

Reducing noise in protein multialignments

Gökçen Eraslan (euSYSBIO / gokcen@kth.se)

Başak Eraslan (euSYSBIO / eraslan@kth.se)

Javeria Ali (euSYSBIO / javeriaa@kth.se)

Project URL: github.com/gokceneraslan/msa_noise_reduction

December 25, 2012

Contents

1	Introduction	3
2	Project Details	3
2.1	Implementation details and usage of reduce_noise Python script	3
2.2	Validation of reduce-noise Python script	5
2.3	Running reduce-noise Python script on test data set	5
2.4	Calculating the symmetric distances between generated newick trees and the reference trees	6
3	Results and Discussion	9
4	Appendices	11
4.1	Appendix A	11
4.1.1	Plots of symmetric distance values	11
4.1.2	Python Codes	14

1 Introduction

In this project a method for noise reduction in multialignments is implemented and evaluated with respect to its effect on phylogeny inference. In this method, a column is considered as noisy if:

- there are more than 50% indels or
- at least 50% of amino acids are unique or
- no amino acid appears more than twice.

In order to validate the method implementation, null data is used as input and the outputs are verified. Thereafter, the method is applied on data which consists of six directories that each contains the reference tree and 300 multialignments derived from that tree by simulation. After generating the noise-reduced counterparts of the multialignments, newick trees corresponding to both the original multialignments and the noise-reduced multialignments are created by using fasttree submodule of PyCogent module. Finally, the symmetric distances between these newick trees with the reference trees are computed by using DendroPy Python module and then the statistics on these distance values are worked out.

2 Project Details

2.1 Implementation details and usage of reduce_noise Python script

In the first step of the project, a Python script which generates the noise-reduced multialignments and produces the newick trees of both the original and noise-reduced alignments, is implemented. It takes a multialignment file as input (in formats: fasta, clustal, phylip, stockholm, emboss) and outputs:

1. A file (in formats: newick, xml) which contains the newick tree of the original multialignment. Additionally, a file (in formats: png, pdf, svg) which displays this phylogenetic tree visually can be generated optionally using ete2 Python module.
2. A file (in formats: fasta, clustal, phylip, stockholm, emboss) which contains the noise-reduced counterpart multialignment of the original input multialignment
3. A file (in formats: newick, xml) which contains the newick tree of the noise-reduced multialignment. Additionally, a file (in formats: png, pdf, svg) which displays this phylogenetic tree visually can be generated optionally.

The usage and optional arguments of the script can be displayed if it is invoked with “-h” argument:

```
usage: reduce_noise [-h] [-i INPUT_ALIGNMENT_FILE] [-o
    OUTPUT_ALIGNMENT_FILE] [-I INPUT_ALIGNMENT_FORMAT] [-O
    OUTPUT_ALIGNMENT_FORMAT] [-d REDUCED_OUTPUT_TREE_FILE] [-g
    ORIGINAL_OUTPUT_TREE_FILE] [-T TREE_FORMAT] [-R
    TREE_RENDERING_FORMAT] [-b TREE_GENERATOR]

Reducing noise in protein multialignments

optional arguments:
  -h, --help
            show this help message and exit
  -i INPUT_ALIGNMENT_FILE, --input INPUT_ALIGNMENT_FILE
            Input file from which the noise to be reduced. By default,
            stdin is used.
  -o OUTPUT_ALIGNMENT_FILE, --output OUTPUT_ALIGNMENT_FILE
            Name of the output file to be created upon the noise
            reduction process. stdout is used if not specified.
  -I INPUT_ALIGNMENT_FORMAT, --input-format
            INPUT_ALIGNMENT_FORMAT
            Input file format, "fasta" is the default. Available
            formats: clustal, phylip, stockholm, emboss.
  -O OUTPUT_ALIGNMENT_FORMAT, --output-format
            OUTPUT_ALIGNMENT_FORMAT
            Output file format, "fasta" is the default. Available
            formats: clustal, phylip, stockholm, emboss.
  -d REDUCED_OUTPUT_TREE_FILE, --output-tree-reduced
            REDUCED_OUTPUT_TREE_FILE
            Phylogenetic tree is generated from the noise-reduced
            alignment file.
  -g ORIGINAL_OUTPUT_TREE_FILE, --output-tree-original
            ORIGINAL_OUTPUT_TREE_FILE
            Phylogenetic tree is generated from the original alignment
            file.
  -T TREE_FORMAT, --tree-format TREE_FORMAT
            File format of output tree files. Default is newick,
            alternative format is xml.
  -R TREE_RENDERING_FORMAT, --tree-rendering-format
            TREE_RENDERING_FORMAT
            Image format to be used in rendering generated trees. This
            option is used with the --output-tree-* options.
            Default value is png. Alternative formats are pdf and
            svg.
  -b TREE_GENERATOR, --tree-generator TREE_GENERATOR
            Use given program to generate phylogenetic trees. Default
            is fasttree. Alternatives are: clearcut, clustalw,
            muscle and raxml.
```

2.2 Validation of reduce-noise Python script

2.3 Running reduce-noise Python script on test data set

The test data contains six directories: asymmetric_0.5, asymmetric_1.0, asymmetric_2.0, symmetric_0.5, symmetric_1.0, and symmetric_2.0. Each directory contains one tree file having a reference tree, and 300 alignments that were created by evolving sequences along the reference tree and then using muscle programme to align the resulting sequences. The number in the filename denotes the average amount of mutations per site in the sequences. For example, the asymmetric_2.0 and symmetric_2.0 directories has alignments with on average two mutations per sequence position, from the root to the leaves.

