

# SplitDist—Calculating Split-Distances for Sets of Trees

Thomas Mailund

Bioinformatics Research Center (BiRC),  
University of Aarhus, Ny Munkegade, Bldg. 540, DK-8000 Århus C

## ABSTRACT

**Summary:** We present a tool for comparing a set of input trees, calculating for each pair of trees the split-distances, i.e., the number of splits in one tree not present in the other.

**Availability:** The source code for SplitDist is available under the GNU Public License (GPL) from <http://www.daimi.au.dk/~mailund/split-dist.html>.

**Contact** mailund@birc.dk

## INTRODUCTION

For any tree, the edges of the tree can be thought of as splitting the set of leaves into two sets: the leaves on either sides of the edge. In this regard, there is a close relationship between the edges of a tree and the possible splits of the leaves into two sets. Each edge in the tree corresponds to a split of the leaves—a unique split if all nodes in the tree has degree at least three—and any unrooted tree with degree at least three for all nodes can be re-constructed from its set of splits [Gusfield, 1991].

Consider the tree with leaves A, B, C, D, E shown on the left below. Besides the trivial splits, separating a leaf from the rest of the tree, this tree contains two splits: A, B | C, D, E and A, B, C | D, E, corresponding to the edges from inner nodes  $i_1$  to  $i_2$  and  $i_2$  to  $i_3$  respectively.



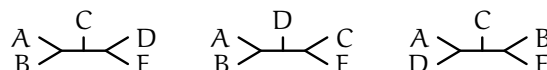
When comparing two trees with the same set of leaves, one question to ask is: which edges—or splits—are found in both trees, and which edges are only found in one of the two? Comparing the two trees above, we see that the two trees share all trivial splits—which will always be

the case—and the split A, B | C, D, E, but not the splits A, B, C | D, E—only found in the first tree—and A, B, D | C, E—only found in the second.

We have built a tool for answering this and related questions; calculating the number of splits shared or differing between pairs of trees, calculating the union and intersection of splits in a set of trees, calculating the frequency the splits appear with in the set of trees, and annotating the trees' edges with the frequencies to show the degree to which the individual edges are supported by the total set of edges.

## FEATURES

Given a set of input trees, our tool computes, for each pair  $t_i, t_j$ , the number of splits in  $t_i$  not found in  $t_j$ . For  $n$  input trees, this is output as a  $n \times n$  matrix. Consider the three trees shown below. As already mentioned, the first two trees differs on one split; the third differs from the other two on both non-trivial splits.



Running the SplitDist tool on these three trees will produce the distance matrix:

```
> sdist t1.tree t2.tree t3.tree
Distance Matrix:
t1.tree :   -   1   2
t2.tree :   1   -   2
t3.tree :   2   2   -
```

The absolute number of different splits between two trees is not necessarily a good measure of the relatedness of the two trees; two large but closely related trees can easily have a larger absolute split distance than two unrelated smaller trees, simply because larger trees have

more splits. To alleviate this problem, our tool can also output the *normalised* distance:  $d_{ij}/T_i$  for each pair  $t_i, t_j$ , where  $d_{ij}$  is the absolute split distance between  $t_i$  and  $t_j$ , and where  $T_i$  is the total number of splits in  $t_i$ . A measure closely related to the normalised distance between two trees, also computed by our tool, is the *similarity* between the two trees:  $s_{ij}/T_i$  where  $s_{ij}$  is the number of splits shared between  $t_i$  and  $t_j$ . Using SplitDist to calculate these values of the three trees from before produces the following output:

```
> sdist --print-norm --print-sim \
      t1.tree t2.tree t3.tree
Normalised Distance Matrix:
t1.tree :      - 0.142857 0.285714
t2.tree : 0.142857      - 0.285714
t3.tree : 0.285714 0.285714      -
Similarity Matrix:
t1.tree :      - 0.857143 0.714286
t2.tree : 0.857143      - 0.714286
t3.tree : 0.714286 0.714286      -
```

For each split in tree  $t_i$ , we can count how often this split is found in the total set of trees. If a large fraction of the trees contains a given split, that split is more supported by the input than splits found in a smaller fraction of the trees; if the trees are built using tree reconstruction heuristics, or from re-sampling using e.g. the bootstrap method [Nei and Kumar, 2000], we have higher confidence in the edges corresponding to splits with high support than the edges corresponding to splits with low support. Using SplitDist, we can calculate the support of each edge in the input trees, and output the trees where each edge is annotated with its support. For the three trees from before, the annotated trees will look like this:

```
> sdist --print-trees t1.tree t2.tree t3.tree
Annotated Trees:
t1.tree : (( 'A' 1, 'B' 1) 0.666667, 'C' 1,
           ('D' 1, 'E' 1) 0.333333)
t2.tree : (( 'A' 1, 'B' 1) 0.666667), 'D' 1,
           ('C' 1, 'E' 1) 0.333333
t3.tree : (( 'A' 1, 'D' 1) 0.333333), 'C' 1,
           ('B' 1, 'E' 1) 0.333333
```

SplitDist can also calculate the union of splits in the input set, e.g. the set of splits appearing in any input tree, and the intersection of the splits in the trees, e.g. the set of splits appearing in all input trees. On the example set, it will produce the following:

```
> sdist --print-shared-splits \
      --print-all-splits \
      t1.tree t2.tree t3.tree
Shared Splits:
All Splits:
A B | C D E : 2
A B C | D E : 1
A B D | C E : 1
A C D | B E : 1
A D | B C E : 1
```

Only non-trivial splits are printed, and, as shown, the three example trees do not all share any non-trivial split; the split  $A, B | C, D, E$  appears in two of the trees, but the remaining only appears in one each.

## IMPLEMENTATION

SplitDist implements the above features in C++ which should compile on any platform supporting autoconf, make, and gcc. The program takes as input a set of unrooted trees in Newick format, and outputs distance matrices, annotated trees, and the union or intersection of the splitsets found in the trees, depending on the options given to the tool. Use the `--help` option for more information.

## REFERENCES

- [Gusfield, 1991] Gusfield, D. (1991). Efficient Algorithms for inferring Evolutionary Trees. *Networks*, 21:19–28.
- [Nei and Kumar, 2000] Nei, N. and Kumar, S. (2000). *Molecular Evolution and Phylogenetics*, chapter 9.3, pages 171–174. Oxford University Press.