Python program for alignment free phylogenomics

Lucas van Dijk

July 24, 2015

1 Introduction

Using alignment-free methods to calculate the distances between sequences has gotten quite a bit of attention recently. In an alignment-free method, shared properties of the sequences (such as number of exact or inexact shared k-length subsequences) are the basis for the distance.

One of the possible applications is constructing phylogenetic trees, where multiple sequence alignment has been the standard for calculating the distances of the sequences. However, multiple sequence alignment is an NP-complete problem [1], and thus does not scale very well with larger sequences or more species.

In [2] Chat et al. used several variants of the D_2 statistic (number of shared k-mers) to construct phylogenetic trees and showed that alignment-free methods definitely have potential for large scale phylogenomics.

In this report we will discuss a Python implementation for constructing phylogenetic trees using the D_2 and D_2-neighbourhood statistics. For evaluation, we compare the generates trees using the state of the art multiple alignment tool MAFFT [3].

2 Method

As mentioned before, our Python program can use the D_2 and D_2-neighbourhood statistic to construct phylogenetic trees [2]. We summarize the algorithm below:

1. Construct distance matrix between the sequences, using the statistic of choice.
2. Construct the tree using neighbour-joining.

The program has a few statistics built in:

- D_2, which is based on the number of shared k-mers
- D_2^N (D_2-neighbourhood), which also takes the neighbourhood (n point mutations of a k-mer) into account.

D_2 statistic

The D_2 statistic is defined as follows:

$$D_2 = \sum_{w \in A^k} X_w Y_w$$

Where A^k is the set of all possible k-length words using alphabet A, X_w is the word count of word w in sequence X, and Y_w is the word count of word w in sequence Y.

This statistic is covered in more detail in Song et al. [4].

D_2^n, D_2 with neighbourhood

The D_2-neighbourhood statistic, or D_2^n, also takes the *neighbourhood* of each k-length word w into account. The neighbourhood of a word w is defined as alle possible n point mutations of the word w. For example, when $w =$ AAA ($k = 3$), and $n = 1$, then the neighbourhood of w consists of: AAA, AAC, AAT, AAG, ACA, ATA, AGA, CAA, TAA, GAA. All these words are recorded as k-mers found in the sequence. This statistic was introduced by Chan et al. [2], and is defined as follows:

$$D_2^n = \sum_{w \in A^k} X'_w Y'_w$$

Where X'_w is word count of word w in the sequence X with the neigbourhood taken into account, and the same for Y'_w.

Distance normalization

To transform the scores of one of the above statistics to an actual distance, we use the logarithmic representation of the geometric mean.

$$D_{ab} = \left| \ln\left(\frac{S_{ab}}{\sqrt{S_{aa} \cdot S_{bb}}} \right) \right|$$

Where D_{ab} is the actual distance between sequences a and b, S_{ab} is the score using one of the above statistics between sequences a and b, and S_{aa} and S_{bb} are the self matching scores.

3 Python Implementation

Our Python program depends on a few external packages:

- NumPy
- Scikit-bio, which in turn depends on
  - Pandas
  - SciPy
  - matplotlib

NumPy is used for fast numerical computation, and scikit-bio provides us with a lot of the file format readers and writers, tree representations, and an implementation for neighbour-joining.

With scikit-bio we can easily read the sequences in a PHYLIP or FASTA file into their corresponding `SequenceCollection` or `Sequence` objects. We've added some custom additions to scikit-bio: limited support for reading PHYLIP files, and a sliding window iterator for sequences.

Data Preparation

PAML is a toolbox for phylogenetic analysis, and includes functionality to generate synthetic trees [5]. The output file contains several sets of sequences where each set is evolutionary related, in concatenated PHYLIP format.

For easier analysis of the sequences, we split each collection of sequences into their own files, and also store them in FASTA format. This is an easy task using scikit-bio. All these sequence collections are stored in the folder data/sequences.

Distance calculation

To generate the k-mers of a sequence, we let a sliding window walk over the sequence. This iterator is passed to the built in Python `collections.Counter` class, which counts each returned element by a given iterator. It uses internally an optimized C implementation, so it's very fast.

