PhyloNet: Phylogenetic Networks Toolkit

User Documentation

The BioInformatics Group
Department of Computer Science
Rice University
http://bioinfo.cs.rice.edu

Development and maintenance of the PhyloNet software package is made possible through generous support from the Department of Energy (grant DE-FG02-06ER25734), the National Science Foundation (grant CCF-0622037), the George R. Brown School of Engineering (Roy E. Campbell Faculty Development Award), and the Department of Computer of Science at Rice University.

Copyright

Contents

Chapter 1

Introduction

PhyloNet is a collection of tools designed mainly for analyzing, reconstructing, and evaluating reticulate (or non-treelike) evolutionary relationships, generally known as *phylogenetic networks*. Various methods that we have developed make use of techniques and tools from the domain of phylogenetic trees, and hence the PhyloNet package includes several tools for phylogenetic tree analysis.

PhyloNet is released under the GNU General Public License. For the full license, see the file `GPL.txt` included with this distribution. Though the source code has not yet been posted, it is available by request.

1.1 Contributors

PhyloNet is designed, implemented, and maintained by the BioInformatics Group, which currently includes Professor Luay Nakhleh (`nakhleh@cs.rice.edu`) and Ph.D. students Derek Ruths (`druths@cs.rice.edu`) and Cuong Than (`cvthan@cs.rice.edu`). The group is affiliated with the Department of Computer Science at Rice University.

1.2 Contact Info

Feel free to contact us by mail, email, phone or fax:

The BioInformatics Group
c/o Luay Nakhleh
Department of Computer Science
Rice University
6100 Main Street
Houston, TX 77005
Email: `nakhleh@cs.rice.edu`
Phone: 713-348-3959
Fax: 713-348-5930

Chapter 2

Installation

System Requirements In order to run the PhyloNet toolkit, you must have Java 1.5.0 or later installed on your system. All references to the *java* command assume that Java 1.5 is being used.

Downloading `phylonet.jar` Acquire the current release of PhyloNet by downloading the most recent version of the PhyloNet JAR file. You will have a file named `phylonet_vX.Y.jar`, where X is the major version number and Y is the minor version number.

Installing the file Place this in the desired installation directory. The remainder of this document assumes that it is located in `$PHYLONET_PATH/jar`.

Installation is now complete. In order to run PhyloNet, you must execute the file `phylonet_vX.Y.jar`, as described in the next section.

2.1 Usage Instructions

All tools are run by executing the jar, `phylonet_vX.Y.jar`, and specifying the tool name and arguments.

Executing `phylonet_vX.Y.jar` The downloaded file, `phylonet_vX.Y.jar`, is an executable jar. On the command-line, type

```
java -jar $PHYLONET_PATH/jar/phylonet_vX.Y.jar -h
```

This will run the tool and print the help message. The help message will include a listing of the tools available for execution.

Running tool `<tool>` To run the specific tool, `<tool>`, type

```
java -jar $PHYLONET_PATH/jar/phylonet_vX.Y.jar <tool>
```

Since each tool requires different input data and parameters, it is helpful to review the help for a given tool before using it. To see the help information, type

```
java -jar $PHYLONET_PATH/jar/phylonet_vX.Y.jar <tool> -h
```

For more detailed help information see the Tool Reference section of this document.

Chapter 3

Software Conventions

3.1 Phylogenetic Tree Representation: The Newick Format

Many tools require trees as input or produce trees as output. All of which will be in Newick format [2]. In brief, Newick format uses nested parentheses to model trees. A single tree is always terminated by a semi-colon.

Name and Distance Properties Every node can have two properties: a name and a distance from its parent. The name of a node must be a string with using only alphanumeric characters. This distance can be any positive, real number. In Newick format, these properties are separated by a ":" character. Thus, the node "foo" that is 0.63 units away from its parent is written

```
foo:0.63
```

In practice, sometimes both the distance and bootstrap value need to be specified. Our tool allows this with the following convention: node name, followed by a colon and distance, and followed by a colon and bootstrap. For example, the edge connecting node "foo" with its parent has distance 0.63 and bootstrap 70%. Then, it can be written

```
foo:0.63:70
```

The distance and the bootstrap can be omitted, it can be written as: (1) `foo` (node "foo" does not specify the distance and bootstrap value for the edge connecting it with its parent); (2) `foo:0.63` (distance is specified, but not bootstrap); and (3) `foo::70` (bootstrap is specified, but not distance—note the two colons preceding the bootstrap value).

Specifying Children Internal nodes have other nodes as their children. This is specified by placing a parenthetical clause before the name and distance pair:

```
(...)foo:0.63
```

Within the parentheses, the node's children are written, separated by commas. The listing does not specify order:

```
(childA:0.1, childB:0.3)foo:0.63
```

This specifies that the node "foo" has three children, one of which is unnamed and one of which has no distance specified.

Roots By default, a tree is rooted. Therefore, the tree:

```
(A,(B,C)I1)ROOT;
```

is rooted at the node named "ROOT". To override this and specify an unrooted tree, the entire tree specification must be preceded by the string `[&U]`. Therefore the following tree is not rooted:

```
[&U] (A,(B,C)I1)D;
```

3.2 Phylogenetic Network Representation: The eNewick Format

The Newick format for representing and storing phylogenetic trees was adopted in 1986 [2], and it has been the standard for almost all phylogeny software packages ever since. This format captures an elegant correspondence between leaf-labeled trees and matched parentheses, where the the leaves are represented by their names and the internal nodes by a matched pair of parenthesis that contains a list of the Newick representation of all its children. Shown in Figure 3.2 are three trees along with their representations in the Newick format.

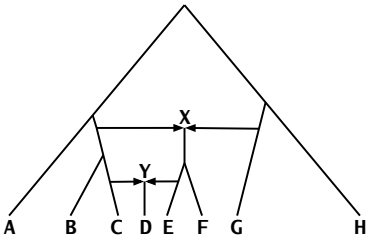
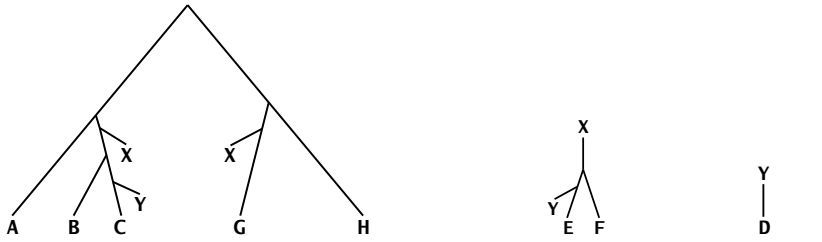


Figure 3.1: A phylogenetic network N with eight leaves (labeled $A, \ldots, H$) and two network nodes X and Y. Shown are the orientation of the network edges; all other edges are directed away from the root (toward the leaves).



$N' = ((A, (X, (B, (C, Y)))), ((X, G), H))$ $\quad X = (F, (E, Y))$ $\quad Y = (D)$

Figure 3.2: Three trees, N', X, and Y, along with their Newick representation. These three trees form the tree decomposition $\mathcal{F}$ of the phylogenetic networks N in Figure 3.1. The eNewick representation of N is the triplet $\langle N'; X; Y \rangle$.

