https://github.com/msarrias/protein_multial_noise_reduction

Reducing noise in protein multialignments

Marina Herrera Sarrias

January 6, 2019

Abstract

In this research a noise-reduction method is implemented in protein multi-aligments to evaluate its impact on phylogenic inference. The data used in this project is a reduced data set from the original data used by the creators of TrimAI. To test the data the program fastprot was used to obtain the distance matrices and fnj to infer a phylogenic tree for each alignment.Dendropy has been used to measure the symmetric distance between the inferred tree and the reference tree. The results obtained show that although in most of the cases the effect of reducing the noise from the alignments was ineffective, when it was it had a reducing effect on the distance, with a greater effect on symmetric alingments with greater average amount of mutations per site. This study focuses in determining whether the noise reduction makes any effect in the symmetric distance calculation.

Contents

Abstract **i**

1 Introduction **1**

2 Results and Discussion **1**

3 Methods and Materials **3**

3.1 Running: run_program.py 3

3.2 run_program.py: Error handling and Control cases 6

1 Introduction

The implemented noise-reduction method evaluates a multialigment column as noisy if there are more than 50% indels, at least 50% of the amino acids are unique and non of the amino acids appear more than twice. Each column takes the form of a sequence of amino acids given a position of the aligned sequences.

The reduced data set used in this project is composed by six different sub-directories, each sub-directory contains a reference tree and 300 alignments created by evolving sequences along the reference tree. Each pair of sub-directories present an average amount of mutations of 0.5, 1.0 and 2.0 per sequence site. Also, per each mutation rate there is one symmetric reference tree and one asymmetric reference tree.

For validating the effect of reducing reducing

2 Results and Discussion

After verifying the correctness of the program and processing the six data sets Table 1 is the resulting frequency table of the six possible symmetric distance scenarios.

In most of the cases the original tree and the noise reduced alignment tree are the same, this means that the method applied did not have any effect in most of the alignments, although the cases in which the effect is not the same between the original and noise reduced inferred tree, the effect of reducing noise in the alignments makes the new noise reduced data to have a smaller symmetric distance with the reference tree, while there are fewer cases in which the effect is the opposite.

The reference tree is recovered with a greater frequency when the tree is symmetric and the average amount of mutations per sequence site is smaller, meaning that the symmetric difference is equal to 0. According to the frequency values the reference tree was recovered 3 times by the noise reduced tree while it was recovered 5 times by the original multialignment inferred tree.

Table 2 gives statistical information about the six data sets. In most of the cases, both inferred trees take the same values, except for the means

of the noise reduced inferred trees symmetric difference which is slightly smaller and the maximum noise reduced inferred tree of the Asymmetric 2.0 sub-directory. Although both frequency values are mostly the same on all of the original and noise reduced cases, the values tend to be smaller for the symmetric sets than for the asymmetric sets, the values tend to increment as the average amount of mutations increments as well.

	Asymm_0.5	Asymm_1.0	Asymm_ 2.0	Symm_0.5	Symm_1.0	Symm_2.0
Original >Reduced	45	60	79	44	59	83
Original <Reduced	48	54	45	28	36	54
Original = Reduced	206	186	176	214	201	162
Original = 0	0	0	0	4	1	0
Reduced = 0	0	0	0	2	1	0
Original & Red = 0	1	0	0	8	2	1
Total	300	300	300	300	300	300

Table 1: Data sets frequencies referring to the symmetric distance of the original and reduced newick trees with the reference tree.

	Minimum		Maximum		Mean		Median	
	Original	Reduced	Original	Reduced	Original	Reduced	Original	Reduced
Asymmetric 0.5	2	2	18	18	7.63	7.62	8	8
Asymmetric 1.0	2	2	18	18	9.43	9.39	10	10
Asymmetric 2.0	4	4	22	20	12.34	11.99	12	12
Symmetric 0.5	2	2	10	10	4.66	4.55	4	4
Symmetric 1.0	2	2	12	12	6.23	6.05	6	6
Symmetric 2.0	2	2	16	16	8.91	8.65	8	8

Table 2: Data sets statistics referring to the symmetric distance of the original and reduced newick trees with the reference tree.

Referring to Figure 3 the noise reduction performance was slightly higher on the symmetric sets than in the asymmetric, the noise reduction average also tends to grow as the average amount of mutations increments as well.

