NEXUS: An Extensible File Format for Systematic Information

Author(s): David R. Maddison, David L. Swofford and Wayne P. Maddison

Source: *Systematic Biology*, Vol. 46, No. 4 (Dec., 1997), pp. 590-621
Published by: Oxford University Press for the Society of Systematic Biologists
Stable URL: https://www.jstor.org/stable/2413497
Accessed: 21-04-2025 08:55 UTC

Syst. Biol. 46(4):590–621, 1997

NEXUS: AN EXTENSIBLE FILE FORMAT FOR SYSTEMATIC INFORMATION

DAVID R. MADDISON,[1] DAVID L. SWOFFORD,[2] AND WAYNE P. MADDISON[3]

[1]*Department of Entomology, University of Arizona, Tucson, Arizona 85721, USA; E-mail: beetle@ag.arizona.edu*
[2]*Laboratory of Molecular Systematics, MRC 534, MSC, Smithsonian Institution, Washington, D.C. 20560, USA*
[3]*Department of Ecology and Evolutionary Biology, University of Arizona, Tucson, Arizona 85721, USA*

Abstract.—NEXUS is a file format designed to contain systematic data for use by computer programs. The goals of the format are to allow future expansion, to include diverse kinds of information, to be independent of particular computer operating systems, and to be easily processed by a program. To this end, the format is modular, with a file consisting of separate blocks, each containing one particular kind of information, and consisting of standardized commands. Public blocks (those containing information utilized by several programs) house information about taxa, morphological and molecular characters, distances, genetic codes, assumptions, sets, trees, etc.; private blocks contain information of relevance to single programs. A detailed description of commands in public blocks is given. Guidelines are provided for reading and writing NEXUS files and for extending the format. [Computer program; file format; NEXUS; systematics.]

NEXUS is a file format designed to house systematic data. Although it is currently in use by several computer programs, including MacClade 3.07 (Maddison and Maddison, 1997), PAUP 3.1.1 (Swofford, 1993) and the forthcoming PAUP* 4, COMPONENT 2 (Page, 1993), SplitsTree (Huson and Wetzel, 1994), and Genetic Data Analysis (Lewis and Zaykin, 1996), the file format and the principles underlying its design and governing future expansion have never been formally described. This document provides the description of the file format.

This paper serves several purposes. First, we seek to promote our goal of a common file format to be shared by phylogenetic computer programs. Virtually every new computer program in systematic biology has used a unique file format. Unique file formats are easier for the programmer because they contain no information extraneous to the particular program, but they often transfer to users the task of reformatting their files for use with different programs. Second, we hope to convey the general principles of NEXUS files so that programmers and users may better understand them. Third, we describe the technical details of the file format and give guidelines for extending it, so that those writing computer programs that read and write NEXUS files can abide by the standard. If our goal of a shared file format is to be reached, it is important that the standard be followed faithfully and that those inventing extensions to the format do so in such a way as to maintain ease of shared use.

Modularity is the primary design feature of a NEXUS file. A NEXUS file is composed of a number of blocks, such as TAXA, CHARACTERS, and TREES blocks. Following is a NEXUS file containing a short DNA sequence for each of four taxa, as well as a single tree:

```
#NEXUS
BEGIN TAXA;
 DIMENSIONS NTAX=4;
 TAXLABELS fish frog snake mouse;
END;
BEGIN CHARACTERS;
 DIMENSIONS NCHAR=20;
 FORMAT DATATYPE=DNA;
 MATRIX
  fish   ACATA GAGGG TACCT CTAAG
  frog   ACTTA GAGGC TACCT CTACG
  snake  ACTCA CTGGG TACCT TTGCG
  mouse  ACTCA GACGG TACCT TTGCG;
END;
BEGIN TREES;
 TREE best=(fish, (frog,
  (snake, mouse)));
END;
```

Other blocks can be added to the file to house various kinds of data, including discrete and continuous morphological characters, distance data, frequency data, and information about protein-coding regions, genetic codes, assumptions about weights, trees, etc. Images can be included. This modular format allows a computer program reading the file to ignore safely the unfamiliar parts of the file, permits sharing of the file by various programs, and permits future expansion to encompass new information.

History of NEXUS File Format

The development of NEXUS began in 1987 for joint use by MacClade 3 (Maddison and Maddison, 1992) and PAUP 3 (Swofford, 1989). MacClade is a program for editing systematic data and conducting phylogenetic analysis of character evolution. PAUP is a program primarily for inference of phylogenies. During the long gestation period of the two programs, the file format evolved. Most of these changes took place before PAUP 3 was released, although some significant modifications were made in the early 1990s.

