Project 4 Documentation

Group 27

Mateo Sokac, AU 589901

Freja Kelleris, AU504503

4/10/2017

Revision Sheet

		Revision Description
		Initial Documentation

Project 4 Documentation

Group 27
Mateo Sokac, AU 589901
Freja Kelleris, AU504503

4/10/2017

Revision Sheet

Release No.	Date	Revision Description
Rev. 0	4/10/2017	Initial Documentation

Project 3 Documentation

TABLE OF CONTENTS

1.0 INTRODUCTION

Problem

You should implement an algorithm for computing the RF distance between two unrooted evolutionary trees over the same set of species. The algorithm can e.g. be Day's algorithm as explained in class. You should make a program called rfdist which as input takes two evolutionary trees in Newick format (also referred to as 'New Hampshire format'), and outputs the RF distance between them.

Our program contains two files; rfdist.py and main.py.

rfdist.py
Arguments: 2 newick files and boolean for generating trees or not
Example: python rfdist.py test.newick test.newick true
Description: Implementation of Day's depth first, using library ete3

Main.py
Arguments: "permuted" or "original" && tree or notree
Example: python main.py permuted tree
Description: Implementation of Experiment1. Will produce 8x8 table and store it file named table_permuted.txt or table_original.txt. We store the table in the file since it is too big to visualize it in terminal. Tree argument will produce tree for each input.
***You will find our folder architecture in a zip but if you want to run the program on Your own, make sure that your files contain word "permuted" somewhere in the name (for creating 8x8 table with permuted results).
***Depends on PrettyTable library, so make sure you install it with pip install PrettyTable

We validated our rfdist algorithm by inputting two testing newick trees and getting distance of 8. Each time algorithm picks different middle node by pure randomness and each time we got the same results. Also goes for bigger examples but we don't know the exact value of distance we are expecting.

Implementation:
The rfdist program takes two newick file names as its input and using the Tree function from the ete3 package to parse these files into two trees. A random node is chosen, and the two trees are reconstructed using this node as the root (using the set_outgroup function and taking the children of the resulting tree)and from there the day's algorithm is applied to obtain the rf-distance between the two trees. The depth first search of the first tree is done with the use of the traverse("postorder") search function ete3 provides. For each leaf in the tree the DFS number is stored in the variable 'order; after

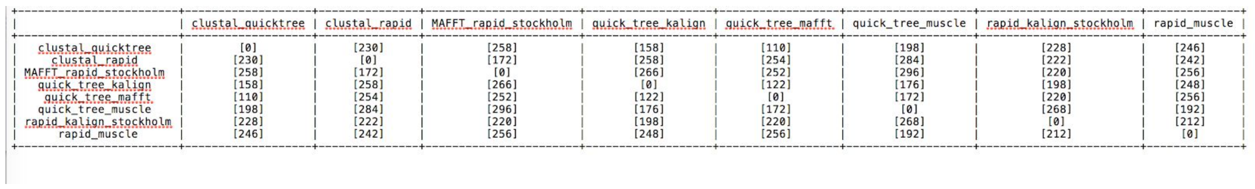
which the order found for the first tree is applied to the corresponding nodes on the second tree. The node interval for the internal nodes are computed for both trees and the shared splits are computed from there.

2.0 EXPERIMENTS

- Experiment1:
  For each alignment method (Clustal Omega, Kalign, MAFFT, MUSCLE), you build a NJ tree using constructed trees. (You might want to use the program Dendroscope to visualize the constructed trees.)

