# PluMiST - Plus and Minus SuperTrees

Manual Version 1.1 (March 11, 2011)

**Anne Kupczok**
Current address:
Institute of Science and Technology, Austria
Am Campus 1
3400 Klosterneuburg, Austria
email: `anne.kupczok@ist.ac.at`

## Contents

## 1 Legal Stuff

## 2 Introduction

`PluMiST` is a computer program for reconstructing and evaluating majority rule supertrees: MR(-), MR(+) and MR(+)g (Cotton and Wilkinson, 2007; Dong and Fernández-Baca, 2009). The tree space is searched by NNI (nearest-neighbor interchange) and TDR (taxa-deletion-reinsertion). Only fully resolved input trees can be considered for MR(+) and MR(+)g. Bifurcating supertrees are scored, while multifurcating trees may be returned as the strict consensus of equally best scoring trees.

The details of the implemented algorithm and a simulation study using this program can be found in Kupczok (2011). For an input tree $\mathcal{G}$ and a supertree $\mathcal{S}$ with $n_{\mathcal{G}}$ and $n$ taxa, respectively, the distances are computed as follows:

$$d^{-}(\mathcal{S},\mathcal{G}) = 2 \times C, \qquad d^{+}(\mathcal{S},\mathcal{G}) = 2 \times B, \qquad d^{+}_{g}(\mathcal{S},\mathcal{G}) = C + B,$$

where $B$ is the number of splits in $\mathcal{S}$ that are incompatible to at least one split in $\mathcal{G}$ and $C$ is the number of splits in $\mathcal{G}$ that are incompatible to at least one split in $\mathcal{S}$. The sum over the distances of all input trees is called the *score* of a supertree with the respective supertree method. Note that the multiplication with two in the distance computations of $d^{-}$ and $d^{+}$ is omitted in the output of PluMiST. In detail, the sum over $C$ is considered as the score of MR(-); the sum over $B$ is considered as the score of MR(+); and the sum over $B + C$ is considered as the score of MR(+)g.

## 3 Installation

The program is freely available from http://www.cibiv.at/software/plumist/. The command-line program is written in python and should run on every computer with python version 2.4 or newer versions of python 2.x. It will not run with python 3.x. Python can be downloaded from http://www.python.org/.

First unzip plumist-1-1.zip, then change into the directory PluMiST and type
python plumist.py [options] to run the program.

## 4 Command-line options

Run python plumist.py -h to print out a short description of available options:

```
Usage: plumist.py [options]

Options:
  -h, --help            show this help message and exit

  Input options:
    -s SOURCE, --source=SOURCE
                        Filename for input trees in newick format (required)
    -t TAXA, --taxa=TAXA
                        Taxafilename: each taxon one line (default no file,
                        read out of gene trees)

  Output options:
    -o HEADER, --head=HEADER
                        Header for output (default 'plumist_'{alg})

  Type of computation:
    -e, --eval          Only evaluate scores of all trees in stree file
                        without treesearch (default off). Note that
                        multifurcating trees are resolved and the minimum
                        distance is returned (-1 returned in case of too many
                        multifurcations)
    -p POST, --post=POST
                        Type of Analysis: '0' - complete Algorithm (default),
                        '1' - only starting tree and heuristic tree search,
                        '2' - skip treesearch and use starting trees for
                        consensus (extract minimal score first)

  Algorithm options:
    -r STREE, --tree=STREE
                        Starting tree: random ('r'), step-wise addition tree
                        ('m', default) or filename with tree in newick format
    -a OPT, --alg=OPT   Type of optimizing function: 'm' - minus (default),
                        'p' - plus, 'g' - plusg
    -f FRAC, --frac=FRAC
                        Fraction of taxa deleted in optimization step,
                        default: 0.25, if set to 0, only NNI optimization
```

## 4.1 Examples

```
python plumist.py -s examples/source6.trees -r examples/trees_6 -o matrep6p -t
examples/taxa -a p -e
python plumist.py -s examples/source6.trees -r examples/trees_6 -o matrep6m -t
examples/taxa -a m -e
python plumist.py -s examples/source6.trees -r examples/trees_6 -o matrep6g -t
examples/taxa -a g -e
```

For testing, tree these commands and compare the resulting files to the ones in the `example`-directory. Note that both commands only evaluate the trees, thus no heuristic is involved.

```
python plumist.py -s examples/source.trees
```

This command searches for an MR(-) tree. Files starting with `plumist_minus` will be generated, they should be similar to those in the example directory. Adding the option `-a p` and `-a g`, will compute MR(+) and MR(+)g trees, respectively.

