Chapter 3
Input files

This chapter describes the input file format used by COMPONENT. This format is an extension of the NEXUS format developed for the programs PAUP 3.0 and MacClade by David Swofford, Wayne Maddison, and David Maddison (Swofford, 1990; Maddison and Maddison, 1992). The NEXUS format is designed to be a flexible and expandable file format for use in phylogenetic programs. A goal of the format is to allow different programs to be able to readily exchange data, instead of each program having its own format.

NEXUS format input files for COMPONENT can be created using the program's own editing facilities (see Chapter 1), or by using any other text editor that can output standard text files (e.g., Windows Notepad.)

Because of its import and export facilities COMPONENT does not restrict you to just the NEXUS format. The program can read and write Hennig86 and PHYLIP tree files, as well as some other less common formats (e.g., CONTREE and FREQPARS). Hence whatever program you are using you should be able to input trees created by that program into COMPONENT.

The NEXUS format

NEXUS format files consist of the #NEXUS directive at the beginning of the file, followed by a series of "blocks" enclosing groups of commands. Some blocks (such as the TREES block) are shared by a number of programs, other blocks may be specific to a single program. A simple NEXUS file might look like this:

```
#nexus

[ A simple NEXUS file]

begin trees;
      utree PAUP_1 = (a,(b,(c,(d,(e,f)))));
      utree PAUP_2 = (a,(b,((c,d),(e,f))));
endblock;
```

Blocks

Every block has the format

```
BEGIN block-designator;
      <one or more commands>
ENDBLOCK;
```

where *block-designator* is the name of a valid NEXUS block. Each command within the block is terminated by a semicolon (;). Within each block the commands must appear in a logical order, that is, a command that affects another command must precede the second command in the file.

The files are free format. Blanks, tabs, and new lines may be placed anywhere in the file. COMPONENT is not case sensitive for commands, so they can be entered in upper or lower case, or a mixture of both.

Comments

Comments may be included anywhere in the output file by enclosing them in square brackets. For example,

```
[This is a comment]
```

Unless the first character following the "[" is an exclamation point (!), the comment is ignored by the program. If the comment begins with a "[!", as in

```
[!This is an output comment]
```

then COMPONENT will output the comment to the display buffer exactly as it appears. If an output comment occurs inside a TREES block then COMPONENT can display the comment using the **About** command from the **Trees** menu (see Chapter 2).

Comments cannot be nested. Comments that begin with an asterix (*) have a special meaning in MacClade and hence should be avoided in COMPONENT.

Conventions

The following conventions are used in the syntax descriptions that follow. NEXUS commands and reserved words are in upper case. Italicised items — e.g., *x* — represent items to be substituted by the user. Items inside square brackets — e.g., [X] — are optional. Items inside curly braces and separated by vertical bars — e.g., {X |Y |Z} — are mutually exclusive options; the default option, if any, is underlined — e.g., X. An item followed by an ellipsis (…) means that it may be repeated one or more times.

Tokens such as taxon names and file names are entered as strings of characters bounded by whitespace ( ) or punctuation (()[]{}'=;). Examples include

```
Eucalyptus
A._australis
'John''s best tree'
'c:\hennig\peg.tre'
```

Unless a token is enclosed in single quotes (') it can only contain alphanumeric characters (A..Z, a..z, 0..9), the period (.), and the underscore (_). A token that starts with a single quote ends with the next single quote (unless that single quote is in a pair of consecutive single quotes; if so, then the token ends at the next single quote after all such pairs of quotes). For output purposes, tokens enclosed in single quotes have those quotes removed, and any pairs of quotes will be made single. Hence the token `'John''s best tree'` is output as `John's best tree`.

The TAXA block

The TAXA block specifies information about the taxa or areas. There can be only one TAXA block in a file, and it must come before any TREES block. A TAXA block contains only a DIMENSIONS command and a TAXLABELS command.

```
BEGIN TAXA;
      DIMENSIONS NTAX=number-of-taxa;
      TAXLABELS taxon-name ...;
ENDBLOCK;
```

The TREES block

The TREES block is used to input user-defined trees into COMPONENT. Trees can be individually described by a TREE or UTREE command, imported from another file (IMPORT), or generated from a distribution of trees either exhaustively (ALL) or by random sampling (RANDOM).