However, no similar format exists for phylogenetic networks; instead, existing phylogenetic network software tools store these networks as adjacency lists of their underlying

graphs, which are usually very large and necessitate translation of representations among the different tools. We propose a new format, which we call the *extended Newick*, or eNewick, format. In this format, a phylogenetic network is decomposed into trees, each of which is represented using the Newick format. Figure 3.2 shows the tree decomposition and eNewick representation of the network N in Figure 3.1.

The eNewick representation of phylogenetic network is the format used in the phylogenetic network characterization and comparison utilities in PhyloNet (described below).

Phylogenetic Network Representation: The Edge-list Format

The RIATA-HGT method outputs phylogenetic networks in the *edge-list* format. In the case of horizontal gene transfer, a species tree is almost always assumed, and hence the phylogenetic network can be represented by a pair $\langle ST, X \rangle$, where ST is the species tree in Newick format (with names assigned to internal nodes), and X is the list of edges posited between the edges of ST.

Chapter 4

Tool Reference

Descriptions of the tools available in the PhyloNet package are provided here. Tools are listed in alphabetical order.

General Usage When running a tool, the tool name and arguments must be preceded by:

```
java -jar $PHYLONET_PATH/jar/phylonet_vX.Y.jar
```

4.1 `countcoal`

Description This `countcoal` tool computes the number of coalescent scenarios that can explain the incongruence observed between two trees. The tool reads two trees from standard in (one tree per line) and prints the number of coalescent scenarios to standard out. The trees must be specified in newick format.

Usage

```
countcoal [-h] [-f ⟨infile⟩] [-o ⟨outfile⟩]
```

The default behavior described above can be modified by using the following options:

`-h` prints the help message

`-f` ⟨`infile`⟩ reads the pair of trees from the file ⟨infile⟩. This file should contain the trees on the first two lines—one tree per line.

`-o` ⟨`outfile`⟩ writes the output to the file ⟨outfile⟩ rather than the standard out.

Citing If you use this tool in published work, please cite [23].

4.2 `lca`

Description This `lca` tool computes the least common ancestor of a group of nodes in a tree. The tool reads the tree, T, as input first. The tool prints this tree with all internal nodes labeled. If a node is not labeled in the original tree, then the node receives a generated name. Then the tool reads sets of nodes. A node set is a collection of names

belonging to nodes in T. Syntactically, a set of nodes is a set of space delimited names on the same line. For each node set read, `lca` prints their least common ancestor.

Usage

`lca [-h] [-o` *filename*`] [-t` *filename*`] [-n` *filename*`]`

By default, the tree and node set are read from standard in and the results are written to standard out. This behavior can be modified by using the following options:

`-h` prints the help message

`-o filename` writes all the output to the file *filename*

`-t filename` reads the tree from the file *filename*

`-n filename` reads the node set from the file *filename*

4.3 `mast`

Description The `mast` tool computes a Maximum Agreement Subtree of a pair of trees, using the algorithm of Steel and Warnow [18]. The trees must be either all rooted or all unrooted. If the trees are rooted, then the rooted MAST is computed. If the trees are all unrooted, then the unrooted MAST is computed. All trees must also have at least three leaves.

Usage

`mast [-h] [-a] [-i` *filename*`] [-o` *filename*`]`

By default, the trees are read from standard in and the MAST is written to standard out. This behavior can be modified by the following options:

`-h` prints the help message

`-o filename` writes all the output to the file *filename*

`-i filename` reads all the input trees from the file *filename*

To compute *all* MASTs, specify an option `-a` in the command line arguments. However, since the number of MASTs may be exponential in the number of leaves in the trees, using this option can slow down the execution time (in some cases considerably). Plans for modifying the code so that it computes MASTs of more than two trees are currently under way.

4.4 `rf`

Description This tool computes the symmetric difference, also known as the Robinson-Foulds (RF) distance [15], between two trees. The trees do not need to be rooted. The first tree read is the *model* tree. The second is the *experimental* tree.

Usage

`rf [-h] [-m` *filename*`] [-e` *filename*`] [-o` *filename*`]`

The tool reads in two trees. It outputs the number of False Negative edges, the number of False Positive edges, the number of internal edges in the model tree and finally the number of internal edges in the experimental tree. By default, the tool reads the trees from the standard in and print the results to standard out. This behavior can be modified by the following options:

`-h` prints the help message

`-m filename` indicates that the model tree be read from file *filename*

`-e filename` indicates that the experimental tree be read from the file*filename*

`-o filename` writes all the output to the file *filename*

4.5 recomp

Description This tool is an implementation of the RECOMP algorithm for detecting recombination breakpoints along a set of aligned genetic sequences [16, 17]. For a detailed description of the algorithm, see [17]. In brief, the algorithm reads in a set of aligned genetic sequences, a window size, a step size, and a window comparison function. The function is designed in such a way that highly different adjacent windows correlate to windows separated by a recombination breakpoint. The method returns the value of the comparison function evaluated at positions along the length of the sequences.

Usage

`recomp [-h] -w` *WIN_SIZE* `-s` *STEP_SIZE* `[-d RF|SPR] [-i` *IFILE* `[`*FMT*`]] [-o` *OFILE*`] -c` *FXN PAUP_PATH POP_SIZE NUM_ITERS NUM_LVLS*

`-h` prints the help message

`-w WIN_SIZE` specifies the window size

`-s STEP_SIZE` specifies the step size

`-d RF|SPR` specifies what distance measure should be used. The Robinson-Foulds (*RF*) distance [15] is the default. The Subtree-Prune-Reroot (SPR) distance is also supported, and in this case the RIATA-HGT heuristic [11] is used to compute the distance.

`-i` *FILE FMT* specifies the file that should be used as input. If omitted, then input is read from the STDIN. By default, input is read as a fasta file. FMT can be used to override the default. *FASTA* indicates that the input is in FASTA format. *PLAIN* indicates that the input is in a plain format in which each line of the file has a sequence name and the sequence itself. The plain format also supports an optional header declaring the number of sequences and the number of nucleotides in the sequences.

`-o OFILE` specifies the file that will be used for output. If omitted, then STDOUT is used.

`-c` *FXN PAUP_PATH POP_SIZE NUM_ITERS NUM_LVLS* specifies the window comparison function that will be used.

- *PAUP_PATH* is the path to a PAUP executable
- *POP_SIZE* is a parameter used by PAUP. It specifies the number of trees that PAUP will maintain in memory during maximum parsimony search. The recommended value for this is 100.
- *NUM_ITERS* is a parameter used by PAUP, specifying the number of iterations to use during the maximum parsimony search. Values between 25 and 100 tend to yield accurate trees.
- *NUM_LVLS* specifies the number of tree levels that should be included in the window comparison. In [17], this is the k parameter. Taking more levels increases the number of non-optimal trees included in the window analysis. In [17], 3 levels were found to give the best result.