## 4.2 Input options

`-s SOURCE, --source=SOURCE` The filename of a list of input (gene) trees in newick format. The gene trees may contain missing data, but the user must ensure that each tree contains at least one inner split and that the overlap is sufficient. All trees are interpreted as unrooted. If you want to use rooted trees, the root must be added as an extra taxon. See examples `source6.trees` and `source.trees` in the `example`-directory.

`-t TAXA, --taxa=TAXA` A taxa list may be given, where each taxon appears on one line. If the taxa are not given, they are extracted from the gene trees. See example `taxa` in the `example`-directory.

## 4.3 Output options

`-o HEADER, --head=HEADER` Header for output file. Different output files are generated depending on the type of analysis (see section 5).

## 4.4 Type of computation

`-e, --eval` All trees in the `STREE`-file are read and scored. The output of a tree evaluation is described in section 5.1.

`-p POST, --post=POST` The steps of the algorithm can be addressed independently using `-p 1` or `-p 2`. It is possible to run the first part several times using a different `HEADER`. The resulting tree files should then be concatenated and used as `STREE` for the consensus step. The output of a tree search is described in section 5.2.

### 4.5 Algorithm options

`-r STREE, --tree=STREE` List of trees in newick format or flag "r" for a random starting tree or flag "m" for a step-wise addition tree. When a tree search is performed, the first tree in the file is used as the starting tree and the others are ignored. When the trees are evaluated (option `-e`) or summarized (option `-p 2`) the trees are interpreted as possible supertrees and all are scored. See `trees_6` in the `example`-directory.

`-a OPT, --alg=OPT` Here, the optimizing function is specified, `m` searches for MR(-)supertrees, `p` for MR(+)supertrees, and `g` for MR(+)g-supertrees.

## 5 Output specification

Different files may be generated, depending on the options:

### 5.1 Tree evaluation

**eval** The `*.eval` file contains the scores of all trees in the same order as in the `STREE` file.

**tree** The `*.tree` file contains all best scoring trees among the input trees.

### 5.2 Tree search

**start** The `*.start` file contains the starting tree in newick format if it was not given by the user.

**trees** The `*.trees` file contains trees with the minimal scores. It is updated during the run. After the program is finished, it contains all trees of minimal score that were found.

**strict** The `*.strict` file contains the strict consensus of the best-scoring trees. Thereby inner nodes are labeled `x/y`, where `x` is the number of gene trees not contradicting the corresponding split and `y` is the number of gene trees where a nontrivial split supports the corresponding split. Thus $y$ is the number of input trees that *support* the node and $x - y$ is the number of input trees that are *irrelevant* for that node (with the definitions of support and irrelevance used in Wilkinson *et al.* (2005)).

**con** The `*.con` file contains the contracted consensus tree. Splits contradicted by $\geq 50\%$ of the input trees are deleted. Nodes are labeled as for the strict consensus.

**prog** The `*.prog` (progress) file contains information about the run. It reports whenever a tree with a better score is found. It also prints the starting and strict consensus tree as a list of splits. Thereby splits are coded binarily, where the taxon order is given on the top. After each split the following information is printed: whether the split is a terminal split, how many input trees contradict it and how many input trees are irrelevant.

**screen** During the run, several information is printed on the screen. E. g. in each iteration, the number of the iteration and the corresponding score is printed.

## 6 FAQs

1. Can I also do an exhaustive or branch-and-bound search under these criteria?
   No and yes. No, the program will not do an exhaustive search. But yes, you can input a list with all trees of a particular number of taxa and process these with the option `-p 2`.

## 7 Version history

**Version 1.1** Release date: 11 March 2011; one bug fixed concerning the start score; changes in output

**Version 1.0** Release date: 31 August 2010; covers PluMiST as described in Kupczok (2011), now MR(-) also allows for multifurcating input trees.

**Version 0.2** Release date: 26 July 2010; new method MR(+)g, introduction of TDR operation, test version for the official version.

**Version 0.1** Release date: 24 March, 2010; covers PluMiST as described in Kupczok (2010).

## References

Cotton, James A and Mark Wilkinson (2007) Majority-Rule Supertrees. *Syst. Biol.*, **56**(3):445–452.

Dong, Jianrong and David Fernández-Baca (2009) Properties of Majority-Rule Supertrees. *Syst. Biol.*, **58**(3):360–367.

Kupczok, Anne (2010) *Postprocessing Phylogenies: Tree Distances and Supertrees.* Ph.D. thesis, University of Vienna.

Kupczok, Anne (2011) Split-based Computation of Majority-Rule Supertrees. *submitted to BMC Evol. Biol.*

Wilkinson, Mark, Davide Pisani, James A Cotton, and Ian Corfe (2005) Measuring Support and Finding Unsupported Relationships in Supertrees. *Syst. Biol.*, **54**(5):823–831.