**RFS v2.0 Manual**

RFS is a supertree program that takes a set of rooted species trees as input and heuristically searches for a binary supertree that minimizes the total (rooted) Robinson-Foulds distance (i.e. symmetric difference) between the supertree and the input trees. The intuition behind seeking a binary supertree, even though the input trees may be unresolved, is that, under this setting, minimizing the RF distance is equivalent to maximizing the number of shared clades between the input trees and the supertree. Thus, we seek a supertree that is consistent with the largest number of clusters from the input trees. Further details about this method are available in the following paper:

"Robinson-Foulds Supertrees", Mukul S. Bansal, J. Gordon Burleigh, Oliver Eulenstein, and David Fernández-Baca. *Algorithms for Molecular Biology* 2010, 5:18.

RFS implements the fast SPR based local search heuristic described in the above paper. The initial supertree for the search can be built either by a stepwise taxon addition algorithm or randomly. This initial tree can also be provided by the user. RFS then runs the SPR based hill climbing heuristic starting from this initial tree. This heuristic seeks to find a best tree in the SPR neighborhood of the currently best supertree. This procedure is called a local search step. If there are multiple best trees in the neighborhood, then one is chosen randomly and the local search step is repeated with this new tree. The heuristic terminates when no better trees can be found in the local search step. The new version of RFS (v2.0) implements a slightly modified version of the local search heuristic that uses a more relaxed hill-climbing strategy. This modification speeds up the heuristic search significantly, without affecting its accuracy. The gain in speed allows us to perform more ratchet iterations (see the paragraph below). The result is that the new version of the program is not only considerably faster but also more accurate.

To avoid getting caught in local minima, RFS also implements a ratchet search strategy like the one used for parsimony. In general, a ratchet search performs a number of iterations that consist of two local SPR searches: one in which the characters (input trees) are equally weighted, and another in which the set of the characters are re-weighted. RFS re-weights the characters by randomly removing approximately two-thirds of the input trees. The goal of re-weighting the characters is to alter the tree space to avoid getting caught in a globally suboptimal part of the tree space. At the end of each iteration, the best tree from the unweighted SPR search is taken as the starting point of the next iteration. The number of ratchet steps performed by RFS v2.0 is set to 50 (the previous version of RFS performed only 25 ratchet iterations). When the program terminates it outputs all those trees found during the search that have the lowest cost. The number of optimal trees output is at least 1 and at most 51. If more than one tree is output, then each of those trees is equally optimal, and users may find it useful to combine them all into a single consensus tree (we recommend using strict consensus) using other software.

In general, we strongly recommend that users perform several runs of RFS on any data set. Due to the random steps involved in the stepwise taxon addition heuristic as well as the SPR based local search heuristic, each run of the program is likely to result in different trees.

General input/output
---

All input and output is handled as plain ASCII text. White spaces are allowed when they don't affect the syntax.

All input trees must be expressed using the Newick format terminated by a semicolon, and they must be fully binary (fully resolved). Multifurcations in the input trees will trigger a warning or error message. Species names in the input trees must be unique.
e.g. ((speciesA,speciesB),speciesC);

Lables with non-alphabetic characters (e.g spaces) have to be encapsulated in apostrophes or quotation marks.
e.g. speciesA, "species A" or 'species A'

Trees can span multiple lines and contain comments. Comments have to be encapsulated in square brackets.
e.g.

```
(
  (speciesA,speciesB), [example comment text]
  speciesC
);
```

Program options
---

| | |
|---|---|
| -i, --input | Input file |
| -o, --output | Output file |
| -g, --generator 0\|1\|2 | The algorithm used to build the starting supertree.<br>0 - don't generate<br>In this case the first tree in the input file is assumed to be the initial supertree.<br>1 - stepwise taxon addition heuristic [default]<br>We randomly pick three leaves from the species leaf set, and then consider all the triplets that these three leaves can form. We randomly pick a triplet with the best score as the starting supertree. Then, we randomly pick another leaf to add to the current supertree. We try to add this leaf in all possible positions in the current supertree, and store all those positions that give the best score. Then we randomly pick a position from among these best locations and add the leaf at that location. In this way, we incrementally add all species to the supertree.<br>2 - random tree<br>The initial supertree obtains a random topology. |
| -q, --quiet | No processing output. |
| --seed <integer number> | Set a user defined random number generator seed. By default the seed is generated from the local wall clock. |
| -v, --version | Output the version number. |

| | |
|---|---|
| -h, --help | Output a brief help message with information about the various arguments. |

The RF-report tool
---

In addition to RFS, we also provide a tool, RF-report, that takes as input a supertree and the set of input trees, and outputs the total (un-normalized) RF distance from the supertree to the input trees. As with RFS, the input trees must be in newick format. The first tree in the input file is assumed to be the supertree, and the remaining trees are assumed to be the input trees. The input file is specified using the "-i" option.

e.g.

```
RF-report -i input.newick
```

NOTE: This tool works correctly only if the supertree is fully resolved. The other input trees, however, may be unresolved.

Example dataset
---

As an example, we provide the sea-birds dataset mentioned in the paper. This dataset consists of seven input trees. To run RFS on this dataset, the user would type:

```
RFS -i seabirds.newick -o outputfile
```