For an explanation of the different functions, please see [17]. Valid choices of *FXN* are:

- *PARS* - the parsimony difference function
- *MAX* - the average maximum distance
- *MIN* - the average minimum distance
- *INT* - the percent intersection of the window tree sets

Citing If you use this tool in published work, please cite [17].

4.6 `riatahgt`

Description This tool is an implementation of the RIATA-HGT algorithm for detecting and reconstructing horizontal gene transfer events from phylogenetic incongruence. For a detailed description of the algorithm, see [11]. In brief, the algorithm reads in a single species tree and any number of gene trees. It returns the horizontal gene transfer events.

The tool performs some algorithmic techniques to speed up the detection of HGT. One technique is to partition the solutions into independent subsets, and the tool then finds equivalent solutions to for every subset. In this way, the tool decreases the size, and thus decreases the running time, of the search space. The second technique is to handle (non-binary) caterpillars by replacing them with a chain of three leaves. In general, species and gene trees can be non-binary. There are various reasons for this, such as tree reconstruction errors, insufficient information to build trees. The tool maximally refines trees so that it does not introduce new HGT before detecting HGT. A detailed description of those techniques are given in the paper [20].

The tool also computes a support value for HGT it detected, based on the bootstrap values of branches in the gene tree (and the species tree, if those values are present). A key advantage of the method is that it does not require resampling gene sequences, which greatly reduces running time. The method is described in detail in the paper [19].

Usage

`riatahgt [-h] [-u] [-p` *prefix*`] [-i` *filename*`] [-o` *filename*`] [-b] [-e]`

The tool reads a set of trees, either from the standard input or from the input file specified with the `-i` option. For the tool to run, there has to be at least two trees in the input. If it reads k trees ($k \geq 2$), it assumes that the first tree is the species tree and the remaining trees are the gene trees. The tool then proceeds by computing solutions on pairs of species/gene trees. In other words, if the input contains k trees $T_1, \ldots, T_k$, the tool computes solutions for the pairs (T_1, T_2), (T_1, T_3), $\ldots$, (T_1, T_k). Further, the tool prints the solutions for these pairs in this particular order.

For each pair, the tool first prints the species/gene trees, in this order, with the internal nodes labeled (this labeling is needed for printing the HGT edges). The user can specify a particular prefix to name the internal nodes with the `-p` option. For example, if the user species `-p Int`, and assuming there are ℓ internal nodes, then the tool will label the internal nodes of the species tree as $\texttt{Int}_1$, $\texttt{Int}_2$, $\ldots$, $\texttt{Int}_\ell$. The internal node names in the HGT events refer to the names in the species, and not the gene, tree. Because solutions might share common HGT events, we group and print them by components to make the tool's ouput more concise and informative. From the tool's output, one can get a complete solution by selecting a subsolution from each component. For example, let's consider the following species and gene trees:

```
ST = ((e,(f,g)I1)I2,((a,(b,c)I3)I4,d)I0)I5;
GT = (((a:70.0,b:75.0):90.0,c:80.0):60.0,(((e:80.0,f:75.0):95.0,
g:60.0):80.0,d:70.0));
```

HGT events for this pair (`ST`, `GT`) are computed and printed by the tool as:

```
There are 3 component(s), which account(s) for 27 solution(s), each of size 3
-----------------------------------------------------------------------
Component I5:
Subsolution1:
        I2 -> d (70.0)
Subsolution2:
        d -> I2 (80.0)
Subsolution3:
        I5 -> I4 (70.0) [time violation?]
-----------------------------------------------------------------------
        Component I2:
        Subsolution1:
                f -> e (95.0)
        Subsolution2:
                I2 -> g (95.0) [time violation?]
        Subsolution3:
                e -> f (95.0)
-----------------------------------------------------------------------
        Component I4:
        Subsolution1:
                b -> a (90.0)
```

```
        Subsolution2:
                I4 -> c (90.0) [time violation?]
        Subsolution3:
                a -> b (90.0)
**************************************************************************
```

There are 27 possible solutions for this pair of trees, each of which consists of events from sub-solutions of each component. The set {`d -> I2, f -> e, a -> b`} is a solution, for example. Note that an HGT event has the format:

`sn` $\rightarrow$ `tn`

where `sn` and `tn` are the names of nodes in the species tree, and are the source and target, respectively, of an HGT edge. This indicates that an edge should be added from the edge incident into `sn` to the edge incident into `tn` in the species tree.

If the branches in the gene tree has bootstrap values, the tool infers a support value for HGT events. The support value for each HGT event is in parentheses next it. For example, event `f -> e` in component `I2` has support 95%. In the case the species tree branches also have bootstrap values, they are also taken into account when the tool computes the support value for HGT.

The above output format is compact, and so it is very useful for presentation. However, the tool also allows you to display the full solutions, by using the option `-e`. In this case, the tool will present solutions in the form of a network in eNewick format. Please refer to the README file included in this package for a sample output of the tool when the tool is invoked with this option.

By default the all trees are read from standard in and the HGT events are written to standard out. This behavior can be changed by using the following options:

`-h` prints the help message

`-i filename` indicates that the species tree and gene tree(s) are read from file *filename*

`-o filename` writes all the output to the file *filename*

`-p prefix` specifies a name prefix used to label internal nodes. If no prefix is specified, then the default prefix `I` is used.

`-b` is for batch processing. By default, the tool displays a window for visualizing HGT events. In case you want to do batch processing, specifying this option prevents that window from appearing.

Input trees are always refined and contracted before RIATA-HGT computes HGT events. In general, this detects fewer HGT events than when the trees are left intact. In some cases, however, computing HGT events with the refined and contracted trees might produce more events than with the original trees (such cases are rare, though). To prevent the trees from being refined and contracted, use the option `-u`; RIATA-HGT will compute events for the original trees provided.

Citing If you use this tool in published work, please cite [11].

4.7 gencplex

Description This tool generates CPLEX input for a species tree and a set of gene trees. The method for generating CPLEX input is based on the paper [22].

Usage

```
gencplex [-h] stfile gtfile w1 w2
```

The tool accepts two string arguments, which are file names containing the species trees (`stfile`) and gene trees (`gtfile`), and two numbers `w1` and `w2`. These two values specify the weight for the number of deep coalescences and the number of no coalescences. You can have multiple species tree in the file `stfile`, in which case the tool will generate each CPLEX input for each species tree. The tool will generate three files for each pair of a species tree and all gene trees in the file `gtfile`: `input#`, `var#`, and `script#` (`#` indicates which species tree in `stfile` these files are for). The `input#` is a mixed linear integer programming (MILP) formulation that CPLEX will read, and solve. However, because the number of variables and constraints in this file might be very big, and so manually typing commands to query CPLEX the optimization solution is not practical, the tool supplies with a `script#` file, which you can use to ask CPLEX to load the problem, solve it. From the UNIX shell, you type: `cat script# | cplex > output#`, and all of CPLEX output is directed to the file `output#`. The purpose of the file `var#` is to provide a mapping between the species (and gene) trees' nodes to variables in the MILP formulation.

The option `-h` is to allow you to see the usage message for this tool.

4.8 genst

Description This tool generates species tree topologies based on maximal sets of compatible clusters. The algorithm is described in the paper [22].

