Noise reduction 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

January 3, 2013

Contents

1 Introduction

In this project a method for noise reduction in multialignments is implemented and evaluated with respect to its effect on pylogeny 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 consisting of six directories each of which 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 multialignmens and the noise-reduced multialignmens are created by using fast-tree 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 by the help of the ete2 Python module.
2. A file (in formats: fasta, clustal, phylip, stockholm, emboss) which contains the noise-reduced counterpart of the original input multialignment
3. A file (in formats: newick, xml) containing 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 Laboratory notebook

We have written laboratory notebook as a Markdown[1] formatted file and converted this file to a HTML file using a simple Python script (which is also

[1]Markdown is a lightweight markup language allowing people to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid HTML.

included in the Appendix section) and a neat HTML template from the Strapdown.js project. Laboratory notebook file can be accessed from the GitHub project.

2.3 Validation of reduce-noise Python script

To test the correctness of our scripts, various control cases were made, which are as follows:

1. Empty data set
2. A file with a single sequence
3. A fasta file with aligned sequences where most of the columns are with more than 50% indels
4. A fasta file with aligned sequences where most of the columns have at least 50% unique amino acids
5. A fasta file with aligned sequences where in any column none of the amino acids appear more than twice.
6. 10 random fasta files containing 10 protein sequences, 150 a.a in length, each.

Data sets containing one of each type of the noise criteria were created using real protein sequences from Uniprot. The random protein sequences were generated using the following tools:

1. Sequence Manipulation Suite (SMS)
2. RandSeq - Random protein sequence generator (With option: Average amino acid composition computed from Swiss-Prot)

All the control data was succesfully run by our scripts, the results of which are discussed in detail in section 3.

2.4 Running reduce-noise Python script on test data set

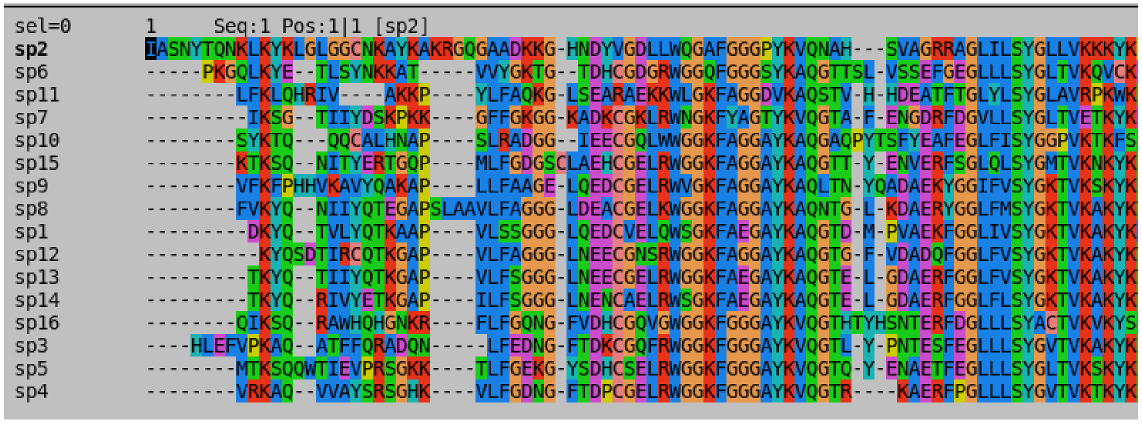
The test data contains six directories: asymmetric_0.5, asymmetric_1.0, asymmetric_2.0, symmetric_0.5, symmetric_1.0, and symmetric_2.0. Each directory contains one tree file having a reference tree and 300 alignments that were created by evolving sequences along the reference tree with further alignment on the resulting sequences by using the muscle programme. 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
```

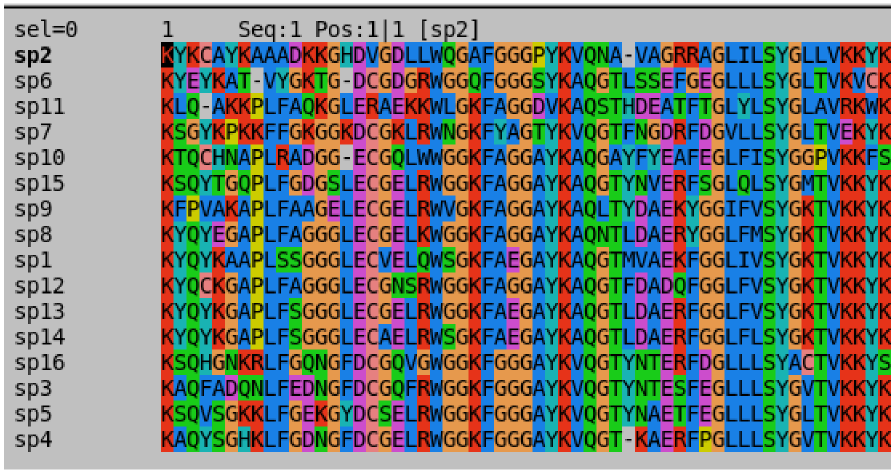
In order to show an example of the outcome columns of noise reduction operation, in Figures 1 and 2, the columns in s001.align.1 data in Asymmetric 2.0 data set is shown in its original form and after the noise reduction operation is applied to them. In this example, we observe that the columns which 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