The files containing the noise reduced multialignments and the files containing the newick trees corresponding to the original and noise reduced multialignments are generated when the reduce-noise Python script is run for all input files with the parameters as follows:

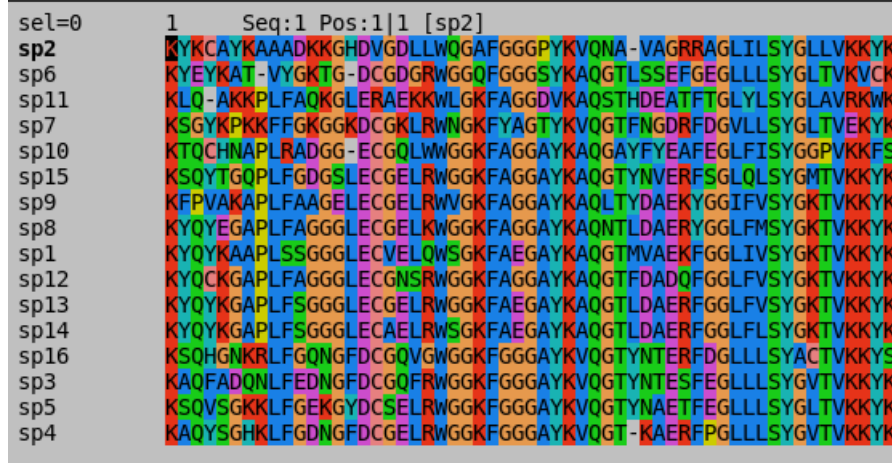
```
$ ./reduce_noise
-i <inAlignmentFileName>
-o <outAlignmentFileName>
-d <reducedOutputTreeFileName>
-g <originalOutputTreeFileName>
-R svg
```

In order to show an example of the outcome columns of noise reducing operation, In Figures 1 and 2, the columns in s001.align.1 data in Asymmetric 2.0 data set is shown in its original form and after the noise reducing operation is applied to them. In this example, we observe that the columns which do not satisfy the given constraints are filtered out as they are considered to be noisy.

Figure 1: Original multialignment of s001.align.1 data in Asymmetric 2.0 data set

seq=0	1	Seq:1 Pos:1 1 [sp2]
sp2	1	ASNYTONKLYKGLGGCNKAYKAKRGGAADKKG-HNDYVGDLLWQAFGGGPYKVONAH--SVAGRRAGLILSYGLLVKKKYK
sp6	----	PKGOLKYE--TLSYNKKAT-----VVGKGTG--TDHCGDGRWGGQFGGGSYKAQGTTSL-VSSEFGEGLLLSYGLTVKQVCK
sp11	-----	LFKLQHRIV---AKKP--YLFAQKG-LSEARAEEKWLKGFAGGDVKAQSTV-H-HDEATFTGLYLSYGLAVRPKWK
sp7	-----	IKSG--TIIYDSKPKK--GFFGKGG-KADKCGKLRWNGKFYAGTYKVOGTA-F-ENGDRFDVLLSYGLTVETKYK
sp10	-----	SYKTQ--QQCALHNAP---SLRADGG--IEECGLWVGKKFAGGAYKAQGAQPYTSFYEAFFGLFISYGGPVKTKFS
sp15	-----	KTQSQ--NITYERTGQP---MLFGDGSCLAEHCGELRWGGKFAGGAYKAQGTI-Y-ENVERFSGQLLSYGMTVKNKYK
sp9	-----	VFKFPHHVKAQYQAKAP---LLFAAGE--LQEDCGELRWVGKFAGGAYKAQLTN-YQADAKEYGGIFVSYGKTVKSKYK
sp8	-----	FVKYQ--NIIYQTEGAPSLAAVLFAGGG-LDEACGELRWGGKFAGGAYKAQNTG-L-KDAERYGGLFMSYGKTVKAKYK
sp1	-----	DKYQ--TVLYQTKAAP---VLSSGGG-LQEDCVELQWSGKFAEGAYKAQGT-D-M-PVAEKFGGLIVSYGKTVKAKYK
sp12	-----	KYQSDTIRCDTKGAP---VLFAAGG-LNEECGSRWGGKFAGGAYKAQGTG-F-VDADQFGGLFVSYGKTVKAKYK
sp13	-----	TKYQ--TIIYQTKGAP---VLFSGGG-LNEECGELRWGGKFAGGAYKAQGTG-L-GDAERFGLFVSYGKTVKAKYK
sp14	-----	TKYQ--RIVYETKGAP---ILFSGGG-LNENCAELRWSGKFAEGAYKAQGTG-L-GDAERFGLFVSYGKTVKAKYK
sp16	-----	QIKSQ--RAWHQHGNKR---FLFGQNG-FVDHCGQVVGWGGKFAGGAYKVOGTHYHSNTERFDGLLLSYACTVKKYK
sp3	----	HLEFVPAQ--ATFFQADON---LFEDNG-FTDKCGQFRWGGKFAGGAYKVOGTL-Y-PNTESFEGLLLSYGVTVKAKYK
sp5	-----	MTKSQQTIEVPRSGKK---TLFGEKG-YSDHCSLWGGKFAGGAYKVOGTQ-Y-ENAETFEGLLLSYGLTVKSKYK
sp4	-----	VKAQ--VVAYSRSCHK---VLFGDNG-FTDPCGELRWGGKFAGGAYKVOGTR---KAERFPGLLLSYGVTVKTKYK

Figure 2: Noise-reduced multialignment of s001.align.1 data in Asymmetric 2.0 data set



2.4 Calculating the symmetric distances between generated newick trees and the reference trees

At the beginning of this step, there exists 1 reference tree file, 300 newick tree files corresponding to original multialignments and 300 newick tree files corresponding to noise reduced multialignments for each data directory. In order to quantify the difference between these inferred newick trees and their corresponding reference trees, the symmetric distances, euclidean distances, and Robinson-Foulds distances are calculated between them with the use of compare-trees Python script.

For instance, these distances are calculated with respect to the reference tree displayed in Figure 3 for Asymmetric 2.0 data set. The symmetric, euclidian and Robionson-Foulds distances of the newick tree (displayed in Figure 4) of original s001.align.1 data in Asymmetric 2.0 data sets are calculated to be 14, 205.45, 715.55 respectively. However, these distances for the newick tree (displayed in Figure 5) of the noise-reduced counterpart of this data are 18, 205.67 and 716.18 respectively. Since the differences between the euclidian and Robinson-Foulds distances are much smaller then symmetric distances for all of the six data sets, only the symmetric distances are taken into account while the statistics in the results and discussion section are prepared.