Usage

```
genst [-h] gtfile stfile
```

The tool has two arguments: `gtfile` contains gene trees, and `stfile` is the file you want to store species tree topologies.

4.9 compute_st

Description We also include in this package a Perl script to compute the best species tree based on the algorithm described in the paper [22]. This script will generate species tree topologies (by using the tool `genst`), run CPLEX to solve an MILP for each species tree topology, and compare the eta value of each tree to get the best one.

Usage This tool first requires that `CPLEX` is successfully installed on your machine. You might also need to look at the script to change the location of this program so that the script can find it. At the beginning of the script, there are 4 string variables you can change. It also has a short description on how to change them.

The script will write the results into a file called `results`. Each line in this file corresponds to a species tree, and it has several fields: `species tree; time of the root; numer of missing clades; number of deep coalescences; number of no coalescences`. The last line of the file indicates which tree is the best one.

4.10 `infer_st`

Description This tool allows users to infer species tree. Several methods are contained, including MDC, GLASS, extended Majority Consensus and Democratic Vote.

Usage The tool accepts several arguments depending on different methods. The method to be used need to be specify first as follows:

```
infer_st [-m MDC|MDC_ILP|MDC_TIME|MDC_UR|GLASS|MC|DV] ...
```

Details of these methods are described below.

4.10.1 `infer_st -m MDC`

Description This tool allows users to infer the species tree using the "Minimize Deep Coalescence" (MDC) criterion. The method for inferring the species tree is described in details in [21].

Usage

```
infer_st -m MDC -i input [-e proportion] [-x] [-b threshold] [-a mapping]
[-ur] [-t time] [-a mapping] [-o output]
```

The users must include the option `-i`, followed by the name of the file that contains the gene trees. Gene trees must be rooted. Gene losses are allowed. The option `-o` allows the users to specify the file to save the result. Users can also use `-b` to set bootstrap threshold, if the input gene trees have bootstrap values.

By default, the method returns the optimal tree. But the option `-e` allows the users to get the optimal tree and a set of sub-optimal trees. If the optimal tree has n extra lineages, all the sub-optimal trees that have extra lineages less than *(1+proportion/100)*n* will be returned with the optimal tree.

By default, the method uses clusters induced from gene trees to infer species tree. However, the option `-x` allows users to specify using all possible clusters to infer species tree.

By default, the method will always return a binary species tree. But users can use option `-ur` to allow non-binary species tree. If the gene trees are not binary and the degree of resolution are low, it is recommended to use this option. Otherwise, the program will do some exhaustive search for a binary species tree. In this case, users can also use option `-t` to limit the search time. The time is in the unit of minutes.

By default, it is assumed that only one individual is sampled per species in gene trees. However, the option `-a` allows multiple alleles to be sampled, which is followed by the name of the file that contains the association between gene tree taxa and species tree taxa. For example, if taxa *a1* and *a2* in gene trees map to taxon *a* in species tree and taxa *b1* and *b2* in gene trees map to taxon *b* in species tree, the following associations should be in the mapping file:

```
a:a1,a2;b:b1,b2;
```

The resulting species trees are displayed with the number of extra lineages in each branch. For example, consider the following inferred species tree:

```
((a:0,b:0):2,(c:0,d:0):1):0
```

In this species tree, there are two extra lineages in branch between node *(a, b)* and the root, and one extra lineage in branch between node *(c, d)* and the root. All other branches have 0 extra lineages.

4.10.2 `infer_st -m MDC_ILP`

Description This tool allows users to infer the species tree using the "Minimize Deep Coalescence" (MDC) criterion by integer linear programming (ILP). The method for inferring the species tree is described in details in [21].

Usage This tool requires CPLEX is installed on the user's computer before it can be run. The tool accepts three arguments as follows:

```
infer_st -m MDC_ILP -i input -p cplex [-o output]
```

The users must include the option `-i` followed by the file that contains gene trees, and option `-p` followed by the path to the executable CPLEX program. The third option *-o* allows users to specify the file to store the result.

4.10.3 `infer_st -m MDC_TIME`

Description This tool allows users to infer the species tree using MDC criterion but considering time.

Usage

```
infer_st -m MDC_TIME -i input [-a mapping] [-b bootstrap] [-o output]
```

The users must include the option `-i`, followed by the name of the file that contains the gene trees. Gene trees must be rooted and have branch length. Gene trees are allowed to have gene losses. The option `-o` allows the users to specify the file to save the result.

By default, it is assumed that only one individual is sampled per species in gene trees. However, the option `-a` allows multiple alleles to be sampled, which is followed by the name of the file that contains the association between gene tree taxa and species tree taxa. The format of the mapping file is the same as what is mentioned above.

Users can also use `-b` to set bootstrap threshold, if the input gene trees have bootstrap values.

The resulting species trees are displayed with the number of extra lineages in each branch, which has been mentioned above.

4.10.4 `infer_st -m MDC_UR`

Description This tool allows users to infer the species tree from unrooted gene trees using MDC criterion [25]. .

Usage

```
infer_st -m MDC_UR -i input [-e proportion] [-x] [-a mapping] [-b threshold]
[-ur] [-t time] [-o output]
```

The users must include the option `-i`, followed by the name of the file that contains the gene trees. Gene trees must be unrooted. The option `-o` allows the users to specify the file to save the result. Users can also use `-b` to set bootstrap threshold, if the input gene trees have bootstrap values.

By default, the method returns the optimal tree. But the option `-e` allows the users to get the optimal tree and a set of sub-optimal trees. If the optimal tree has n extra lineages, all the sub-optimal trees that have extra lineages less than *(1+proportion/100)*n* will be returned with the optimal tree.

By default, the method uses clusters induced from gene trees to infer species tree. However, the option `-x` allows users to specify using all possible clusters to infer species tree.

By default, the method will always return a binary species tree. But users can use option `-ur` to allow non-binary species tree. If the gene trees are not binary and the degree of resolution are low, it is recommended to use this option. Otherwise, the program will do some exhaustive search for a binary species tree. In this case, users can also use option `-t` to limit the search time. The time is in the unit of minutes.

By default, it is assumed that only one individual is sampled per species in gene trees. However, the option `-a` allows multiple alleles to be sampled, which is followed by the name of the file that contains the association between gene tree taxa and species tree taxa. The format of the mapping file is the same as what is mentioned above.

4.10.5 `infer_st -m GLASS`

Description This tool allows users to infer the species tree by GLASS. The details of this method are described in [9].

Usage

```
infer_st -m GLASS -i input [-t tree|matrix] [-a mapping] [-o output]
```

The users must include the option `-i`, followed by the name of the file that contains input which is either gene trees or distance matrix of species taxa. The option `-o` allows the users to specify the file to save the result.

By default, the method takes gene trees as input. Gene trees must have branch length. The users can also specify the input to be distance matrix of species taxa. The input file contains distance matrix must be in certain format. It should contain a line containing the number of taxa in species tree, followed by a list of taxa, then the distance matrix. For example:

```
4
a b c d
1 2 3
4 5
6
```

In this example, there are four taxa a, b, c and d in species tree. And the distance between a and b is 1, a and c is 2, a and d is 3, b and c is 4, b and d is 5, c and d is 6.