  Newick files are in corresponding folder name with corresponding file name.
  You can run our program in order to get the result by running following command:
  ***make sure that files are in folder named “results”

  python main.py “original”



```
+-----------------------+-------------------+---------------+-----------------------+-------------------+------------------+-------------------+------------------------+---------------+
|                       | clustal_quicktree | clustal_rapid | MAFFT_rapid_stockholm | quick_tree_kalign | quick_tree_mafft | quick_tree_muscle | rapid_kalign_stockholm | rapid_muscle |
+-----------------------+-------------------+---------------+-----------------------+-------------------+------------------+-------------------+------------------------+---------------+
|   clustal_quicktree   |        [0]        |     [230]     |         [258]         |       [158]       |      [110]       |       [198]       |         [228]          |     [246]     |
|     clustal_rapid     |       [230]       |      [0]      |         [172]         |       [258]       |      [254]       |       [284]       |         [222]          |     [242]     |
| MAFFT_rapid_stockholm |       [258]       |     [172]     |          [0]          |       [266]       |      [252]       |       [296]       |         [220]          |     [256]     |
|   quick_tree_kalign   |       [158]       |     [258]     |         [266]         |        [0]        |      [122]       |       [176]       |         [198]          |     [248]     |
|    quick_tree_mafft   |       [110]       |     [254]     |         [252]         |       [122]       |       [0]        |       [172]       |         [220]          |     [256]     |
|   quick_tree_muscle   |       [198]       |     [284]     |         [296]         |       [176]       |      [172]       |        [0]        |         [268]          |     [192]     |
| rapid_kalign_stockholm|       [228]       |     [222]     |         [220]         |       [198]       |      [220]       |       [268]       |          [0]           |     [212]     |
|      rapid_muscle     |       [246]       |     [242]     |         [256]         |       [248]       |      [256]       |       [192]       |         [212]          |      [0]      |
+-----------------------+-------------------+---------------+-----------------------+-------------------+------------------+-------------------+------------------------+---------------+
```

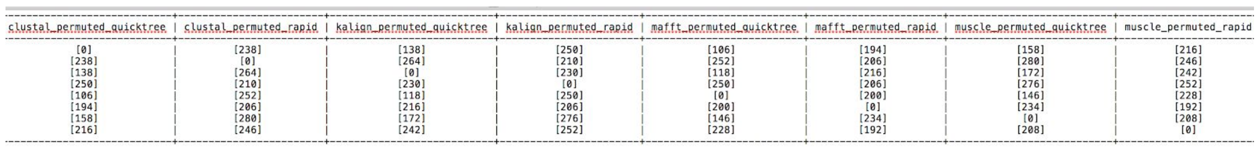
  ***Tables are too big to fit in this doc file, so in order to read it please open attached file named ”table_original.txt”

- Experiment2:
  Redo the above experiment where you use 395 input sequences in patbase_aibtas_permuted.fasta. This yields another 8x8 table.

  Newick files are in corresponding folder name with corresponding file name.
  You can run our program in order to get the result by running following command:
  ***make sure that files are in folder named “results” contain word “permuted” somewhere in name

  python main.py “permuted”

  This table is huge so we cut off the names of sequences



```
---------------------------+------------------------+---------------------------+-----------------------+--------------------------+----------------------+---------------------------+----------------------
clustal_permuted_quicktree | clustal_permuted_rapid | kalign_permuted_quicktree | kalign_permuted_rapid | mafft_permuted_quicktree | mafft_permuted_rapid | muscle_permuted_quicktree | muscle_permuted_rapid
---------------------------+------------------------+---------------------------+-----------------------+--------------------------+----------------------+---------------------------+----------------------
           [0]             |         [238]          |           [138]           |         [250]         |          [106]           |        [194]         |           [158]           |        [216]
          [238]            |          [0]           |           [264]           |         [210]         |          [252]           |        [206]         |           [280]           |        [246]
          [138]            |         [264]          |            [0]            |         [230]         |          [118]           |        [216]         |           [172]           |        [242]
          [250]            |         [210]          |           [230]           |          [0]          |          [250]           |        [206]         |           [276]           |        [252]
          [106]            |         [252]          |           [118]           |         [250]         |           [0]            |        [200]         |           [146]           |        [228]
          [194]            |         [206]          |           [216]           |         [206]         |          [200]           |         [0]          |           [234]           |        [192]
          [158]            |         [280]          |           [172]           |         [276]         |          [146]           |        [234]         |            [0]            |        [208]
          [216]            |         [246]          |           [242]           |         [252]         |          [228]           |        [192]         |           [208]           |         [0]
---------------------------+------------------------+---------------------------+-----------------------+--------------------------+----------------------+---------------------------+----------------------
```

