Project 4: Tree Comparison

- By Cæcilia, Lea & Peter

Introduction

Unfortunately we could not implement Day's algorithm because of technical problems with the newick format. After several failed attempts, we chose to use the *compare* function from the package *ete3*. Besides that we think that we have answered the questions adequately.

Implementation

To implement rfdist we have used, as mentioned, the *compare* function from the package *ete3*. We used this function on one of the trees with the other tree to compare as an input of the function. In order to use the *compare* function, we must specify that the two trees are unrooted. Then we return the rf-value of the function.

```
from ete3 import Tree

def rfdist(t1, t2):
  result = t1.compare(t2, unrooted=True)
  return result["rf"]
```

To read in the trees in Newick-format, we used the function *Tree* from the package *ete3*. This function takes a file in Newick-format and converts it into a tree as long as we specify that the file is a Newick file.

```
clustal_rapidnj = Tree(newick="clustal_rapidnj.newick")
kalign_rapidnj = Tree(newick="kalign_rapidnj.newick")
muscle_rapidnj = Tree(newick="muscle_rapidnj.newick")
clustal_quicktree = Tree(newick="clustal_quicktree.newick")
kalign_quicktree = Tree(newick="kalign_quicktree.newick")
muscle_quicktree = Tree(newick="muscle_quicktree.newick")
```

To validate the correctness of our implementation, we used the test seqs data and compared our results with the expected results.

Experiments

Experiment 1:

A 6x6 table showing the RF-distance between each pair of constructed trees based on the patbase_aibtas.fasta

RF distance	clustal_rapidnj	kalign_rapidnj	muscle_rapidnj	clustal_quicktree	kalign_quicktree	muscle_quicktree
clustal_rapidnj	0	274	242	230	296	284
kalign_rapidnj	274	0	286	290	232	318
muscle_rapidnj	242	286	0	246	298	192
clustal_quicktree	230	290	246	0	206	198
kalign_quicktree	296	232	298	206	0	264
muscle_quicktree	284	318	192	198	264	0

As we can see in the table above, the largest RF distance is between the two different multiple alignment algorithms and tree-building algorithms, which is also what we expected. Equivalently, we can see that when we compare two trees that have the same alignment algorithm or the same tree-building algorithm, the RF distance is smaller as expected.

Experiment 2:

A 6x6 table showing the RF-distance between each pair of constructed trees based on the patbase_aibtas_permuted.fasta

RF distance	clustal_rapidnj	kalign_rapidnj	muscle_rapidnj	clustal_quicktree	kalign_quicktree	muscle_quicktree
clustal_rapidnj	0	280	246	238	298	280
kalign_rapidnj	280	0	268	288	226	298
muscle_rapidnj	246	268	0	216	284	208
clustal_quicktree	238	288	216	0	198	158
kalign_quicktree	298	226	284	198	0	224
muscle_quicktree	280	298	208	158	224	0

As we can see in the table above, the RF distances look more or less the same as in experiment 1, with some distances being a bit larger than in experiment 1 and some being a bit shorter than in experiment 1.

Experiment 3:

A 1x6 table showing the RF-distance between the trees constructed based on the same alignment and NJ method for normal and the permuted dataset.

	clustal_rapidnj	kalign_rapidnj	muscle_rapidnj	clustal_quicktree	kalign_quicktree	muscle_quicktree
RF distance	160	264	216	68	172	158

We are a bit surprised by the results in experiment 3, since we apply the same alignment method and tree-building method. We expected a smaller RF distance like the one for clustal_quicktree. In conclusion it turns out that the permutation of data has a huge impact on the trees constructed.