Our sliding window generator first converts the `Sequence` to a `str` object, because slicing of a native Python string is much faster than slicing on a `Sequence` object. A `Sequence` object is a subclass of `pandas.DataFrame`, and it is quite CPU intensive to instantiate a new object for each window in the sequence.

Throughout the whole code we use iterators are lot, and they're designed in such way that only the data that is required for the current step is loaded into memory. Another benefit is easy data parallelization using the built in Python `multiprocessing` toolbox, to make use of all available CPU cores.

When the distance matrix is completely filled, the phylogenetic tree can be constructed using neighbour-joining. We use the implementation available in scikit-bio. Subsequently, we store the generated tree in Newick format on the disk.

Scripts usage

`phylip-splitter.py` - Split concatenated PHYLIP files

This script can split a PHYLIP file with multiple sets of sequences into separate files again. The script allows you to specify the output directory and the output format. The default output format is FASTA.

Note that this script currently only supports a PHYLIP file in sequential format, which means that the whole sequence needs to be on the same line.

```
usage: phylip-splitter.py [-h] [--format FORMAT] [--prefix PREFIX]
                          infile outdir

Split a concatenated PHYLIP file. Currently only PHYLIP files in sequential
format are supported.

positional arguments:
  infile                The PHYLIP formatted file to read from
  outdir                The output directory

optional arguments:
  -h, --help            show this help message and exit
  --format FORMAT, -f FORMAT
                The output format. See the scikit-bio documentation
                      for more information (the skbio.io module).
  --prefix PREFIX, -p PREFIX
                Filename prefix. Each output filename will be prefixed
                        with this string
```

`af-phylogeny.py` - Construct phylogenetic tree

This is the actual work horse of this Python package. This script creates a phylogenetic tree from a collection of sequences using an alignment free method.

It supports several input and output file formats, please refer to the skbio.io module documentation for more information.[1]

This script is also able to use multiple cores. It tries to be smart about when to enable this: when there are more than 16 sequences in the file, it enables multicore support and will automatically spawn a number of child processes corresponding to the number of cores. This behaviour can be overwritten by setting the number of child processes manually.

```
usage: af-phylogeny.py [-h] [--format FORMAT] [--target FORMAT] [--parallel
N]
                 [--log-level {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                        infile outfile

Construct phylogenetic trees using an alignment free method.

positional arguments:
  infile              The file containing the evolutionary related
                        sequences.
  outfile               The tree output filename.

optional arguments:
  -h, --help            show this help message and exit
  --format FORMAT, -f FORMAT
                 The sequence collection input file format. See the
                scikit-bio documentation for more information about
                        the supported types. Defaults to fasta.
  --target FORMAT, -t FORMAT
                   The tree output file format. See the scikit-bio
                documentation for more information about the supported
                        types. Defaults to newick.
 --parallel N, -p N   Enable multicore support. This option specifies the
                  number of child processes to spawn. By default it
                  automatically detects the number of cpu cores and
                enables multicore support when there are more than 16
                        sequences in the file to analyze.
 --log-level {DEBUG,INFO,WARNING,ERROR,CRITICAL}, -l {DEBUG,INFO,WARNING,ERROR,CRITICAL}
                        Set logging level. Default is info.
```

`af-evaluation.py` - Evaluate the generated trees

This script is useful when you want to compare two sets of tree files to each other.

[1] http://scikit-bio.org/docs/latest/io.html

It expects the tree files generated with *method1* and *method2* in separate directories, with corresponding filenames. It compares tree with the same filename using the Robinson-Foulds metric. For each tree it generates an *evaluation file*, containing ASCII art of both trees and the Robinson-Foulds distance. By default it stores this file in the first directory specified, but you can override the output directory with the corresponding command line option.

```
usage: af-evaluation.py [-h] [--format FORMAT] [--output-dir OUTPUT_DIR]
                [--log-level {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                        dir1 dir2

Compare sets of generated trees to eachother using the Robinson-Foulds metric.
This program compares all tree files in one directory with matching tree files
in another directory.

positional arguments:
  dir1                  First directory containing tree files
  dir2                  Second directory containing tree files

optional arguments:
  -h, --help            show this help message and exit
  --format FORMAT, -f FORMAT
                The format of the tree files. Defaults to newick. See
                the scikit-bio toolkit for more information about the
                        supported formats.
  --output-dir OUTPUT_DIR, -o OUTPUT_DIR
                Output directory for evaluation files. Defaults to the
                        first directory.
 --log-level {DEBUG,INFO,WARNING,ERROR,CRITICAL}, -l {DEBUG,INFO,WARNING,ERROR,CRITICAL}
                        Set logging level. Default is info.
```