Figure 3: Reference newick tree of Asymmetric 2.0 data set.

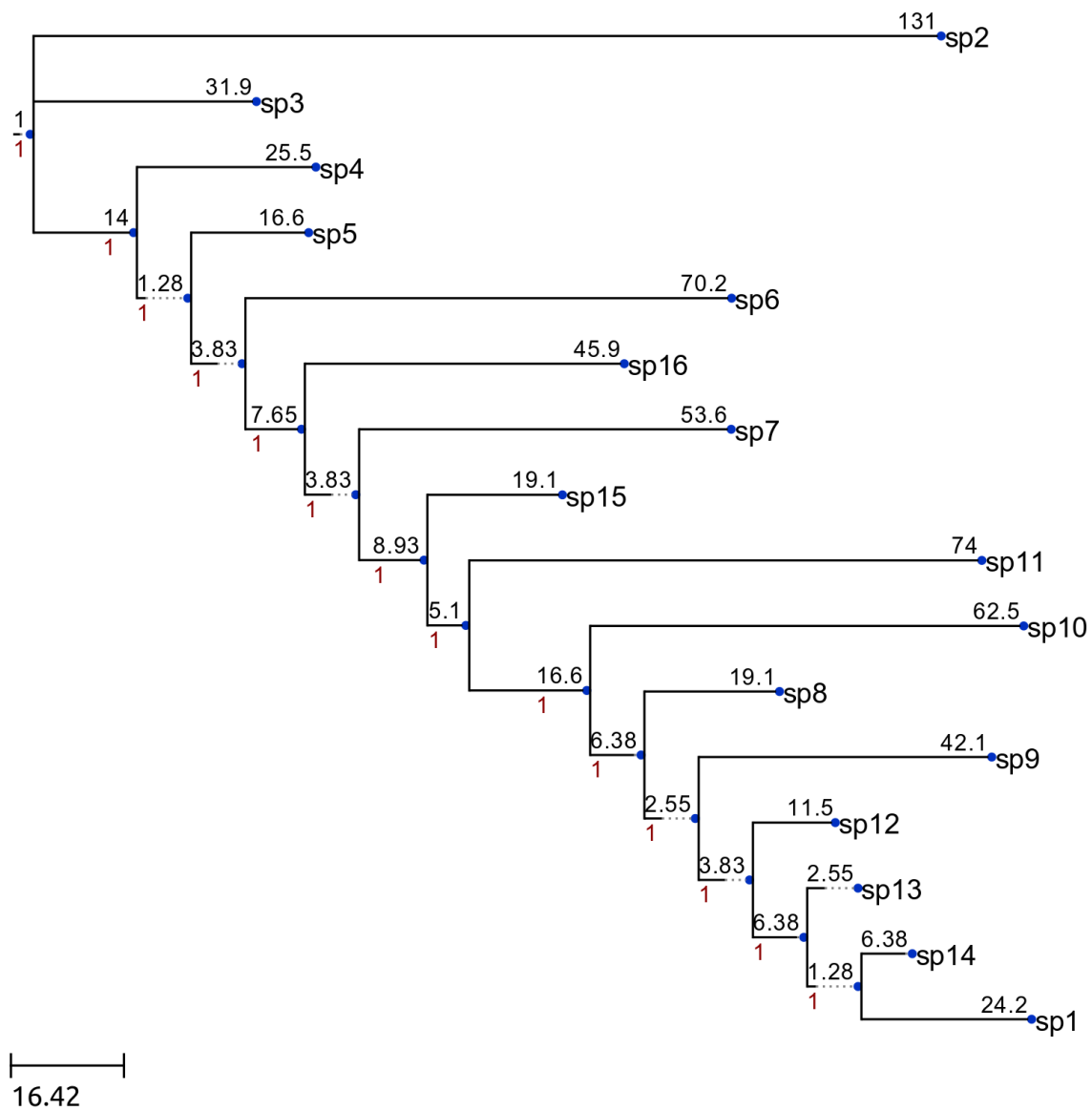


Figure 4: Newick tree corresponding to original s001.align.1 data in Asymmetric 2.0 data set.