If the input is gene trees, by default, it is assumed that only one individual is sampled per species in gene trees. However, the option `-a` allows multiple alleles to be sampled, which is followed by the name of the file that contains the association between gene tree taxa and species tree taxa. The format of the mapping file is the same as what is mentioned above.

4.10.6 `infer_st -m MC`

Description This tool allows users to infer the species tree from gene trees using an extended version of Majority Consensus which can deal with gene trees having multiple alleles in species and unrooted gene trees.

Usage

```
infer_st -m MC -i input [-u] [-a mapping] [-o output]
```

The users must include the option `-i`, followed by the name of the file that contains the gene trees. The option `-o` allows the users to specify the file to save the result.

By default, it is assumed that gene trees are rooted. However, the option `-u` can specify gene trees to be treated as unrooted.

By default, it is assumed that only one individual is sampled per species in gene trees. However, the option `-a` allows multiple alleles to be sampled, which is followed by the name of the file that contains the association between gene tree taxa and species tree taxa. The format of the mapping file is the same as what is mentioned above.

4.10.7 `infer_st -m DV`

Description This tool allows users to infer the species tree from gene trees using Democratic Vote.

Usage

```
infer_st -m DV -i input [-o output]
```

The users must include the option `-i`, followed by the name of the file that contains the gene trees. The option `-o` allows the users to specify the file to save the result. All the gene trees which have the highest frequency are returned.

4.11 `infer_st_bootstrap`

Description This tool allows users to infer the species tree using bootstrap with existing methods.

Usage

```
infer_st_bootstrap -n num [-s threshold] -m MDC|MDC_ILP|MDC_TIME
|MDC_UR|GLASS|MC|DV ...
```

The users must specify the number of repetitions through option `-n`. The option `-s` allows the users to specify the threshold value. The default value for it is 0.5.

The users must specify the method for inferring species tree through option `-m`, including all versions of MDC, GLASS, majority consensus and democratic vote. After specifying the method, other options for a particular method all apply.

4.12 deep_coal_count

Description This tool allows users to count the number of extra lineages contributed by a species tree and a set of gene trees.

Usage

```
deep_coal_count species-tree-file gene-trees-file [-u] [-b threshold]
[-a mapping]
```

The first argument is the file that contains species trees. The second argument is the file that contains gene trees. These two argument are required.

By default, it is assumed that trees are rooted. However, the option `-u` can specify gene trees to be treated as unrooted.

The users can also use `-b` to set bootstrap threshold, if the input gene trees have bootstrap values.

By default, it is assumed that only one individual is sampled per species in gene trees. However, the option `-a` allows multiple alleles to be sampled, which is followed by the name of the file that contains the association between gene tree taxa and species tree taxa. The format of the mapping file is the same as what is mentioned above.

4.13 process_gt

Description This tool allows users to refine and root gene trees with respect to a rooted binary species tree under MDC criterion [25].

Usage

```
process_gt species-trees-file gene-trees-file [-u] [-b threshold]
[-a mapping]
```

The first argument is the file that contains species trees. The second argument is the file that contains gene trees. These two argument are required.

By default, it is assumed that gene trees are rooted. In this case, the method refines the gene trees with respect to the species tree. If the gene trees are unrooted, users can use option `-u`. In this case, the method not only refines and but also roots gene trees.

Users can also use `-b` to set bootstrap threshold, if the input gene trees have bootstrap values.

By default, it is assumed that only one individual is sampled per species in gene trees. However, the option `-a` allows multiple alleles to be sampled, which is followed by the name of the file that contains the association between gene tree taxa and species tree taxa. The format of the mapping file is the same as what is mentioned above.

4.14 `charnet`

Description This tool implements functions to compute the trees, tripartitions and clusters contained in a phylogenetic network [8, 14, 13]. Please note that the clusters and tripartitions returned by this tool will not contain trivial ones, that is, clusters and tripartitions for the network's root and leaves.

Usage

```
charnet [-h] [-i input] [-o output] -m tree|tri|cluster
```

By default, the tool reads the input network from the standard input, and outputs the result to the screen. The options `-i` and `-o` allow the user to specify the files that store the input network and the result.

The user must include the option `-m`, followed by either `tree`, `tri` or `cluster`, to specify which characterization he or she wants. For example, if the user types

`charnet -i net -m cluster`,

then the tool will read the network from the file `net`, and compute all the clusters contained in this network. The result is printed on the standard output.

The option `-h` prints the help message on the usage of the tool. The network must be in the eNewick format.

4.15 `cmpnets`

Description This tool allows the user to compute the distance between two phylogenetic networks, based on their topologies. Three measures are currently implemented:

- Tree-based measure.
- Tripartition-based measure.
- Cluster-based measure.

See [12, 8, 14, 13] for complete description of these measures.

Usage

```
cmpnets [-h] [-i input1 input2] [-o output] -m tree|tri|cluster
```

By default, the tool reads the two networks from the standard input and prints the result to the screen. By default, it is assumed that the network is described using the eNewick format. However, the user has the option of describing the network as either a set of clusters or a set of trees. If the network is described as a set of clusters, then the first line of the network description contains a single word—"`cluster`", followed by the clusters,

each listed on a separate line. The elements of a cluster are separated by white space. Similarly, if the network is described as a set of trees, then the first line of the network description contains a single word—"`tree`", followed by trees, each listed on a separate line using the Newick format.

For the two input networks, the only combinations of formats allowed are:

1. Both networks are in eNewick format. In this case, any of the three measures can be invoked.
2. Both networks are described in terms of their constituent trees. In this case, only the tree-based measure can be invoked.
3. Both networks are described in terms of their constituent clusters. In this case, only the cluster-based measure can be invoked.
4. One network is in eNewick format, and the other is described in terms of its constituent trees. In this case, only the tree-based measure can be invoked.
5. One network is in eNewick format, and the other is described in terms of its constituent clusters. In this case, only the cluster-based measure can be invoked.

When at least one of the two networks is not in eNewick format, and since in this case the type of measure used is automatically determined, entering the measure type is optional (actually, it will be ignored by the tool).

If the options `-i` and `-o` are used, the networks will be read from the two specified input files and the result is stored in the output file. In case the two networks are read from the standard input, they must be separated by a blank line. After the inputs are successfully parsed and the networks are constructed, the tool computes their distance. For example, if the user types

`cmpnets -o result -m tree`,

then the tool will wait for the user to enter two networks as input, compute the tree-based distance and then write the result to the file `result`. If the user already has clusters for a network, he/she can type

`cmpnets -i net1 net2`,

where `net1` contains a list of clusters for a network and a line ``` ``cluster'' ``` at the beginning of the file, and `net2` contains eNewick representation for another network. The tool will compute the cluster-based distance between the two networks, and prints the result to the screen. As in the tool `charnet`, input clusters for a network will not contain trivial clusters, i.e., clusters that have either one taxon or all the taxa in the network, since these clusters are present in all networks.