	Average Noise Reduction
Asymmetric 0.5	0.146
Asymmetric 1.0	0.199
Asymmetric 2.0	0.252
Symmetric 0.5	0.172
Symmetric 1.0	0.221
Symmetric 2.0	0.282

Table 3: Average noise reduction performed per data set

3 Methods and Materials

The project was developed working in three different major tasks separately which were later joint together to consolidate the program and reproduce the exploratory data analysis. The main objective for creating three different scripts, for each functionality was to initially test the data and the control cases, before unifying them all in a single script and running the program on the fixed data set.

A Github repository was created in order to version control the program. The Laboratory notebook contains all details regarding the projects evolution, it was created using a Jupyter notebook Markdown and later compiled into a HTML file. It can be found on the Github repository

Following the course centerpiece guideline, controls were introduced along the project as well error handling. When an error occurs the program aborts and prints a message to standard error. All standard-in files are read by robust libraries. A sub-directory containing a subset of the fixed data set was created in order to test the program, controls for different scenarios were also introduced to ensure the program functionality. No input is required for testing the data, either for running the main program.

3.1 Running: run_program.py

run_program.py performs all the project's required tasks in different stages:

1. Reduce protein multialignment noise: The program will firstly takes an alignment from standard-in, filter the columns and write a new file

containing the noise reduced multi-alignment. In order to illustrate the functionality of the program, Seaview was used as a graphical interface for the following example.

Figure 1 shows the original alignment for the s001.align.1.msl file, corresponding to the asymmetric_0.5 subdirectory. Figure 2 shows the alignment after the noise reduction was performed. In this particular case the reduced noise represented a 17.08% of the original sequence.

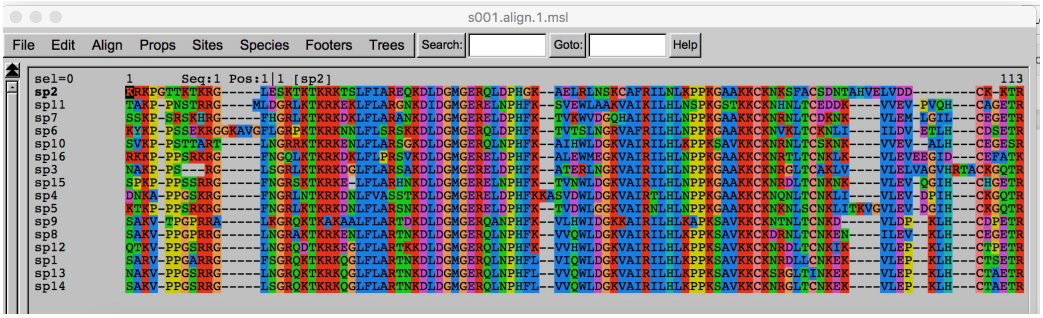

Figure 1: Original s001.align.1.msl

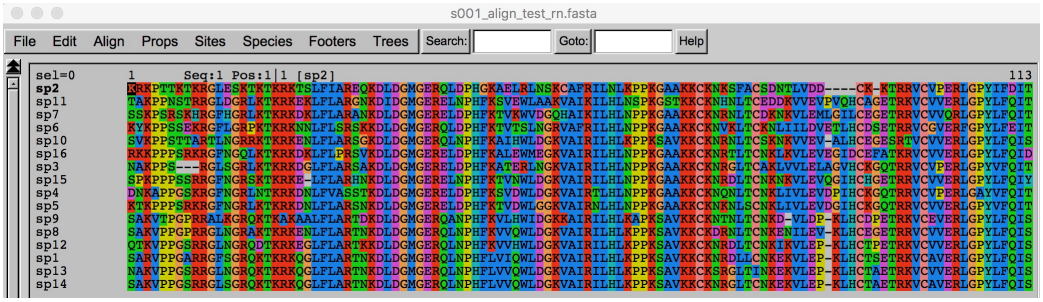

Figure 2: s001.align.1.msl after noise reduction

2. Generate phylogenetic trees: On the second stage, the program will pars through all the alignments available in the data directory (original and noise reduced), and using the fastphylo programs (fastprot and fnj) will compute a distance matrix and generate a phylogenetic tree

per alignment. In order to accomplish this task the program uses a sub-process to pipe the matrix creation and generate the tree. As the symmetric distance (computed in next stage) only requires the tree's topology it will not be necessary to compute the branch length information.