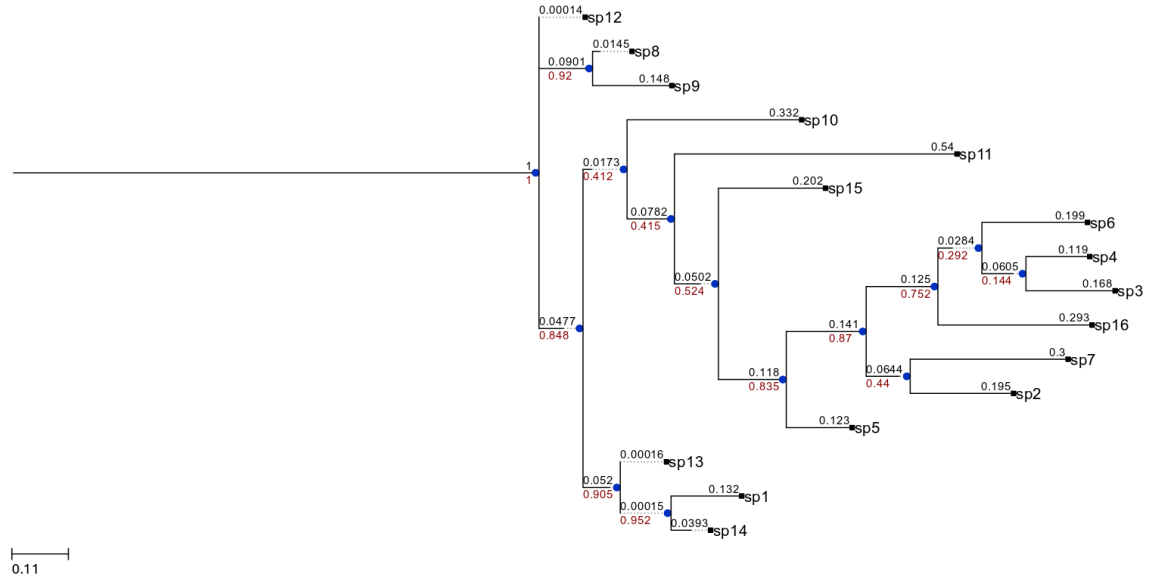
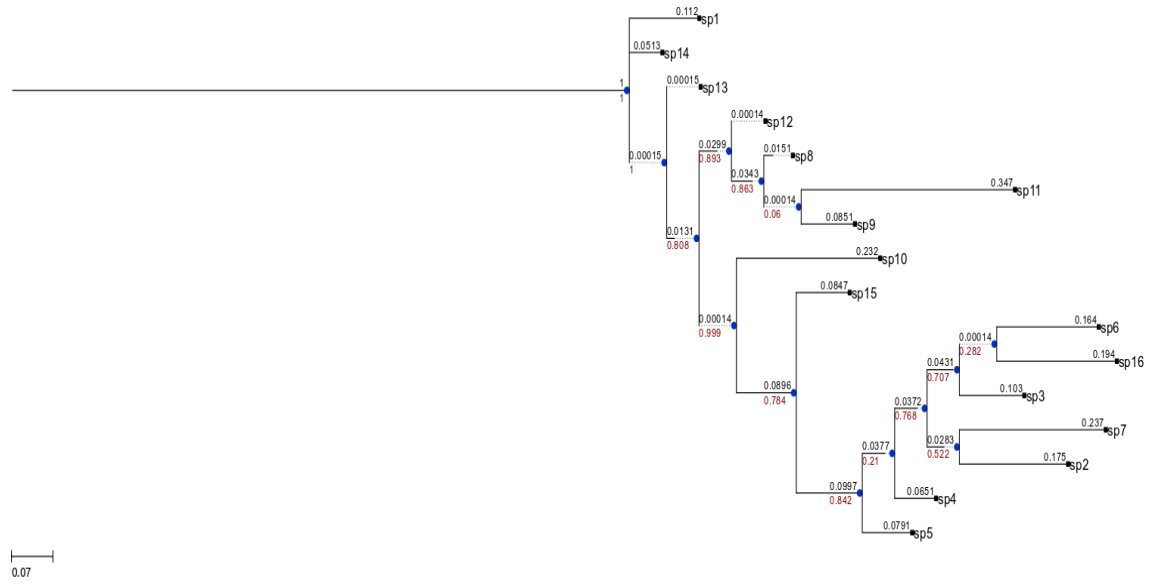


Figure 5: Newick tree corresponding to noise-reduced s001.align.1 data in Asymmetric 2.0 data set.



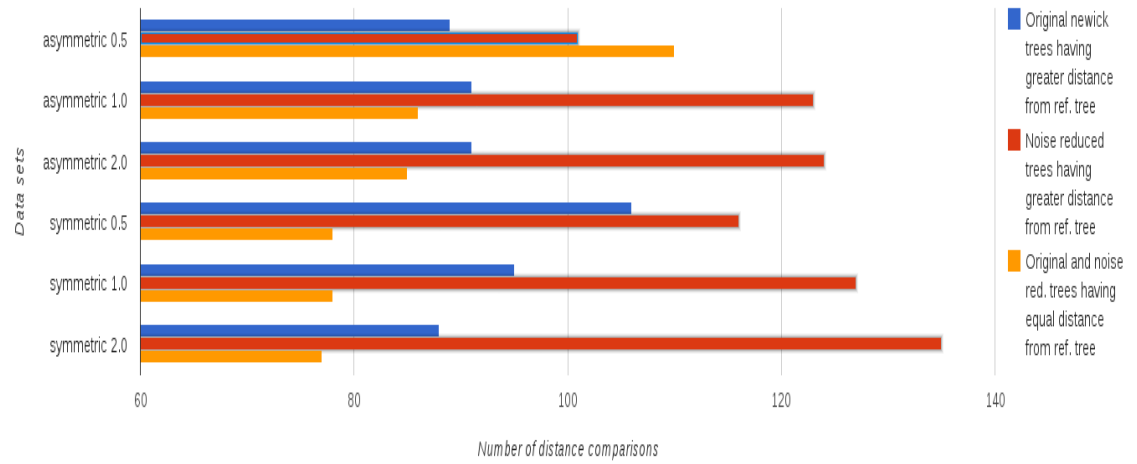
3 Results and Discussion

Even though euclidian and Robinson-Foulds distances of the concerned newick trees are also calculated, since it is observed that the differences between the euclidian and Robinson-Foulds distances are much smaller than symmetric distances for all of the six data sets, only the statistical results concluded by the examination of the symmetric distance values are presented in this section. In Table 1 the maximum, minimum, mean and median symmetric distance values are presented for the newick trees of both the original and the noise-reduced multialignments in the six data sets. This table displays that in general the newick trees of the noise reduced multialignments have slightly bigger symmetric distances with the reference trees than the newick trees of the original multialignments have. Additionally, Figure 6 presents for how many comparisons the original multialignment trees have greater symmetric distance than the noise-reduced multialignment trees and vice versa. Again it is observed that the newick trees of the noise-reduced multialignments generally have bigger symmetric distance values for all of the six data sets. The graphics comparing these distance values are available in Appendix A.

Table 1: Statistics regarding the symmetric distance values of original and noise reduced multialignment data sets.