The option `-h` prints the help message on the usage of the tool.

4.16 `sim_gtinnetwork`

Description This tool simulates gene trees under the phylogenetic network model in [24].

`Usage`

`sim_GTinNetwork [-h]` t_1 t_2 γ n `[-o output]`

The parameters t_1, t_2, γ and n are required. The definitions of t_1, t_2, γ can be found in [24]. They are used to characterize the network. Users can specify the number of gene trees to be simulated by n. The option `-o` allows the users to specify the file to save the result. By default, the tool will write the simulated gene trees to the standard output.

The option `-h` prints the help message on the usage of the tool.

4.17 netpars

Description This tools computes the parsimony score of a phylogenetic network, given the sequences at its leaves, using the formulation of [10] (reviewed briefly below). The sequences are split into blocks of equal length (with the exception of the last block, which many not necessarily contain the same number of sites as the ones before) which is specified by the user. The parsimony score is computed based on these blocks as well as the trees contained inside the network.

`Usage` To compute the parsimony, type:

```
netpars [-h] [-i netfile1 netfile2 seqfile] -b block-size [-o output]
```

By default, the tool will read the networks and sequences from the standard input. In this case, the input for the networks and the sequences must be separated by a blank line. The tool can also accept input from files, if the option `-i` is enabled. In this case, you specify three file names, two of which are for the networks and the other for the sequences, as arguments for the option `-i`. The score will be printed on the standard ouput, but it can be directed to a file if the option `-o` is enabled.

The tool can read the networks either in the eNewick format, or in the form of a tree followed by a set of HGT events. Refer to the subsection 3.2 for the specification of the eNewick format. The sample files also have sample input networks in both formats. For the file that contains the sequences, the tool expects it in the PHYLIP format:

```
#taxa sequence-length [I (optional)]
taxon-name1 sequence1
taxon-name2 sequence2
...
```

The first line in the file specifies the number of taxa and the length of every sequence. An option `I` can be specified to indicate that the sequences are interleaved. The number of taxa should match the number of leaves in the network. Each subsequent line contains a taxon name followed by the DNA sequence of this taxon. Note that all sequences must have the same length; otherwise, the tool will throw an I/O exception.

The tool currently does not handle weighted parsimony, a feature that will be added in the near future.

4.17.1 Maximum parsimony of phylogenetic networks

This relationship between a phylogenetic network and its constituent trees is the basis for the MP extension to phylogenetic networks described. We now briefly review the definitions of [10].

Definition 1 *The Hamming distance between two equal-length sequences x and y, denoted by $H(x, y)$, is the number of positions j such that $x_j \neq y_j$.*

Given a fully-labeled tree T, i.e., a tree in which each node v is labeled by a sequence s_v over some alphabet Σ, we define the Hamming distance of an edge $e \in E(T)$, denoted by $H(e)$, to be $H(s_u, s_v)$, where u and v are the two endpoints of e. We now define the parsimony score of a tree T.

Definition 2 *The parsimony score of a fully-labeled tree T, is $\sum_{e \in E(T)} H(e)$. Given a set S of sequences, a maximum parsimony tree for S is a tree leaf-labeled by S and assigned labels for the internal nodes, of minimum parsimony score.*

Given a set S of sequences, the MP problem is to find a maximum parsimony phylogenetic tree T for the set S. Unfortunately, this problem is NP-hard, even when the sequences are binary [1, 4]. One approach that is used in practice is to look at as many leaf-labeled trees as possible, and choose one with a minimum parsimony score. The problem of computing the parsimony score of a fixed leaf-labeled tree is solvable in polynomial time [3, 5].

In the context of phylogenetic networks, the evolutionary history of a single (non-recombining) gene is modeled by one of the trees contained inside the phylogenetic network of the species containing that gene. Therefore the evolutionary history of a site s is also modeled by a tree contained inside the phylogenetic network. A natural way to extend the tree-based parsimony score to fit a dataset that evolved on a network is to define the parsimony score for each site as the minimum parsimony score of that site over all trees contained inside the network.

Definition 3 *([6, 7, 10]) The parsimony score of a network N leaf-labeled by a set S of taxa, is*

$$NCost(N, S) := \sum_{s_i \in S} (\min_{T \in \mathcal{T}(N)} TCost(T, s_i))$$

where $TCost(T, s_i)$ is the parsimony score of site s_i on tree T.

Notice that as usually large segments of DNA, rather than single sites, evolve together, Definition 3 can be extended easily to reflect this fact, by partitioning the sequences S into non-overlapping blocks b_i of sites, rather than sites s_i, and replacing s_i by b_i in Definition 3. This extension may be very significant if, for example, the evolutionary history of a gene includes some recombination events, and hence that evolutionary history is not a single tree. In this case, the recombination breakpoint can be detected by experimenting with different block sizes.

Chapter 5

The Graphical Interface

Since version 2.0, Phylonet includes a GUI, or graphical user interface, as an alternative to the command-line-based interface described above. Most of the features available in the command-line version of PhyloNet are also available in the graphical version. This chapter describes the basic features of PhyloNet's GUI.

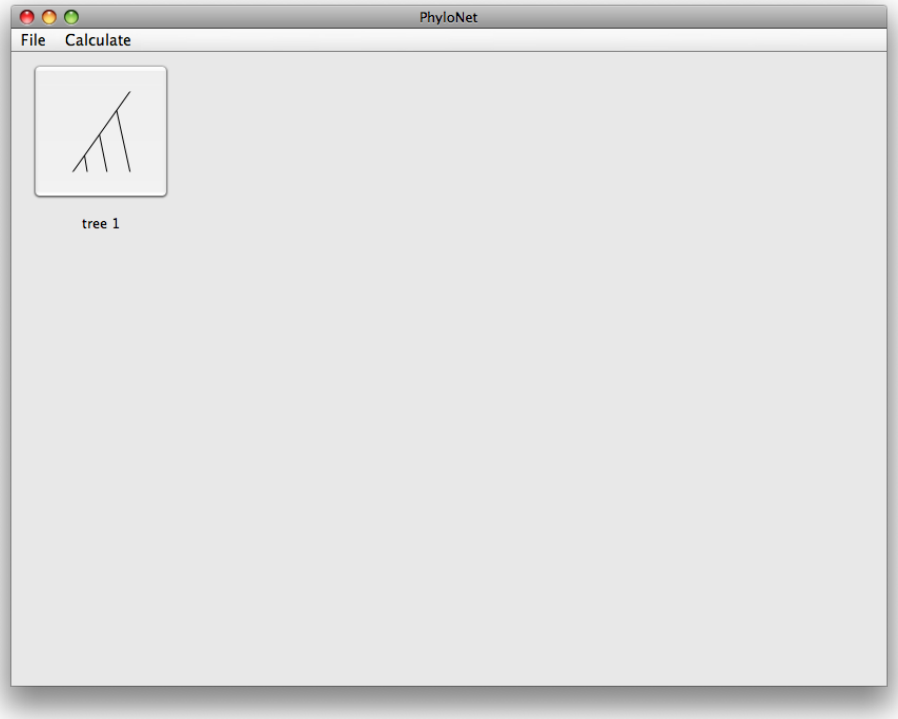


Figure 5.1: The main window, showing a workspace with a single imported datum. The stylized tree icon indicates that the datum is a tree, and the text below the icon indicates that the user has entitled it `tree 1`.