Following the s001.align.1.msl alignment example both, the noise reduced and original alignment have the same phylogenetic tree shown in Figure 4 while the reference tree is represented by Figure 3.

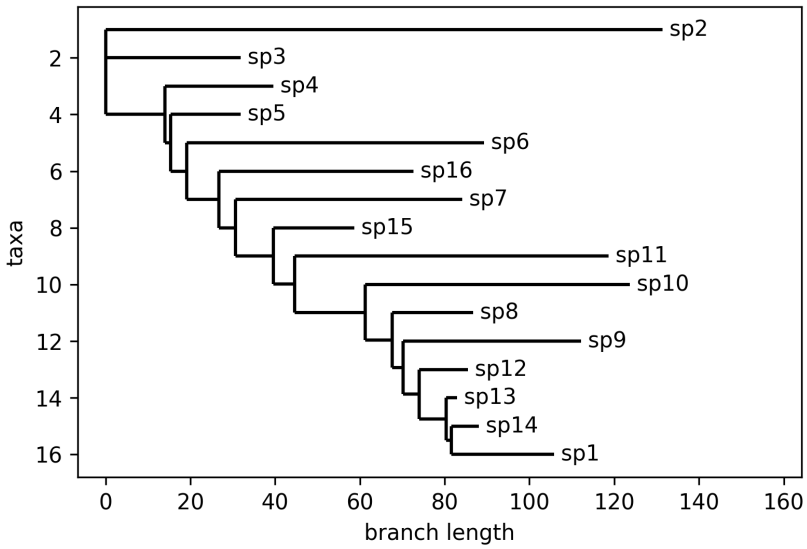

Figure 3: asymmetric_0.5.tree reference tree

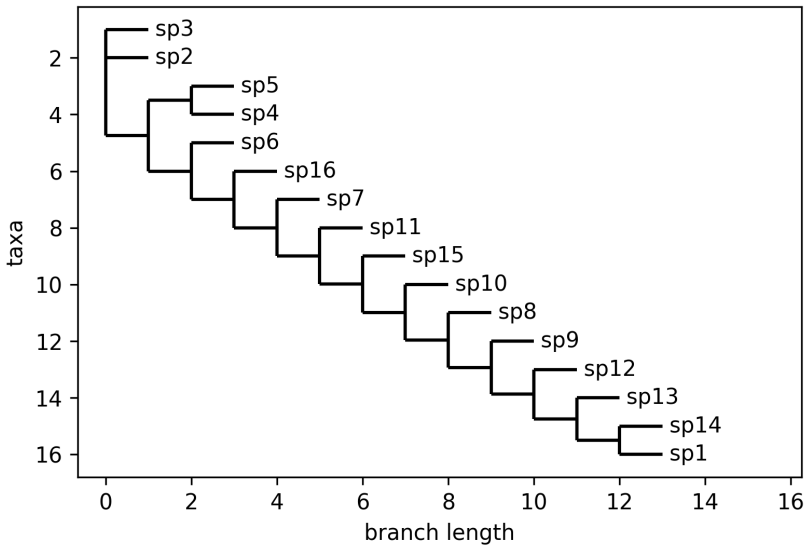

Figure 4: s001.align.1.msl original and noise reduced tree

3. Compute trees distances: At the final stage the program will compute the symmetric distance between the reference tree and the original and noise reduced tree using Dendropy. The symmetric distance is a count of how many partitions there are among the two trees, that are on one tree and not on the other. Following the s001.align.1.msl, alignment example, as the original tree and noise reduced tree are the same the distance with the reference tree will be therefore the same, in this care 4.

3.2 run_program.py: Error handling and Control cases

run_program.py will abort if any of the following cases occur if:

- There is an input introduction when executing the program.
- The data directory is empty.
- The data sub-directories are empty. (e.g asymmetric_0.5).
- Any of the standard-in file (.msl,.tree) is empty.

- All the columns in the alignment are noisy.

The program will print a warning message but will not abort processing if:

- The alignment do not have noisy columns.
- The noise reduction rate is greater than 0.5 times the original alignment.

All of the control files used for testing the accuracy of the script can be found in the Github repository, results directory. Where a file for each case exposed before can be found.