The syntax for the trees block is:

```
BEGIN TREES;
      [TRANSLATE  token token-used-in-tree-specification
            [,token token-used-in-tree-specification]¼;]
      [TREE [*] tree-name = tree-specification;]
      [UTREE [*] tree-name = tree-specification;]
      [IMPORT [FILETYPE = {COMPONENT|CONTREE|
            HENNIG86|NEXUS|PHYLIP}]
            FILE='file-name';]
      [RANDOM
            MODEL={MARKOVIAN|EQUIPROBABLE
                        |UNLABELED}
            NTREES=number-of-trees
            [UNROOTED]
            [NTAX=number-of-taxa]
            [SEED=integer];]
      [ALL [MODEL={EQUIPROBABLE|UNLABELED}]
            [UNROOTED]
            [NTAX=number-of-taxa];]
ENDBLOCK;
```

The TRANSLATE command can be used to define a translation table that maps arbitrary labels in the tree specification onto valid taxon labels. If no explicit translation table is provided, a default table is defined that maps the integers 1 through NTAX to the corresponding taxon labels, so that integer values rather than taxon labels may be used in the tree specifications.

The TREE and UTREE commands are used to input rooted and unrooted trees, respectively. You cannot mix TREE and UTREE commands in the same file — all trees must be either rooted or unrooted.

The IMPORT command is used to read trees stored in another file in another format. For example, a file of trees output by Hennig86 can be read by COMPONENT using the IMPORT command.

The RANDOM command generates NTREES random trees using the specified model. The ALL command generates all possible trees.

The DISTRIBUTION block

The DISTRIBUTION block is unique to COMPONENT and describes the distribution and relationships of a single clade of entities (organisms or genes) that are associated in some sense with the entities described in the TAXA block. For example, the DISTRIBUTION block may contain the distribution and relationships of a group of parasitic taxa. An input file may have up to 10 DISTRIBUTION blocks.

The syntax for the DISTRIBUTION block is:

```
BEGIN DISTRIBUTION;
      [TITLE = 'title-string';]
      NTAX = number-of-taxa;
      RANGE
            taxon-1 : ahost [another-host ...],
            [taxon-2 : ahost [another-host ...],]
            taxon-ntax : ahost [another-host ...];
      [TREE [*] tree-name = tree-specification;]
ENDBLOCK;
```

The optional TITLE command provides a convenient way of referring to individual distribution blocks.

The NTAX specifies the number of taxa in the distribution block.

The RANGE command describes the distribution of the NTAX taxa.

The TREE command describes the relationships of the NTAX taxa. You can have more than one TREE command in the block.

Command reference

This section describes the syntax for all commands used in the input files.

ALL (TREES block)

The ALL command generates all possible trees.

Syntax

```
[ALL [MODEL={EQUIPROBABLE|UNLABELED}]
      [UNROOTED]
      [NTAX=number-of-taxa];]
```

Example

```
ALL MODEL=UNLABELED UNROOTED;
```

Remarks

If MODEL is not specified then all possible trees, represented one each, will be generated. This is equivalent to MODEL=EQUIPROBABLE. If the UNLABELED model is chosen the all distinct unlabelled trees (i.e., shapes) will be produced.

By default the trees produced are rooted. You can request unrooted trees by using the UNROOTED subcommand.

The NTAX subcommand is only necessary if the number of taxa have not already been defined (for example in the TAXA block). Do not use the NTAX subcommand if your file has a TAXA block.

For the algorithms used by COMPONENT to generate all possible trees see Chapter 6 "Random Trees."

DIMENSIONS (TAXA block)

The DIMENSIONS command specifies how many taxa (or areas) are in an input file.

Syntax

```
DIMENSIONS NTAX=number-of-taxa;
```

Example

```
DIMENSIONS NTAX=10;
```

IMPORT (TREES block)

Imports trees from a file other than the input file. Trees in the import file need not be NEXUS format trees, hence this command allows you to read files created by other programs.

Syntax

```
IMPORT [FILETYPE={COMPONENT |CONTREE |HENNIG86 |
            NEXUS |PHYLIP}] FILE = 'file-name';
```

Example

```
IMPORT FILETYPE=HENNIG86 FILE='c:\hennig\my.tre';
```

Remarks

To import trees from a file you must specify the file's name and type (if you omit the FILETYPE subcommand the program will assume that the file being imported is a NEXUS tree file. The file name must be enclosed in single quotes, and can include drive and path information.

Currently COMPONENT supports these formats:

- CONTREE files created by PAUP 2.4.1 using the CONFILE command.
- COMPONENT 1.5 tree files.
- Hennig86 1.5 files created by the tsave command.
- NEXUS format tree files.
- PHYLIP 3.*x* tree files created with the 'Y' option

When COMPONENT imports trees from Hennig86 or PHYLIP files it ensures that the root of the each tree is binary. Hennig86 trees are always regarded as unrooted trees, PHYLIP trees are regarded as unrooted only if they originally contained a basal trifurcation (e.g., the trees output by FITCH). For more details on these formats and how COMPONENT interprets the trees see Appendix A.