Roderic Page adopted NEXUS for version 2 of COMPONENT (Page, 1993). COMPONENT is a program for comparing trees and includes commands for reconciling associated phylogenies (e.g., host and parasite trees). COMPONENT's requirements forced some rethinking of the format and were the major reason the TAXA block was invented. The general structure of a NEXUS command was improved to allow easy avoidance of foreign commands.

A number of other projects have housed data using the NEXUS file format. Huson and Wetzel (1994) have adopted the NEXUS format for use in SplitsTree, a program that conducts a "split decomposition" analysis (Bandelt and Dress, 1992; Wetzel, 1994). The NEXUS format is used in Genetic Data Analysis (Lewis and Zaykin, 1996) and in portions of the TreeBASE system (Donoghue et al., 1996).

The NEXUS format will continue to coevolve with systematics computer programs. Commands will be added to accommodate additional kinds of data and assumptions; within this document, some commands are described that are not yet used by any existing computer program.

Goals

In designing the NEXUS format, we had four goals.

1. Expandability.—New elements can be added to the file format with minimum disruption. Programs that were not designed to read these new elements should still be able to acquire as much other information as possible from the file. This goal is achieved by commands of standardized structure organized into labeled blocks, allowing programs to recognize relevant elements and disregard irrelevant ones.
2. Inclusivity.—The file format ideally encompasses all information a systematist or phylogenetic biologist might wish to use, including character and distance data, assumptions, trees, etc.
3. Portability.—The file format can be used by programs running on many different computer operating systems and hardware. The use of simple text files and the insensitivity to varied newline text characters allow for broad portability.
4. Processibility.—Programs can easily read the information they desire and can easily detect and skip other information. A program should be able to tell when it detects an element in the file it does not understand whether or not that lack of knowledge will be fatal to its reading of the file. The uniform definition of words, punctuation, and whitespace and the use of a modular structure enhance processibility.

Basic Elements of a NEXUS File

NEXUS files consist of a string of standard text characters. The text characters form words, whitespace, punctuation, and comments (definitions of these and other terms are given in the Appendix). Words in a NEXUS file are equivalent to words in

a sentence, whitespace is equivalent to blank spaces, and punctuation is equivalent to commas, periods, and the like. (To avoid confusion between the text characters of which a text file is composed and the characters of an organism, we will always use the adjective "text" when referring to the former or, in the special case of a text character indicating the end of a line, "newline character.") For example, in the following line

```
BEGIN TREES;
```

there are two words (BEGIN and TREES) separated by whitespace and followed by a punctuation mark and then by whitespace (in this case a newline character, which is not evident but is present in the file). Punctuation marks and words are also called tokens. Comments can be added by enclosing text with brackets:

```
BEGIN TREES; [some trees follow]
```

Comments can contain information that might be processed, including commands, but also can safely be skipped.

The first token in a standard NEXUS file must be

```
#NEXUS
```

A program reading a NEXUS file seeks to process, in sequence, the tokens it encounters. For the most part, whitespace, including newline characters, is ignored, with two exceptions: (1) whitespace indicates boundaries between words; (2) in interleaved matrices, newline characters indicate the boundary between data of different taxa (for nontransposed matrices) or characters (for transposed matrices) (see descriptions in the CHARACTERS and DISTANCES blocks). For example, a program reading the following file,

```
#NEXUS
BEGIN TREES;
 TREE best=(fish, (frog,
  (snake, mouse)));
END;
```

would first encounter the token #NEXUS, then the words BEGIN and TREES, then the semicolon, then the word TREE, and so on, until the last semicolon was found. In total, the program would read 23 tokens.

The tokens in a NEXUS file are organized into commands, which are in turn organized into blocks. Commands follow a simple format: the first token in the command is the command name, which is followed by a series of tokens and whitespace; the command is terminated by a semicolon. Thus, each command is of the form

```
command-name token token ...;
```

All elements of a NEXUS file are organized into such commands, except for comments (see Appendix) and the #NEXUS token at the start of the file.

Blocks are a series of commands, beginning with a BEGIN command and ending with an END command. Typically a block holds a particular sort of infomation, such as the definition of taxa or the specification of assumptions. A block has the following basic form

```
BEGIN block-name;
 command-name token ...;
 command-name token ...;
 ...
END;
```

In the above example, as in the rest of this article, predefined NEXUS words are shown in small caps, and words to be replaced by relevant tokens are shown in italics.