```
sel=0       1     Seq:1 Pos:1|1 [sp2]
sp2         IASNYTQNKLKYKLGLGGCNKAYKAKRGQGAADKKG-HNDYVGDLLWQGAFGGGPYKVQNAH---SVAGRRAGLILSYGLLVKKKYK
sp6         -----PKGQLKYE--TLSYNKKAT-----VVYGKTG--TDHCGDGRWGGQFGGGSYKAQGTTSL-VSSEFGEGLLLSYGLTVKQVCK
sp11        --------LFKLQHRIV----AKKP----YLFAQKG-LSEARAEKKWLGKFAGGDVKAQSTV-H-HDEATFTGLYLSYGLAVRPKWK
sp7         ---------IKSG--TIIYDSKPKK----GFFGKGG-KADKCGKLRWNGKFYAGTYKVQGTA-F-ENGDRFDGVLLSYGLTVETKYK
sp10        --------SYKTQ---QQCALHNAP----SLRADGG--IEECGQLWWGGKFAGGAYKAQGAQPYTSFYEAFEGLFISYGGPVKTKFS
sp15        --------KTKSQ--NITYERTGQP----MLFGDGSCLAEHCGELRWGGKFAGGAYKAQGTT-Y-ENVERFSGLQLSYGMTVKNKYK
sp9         --------VFKFPHHVKAVYQAKAP----LLFAAGE-LQEDCGELRWVGKFAGGAYKAQLTN-YQADAEKYGGIFVSYGKTVKSKYK
sp8         --------FVKYQ--NIIYQTEGAPSLAAVLFAGGG-LDEACGELKWGGKFAGGAYKAQNTG-L-KDAERYGGLFMSYGKTVKAKYK
sp1         ---------DKYQ--TVLYQTKAAP----VLSSGGG-LQEDCVELQWSGKFAEGAYKAQGTD-M-PVAEKFGGLIVSYGKTVKAKYK
sp12        ----------KYQSDTIRCQTKGAP----VLFAGGG-LNEECGNSRWGGKFAGGAYKAQGTG-F-VDADQFGGLFVSYGKTVKAKYK
sp13        ---------TKYQ--TIIYQTKGAP----VLFSGGG-LNEECGELRWGGKFAEGAYKAQGTE-L-GDAERFGGLFVSYGKTVKAKYK
sp14        ---------TKYQ--RIVYETKGAP----ILFSGGG-LNENCAELRWSGKFAEGAYKAQGTE-L-GDAERFGGLFLSYGKTVKAKYK
sp16        --------QIKSQ--RAWHQHGNKR----FLFGQNG-FVDHCGQVGWGGKFGGGAYKVQGTHTYHSNTERFDGLLLSYACTVKVKYS
sp3         ----HLEFVPKAQ--ATFFQRADQN-----LFEDNG-FTDKCGQFRWGGKFGGGAYKVQGTL-Y-PNTESFEGLLLSYGVTVKAKYK
sp5         --------MTKSQQWTIEVPRSGKK----TLFGEKG-YSDHCSELRWGGKFGGGAYKVQGTQ-Y-ENAETFEGLLLSYGLTVKSKYK
sp4         --------VRKAQ--VVAYSRSGHK----VLFGDNG-FTDPCGELRWGGKFGGGAYKVQGTR----KAERFPGLLLSYGVTVKTKYK
```

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