Data set	Min		Max		Mean		Median	
	Original	Noise-reduced	Original	Noise-reduced	Original	Noise-reduced	Original	Noise-reduced
Asymmetric 0.5	6	8	26	26	19.49	19.67	20	20
Asymmetric 1.0	10	8	26	26	20.28	20.59	20	22
Asymmetric 2.0	10	10	26	26	21.27	21.77	22	22
Symmetric 0.5	0	4	26	26	17.58	17.8	18	18
Symmetric 1.0	4	4	26	26	18.27	18.72	18	20
Symmetric 2.0	10	10	26	26	19.83	20.67	20	22

Figure 6: Results of the symmetric distance comparisons between original and noise-reduced multialignments



4 Appendices

4.1 Appendix A

4.1.1 Plots of symmetric distance values

Figure 7: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Asymmetric 0.5

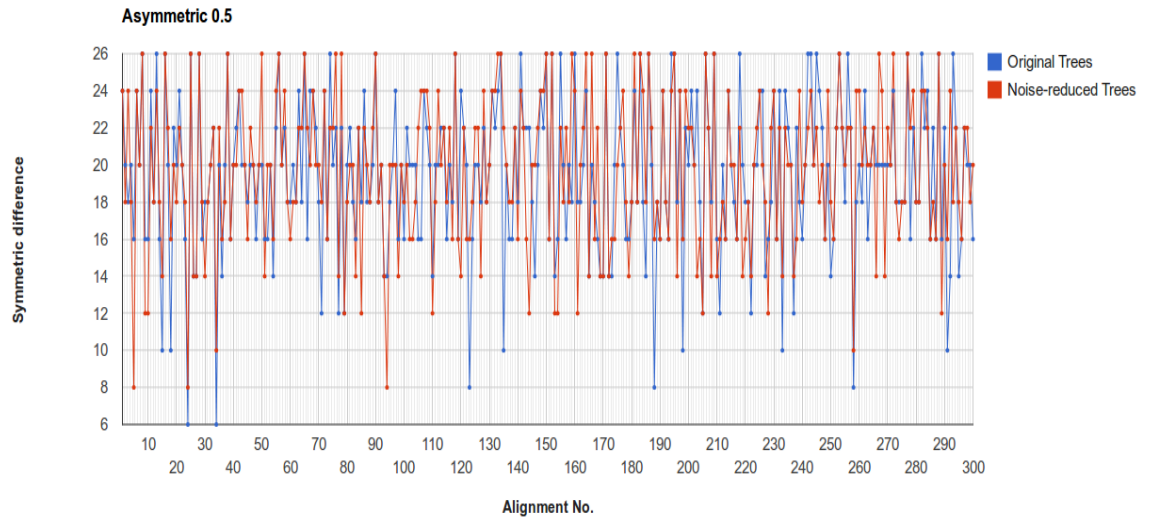


Figure 8: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Asymmetric 1.0

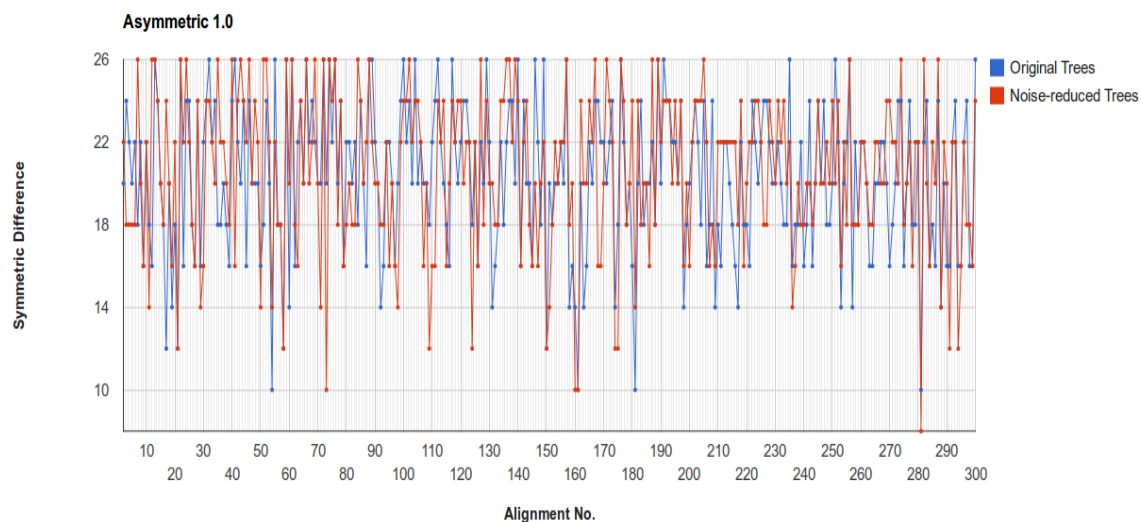


Figure 9: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Asymmetric 2.0

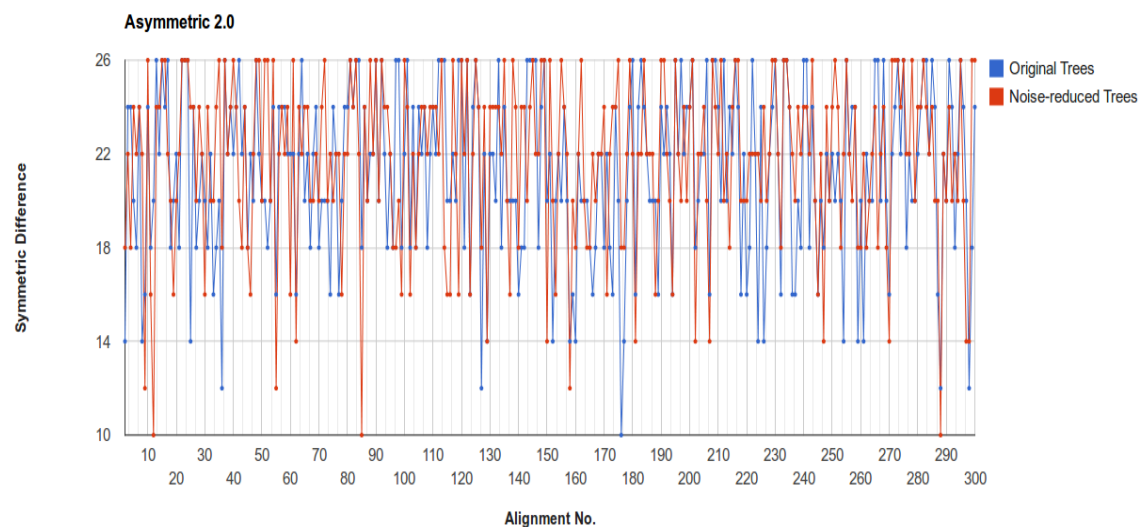


Figure 10: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Symmetric 0.5

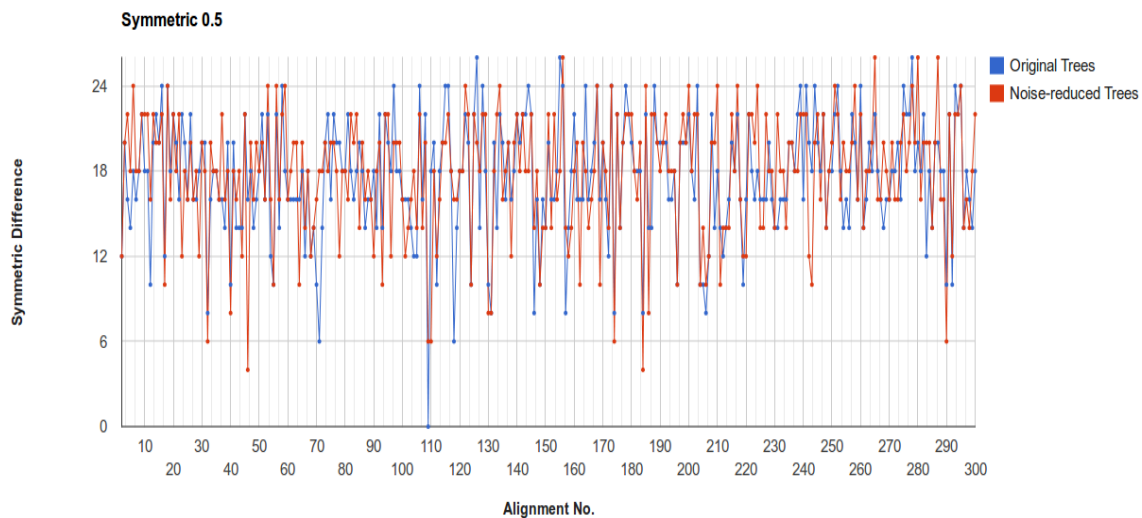


Figure 11: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Symmetric 1.0

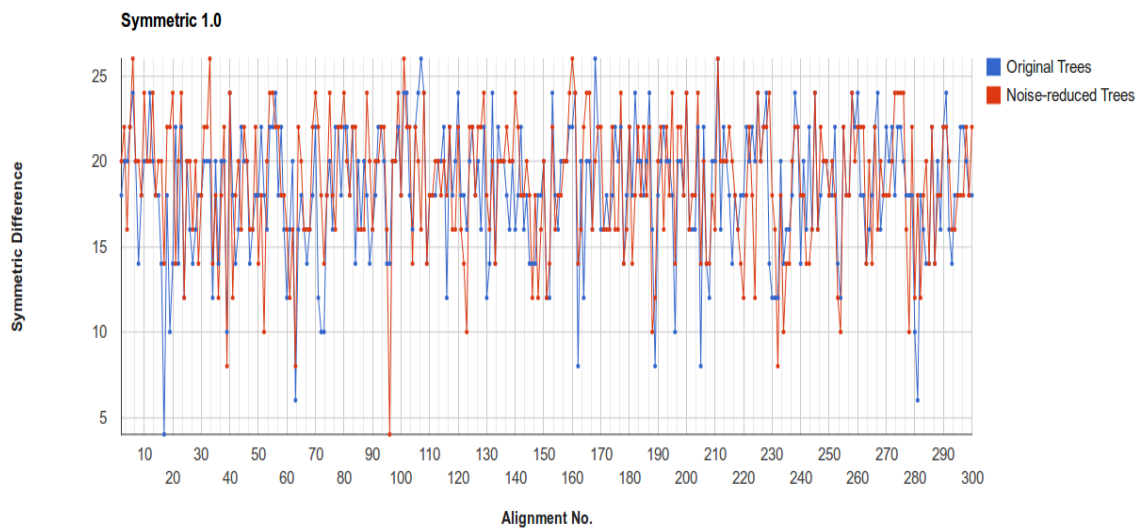
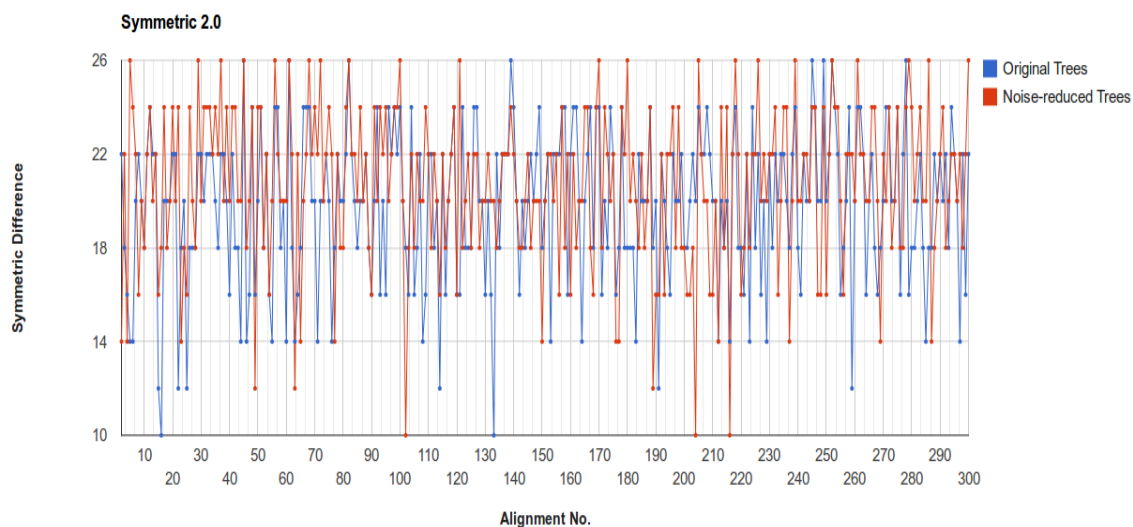


