CAFE: Computational Analysis of gene Family Evolution

Tutorial

Nov 19, 2016

Contents

1 This tutorial **3**

1.1 Dependencies 3

2 Preparing the input **4**

2.1 Downloading the data 4

2.2 Finding gene families 4

2.2.1 Moving all longest isoforms into a single file 5

2.2.2 All-by-all BLAST 5

2.2.3 Clustering sequences with mcl 6

2.2.4 Final parsing of mcl's output 6

2.3 Estimating a species tree 7

2.3.1 Making the species tree ultrametric 8

3 Running CAFE **9**

4 Acknowledgments **10**

1 This tutorial

This document contains a tutorial that should help you get your hands dirty with CAFE. The tutorial is divided into two parts:

1. ***Preparing an input dataset that CAFE understands***: this is most of the work, and makes use of auxiliary Python scripts (which we provide) and a few other programs;

2. ***Running CAFE***: performing basic evolutionary inferences about gene family evolution.

1.1 Dependencies

The tutorial assumes you are running a *Unix-based operating system*. It also assumes you have a *local working version of CAFE* (please see CAFE's manual for instructions on how to install it), but also of a few other software that are necessary for the first part of the tutorial:

- Python 2.7.x;
- BLAST;
- mcl;
- r8s.

2 Preparing the input

2.1 Downloading the data

This tutorial and the scripts we provide assume you will use sequences in FASTA format (`.fa`) downloaded from Ensembl using the Biomart tool. To download the protein sequences from, say, cat (*Felis catus*), you must navigate Biomart:

1. *CHOOSE DATABASE* → *Ensembl Genes 87* → *CHOOSE DATASET* → *Cat genes*;
2. Then click *Attributes* → *Sequences: Peptide* → *Header information: Gene ID + CDS length* (uncheck *Transcript ID*);
3. Finally, click *Results*. If you have a good internet connection, choose *Compressed file (.gz)*, otherwise choose *Compressed web file* and provide your email address.

We are going to analyze data from 12 species: mouse, rat, cow, horse, cat, marmoset, macaque, gibbon, baboon, orangutan, chimpanzee, and human. It should not take too long to download the 12 files, but we are providing you with a tarball (`twelve_spp_proteins.tar.gz`) containing all of them.

In order to decompress `twelve_spp_proteins.tar.gz` (it will create its own folder), open your shell, move to the folder into which you downloaded the file, and enter:

```
$ tar -zxvf twelve_spp_proteins.tar.gz
$ cd twelve_spp_proteins
$ for i in `ls -1 *.tar.gz`; do tar -zxvf $i; done
```

2.2 Finding gene families

Finding gene families within and among species requires a few steps. First, we need to deal with alternative splicing and redundant gene entries by removing all but the longest

isoform for each gene. After this is done for all 12 species, we shall move all sequences to a single file and prepare a database for BLAST. BLAST will allow us to find the most similar sequence for each sequence in the database (all-by-all blastp). Then we employ a clustering software, mcl, to find groups of sequences (gene families) that are more similar among themselves then with the rest of the dataset. Finally, we parse mcl's output to use as input for CAFE.

2.2.1 Moving all longest isoforms into a single file

In order to keep all but the longest isoforms, and place all sequences from all species into a single `.fa` file for the next tutorial step, run the following commands in your shell:

```
$ cd /path/to/twelve_spp_proteins/
$ python /path/to/cafetutorial_longest_iso.py ./
$ cat longest*.fa > makeblastdb_input.fa
```

2.2.2 All-by-all BLAST

With `makeblastdb_input.fa` in hand, we can now prepare a database for BLAST, and then run `blastp` on it, all sequences against all sequences. This step gives us, for each sequence, the most similar sequence (in addition to itself) in the dataset. We can then find clusters of similar sequences from these similarity scores (see next step). To prepare the database, move to the directory containing `makeblastdb_input.fa` and run the following command on your shell:

```
$ makeblastdb -in makeblastdb_input.fa -dbtype prot -out blastdb
```

This command should create a few files. From here, we can actually run `blastp` (on, say, four threads) with:

```
$ blastp -num_threads 4 -db blast.db -query makeblastdb_input.fa
-outfmt 7 -seg yes > blast_output.txt
```

The `-seg` parameter filters low complexity regions (aminoacids coded as X) from sequences. Because this job can take many hours, we will provide you with its output (a tarball named `blast_output.tar.gz`) so you can keep on doing the tutorial.

2.2.3 Clustering sequences with mcl

Now we must use the output of BLAST to find clusters of similar sequences. These clusters will essentially be the gene families we will analyse with CAFE. Clustering is done with a program called mcl, running a few commands:

```
$ grep -v "#" blast_output.txt | cut -f 1,2,11 > blast_output.abc
$ mcxload -abc blast_output.abc --stream-mirror --stream-neg-log10 -
   stream-tf
'ceil(200)' -o blast_output.mci -write-tab blast_output.tab
$ mcl blast_output.mci -I 3
$ mcxdump -icl out.blast_output.mci.I30 -tabr blast_output.tab -o
dump.blast_output.mci.I30
```

In the first command above, we convert the blast output into ABC format, which mcl understands. Then we give the .abc file to `mcxload`, which creates a network and a dictionary file (`.mci` and `.tab`, respectively). These two files are then used by mcl to perform the clustering, which is done with the last two commands.