```
sel=0       1     Seq:1 Pos:1|1 [sp2]
sp2         KYKCAYKAAADKKGHDVGDLLWQGAFGGGPYKVQNA-VAGRRAGLILSYGLLVKKYK
sp6         KYEYKAT-VYGKTG-DCGDGRWGGQFGGGSYKAQGTLSSEFGEGLLLSYGLTVKVCK
sp11        KLQ-AKKPLFAQKGLERAEKKWLGKFAGGDVKAQSTHDEATFTGLYLSYGLAVRKWK
sp7         KSGYKPKKFFGKGGKDCGKLRWNGKFYAGTYKVQGTFNGDRFDGVLLSYGLTVEKYK
sp10        KTQCHNAPLRADGG-ECGQLWWGGKFAGGAYKAQGAYFYEAFEGLFISYGGPVKKFS
sp15        KSQYTGQPLFGDGSLECGELRWGGKFAGGAYKAQGTYNVERFSGLQLSYGMTVKKYK
sp9         KFPVAKAPLFAAGELECGELRWVGKFAGGAYKAQLTYDAEKYGGIFVSYGKTVKKYK
sp8         KYQYEGAPLFAGGGLECGELKWGGKFAGGAYKAQNTLDAERYGGLFMSYGKTVKKYK
sp1         KYQYKAAPLSSGGGLECVELQWSGKFAEGAYKAQGTMVAEKFGGLIVSYGKTVKKYK
sp12        KYQCKGAPLFAGGGLECGNSRWGGKFAGGAYKAQGTFDADQFGGLFVSYGKTVKKYK
sp13        KYQYKGAPLFSGGGLECGELRWGGKFAEGAYKAQGTLDAERFGGLFVSYGKTVKKYK
sp14        KYQYKGAPLFSGGGLECAELRWSGKFAEGAYKAQGTLDAERFGGLFLSYGKTVKKYK
sp16        KSQHGNKRLFGQNGFDCGQVGWGGKFGGGAYKVQGTYNTERFDGLLLSYACTVKKYS