5.1 Accessing the GUI

If your computer is configured to recognize executable JARs, then the GUI is accessible by double-clicking the JAR file included in the distribution.

Alternatively, you may enter

```
java -jar phylonet_vX_Y.jar
```

at the command prompt.

5.2 Overview

PhyloNet's graphical interface consists of a *workspace*, a semi-permanent working area into which the user can import a number of *data* to examine.

5.3 Importing Data

Before you can perform any calculations, you must import one or more pieces of data into the workspace.

To do so, open the **File** menu, then select **Import Data** and choose the type of datum you wish to import. Then browse to and select the file containing the datum you wish to import. PhyloNet will attempt to parse the specified file. If it is successful, and if the file contains only a single datum, you will be prompted to enter a unique name for the datum. This name will be the identifier by which you refer to the datum in each of PhyloNet's tools. Press **OK** to assign the name and finish the import process, or press **Cancel** to cancel the import. Note that once a datum has been imported into PhyloNet, it is independent of the file from which it originated—all changes to the datum are contained within PhyloNet and are not saved permanently until you export the datum (see Section 5.6).

PhyloNet currently supports importing trees (with or without bootstrap values) in the Newick format, networks in the eNewick format, and sequence alignments.

5.3.1 Batches of Data

PhyloNet provides support for importing multiple data from a single file—a so-called "batch" of data.

If, while importing a datum, you select a file containing multiple data, a dialog like Fig. 5.2 will appear, offering you the choice to import the data individually or as a batch. If you choose to import them individually, you must supply a unique name for each datum in the file; otherwise, you need only supply a name for the batch.

Importing data as a batch allows you to more conveniently perform certain calculations that require many input data. A batch is less flexible than a multitude of individual data, however, as you may only refer to the batch as a whole—for example, you cannot perform calculations on individual data within a batch.

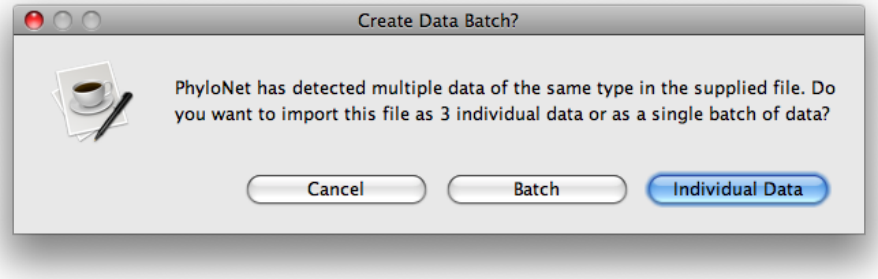


Figure 5.2: The data batch dialog. Selecting **Individual Data** will import the data file as many individual data, requiring the user to enter a name for each one. Selecting **Batch** will import the data as a single "batch" of data, requiring only one title. Pressing **Cancel** terminates the import process.

5.4 Viewing Data

Each imported datum appears in the user's workspace as an icon and its associated name. To view a datum, simply click on its icon. This brings up a viewing window for that datum.

From the **File** menu of the viewing window, the user may rename the datum, delete the datum from the workspace, or export the datum to a text file. (See Section 5.6 for more information on exporting data.) Depending on the datum's type, other options may be available—for example, when viewing a Tree datum, the user can display the number of nodes in the tree with the **Count Nodes** option in the **Tree** menu.

5.5 Performing Calculations

PhyloNet's GUI provides access to all of the tools available through the command-line interface. We have put effort into making each tool's input and output dialogs as understandable as possible—if there is ever confusion regarding the interface, please contact us.

All of PhyloNet's tools are accessible via the **Calculate** menu of the main workspace view. In general, each tool in the GUI corresponds to the tool of the same name in the command-line interface. To see a brief description of a given tool, let the mouse cursor hover over that tool's name for a moment. (For more detailed descriptions of each tool, see Section 4.)

Each tool requires some form of input from the user's workspace, and only those tools for which valid inputs currently reside within the workspace are enabled—all others are "grayed out". For example, the `MAST` tool requires that at least two Tree data be imported into the workspace before it will become enabled.

Once the user selects a tool from the **Calculate** menu, that tool's Parameters dialog will appear (see Fig. 5.4). From this dialog the user must specify all the parameters (inputs) to the tool. Since these inputs are generally the same as those provided to the command-line interface, this section will not attempt to re-describe the inputs for each tool.

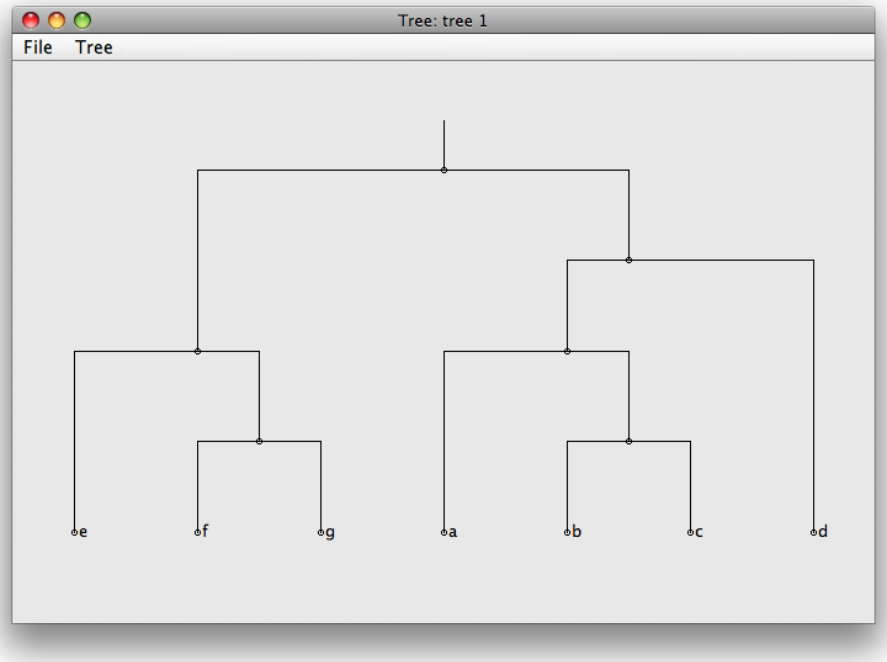


Figure 5.3: The viewing window for a Tree datum named `tree 1`.

After entering all the necessary parameters and pressing **OK**, PhyloNet will run the specified calculation. If no errors are encountered, a Results dialog will appear, summarizing the calculation's results and giving you the option to **Export** the results. Doing so generates a text file containing the various inputs and outputs of the calculation, as well as the time it was completed.