PUBLIC BLOCKS

Public blocks will be used by a number of current and future programs. All of the blocks defined in this document are considered public. In contrast, private blocks might be designed to contain information that would be relevant only for a single program. The MACCLADE block and PAUP block are two examples of private blocks.

The eight primary public blocks are TAXA, CHARACTERS, UNALIGNED, DISTANCES, SETS, ASSUMPTIONS, CODONS, TREES, and NOTES. The TAXA block defines taxa and gives them names. The block also establishes the order (numbering) of the taxa. The CHARACTERS block contains in-

formation about discrete and continuous data, including that for morphological structure and molecular sequences. Polymorphisms and frequency data can be accommodated. Names can be given to the characters and their states. The UNALIGNED block is similar to a CHARACTERS block, but it contains unaligned molecular sequence data. The SETS block contains descriptions of collections of objects. These objects include characters (so that a collection of characters can be referred to and manipulated easily), taxa (perhaps to allow enforcing monophyly of a group of taxa or to remove them quickly from the tree), trees (perhaps to select quickly a set of trees for study), states, and kinds of changes (perhaps to allow study of hydrophobic to hydrophobic changes in a protein). In addition, partitions of characters, taxa, and trees can be formed. The ASSUMPTIONS block contains assumptions about the data. These can include assignment of weights to various characters, specification of the nature of character changes (e.g., the costs of state-to-state changes for parsimony analysis), exclusion of particular characters, and designation of ancestral states. The CODONS block designates the sites in nucleotide sequence data that are protein coding and the position within a codon of each site and assigns a genetic code to molecular sequence data. Custom genetic codes can also be defined within this block. The NOTES block allows attachment of additional information (text, pictures, etc.) to various objects (trees, taxa, characters, etc.) in the file. Full details of the commands available in each public block are provided in the Technical Description.

BLOCKS IN TYPICAL FILES

The blocks encountered in a typical NEXUS file will vary, depending upon the purpose of the file. Three objects commonly defined in NEXUS files are taxa, characters, and trees, defined in blocks of the same name. Taxa consist of the entities (biological species, haplotypes, manuscripts, etc.) whose attributes might be recorded in characters and whose relationships are described in trees. Most programs will therefore require blocks defining or describing one or more of these classes of objects. A simple data file may contain TAXA and CHARACTERS blocks, and a tree file may contain TAXA and TREES blocks. Additional public blocks (e.g., ASSUMPTIONS, SETS) and private blocks (e.g., MACCLADE, PAUP) will often be present.

READING AND WRITING NEXUS FILES

Although the NEXUS format contains many elements, some of them complex, it is easier to use than it may at first appear, largely because of its consistent structure and modular approach. Here we give some suggestions for reading and writing NEXUS files. If a program adopts the NEXUS format, it not only gains compatibility with other computer programs, but the modular format allows easier future expansion of the program's own capabilities.

Reading NEXUS Files

Because NEXUS files are organized into tokens (words and punctuation), along with whitespace and comments, the most basic component of the code to read a file is a routine that returns the next punctuation or word in a file. This routine needs to recognize boundaries between NEXUS tokens and to distinguish words from punctuation.

The first token after the #NEXUS word or after the semicolon that terminates a command is the name of a NEXUS command. This command will indicate either the start or end of a block or of a command within a block. Routines need to be written to handle the commands and blocks of relevance to the program and to skip those irrelevant to the program. A routine to handle a command needs to recognize its first token, which indicates what sort of command it is, and subsequent tokens, which specify details of the command. If the first token indicates it is a command that is not relevant to the program, the program skips it by using its token-reading routine to move to the next semicolon. A routine to handle a block needs to recognize the sort of block it is and then pro-

cess the commands one after another. If the block is not relevant to the program, it is ignored by skipping command after command until the END or ENDBLOCK command is reached.

Programs that read NEXUS files do not have to be able to understand all aspects of the file format; in fact, no program at this time can understand more than about 60% of the elements described in this document. It is thus critical for a program reading a NEXUS file to be able to deal well with elements that are foreign to the program. For example, if MacClade were to encounter a DISTANCES block, it must be able to skip successfully over this foreign block. Individual foreign commands within a block must also be gracefully ignored.

Sample code for reading a NEXUS file is presented on the NEXUS web site, at http://phylogeny.arizona.edu/nexus/nexus.html.