***Tables are too big to fit in this doc file, so in order to read it please open attached file named ”table_permuted.txt”

- Experiment3:
  Compute the RF-distance between the trees produced in 'Experiment 1' and 'Experiment 2' using the same alignment and tree reconstruction method. This yields 8 distances.

  We used quicktree output for permuted and original data

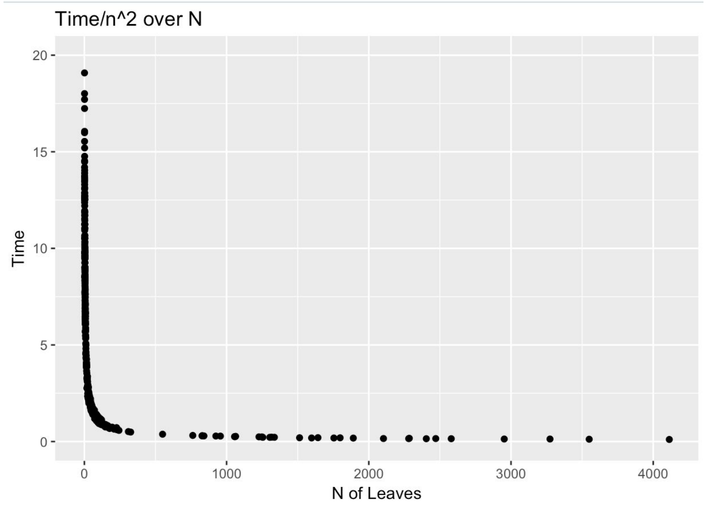
```
+-----------+--------+--------+-------+---------+
|           | Kalign | Muscle | Mafft | Clustal |
+-----------+--------+--------+-------+---------+
|  RapidNJ  | [200]  | [216]  | [170] |  [160]  |
| QuickTree |  [62]  | [158]  |  [44] |   [68]  |
+-----------+--------+--------+-------+---------+
```

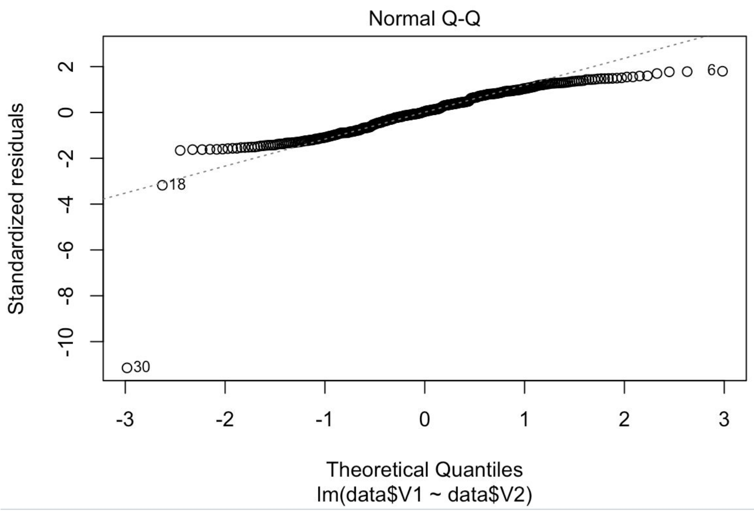
- Experiment4 (optional):
  Redo experiment 1-3 with quartet distance instead of RF-distance. You can e.g. use tqDist to compute the quartet distance.