5.6 Saving Data

Normally, a workspace persists only as long as PhyloNet is running; if you restart PhyloNet, you must re-import all your data. However, any workspace (and the data it contains) can be saved for later use with the **Save Workspace...** option in the **File** menu of the main window. Note that the resulting file is not human-readable or even readable by other programs—it is a PhyloNet-specific workspace file.

To load a previously-saved workspace file, select the **Load Workspace...** option from the main window's **File** menu, and browse to the workspace file on your system. PhyloNet will parse the file and load the workspace it represents, including all data and their names.

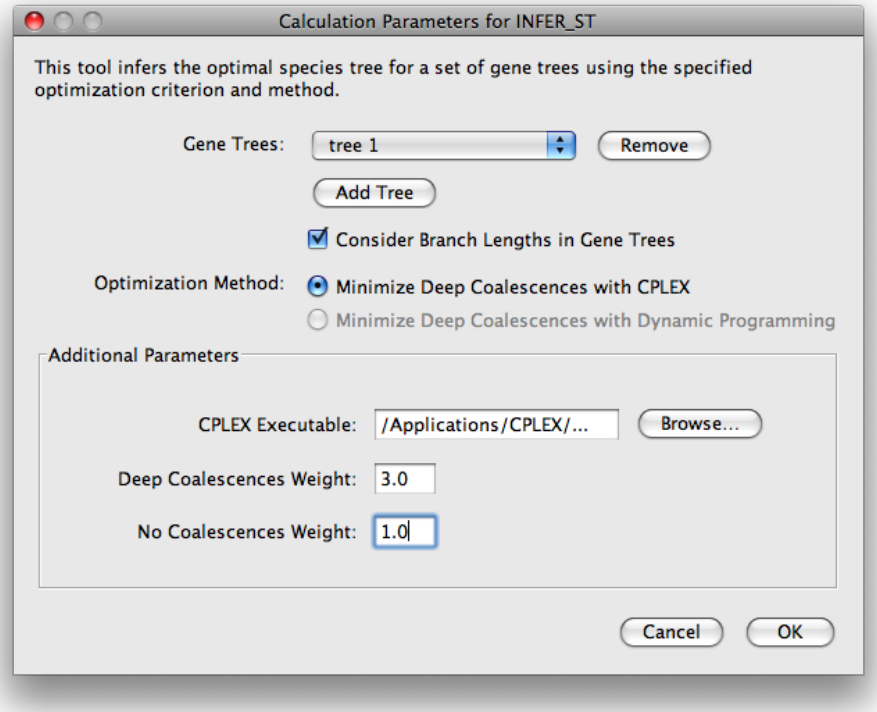


Figure 5.4: The Calculation Parameters dialog for `infer_st`. Here the user has the option to specify the inference method and, if applicable, any additional parameters.

Bibliography

[1] W.H.E. Day. Computationally difficult parsimony problems in phylogenetic systematics. *Journal of Theoretical Biology*, 103:429–438, 1983.

[2] J. Felsenstein. The newick tree format, 1986. http://evolution.genetics.washington.edu/phylip/newicktree.html.

[3] W. Fitch. Toward defining the course of evolution: minimum change for a specified tree topology. *Syst. Zool*, 20:406–416, 1971.

[4] L.R. Foulds and R.L. Graham. The steiner problem in phylogeny is NP-Complete. *Adv. Appl. Math.*, 3:43–49, 1982.

[5] J.A. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29:53–65, 1973.

[6] J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98:185–200, 1990.

[7] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36:396–405, 1993.

[8] B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):13–23, 2004.

[9] Elchanan Mossel and Sebastien Roch. Incomplete lineage sorting: Consistent phylogeny estimation from multiple loci. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2008.

[10] L. Nakhleh, G. Jin, F. Zhao, and J. Mellor-Crummey. Reconstructing phylogenetic networks using maximum parsimony. *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference (CSB2005)*, pages 93–102, August 2005.

[11] L. Nakhleh, D. Ruths, and L.S. Wang. RIATA-HGT: A fast and accurate heuristic for reconstrucing horizontal gene transfer. In L. Wang, editor, *Proceedings of the Eleventh International Computing and Combinatorics Conference (COCOON 05)*, pages 84–93, 2005. LNCS #3595.

[12] L. Nakhleh, J. Sun, T. Warnow, R. Linder, B.M.E. Moret, and A. Tholse. Towards the development of computational tools for evaluating phylogenetic network reconstruction methods. In *Proc. 8th Pacific Symp. on Biocomputing (PSB03)*, pages 315–326. World Scientific Pub., 2003.

[13] L. Nakhleh and L.S. Wang. Phylogenetic networks: properties and relationship to trees and clusters. *LNCS Transactions on Computational Systems Biology II*, pages 82–99, 2005. LNBI #3680.

[14] L. Nakhleh and L.S. Wang. Phylogenetic networks, trees, and clusters. In *Proceedings of the 2005 International Workshop on Bioinformatics Research and Applications (IWBRA 05)*, pages 919–926, 2005. LNCS #3515.

[15] D.R. Robinson and L.R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[16] D. Ruths and L. Nakhleh. Recombination and phylogeny: Effects and detection. *International Journal of Bioinformatics Research and Applications (IJBRA)*, 1(2):202–212, 2005.

[17] D. Ruths and L. Nakhleh. Recomp: A parsimony-based method for detecting recombination. In *Proc. 4th Asia-Pacific Bioinformatics Conference*, pages 59–68, 2006.

[18] M. Steel and T. Warnow. Kaikoura tree theorems: computing the maximum agreement subtree. *Information Processing Letters*, 48:77–82, 1993.

[19] C. Than, G. Jin, and L. Nakhleh. Integrating sequence and topology for efficient and accurate detection of horizontal gene transfer. In *Proceedings of the Sixth RECOMB Comparative Genomics Satellite Workshop. Lecture Notes in Bioinformatics (LNBI #5267)*, pages 113–127, 2008.

[20] C. Than and L. Nakhleh. SPR-based tree reconciliation: Non-binary trees and multiple solutions. In *Proceedings of the Sixth Asia Pacific Bioinformatics Conference (APBC)*, 2008.

[21] C. Than and L. Nakhleh. Species tree inference by minimizing deep coalescences. *PLoS Computational Biology*, 5(9):e1000501, 2009.

[22] C. Than, R. Sugino, H. Innan, and L. Nakhleh. Efficient inference of bacterial strain trees from genome-scale multi-locus data. *The 16th Annual International Conference on Intelligent Systems for Molecular Biololgy (ISMB). Bioinformatics*, 24:i123–i131, 2008.

[23] Cuong Than, Derek Ruths, Hideki Innan, and Luay Nakhleh. Confounding factors in hgt detection: Statistical error, coalescent effects, and multiple solutions. *Journal of Computational Biology*, 14:517–535, 2007.

[24] Y. Yu, C. Than, J. Degnan, and L. Nakhleh. Coalescent histories on phylogenetic networks and detection of hybridization despite incomplete lineage sorting. *Systematic Biology*, 2010.

[25] Y. Yu, T. Warnow, and L. Nakhleh. Algorithms for mdc-based multi-locus phylogeny inference. *The 15th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 531–545, 2011. LNBI 6577.