sp3         KAQFADQNLFEDNGFDCGQFRWGGKFGGGAYKVQGTYNTESFEGLLLSYGVTVKKYK
sp5         KSQVSGKKLFGEKGYDCSELRWGGKFGGGAYKVQGTYNAETFEGLLLSYGLTVKKYK
sp4         KAQYSGHKLFGDNGFDCGELRWGGKFGGGAYKVQGT-KAERFPGLLLSYGVTVKKYK
```

2.5 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 which are represented in the results and discussion section are prepared.

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

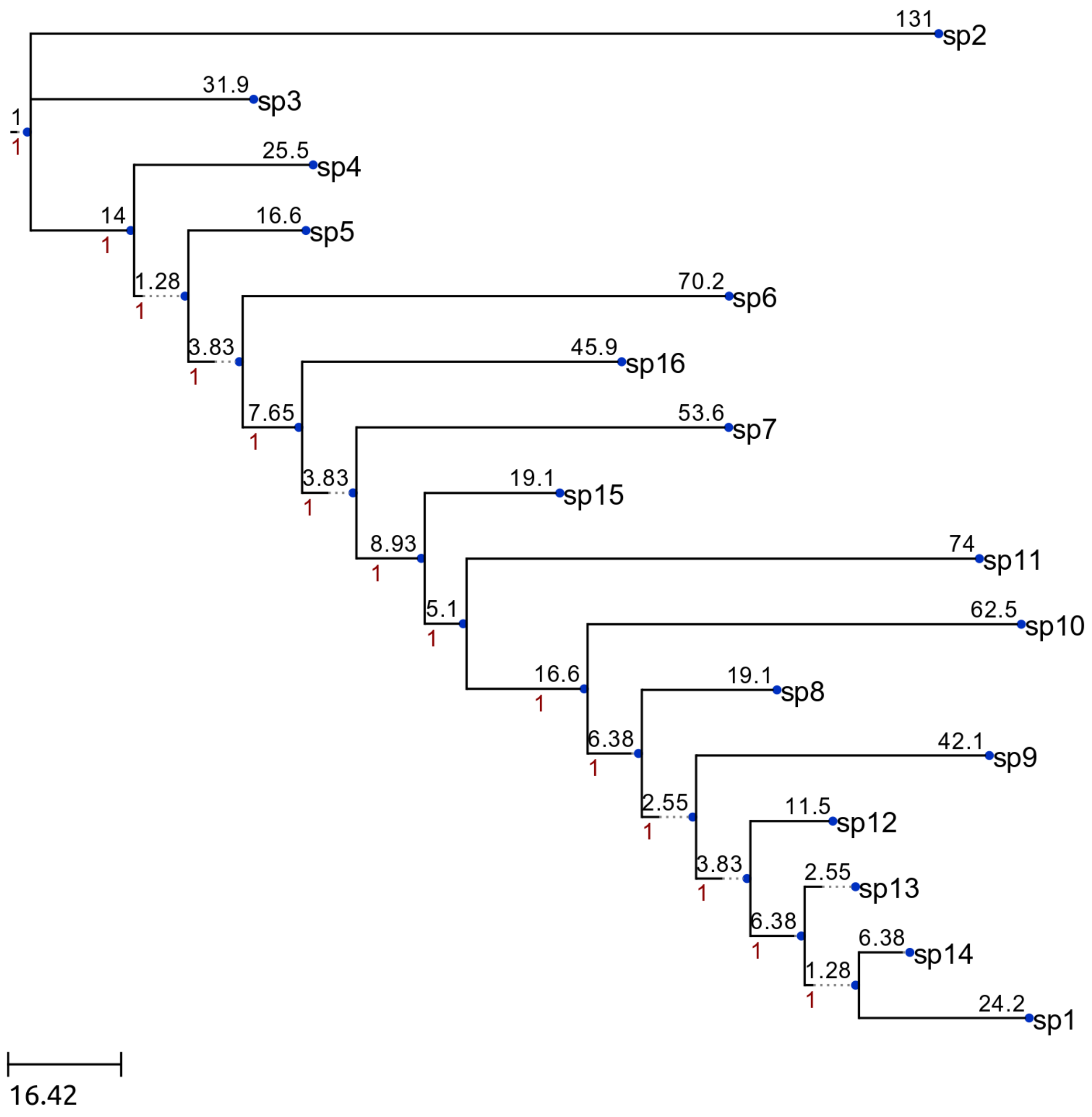


Figure 4: Newick tree corresponding to original s001.align.1 data in Asymmetric 2.0 data set.

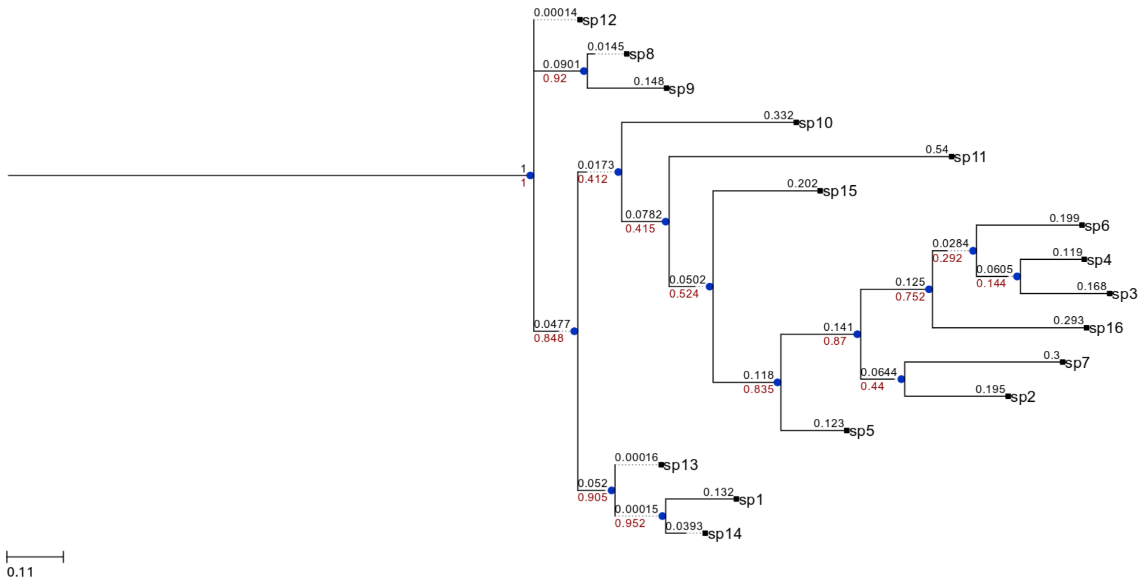
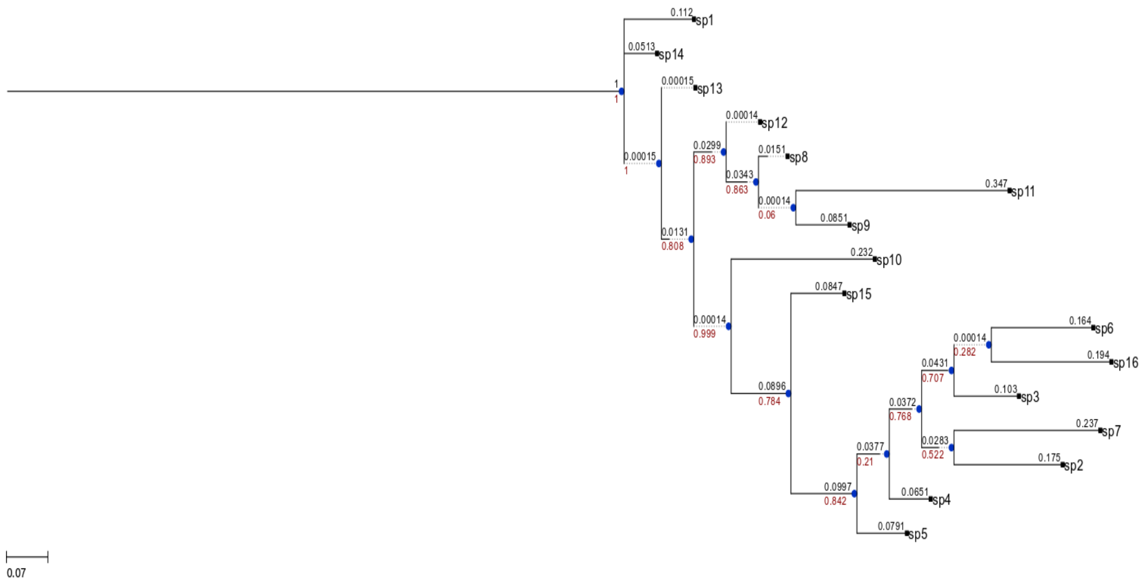


Figure 5: Newick tree corresponding to noise-reduced s001.align.1 data in Asymmetric 2.0 data set.



3 Results and Discussion

The purpose of using controls is to ensure not only that our program is working properly, but to see if our results are understadable. The corner cases such as the empty file as well as the