The `-I` (inflation) parameter determines how granular the clustering will be. Lower numbers means denser clusters, but again, this is an arbitrary choice. A value of 3 usually works, but you can try different values and compare results. Ideally, one wants to be able to identify clusters containing the same number of genes as there are species, as these are likely to represent correct one-to-one ortholog identifications (in our case, we want clusters with 12 species). Furthermore, we want to minimize the number of clusters with just a single sequence.

2.2.4 Final parsing of mcl's output

The file obtained in the last section (the dump file from mcl) is still not ready to be read by CAFE: we need to parse it and filter it. Parsing is quite simple and just involves tabulating the number of gene copies found in each species for each gene family. We provide a script that does it for you. From the directory where the dump file was written, run the following commands:

```
$ python cafetutorial_mcl2cafe.py dump.blast_output.mci.I30 -o
unfiltered_cafe_input.txt -sp "ENSG00 ENSPTR ENSPPY ENSPAN ENSNLE
ENSMMU ENSCJA ENSRNO ENSMUS ENSFCA ENSECA ENSBTA"
```

Then there are two final filtering step we must perform. Gene families that have large gene copy number variance can cause parameter estimates to be non-informative. You can remove gene families with large variance from your dataset, but we found that putting aside the gene families in which one or more species have ≥ 100 gene copies does the trick. In addition, when CAFE performs ancestral state reconstruction on gene family sizes, it also requires that for each tree internal node of interest, at least two descendant species have non-zero family sizes. You can do both filtering steps with another script we provide, but in this tutorial we shall only do the former:

```
$ python cafetutorial_clade_and_size_filter.py -i
unfiltered_cafe_input.txt -o filtered_cafe_input.txt -s
```

As you will see, the script will have created two files: `filtered_cafe_input.txt` and `large_filtered_cafe_input.txt`. The latter contains gene families where one or more species had ≥ 100 gene copies. We can now run CAFE on `filtered_cafe_input.txt`, and use the estimated parameter values to analyse the large gene families that were set apart in `large_filtered_cafe_input.txt`.

2.3 Estimating a species tree

Estimating a species tree takes a number of steps. If genome data is available for all species of interest, one will need sequence alignments (with one sequence per species, from hopefully many genes) and then choose one among the many available species tree estimation methods. Obtaining alignments usually requires finding one-to-one ortholog clusters with mcl (see previous section), but other procedures exist. Alternatively, pre-aligned one-to-one ortholog data from Ensembl or UCSC Genome Browser can sometimes be found and readily used.

With alignments in hand, one could concatenate all alignments and infer a non-ultrametric species tree with a maximum-likelihood (e.g., RAxML or PhyML) or Bayesian phylogenetic software (e.g., MrBayes). This approach might fail if many species you are working with are very closely related and incomplete lineage sorting is rampant. In these cases, coalescent-based methods can be used instead (e.g., "shortcut" methods such as MP-EST and Astral-2, or full coalescent methods such as BPP and *BEAST).

The species tree from this tutorial should not be problematic, but estimating it from genome scale data would take time that we do not have. So we provide you with a

maximum-likelihood tree from concatenated data (in NEWICK format) below:

```
(((ENSBTA:0.09289,(ENSFCA:0.07151,ENSECA:0.05727):0.00398):0.02355,((((
    ENSPPY:0.01034,(ENSPTR:0.00440,ENSP00:0.00396):0.00587):0.00184,
    ENSNLE3:0.01331):0.00573,(ENSMMU:0.00443,ENSPAN:0.00422):0.01431)
    :0.01097,ENSCJA:0.03886):0.04239):0.03383,(ENSRNO:0.04110,ENSMUS
    :0.03854):0.10918);
```

We must now make this tree ultrametric, which can be done using the program r8s.

2.3.1 Making the species tree ultrametric

There are many ways to obtain ultrametric trees (also known as timetrees, these are phylogenetic trees scaled to time, where all paths from root to tips have the same length). Here, we use a fast program called r8s. You will need to know the number of sites in the alignment used to estimate the species tree (the one you want to make ultrametric), and then you can specify one or more calibration points (ideally, the age or age window of a documented fossil) to scale branch lengths into time units. We provide you with a script that prepares the control file for running r8s on the species tree above (we give you the number of sites: 35157236). On your shell, type:

```
$ python cafetutorial_prep_r8s.py -i
    twelve_spp_raxml_cat_tree_midpoint_rooted.txt -o r8s_ctl_file.txt -
    s 35157236 -p 'ENSG00,ENSFCA' -c '94'
```

Then you can finally run r8s, and parse its output with:

```
$ r8s -b -f r8s_ctl_file.txt > r8s_tmp.txt
$ tail -n 1 r8s_tmp.txt | cut -c 16- > twelve_spp_r8s_ultrametric.txt
```

3 Running CAFE

4 Acknowledgments

Thanks to Gregg W. Thomas, who wrote the original walkthrough that served as a basis for this tutorial[1].

[1] If you have any suggestions to help improve this tutorial, please write to fkmendes@indiana.edu