Writing NEXUS Files

Many programs using the NEXUS format will be designed to read the format but not to write automatically files of NEXUS format. However, if a program generates NEXUS format files, then these files should conform to the rules of the NEXUS format. For example, elements of the file need to be written as NEXUS words, punctuation, and whitespace: if the user types in the taxon name "Bembidion velox" then the program should write the blank space as an underscore (Bembidion_velox) to form a standard NEXUS token. In addition, the program should not discard information that had been in the file when read, such as unfamiliar blocks or commands. In rewriting the file, the program should attempt to place the comments, blocks, and commands intact back into the file, preferably in the same order in which they were found. If the program cannot restore foreign blocks, commands, or comments, then it should attempt to leave untouched the original copy of the file.

EXTENDING THE NEXUS FORMAT

Extensions to the NEXUS format should be made with due consideration of the goals of the format.

1. When extensions are designed, care should be taken not to restrict possible future uses. New blocks and new commands for housing a new class of information should be designed with enough flexibility to contain not just the information needed for one version of one particular program but also additional information that might be needed by future versions or other programs. For example, the CODESET command in the CODONS block was designed to allow one to assign different genetic codes to different taxa in the matrix. No program currently accepts files using this feature, but one can easily imagine some future program that analyzes evolution of the genetic code requiring this capability. Because the CODESET command requires that genetic codes be given names, the design of a command (GENETICCODE) that is used by current programs to create custom genetic codes was altered to accommodate names, even though these names are not considered by existing programs.
2. Information should be placed in public blocks whenever possible. In general, private blocks should be used to store information that probably will not be used by other programs; other information that may be more generally useful should be placed in public (e.g., CHARACTERS, ASSUMPTIONS, TREES) blocks. This recommendation applies to extensions to the format and use of the current format.
3. Information contained within a public block should be as uniform as possible. For example, CHARACTERS blocks should contain information about organismal characteristics, not about assumptions. Blocks containing data should contain just one class of data (e.g., do not invent a new type of block that contains allozyme frequency data, sequence data, and geographic data). A future program might want to deal with one of the sorts of information but not the other, and separating them into different blocks will allow the pro-

gram more easily to find what it needs. This rationale is behind the separation of taxon definition and character information into two separate blocks.

4. All similar information should reside in one type of block. For example, all tree descriptions should be housed in a TREES block.
5. New commands in public blocks can be added, but they must be clearly distinguished from the existing ones. Some programs (e.g., PAUP) allow use of abbreviated command names. Abbreviations for new command names should not duplicate abbreviations for existing command names. The formats of any new commands should follow as closely as possible those described in this document. New command or subcommand names that are no more than singular versions of previously existing plural names are not allowed as new command or subcommand names nor are plural versions of existing singular names. Commands or subcommands that differ only in case are homonymous.
6. All commands defined in this document are reserved command names, in the sense that they cannot be used with another meaning. (However, subcommands can be added to change the format of those commands.) In addition, the following tokens are reserved for future use for names of blocks: INTERNATIONAL, NETWORK, NODES, COORDINATES, ASSOCIATION, CLASSIFICATION, and FORMAT. The following command names are reserved for future use: INCLUDEFILE (anywhere in the file), TITLE and ENTITLE (in any block), SOUND and VIDEO (in the NOTES block), and GRAPH (in the TREES block).
7. New commands should be designed to follow the format of an existing command whenever possible. For example, to name and describe some object, the format of an object definition command should be used.
8. If a new MATRIX format is devised for a block containing data, this new format should be indicated by a subcommand in the block's FORMAT command. All of the information necessary to read the matrix should be contained in the DIMENSIONS command and the FORMAT command and in any commands referenced in the FORMAT command (e.g., the CHARSTATELABELS command if the FORMAT subcommand TOKENS is used with DATATYPE =STANDARD).
9. New command comments should be invented with the same care as normal NEXUS commands. Except in the TREE command, command comments should not contain information that is critical to an analysis.
10. All dark characters in a file must be (1) #NEXUS at the start of the file, (2) a comment, or (3) a command consisting of NEXUS tokens, terminated by a semicolon.
11. Whenever possible, newline characters should have no special meaning in addition to the standard meaning of whitespace.
12. Additions to the format should be made in consultation with the authors of this document.

TECHNICAL DESCRIPTION OF NEXUS FORMAT

This description is divided into several parts. First, general issues having to do with computer operating system, file components and structure, objects, and text display are covered. Second, the public blocks are described.

Any particular implementation of the NEXUS format, even by authors of this initial description, should not necessarily be considered as complying exactly with the NEXUS standard.

NEXUS Files and Computer Operating Systems

All NEXUS-compatible programs should be able to read standard text files. On the many computers that use the ASCII character set (American National Standards Institute, 1986), standard ASCII files

should be supported (files of text characters composed of ASCII characters 0–127). Ability to read extended ASCII (text characters 128–255) files is optional. Programs on computers that use character sets other than ASCII should support the native character set.