RANDOM (TREES block)

Generates trees sampled at random from a uniform distribution of trees.

Syntax

```
RANDOM MODEL={MARKOVIAN|EQUIPROBABLE|UNLABELED}
       NTREES=number-of-trees
       [UNROOTED]
       [NTAX=number-of-taxa]
       [SEED=integer];
```

Example

```
RANDOM MODEL=EQUIPROBABLE NTREES=1000 SEED=2047;
```

Remarks

RANDOM produces NTREES random trees sampled from the distribution specified by the MODEL subcommand.

The value for SEED must be in the range 1 to 2,147,483,647. If you don't supply a seed COMPONENT will use a value from the system clock. The value of the seed used will be shown in the display buffer.

By default the trees produced are rooted. You can request unrooted trees by using the UNROOTED subcommand (this command is ignored when MODEL = MARKOVIAN).

The NTAX subcommand is only necessary if the number of taxa have not already been defined (for example in the TAXA block). Do not use the NTAX subcommand if your file has a TAXA block.

For the algorithms used by COMPONENT to generate random trees see Chapter 6 "Generating Random Trees."

☞ *COMPONENT does not support the CONSTRAINT subcommand.*

RANGE (DISTRIBUTION block)

Specifies the distribution of a group of entities such as taxa or genes.

Syntax

```
RANGE
      taxon-1 : ahost [another-host ...],
      [taxon-2 : ahost [another-host ...],]
      taxon-ntax : ahost [another-host ...];
```

Example

```
RANGE
      attenuata         : A6,
      jonesi            : A1,
      litoperas         : A9,
      obliqua           : A45,
      anzuetoi          : A10,
      cataractae        : A7,
      dirempta          : A8,
      bimaculata        : A2 A3;
```

Remarks

In the example above the distributions of eight taxa (*attenuata* to *bimaculata*) found in areas A1 to A10 are described. Taxon *bimaculata* occurs in two areas (A2 and A3) and the other taxa are all endemics.

The distribution of a taxon can be described by listing either the area (or host) labels or the number corresponding to the order the area (or host) is listed in the TAXLABELS command in the TAXA block.

TAXLABELS (TAXA block)

Specifies the names of the taxa (or areas) in the input file.

Syntax

```
TAXLABELS taxon-name ...;
```

Example

```
TAXLABELS Ape Human Monkey;
```

TITLE (TAXA, TREES, DISTRIBUTION blocks)

An optional command to give a block a title.

Syntax

```
TITLE = 'title';
```

Example

```
TITLE = 'My favourite trees';
```

TRANSLATE (TREES block)

Translates labels in tree descriptions.

Syntax

```
TRANSLATE  token token-used-in-tree specification
            [,token token-used-in-tree-specification]...;
```

Example

```
TRANSLATE
      H Human,
      A Ape,
      M Mouse;
```

Remarks

To use the TRANSLATE command the input file must also have a TAXA block with a TAXLABELS command.

Translate lets you use abbreviated labels in your tree descriptions. Since COMPONENT will automatically translate numerical labels in tree descriptions you don't need to provide a translation table for numbers.

TREE, UTREE (TREES block)

Describes a rooted (TREE) or an unrooted (UTREE) tree.

Syntax

```
[U]TREE [*] tree-name = tree-specification;
```

Example

```
UTREE PAUP_1 = ((1,2),(3,(4,5)));
```

Remarks

An asterix (*) marks that tree as the default tree. This designation is ignored by COMPONENT. Although COMPONENT requires the trees to be named it does not store the names and trees are always referred to the order in which they occur in the input file.

Tree specifications

Trees are described using a standard parenthetical notation that will be familiar to users of MacClade, PAUP, and PHYLIP. In this notation each cluster in the tree is enclosed by a pair of parentheses ("()").

To write a tree description visit all the nodes in the tree, starting at the root, and follow these rules:

If the node is a leaf

Write the node's label, then return to the node's immediate ancestor.

If the node is an internal node:

1. If you're visiting the node for the first time, write a left parenthesis ("("), then visit the node's *leftmost* child.

2. If you've already visited the node before, but haven't yet visited all that node's descendants, write a comma (","), then visit the next descendant of the node (going from left to right).

3. If you've already visited the node before, and you've visited *all* the node's descendants, write a right parenthesis (")") and visit the node's immediate ancestor (if any). If the current node is the root then stop.

