Paula Rodrigo, Genona Torruella, Bjarke Pedersen

TREE COMPARISON

Status of work: The *rfdist* algorithm works and passes the tests. We used the permuted and non permuted trees from our classmates repository (Curro, Noa, Andrés, Razvan), because we were not able to install the rapidnj and quicktree on our (Windows) computers.

A short description of your implementation of rfdist, explaining what algorithm you implemented, how you read in trees in Newick-format, how you validated the correctness of your implementation, and how to access and run your program rfdist.

The Newick parser is implemented using Phylo from Biopython.

The Rfdist implementation works by representing splits as binary vectors. The Basic idea behind it is to consider that: any edge ‘e ‘ in a tree corresponds to a split of the species into two disjoint sets. To calculate the rf dist we only have to consider the non-trivial splits, in each tree, which corresponds to splits in an internal edge. Finding each non-trivial split is done using get_subtrees(). Get_subtrees() works by finding all splits in internal edges and representing the corresponding split by a binary vector ‘1 ‘if the ‘species’ is present in the split and ‘0’ if not, then we compute unique binary vectors for all splits. The rf_dist is hereafter computed by subtracting all ‘shared subtrees’ from a list containing ‘all subtrees’. The leftover subtrees which are unique to tree1 or tree2 are counted and the rf_distance is returned.

We validated the correctness of our algorithm empirically using both the test trees from brightspace, which did return the correct value, and some simple cases considering two identical trees, both of the completely same topology and trees that were overall identical but the internal branches turned e.g ((a,b),c) and ((b,a),c) and variations of this.

The program can be executed by running in the command line:

```
python3 rfdist.py tree1.new tree2.new
```

References to the 6 alignments (in Stockholm-format) and trees (in Newick format) that you have produced in Experiment 1 and 2.

All files can be found in the github.

Paula Rodrigo, Genona Torruella, Bjarke Pedersen

Your results of Experiment 1-3, i.e. an 6x6 table showing the RF-distance between each pair of constructed trees based on the patbase_aibtas.fasta, an 6x6 table showing the RF-distance between each pair of constructed trees based on the patbase_aibtas_permuted.fasta, and 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. Comment on your results. Is everything as expected, or are you surprised?

We get slightly lower RF-distances in the permuted trees analysis than in the non permuted one. That either means: applying permutation to the construction of the tree gets closer results to the real tree or that the results are more similar because the same error is being committed in all of them (if you bake a cake wrong several times, they will be similar, but not well done).

	clustal_aib_quicktree	clustal_aib_rapidnj	kalig_aib_quicktree	kalig_aib_rapidnj	muscle_aib_quicktree	muscle_aib_rapidnj
clustal_aib_quicktree	0	264	206	318	198	246
clustal_aib_rapidnj	264	0	316	274	308	272
kalig_aib_quicktree	206	316	0	258	264	298
kalig_aib_rapidnj	318	274	258	0	338	310
muscle_aib_quicktree	198	308	264	338	0	192
muscle_aib_rapidnj	246	272	298	310	192	0

*non permuted trees

	clustal_aib_quicktree	clustal_aib_rapidnj	kalig_aib_quicktree	kalig_aib_rapidnj	muscle_aib_quicktree	muscle_aib_rapidnj
clustal_aib_quicktree	0	268	198	288	158	216
clustal_aib_rapidnj	268	0	318	300	300	272
kalig_aib_quicktree	198	318	0	226	224	284
kalig_aib_rapidnj	288	300	226	0	298	268
muscle_aib_quicktree	158	300	224	298	0	208
muscle_aib_rapidnj	216	272	284	268	208	0

*permuted trees

In the comparison between all trees (both permuted and non permuted), we notice higher RF-distances for trees constructed by different algorithms. For instance, the RF-distance between kalig_permuted_quicktree and clustal_permuted_quicktree is smaller (198) than the RF-distance of kalig_permuted_rapidnj and clustal_permuted_quicktree (288). The only difference between them is the algorithm used to construct the tree and it applies to all the examples.

	clustal_aib_quicktree	clustal_aib_rapidnj	kalig_aib_quicktree	kalig_aib_rapidnj	muscle_aib_quicktree	muscle_aib_rapidnj
Distance permuted vs nonpermuted	72	162	172	290	158	216

You can find the experiments notebook here.