`mafft-tree-fixer.py` - Fix tree files generated by MAFFT

For our internal evaluation we compared the trees generated by our alignment free method to tree files generated with MAFFT [3]. MAFFT is able to output a tree file, but scikit-bio is quite strict in its Newick formatting, and is unable to read it because the generated tree by MAFFT misses a closing semicolon. Furthermore, MAFFT also modifies the sequence names, and to be able to compare two trees the node names must be the same.

This script fixes all of that: it restores the original sequence names, and outputs a properly formatted Newick file.

```
usage: mafft-tree-fixer.py [-h] infile [outfile]

Fix the tree files generated by MAFFT. Tree files generated by MAFFT do not
```

```
contain a closing semi-colon, and therefore scikit-bio is unable to read them.
They also modify the name of the sequences, so we revert that change to be
able to compare the trees using Robinson-Foulds.

positional arguments:
  infile      The tree file to fix
 outfile   The output file. If not specified file will be modified inplace.

optional arguments:
  -h, --help  show this help message and exit
```

Workflow

To automate most tasks a Makefile is included. Before we explain how to use this Makefile let us explain the directory structure:

- af-phylogeny/ - Container folder
  - afphylogeny/ - Python package source code
  - data/ - Directory holding all data files (sequences, trees)
    - af/ - Output directory for tree files created with out alignment free method.
    - mafft/ - Output directory for tree files created with MAFFT-LINSI.
    - sequences/ - The folder containing all sequence collections.

Using the Makefile you can easily run MAFFT or our own alignment free method on all the sequence collections.

`make mafft` - Create tree files using MAFFT

For each FASTA file with multiple sequences in the data/sequences/ folder it runs MAFFT-LINSI on that file, and outputs a Newick formatted tree file in data/mafft/. It also automatically runs the `mafft-tree-fixer.py` script, so you don't need to worry about that. It assumes that the program `mafft-linsi` is available in your `PATH` environment variable.

`make af` - Create tree files using our Alignment Free method

For each FASTA file with multiple sequences in the data/sequences/ folder it runs the `af-phylogeny.py` script, and the tree is stored in data/af/.

`make all` - Create all trees

Runs both `make mafft` and `make af`.

Multicore support

Make is also able to run multiple builds in parallel. This is useful when you have a lot of small trees which are quickly generated (and for which `af-phylogeny.py` does **not** enable multicore support).

To enable multicore support in Make, use the `-j` command line option:

```
make -j4 all
```

This runs four builds in parallel.

4 Results

Tree evaluation

We use the trees generated by MAFFT as reference. Running our own AF tool, and then performing a Robinson-Foulds comparison for each tree, we get a average RF score of 0.67.

This is a lot worse than in the original paper [2], and we're not sure what the reason for this is.

Runtime evaluation

Using the *real.fasta* sequence collection, containing 144 sequences, we get an average runtime of 11 minutes and 14 seconds.

If we include only half of the sequences in *real.fasta*, we get an average runtime of 2 minutes and 52 seconds.

This is expected as the D_2 statistics are quadratic in the number of sequences for their time complexity.

References

1. Wang L, Jiang T: **On the complexity of multiple sequence alignment**. *Journal of computational biology* 1994, **1**:337–348.

2. Chan CX, Bernard G, Poirion O, Hogan JM, Ragan MA: **Inferring phylogenies of evolving sequences without multiple sequence alignment**. *Scientific reports* 2014, **4**.

3. Katoh K, Standley DM: **MAFFT multiple sequence alignment software version 7: Improvements in performance and usability**. *Molecular biology and evolution* 2013, **30**:772–780.

4. Song K, Ren J, Reinert G, Deng M, Waterman MS, Sun F: **New developments of alignment-free sequence comparison: Measures, statistics and next-generation sequencing**. *Briefings in bioinformatics* 2013:bbt067.

5. Yang Z: **PAML: A program package for phylogenetic analysis by maximum likelihood**. *Computer applications in the biosciences* 1997, **13**:555–556.