file with the single aligned sequence which generates an exception, are handled successfully by the program. In the remaining control test cases, upon running the noise reduction script, it is observed that the program correctly filteres out the noisy columns. When all of the columns are filtered as noise, the program gives a meaningful error message. The outcomes after running our scripts on the control data are as expected, therefore the correctness of our program is concluded.

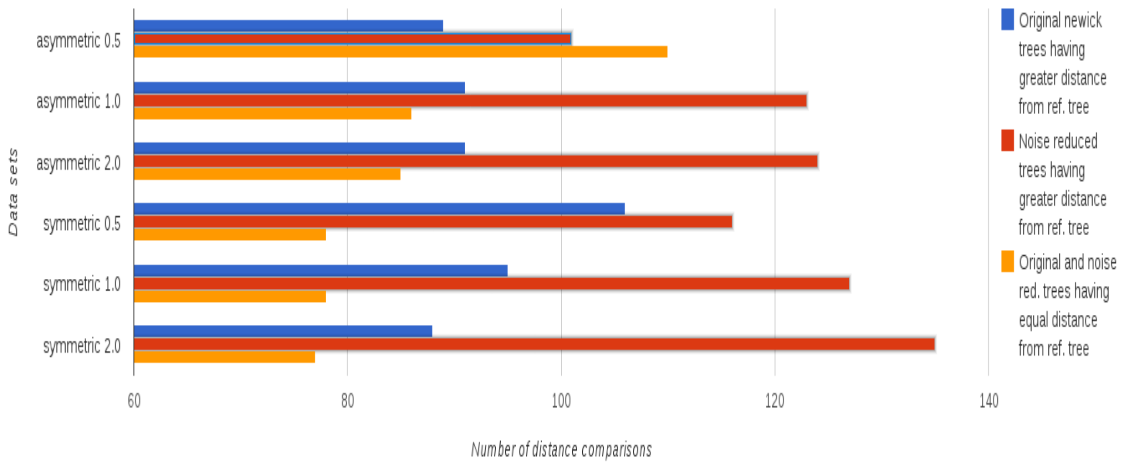
Even though euclidian and Robinson-Foulds distances of the concerned newick trees are also calculated, since it is observed that the differences between the euclidian and Robinson-Foulds distances are much smaller then symmetric distances for all of the six data sets, only the statistical results concluded 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. A reason for this may be due to the fact that the original