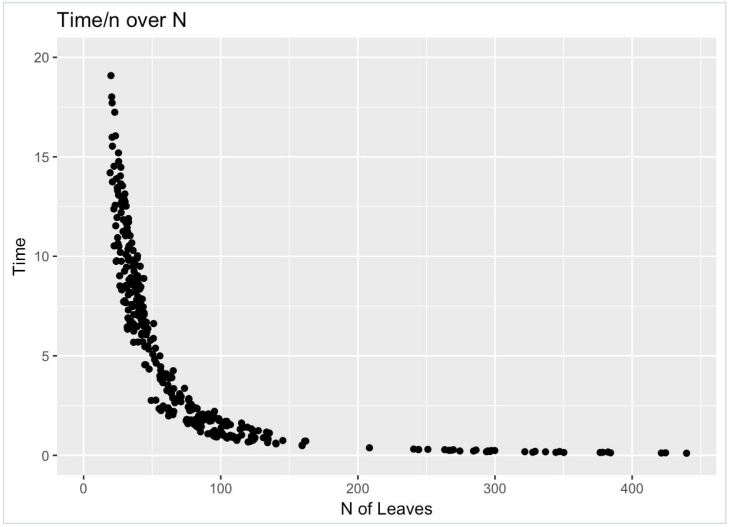
- Experiment 5:
  Make an experiment that shows the running time of your implementation of rfdist. You must choose test data yourself. If the running time is not linear, you should explain why.

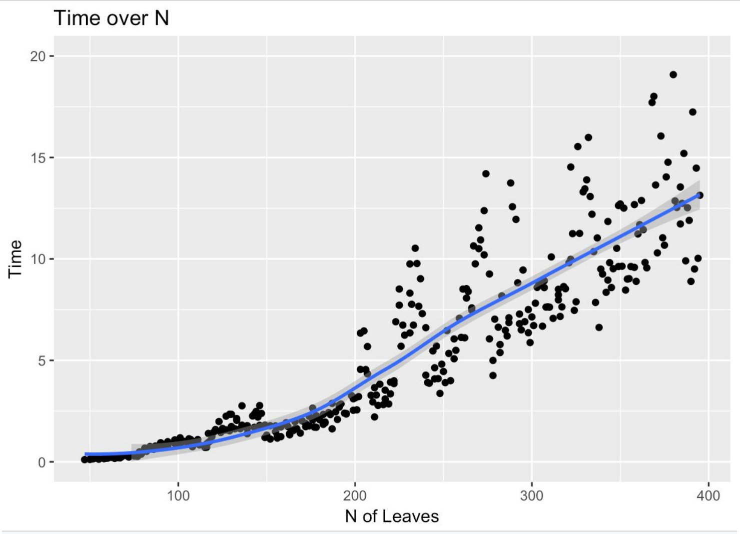
  It seems that something is dragging our running time, so it looks exponential and after some research we concluded that it is because of sorting algorithm. We used ete3 library for performing depth first search and after looking at their code we found they used default python function sorted(). After some investigation online we found out that python default sorting algorithm is hybrid between merge sort and insert sort. Since, merge sort takes O(n log(n)) time and insertion takes O(n) time in best case scenario and O(n^2) in worst case scenario, we can conclude that this is what affects our results. We should use linear sorting algorithm like advanced Radix sort in order to perform linear time results.

  References:
  https://stackoverflow.com/questions/10948920/what-algorithm-does-pythons-sorted-use
  https://www.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/analysis-of-insertion-sort


Time/n^2 over N
Time
N of Leaves
0
5
10
15
20
0
1000
2000
3000
4000


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(data$V1 ~ data$V2)
18
30
6
2
0
-2
-4
-6
-8
-10
-3
-2
-1
0
1
2
3


Time/n over N
Time
N of Leaves
0
5
10
15
20
0
100
200
300
400


Time over N
Time
N of Leaves
0
5
10
15
20
100
200
300
400