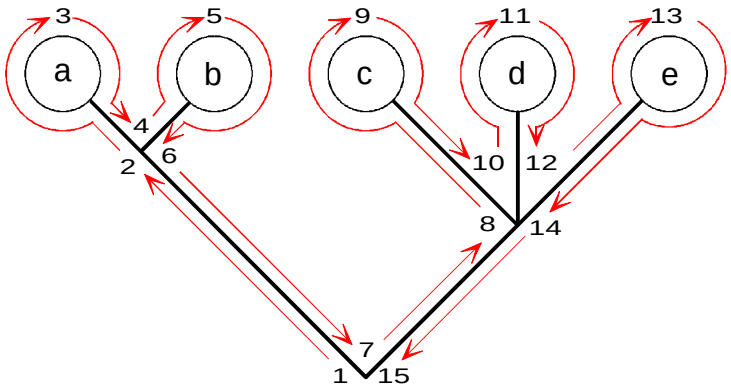


Figure 3.1 The order in which a tree is traversed to create the tree specification

Figure 3.1 shows the order in which a tree is traversed while describing it. Applying the rules to this tree, the description evolves as follows:

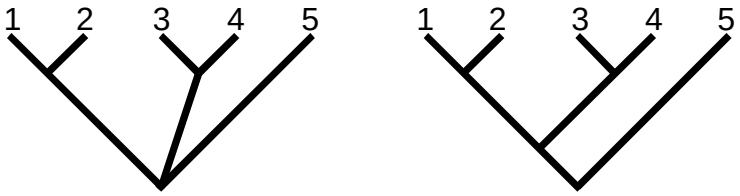
Step	Tree description
1	(
2	((
3	((a
4	((a,
5	((a,b
6	((a,b)
7	((a,b),
8	((a,b),(
9	((a,b),(c
10	((a,b),(c,
11	((a,b),(c,d
12	((a,b),(c,d,
13	((a,b),(c,d,e
14	((a,b),(c,d,e)
15	((a,b),(c,d,e))

Rooted and unrooted trees

The tree description produced by the above method defines a rooted tree. This raises the problem of representing unrooted trees. Rooted trees differ from unrooted trees in having one extra internal node (the root), and some programs represent unrooted trees as rooted trees with this node deleted. For example, Hennig86 and PHYLIP both represent unrooted binary trees as rooted trees with a basal trichotomy.

PAUP 3.0 outputs unrooted binary trees as if they were fully resolved rooted trees, but allows the user to enter unrooted trees using either representation. The example below (after Swofford, 1990) shows two trees that are identical as *unrooted* trees and their corresponding tree descriptions, which are different:

Figure 3.2 Two identical unrooted trees and their corresponding tree specifications, which are different



((1,2),(3,4),5) (((1,2),(3,4)),5)

To avoid such ambiguity COMPONENT requires that a binary unrooted tree be described in the same way as a binary rooted tree, hence you should always describe a tree *as if it were rooted*. Failure to do so may give spurious results.

☞ *COMPONENT requires that each tree description contains the same number of taxa. Unlike PAUP, it does not join any missing taxa to the base of the tree, nor does it allow for unequal sized trees as does MacClade. These differences may be important if you are reading files originally intended for either PAUP or MacClade.*

Example input files

This section provides some example data files to help you gain a better understanding of the NEXUS format. Further examples are given in Chapter 7.

Example 1

This is a simple file specifying three rooted trees.

```
#NEXUS

[!
Margush and McMorris (1981) trees for Adams consensus.
]

BEGIN TREES;
      TREE T1 = (((((1,2),(3,4,5)),6),(7,8)),(9,10));
      TREE T2 = (((1,2),(((3,4),5),6)),((7,8,9),10));
      TREE T3 = ((((1,2,3),4),(5,6)),(((7,8),9),10));
ENDBLOCK;
```

Example 2

This file imports trees from a Hennig86 file. The IMPORT command is used to specify the file created by Hennig86 that contains the trees.

```
#NEXUS

BEGIN TREES;
   IMPORT
      FILETYPE=HENNIG86
      FILE='c:\cp2\windows\turtle.';
ENDBLOCK;
```

Example 3

This file generates 1000 random trees for six taxa. You can also generate random trees using the **Random trees** command from the **Generate** menu, however an advantage of using a file is that you can specify the taxon names.

```
#NEXUS

BEGIN TAXA;
      DIMENSIONS NTAX=6;
      TAXLABELS a b c d e f;
ENDBLOCK;

BEGIN TREES;
      RANDOM MODEL=EQUIPROBABLE NTREES=1000;
ENDBLOCK;
```

Example 4

This file generates the 11 possible unlabelled, unrooted trees for 10 taxa. These trees are the 11 possible shapes for a rooted binary tree.

```
#NEXUS

BEGIN TREES;
      ALL MODEL=UNLABELED UNROOTED NTAX=10;
ENDBLOCK;
```