trees are directly evolved from the reference tree, whereas the noise reduced trees are generated by applying extra operations on their evolution.

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

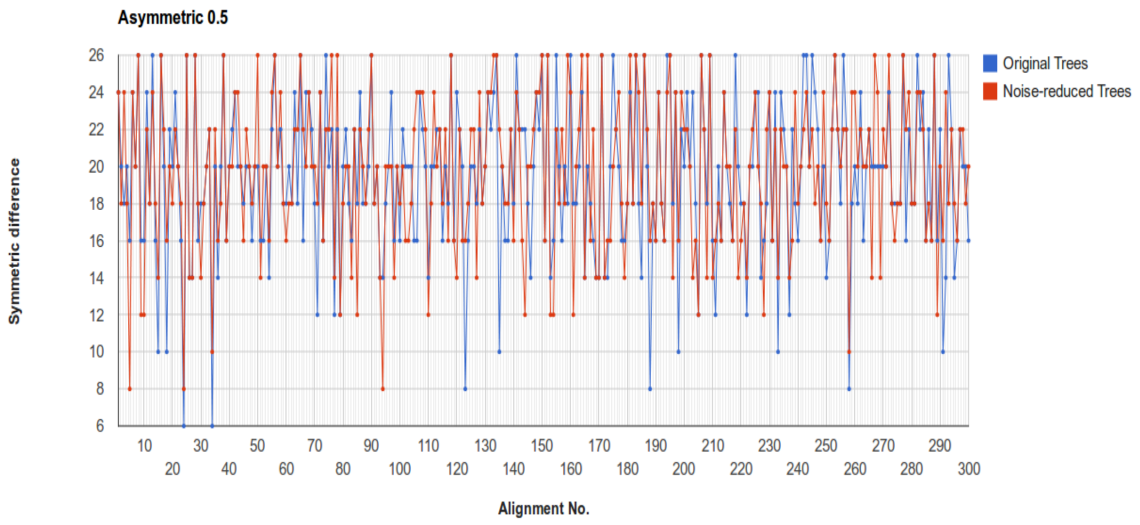


Figure 8: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Asymmetric 1.0

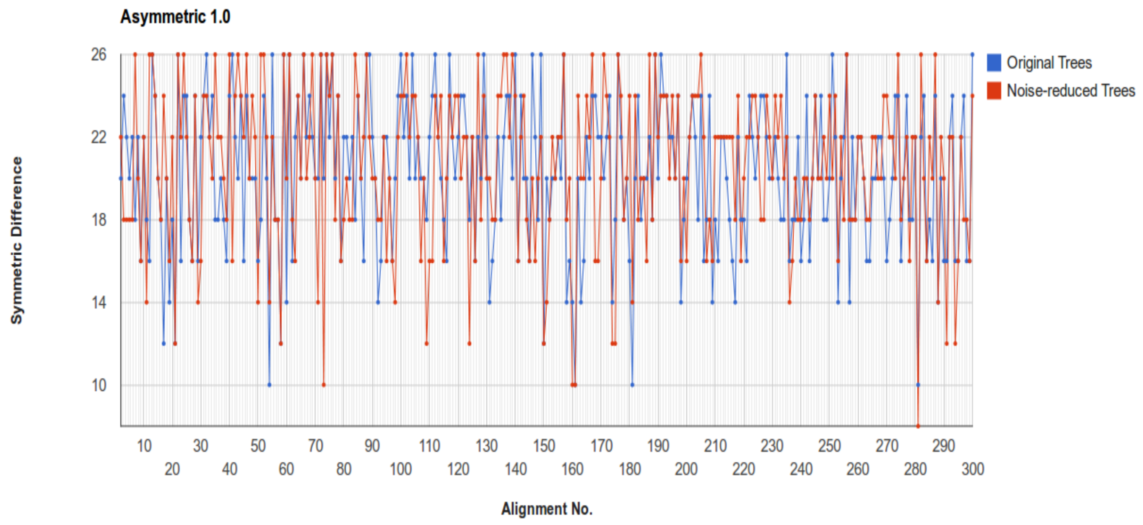


Figure 9: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Asymmetric 2.0

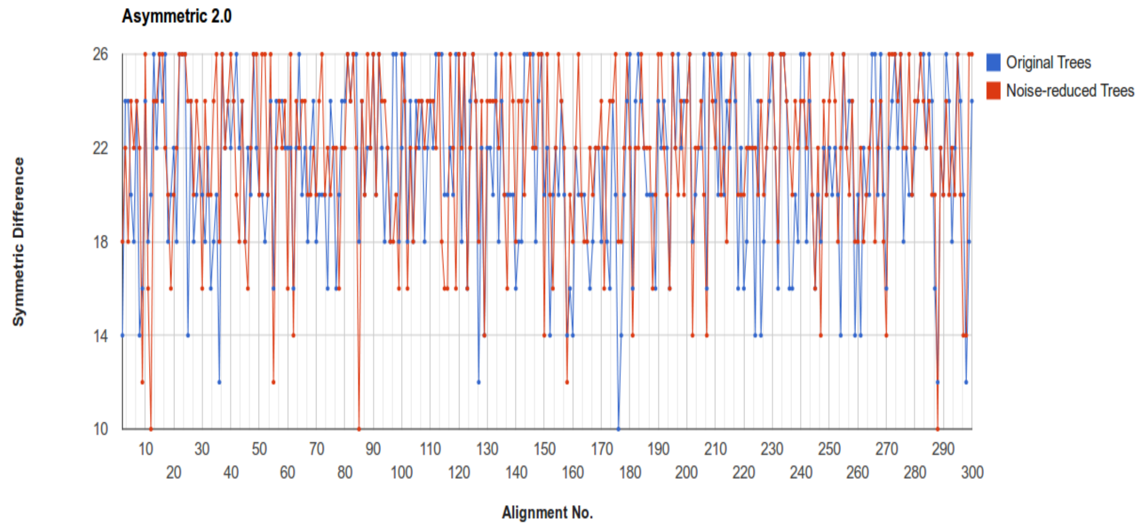


Figure 10: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Symmetric 0.5

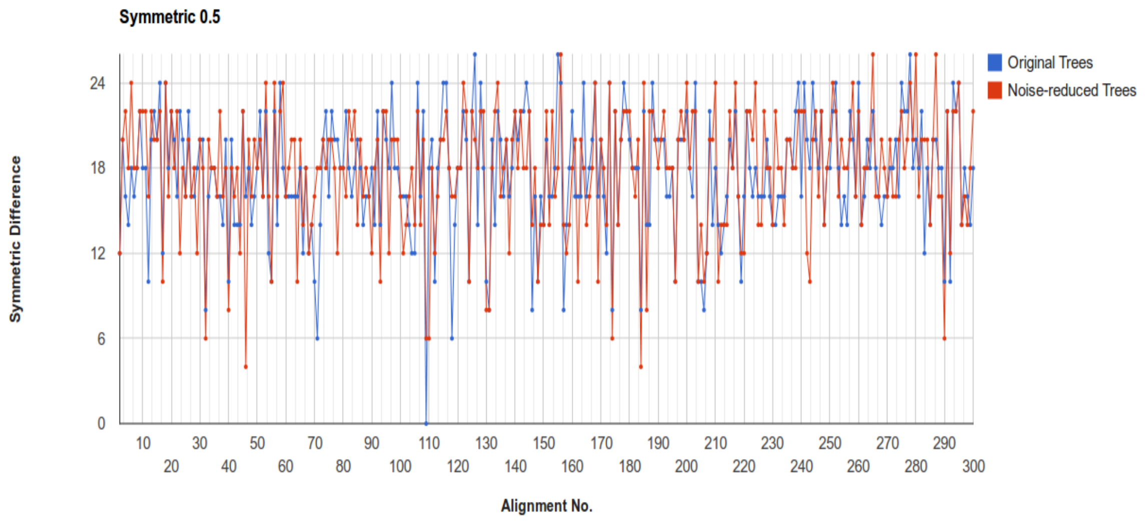


Figure 11: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Symmetric 1.0

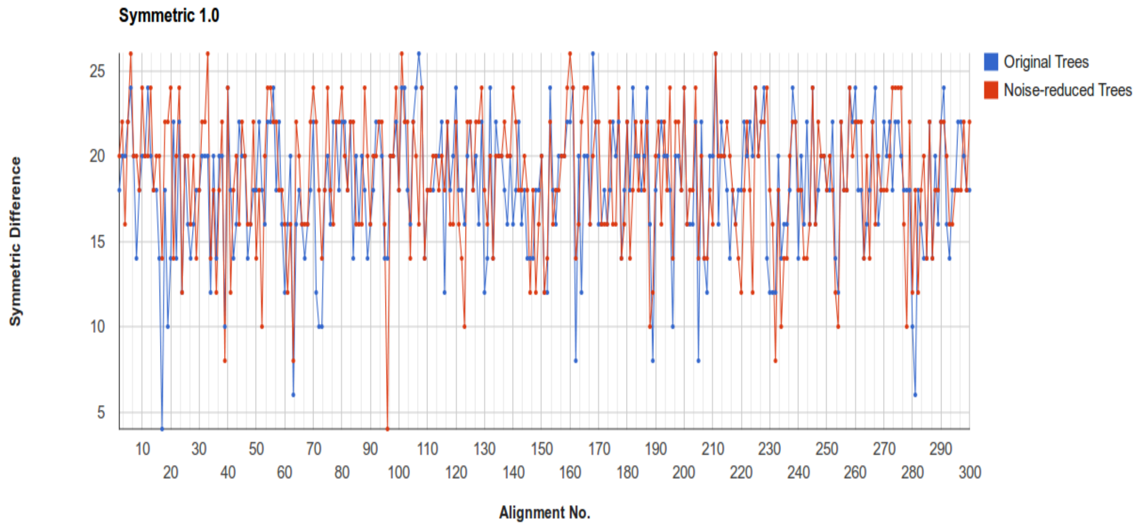
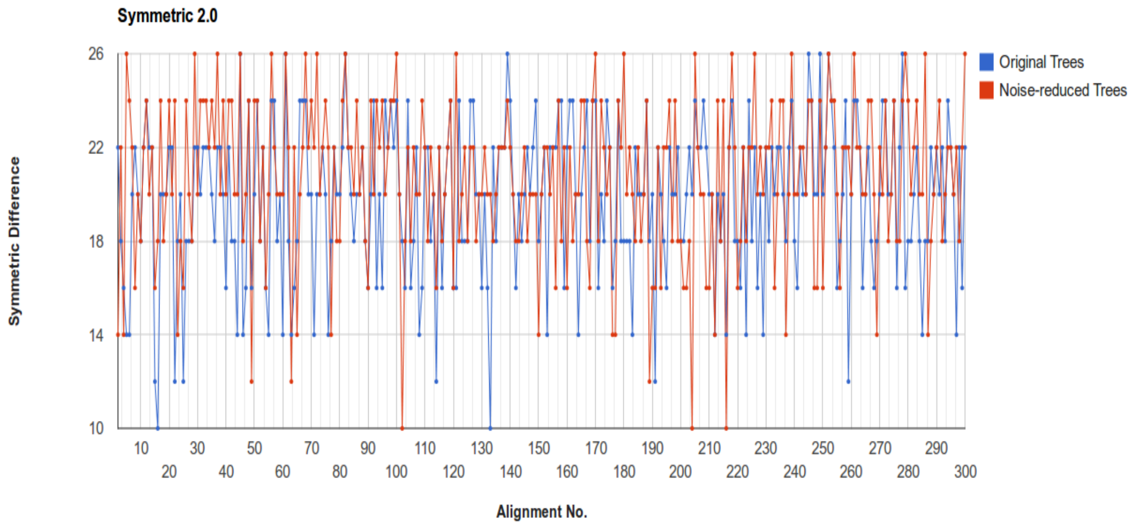


Figure 12: Symmetric distance values between the newick trees of original, noise-reduced multialignments with the reference tree of data set Symmetric 2.0



4.1.2 Project Layout