Figure 12: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Symmetric 2.0



4.1.2 Python Codes

Listing 1: “reduce_noise” script

```

1  #!/usr/bin/env python
2
3  # Reducing noise in protein multialignments
4  # Copyright (C) 2012 Gokcen Eraslan, Basak Eraslan, Javeria Ali
5  #
6  # This program is free software: you can redistribute it and/or modify
7  # it under the terms of the GNU General Public License as published by
8  # the Free Software Foundation, either version 3 of the License, or
9  # (at your option) any later version.
10 #
11 # This program is distributed in the hope that it will be useful,
12 # but WITHOUT ANY WARRANTY; without even the implied warranty of
13 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 # GNU General Public License for more details.
15 #
16 # You should have received a copy of the GNU General Public License
17 # along with this program. If not, see <http://www.gnu.org/licenses/>.
18
19 import sys
20 import argparse
21 import os
22 import importlib
23 from collections import Counter
24

```

```

25 # Biopython is used to parse and manipulate fasta formatted MSA files
26 from Bio import AlignIO, Align, Seq, SeqRecord
27
28
29 # matlab style indexing i.e.: x = 'AGCGAGGC'; index_string(x, [0, 3, 4]-> 'AGA'
30 index_string = lambda string, i: ''.join([string[x] for x in i])
31
32
33 def is_noisy(column):
34     """Determines whether given column is noisy or not"""
35
36     # Counter is a neat class to convert a sequence into a dictionary holding
37     # the count of each item. Here is an example Counter of a msa column:
38     # Counter({'S': 7, 'K': 3, 'N': 2, 'D': 1, 'Q': 1, 'R': 1, 'T': 1})
39
40     counter = Counter(column)
41     col_length = len(column)
42
43     # there are more than 50% indels
44     if counter["-"] > col_length / 2:
45         return True
46
47     # at least 50% of amino acids are unique
48     if len(counter) >= col_length / 2:
49         return True
50
51     if "-" in counter:
52         counter.pop("-")
53
54     # no amino acid appears more than twice
55     if not [v for v in counter.values() if v > 2]:
56         return True
57
58     return False
59
60
61 def filter_noise_in_columns(msa, noise_func):
62     """Filters out the columns if given callable returns False"""
63
64     num_columns = len(msa[0])
65     valid_columns = [col for col in range(num_columns)
66                     if not noise_func(msa[:, col])]
67
68     assert valid_columns, "All columns seem noisy, exiting..."
69
70     new_msa = []
71
72     for record in msa:
73         new_msa.append(SeqRecord.SeqRecord(
74             Seq.Seq(index_string(record.seq, valid_columns)),

```

```

75         record.id, record.name, record.description))
76
77     return Align.MultipleSeqAlignment(new_msa)
78
79
80 def generate_phylo_tree(msa, outfilename, format, image_format, tree_module):
81     """Generates phylogenetic tree from given sequence alignment file."""
82
83     # PyCogent is used to generate phylogenetic tree from the fasta files
84     import cogent
85
86     # ETE toolkit is used to render newick trees as images
87     import ete2
88
89     aln = cogent.LoadSeqs(data=msa,
90                           moltype=cogent.PROTEIN,
91                           aligned=True,
92                           label_to_name=lambda x: x.split()[0])
93
94     assert tree_module in ("fasttree", "clearcut", "clustalw",
95                            "muscle", "raxml"), "Given tree generator is " \
96        "not supported."
97
98     tree_generator = importlib.import_module("cogent.app.%s" % tree_module)
99     tree = tree_generator.build_tree_from_alignment(aln, cogent.PROTEIN)
100    tree.writeToFile(outfilename, format=format)
101
102    ete_tree = ete2.PhyloTree(outfilename, msa)
103    tree_filename, ext = os.path.splitext(outfilename)
104    tree_filename += ".%s" % image_format
105
106    ts = ete2.TreeStyle()
107    ts.show_leaf_name = True
108    ts.show_branch_length = True
109    ts.show_branch_support = True
110
111    ete_tree.render(tree_filename, w=1200, units="px", tree_style=ts)
112
113
114 def main():
115     """Main function of the program. Pretty explanatory, eh."""
116
117     parser = argparse.ArgumentParser(description="Reducing noise in protein "
118                                         "multialignments")
119
120     parser.add_argument("-i", "--input", metavar="INPUT_ALIGNMENT_FILE",
121                        help="Input file from which the noise to be "
122                            "reduced. By default, stdin is used.")
123
124     parser.add_argument("-o", "--output", metavar="OUTPUT_ALIGNMENT_FILE",

