wise alignments (I typically do this with *.*.maf) and the name of the resulting multiple alignment file. A typical `tba` command would be:

```
tba "(((((((human chimp) gorilla) baboon) (rat mouse)) (cow pig)) chicken) \
    fugu)" *.*.maf tba.maf >&tba.log
```

3.3 “Projecting” the Multiple Alignment

The resulting TBA multiple alignment is a complex entity, which may contain inversions, etc. Frequently, it is useful to represent a multiple alignment in the context of some reference sequence, where by all alignment blocks are re-ordered to follow one of the species' sequences in the alignment. This process (performed by using `maf_project`) removes alignment blocks that do not contain the projected species.

Note, that this does not bias the multiple alignment towards the projected sequence; it is still a true multiple alignment. This process merely lets you look at the multiple alignment in the context of one species' sequence.

4 Other Useful Tools

The resulting TBA multiple alignment is now ready for analysis. Several tools (in addition to the above mentioned `maf_project`) are available to aid you in this process and are described in the README. Several are worth noting here:

`maf_order` orders the species in each alignment block to some specified order. It will also remove unwanted species from the multiple alignment (rather than rerunning tba with a subset of species).

`mafFind` extracts a sub-region of the multiple alignment.

`maf2fasta` takes a projected MAF file and re-formats it to either MultiPipMaker format or multi-FASTA format, representing the entire reference species' sequence, but only the alignable portions of all other species' sequences. This program is particularly useful since the MAF specification is relatively new

(see http://genome.ucsc.edu/goldenPath/help/maf.html)

and is not yet as universal as the FASTA format.