```
./bin/
    ./bin/compare_trees*
    ./bin/generate_outputs.sh*
    ./bin/markdown2html*
    ./bin/reduce_noise*
./data/
    ./data/control_data/
    ./data/noise_reduced/
    ./data/original_alignments/
./doc/
    ./doc/project_report/
./output
./results/
     ./results/2012-12-23/
     ./results/lab_notebook.html
     ./results/lab_notebook.markdown
```

4.1.3 Python Codes

reduce_noise script

Listing 1: “reduce_noise” script

```
#!/usr/bin/env python

```

```python
# Reducing noise in protein multialignments
# Copyright (C) 2012 Gokcen Eraslan, Basak Eraslan, Javeria Ali
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.

import sys
import argparse
import os
import importlib
from collections import Counter

# Biopython is used to parse and manipulate fasta formatted MSA files
from Bio import AlignIO, Align, Seq, SeqRecord


# matlab style indexing i.e.: x = 'AGCGAGGC'; index_string(x, [0, 3, 4]-> 'AGA'
index_string = lambda string, i: ''.join([string[x] for x in i])


def is_noisy(column):
    """Determines whether given column is noisy or not"""

    # Counter is a neat class to convert a sequence into a dictionary holding
    # the count of each item. Here is an example Counter of a msa column:
    # Counter({'S': 7, 'K': 3, 'N': 2, 'D': 1, 'Q': 1, 'R': 1, 'T': 1})

    counter = Counter(column)
    col_length = len(column)

    # there are more than 50% indels
    if counter["-"] > col_length / 2:
        return True

    # at least 50% of amino acids are unique
    if len(counter) >= col_length / 2:
        return True

    if "-" in counter:
        counter.pop("-")

    # no amino acid appears more than twice
    if not [v for v in counter.values() if v > 2]:
        return True

    return False


def filter_noise_in_columns(msa, noise_func):
    """Filters out the columns if given callable returns False"""

    num_columns = len(msa[0])
    valid_columns = [col for col in range(num_columns)
                     if not noise_func(msa[:, col])]

    assert valid_columns, "All columns seem noisy, exiting..."

    new_msa = []

    for record in msa:
        new_msa.append(SeqRecord.SeqRecord(
                       Seq.Seq(index_string(record.seq, valid_columns)),
                               record.id, record.name, record.description))

    return Align.MultipleSeqAlignment(new_msa)


def generate_phylo_tree(msa, outfilename, format, image_format, tree_module):
    """Generates phylogenetic tree from given sequence alignment file."""

    # PyCogent is used to generate phylogenetic tree from the fasta files
    import cogent

    # ETE toolkit is used to render newick trees as images
    import ete2

    aln = cogent.LoadSeqs(data=msa,
                          moltype=cogent.PROTEIN,
                          aligned=True,
                          label_to_name=lambda x: x.split()[0])

    assert tree_module in ("fasttree", "clearcut", "clustalw",
                           "muscle", "raxml"), "Given tree generator is " \
                           "not supported."

    tree_generator = importlib.import_module("cogent.app.%s" % tree_module)
    tree = tree_generator.build_tree_from_alignment(aln, cogent.PROTEIN)
    tree.writeToFile(outfilename, format=format)

    ete_tree = ete2.PhyloTree(outfilename, msa)
    tree_filename, ext = os.path.splitext(outfilename)
    tree_filename += ".%s" % image_format

    ts = ete2.TreeStyle()
    ts.show_leaf_name = True
    ts.show_branch_length = True
    ts.show_branch_support = True

    ete_tree.render(tree_filename, w=1200, units="px", tree_style=ts)


def main():
    """Main function of the program. Pretty explanatory, eh."""

    parser = argparse.ArgumentParser(description="Reducing noise in protein "
                                                 "multialignments")

    parser.add_argument("-i", "--input", metavar="INPUT_ALIGNMENT_FILE",
                        help="Input file from which the noise to be "
                        "reduced. By default, stdin is used.")

    parser.add_argument("-o", "--output", metavar="OUTPUT_ALIGNMENT_FILE",
                        help="Name of the output file to be created upon the "
                        "noise reduction process. stdout is used if not "
                        "specified.")

    parser.add_argument("-I", "--input-format", metavar="INPUT_ALIGNMENT_"
                        "FORMAT", help="Input file format, \"fasta\" is the "
                        "default. Available formats: clustal, phylip, "
                        "stockholm, emboss.")

    parser.add_argument("-O", "--output-format", metavar="OUTPUT_ALIGNMENT_"
                        "FORMAT", help="Output file format, \"fasta\" is the "
                        "default. Available formats: clustal, phylip, "
                        "stockholm, emboss.")

    parser.add_argument("-d", "--output-tree-reduced", metavar="REDUCED_"
                        "OUTPUT_TREE_FILE", help="Phylogenetic tree is "
                        "generated from the noise-reduced alignment file.")

    parser.add_argument("-g", "--output-tree-original", metavar="ORIGINAL_"
                        "OUTPUT_TREE_FILE", help="Phylogenetic tree is "
                        "generated from the original alignment file.")