```



```

125         help="Name of the output file to be created upon the "
126         "noise reduction process. stdout is used if not "
127         "specified.")
128
129     parser.add_argument("-I", "--input-format", metavar="INPUT_ALIGNMENT_"
130         "FORMAT", help="Input file format, \"fasta\" is the "
131         "default. Available formats: clustal, phylip, "
132         "stockholm, emboss.")
133
134     parser.add_argument("-O", "--output-format", metavar="OUTPUT_ALIGNMENT_"
135         "FORMAT", help="Output file format, \"fasta\" is the "
136         "default. Available formats: clustal, phylip, "
137         "stockholm, emboss.")
138
139     parser.add_argument("-d", "--output-tree-reduced", metavar="REDUCED_"
140         "OUTPUT_TREE_FILE", help="Phylogenetic tree is "
141         "generated from the noise-reduced alignment file.")
142
143     parser.add_argument("-g", "--output-tree-original", metavar="ORIGINAL_"
144         "OUTPUT_TREE_FILE", help="Phylogenetic tree is "
145         "generated from the original alignment file.")
146
147     parser.add_argument("-T", "--tree-format",
148         help="File format of output tree files. Default is "
149         "newick, alternative format is xml.",
150         default="newick")
151
152     parser.add_argument("-R", "--tree-rendering-format",
153         help="Image format to be used in rendering generated "
154         "trees. This option is used with the --output-tree-* "
155         "options. Default value is png. Alternative formats "
156         "are pdf and svg.",
157         default="png")
158
159     parser.add_argument("-b", "--tree-generator",
160         help="Use given program to generate phylogenetic "
161         "trees. Default is fasttree. Alternatives are: "
162         "cleartcut, clustalw, muscle and raxml. ",
163         default="fasttree")
164
165     args = parser.parse_args()
166
167     if args.input:
168         input_file = open(args.input, "r")
169     else:
170         input_file = sys.stdin
171
172     input_format = "fasta" if not args.input_format else args.input_format
173     output_format = "fasta" if not args.output_format else args.output_format
174

```

```

175     original_msa = AlignIO.read(input_file, input_format)
176     reduced_msa = filter_noise_in_columns(original_msa, is_noisy)
177
178     if args.output:
179         output_file = open(args.output, "w")
180     else:
181         output_file = sys.stdout
182
183     AlignIO.write(reduced_msa, output_file, output_format)
184
185     if args.output:
186         output_file.close()
187
188     tree_format = "newick" if not args.tree_format else args.tree_format
189     tree_rendering_format = ("png" if not args.tree_rendering_format
190                             else args.tree_rendering_format)
191
192     tree_generator = ("fasttree" if not args.tree_generator else
193                      args.tree_generator.lower())
194
195     if args.output_tree_reduced:
196         generate_phylo_tree(reduced_msa.format("fasta"),
197                             args.output_tree_reduced,
198                             tree_format,
199                             tree_rendering_format,
200                             tree_generator)
201
202     if args.output_tree_original:
203         generate_phylo_tree(original_msa.format("fasta"),
204                             args.output_tree_original,
205                             tree_format,
206                             tree_rendering_format,
207                             tree_generator)
208
209
210 if __name__ == '__main__':
211     main()

```

Listing 2: “compare_trees” script

```

1  #!/usr/bin/env python
2
3  # Phylogenetic tree comparison
4  # Copyright (C) 2012 Gokcen Eraslan, Basak Eraslan, Javeria Ali
5  #
6  # This program is free software: you can redistribute it and/or modify
7  # it under the terms of the GNU General Public License as published by
8  # the Free Software Foundation, either version 3 of the License, or
9  # (at your option) any later version.
10 #

```

```

11  # This program is distributed in the hope that it will be useful,
12  # but WITHOUT ANY WARRANTY; without even the implied warranty of
13  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14  # GNU General Public License for more details.
15  #
16  # You should have received a copy of the GNU General Public License
17  # along with this program. If not, see <http://www.gnu.org/licenses/>.
18
19  import argparse
20  import os
21
22  import dendropy
23
24
25  def compare_trees(ref_tree, other, method):
26      """Compare trees using given method"""
27
28      assert method in ("symmetric_difference",
29                        "euclidean_distance",
30                        "robinson_foulds_distance"), "Method not supported."
31
32      comparison_func = getattr(ref_tree, method)
33
34      if not isinstance(other, (list, tuple)):
35          return comparison_func(other)
36      else:
37          return [comparison_func(x) for x in other]
38
39
40  def find_files(directory, extension):
41      """Find files having a particular extension."""
42
43      file_list = []
44
45      for root, dirs, files in os.walk(directory):
46
47          # skip hidden directories
48          dirs[:] = [d for d in dirs if not d.startswith(".")]
49
50          file_list.extend([os.path.join(root, f) for f in files
51                           if os.path.splitext(f)[1] == extension])
52
53      return file_list
54
55
56  def main():
57      """Main function"""
58
59      parser = argparse.ArgumentParser(description="Phylogenetic tree "
60                                          "comparison.")

```

```

61
62     parser.add_argument("reference",
63                          help="Reference tree to be used in the comparison.")
64
65     parser.add_argument("tree_dir", metavar='tree-dir',
66                          help="Directory containing Newick trees that will be "
67                                "compared to the reference tree. File extension of "
68                                "the reference tree is used to detect newick trees.")
69
70     parser.add_argument("output",
71                          help="Name of the output file to be generated.")
72
73     args = parser.parse_args()
74
75     assert os.path.isdir(args.tree_dir), "tree-dir is not a directory."
76     assert os.path.isfile(args.reference), "reference is not a file."
77
78     phylo_ref_tree = dendropy.Tree.get_from_path(args.reference, "newick")
79     trees = sorted(find_files(args.tree_dir, ".tree"))
80     phylo_trees = [dendropy.Tree.get_from_path(t, schema='newick')
81                    for t in trees]
82
83     with open(args.output, "w") as outfile:
84         sym = compare_trees(phylo_ref_tree, phylo_trees,
85                             "symmetric_difference")
86
87         euc = compare_trees(phylo_ref_tree, phylo_trees, "euclidean_distance")
88         rob = compare_trees(phylo_ref_tree, phylo_trees,
89                             "robinson_foulds_distance")
90
91         outfile.write("\t".join(["Reference tree", "Other tree",
92                                   "Symmetric difference", "Euclidean distance",
93                                   "Robinson-Foulds distance"]) + "\n")
94
95         for d1, d2, d3, tree in zip(sym, euc, rob, trees):
96
97             outfile.write("\t".join([args.reference, tree,
98                                       str(d1), str(d2), str(d3)]) + '\n')
99
100 if __name__ == '__main__':
101     main()

```