    parser.add_argument("-T", "--tree-format",
                        help="File format of output tree files. Default is "
                        "newick, alternative format is xml.",
                        default="newick")

    parser.add_argument("-R", "--tree-rendering-format",
                        help="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.",
                        default="png")

    parser.add_argument("-b", "--tree-generator",
                        help="Use given program to generate phylogenetic "
                        "trees. Default is fasttree. Alternatives are: "
                        "clearcut, clustalw, muscle and raxml. ",
                        default="fasttree")

    args = parser.parse_args()

    if args.input:
        input_file = open(args.input, "r")
    else:
        input_file = sys.stdin

    input_format = "fasta" if not args.input_format else args.input_format
    output_format = "fasta" if not args.output_format else args.output_format

    original_msa = AlignIO.read(input_file, input_format)
    reduced_msa = filter_noise_in_columns(original_msa, is_noisy)

    if args.output:
        output_file = open(args.output, "w")
    else:
        output_file = sys.stdout

    AlignIO.write(reduced_msa, output_file, output_format)

    if args.output:
        output_file.close()

    tree_format = "newick" if not args.tree_format else args.tree_format
    tree_rendering_format = ("png" if not args.tree_rendering_format
                             else args.tree_rendering_format)

    tree_generator = ("fasttree" if not args.tree_generator else
                      args.tree_generator.lower())

    if args.output_tree_reduced:
        generate_phylo_tree(reduced_msa.format("fasta"),
                            args.output_tree_reduced,
                            tree_format,
                            tree_rendering_format,
                            tree_generator)

    if args.output_tree_original:
        generate_phylo_tree(original_msa.format("fasta"),
                            args.output_tree_original,
                            tree_format,
                            tree_rendering_format,
                            tree_generator)


if __name__ == '__main__':
    main()
```

compare_trees script

Listing 2: “compare_trees” script

```
#!/usr/bin/env python

# Phylogenetic tree comparison
# Copyright (C) 2012 Gokcen Eraslan, Basak Eraslan, Javeria Ali
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.

import argparse
import os

import dendropy


def compare_trees(ref_tree, other, method):
    """Compare trees using given method"""

    assert method in ("symmetric_difference",
                      "euclidean_distance",
                      "robinson_foulds_distance"), "Method not supported."

    comparison_func = getattr(ref_tree, method)

    if not isinstance(other, (list, tuple)):
        return comparison_func(other)
    else:
        return [comparison_func(x) for x in other]


def find_files(directory, extension):
    """Find files having a particular extension."""

    file_list = []

    for root, dirs, files in os.walk(directory):

        # skip hidden directories
        dirs[:] = [d for d in dirs if not d.startswith(".")]

        file_list.extend([os.path.join(root, f) for f in files
                          if os.path.splitext(f)[1] == extension])

    return file_list


def main():
    """Main function"""

    parser = argparse.ArgumentParser(description="Phylogenetic tree "
                                                 "comparison.")

    parser.add_argument("reference",
                        help="Reference tree to be used in the comparison.")

    parser.add_argument("tree_dir", metavar='tree-dir',
                        help="Directory containing Newick trees that will be "
                        "compared to the reference tree. File extension of "
                        "the reference tree is used to detect newick trees.")

    parser.add_argument("output",
                        help="Name of the output file to be generated.")

    args = parser.parse_args()

    assert os.path.isdir(args.tree_dir), "tree-dir is not a directory."
    assert os.path.isfile(args.reference), "reference is not a file."

    phylo_ref_tree = dendropy.Tree.get_from_path(args.reference, "newick")
    trees = sorted(find_files(args.tree_dir, ".tree"))
    phylo_trees = [dendropy.Tree.get_from_path(t, schema='newick')
                   for t in trees]

    with open(args.output, "w") as outfile:
        sym = compare_trees(phylo_ref_tree, phylo_trees,
                            "symmetric_difference")

        euc = compare_trees(phylo_ref_tree, phylo_trees, "euclidean_distance")
        rob = compare_trees(phylo_ref_tree, phylo_trees,
                            "robinson_foulds_distance")

        outfile.write("\t".join(["Reference tree", "Other tree",
                                 "Symmetric difference", "Euclidean distance",
                                 "Robinson-Foulds distance"]) + "\n")

        for d1, d2, d3, tree in zip(sym, euc, rob, trees):

            outfile.write("\t".join([args.reference, tree,
                                     str(d1), str(d2), str(d3)]) + '\n')

if __name__ == '__main__':
    main()
```

markdown2html script

Listing 3: "markdown2html" script used to generate nice HTML formatted lab notebook

```
#!/usr/bin/env python

# Simple markdown to HTML converter using Strapdown.js
# Copyright (C) 2012 Gokcen Eraslan, Basak Eraslan, Javeria Ali
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.

import argparse
import sys

# A neat HTML template from Strapdown.js project: http://strapdownjs.com
template = """
<!DOCTYPE html>
<html>
%(title)s

<xmp theme="%(theme)s" style="display:none;">

```

```
%(markdown)s

</xmp>

<script src="http://strapdownjs.com/v/0.1/strapdown.js"></script>
</html>
"""

parser = argparse.ArgumentParser(description="markdown2HTML converter.")

parser.add_argument("-t", "--theme", help="HTML theme of the output. "
                    "Possible values: amelia, cerulean, cyborg, journal, "
                    "readable, simplex, slate, spacelab, spruce, superhero, "
                    "united. Default value is united.")

parser.add_argument("-i", "--input", help="Input file in Markdown format.")
parser.add_argument("-o", "--output", help="Output file in HTML format.")
parser.add_argument("-T", "--title", help="Title of the HTML file.")

args = parser.parse_args()

input_file = open(args.input, "r") if args.input else sys.stdin
output_file = open(args.output, "w") if args.output else sys.stdout

template_dict = {}
template_dict["theme"] = args.theme or "united"
template_dict["title"] = ("<title>%s</title>"% args.title) \
                         if args.title else ""

template_dict["markdown"] = input_file.read()

output_file.write(template % template_dict)

if args.output:
    output_file.close()
```