On operating systems that allow designation of file types (e.g., MacOS™), NEXUS files should be designated as type TEXT.

Programs that read NEXUS files should consider the variation in text characters that specify a new line. For example, a program should interpret a single carriage return (ASCII 13), the newline character used on Macintosh computers, as indicating a new line, even if the program runs under the UNIX operating system (which expects new lines to be indicated by a linefeed [ASCII 10]). A NEXUS-compatible program reading a NEXUS file should treat all newline characters (as defined in the Appendix) as equivalent.

Tokens, Words, Punctuation, and Whitespace

Words, punctuation, and whitespace together form a NEXUS file. Tokens are the words and punctuation of a NEXUS command. Words are separated by either whitespace (blank spaces, tabs, etc.) or punctuation. The Appendix contains formal definitions of these terms.

The following general rules apply to NEXUS tokens. Case is insignificant in NEXUS files except where explicitly stated otherwise (e.g., the EQUATE subcommand in the CHARACTERS block). Abbreviations of command, subcommand, and option words are not allowed (although some programs, such as PAUP, may be lenient in this regard).

File Structure

A NEXUS file has the following basic form,

```
#NEXUS
BEGIN block-name;
  command-name token ...;
  command-name token ...;
  ...
END;
BEGIN block-name;
  command-name token ...;
  command-name token ...;
  ...
END;
```

All currently defined commands in a NEXUS file are included within blocks. Any particular block may contain data, trees, assumptions, etc. Blocks cannot be nested. Blocks start with the BEGIN command and end with the END or ENDBLOCK commands. The BEGIN command consists of only two words: BEGIN and a word designating the type of block. Example block names include CHARACTERS, TAXA, and TREES.

The order of blocks is predetermined for some pairs of blocks but not others. For example, most programs will require a CHARACTERS or DATA block to precede the ASSUMPTIONS block so that the characters to which the ASSUMPTIONS block refers will be defined when the ASSUMPTIONS block is encountered. Information must be defined before it is needed (not after).

Although the NEXUS standard does not impose constraints on the number of blocks, particular programs will. For example, MacClade 3.07 does not allow more than one TAXA block in a file.

NEXUS Objects

Many of the commands in a NEXUS file define objects or specify characteristics about them. Currently defined objects include taxa, characters, states, trees, genetic codes, sets (of taxa, characters, states, classes of changes between states, trees), partitions (of taxa, trees, characters), weight sets, types, type sets, character exclusion sets, ancestral states, and codon position sets.

All objects can be labeled (given names). Duplicate names should be avoided if this might cause ambiguity. For example, two taxa should not have the same name nor should a taxon and taxon-set nor a character and a character-set. Names that differ only in case (e.g., Beetles and beetles) should also be avoided because they are

considered homonyms. Names are single NEXUS words; they cannot consist entirely of digits (e.g., a taxon called 123 is illegal).

Taxa, characters, and trees possess an inherent order, determined by their order of definition. This allows them to be referred to by name (if they have been given names) or by number. Thus, in the following matrix,

```
MATRIX
 taxon_A 001020101
 taxon_B 101001022
 taxon_C 001010201;
```

the second taxon could be referred to as taxon_B or as 2. Characters, taxa, and trees can be referred to by their name or number; discrete states can be referred by name or state symbol. All objects other than taxa, characters, states, and trees must be referred to by name, not number.

Although the manner of definition of an object varies among different types of objects, several of the object definitions follow a particular format. Here are two examples of object definition commands:

```
CHARSET larval=1-15;
WTSET myweights (VECTOR)=2 1 2 0 2;
```

The objects that are defined in this way are GENETICCODE, CODONPOSSET, CODESET, CHARSET, STATESET, CHANGESET, TAXSET, TREESET, CHARPARTITION, TAXPARTITION, TREEPARTITION, USERTYPE, WTSET, TYPESET, EXSET, ANCSTATES, and TREE.

Taxa can be defined in TAXA and DATA blocks and in CHARACTERS, UNALIGNED, and DISTANCES blocks (if the NEWTAXA token is included in the DIMENSIONS command).

Display of Text

In general, programs should present text to users as it appears in the NEXUS file. If the NEXUS file contains the token Beetle, then it should be displayed as Beetle, not BEETLE. If the program presents the text outside of a literal display of the NEXUS file itself, however, there are cases in which the program should first alter the text. For example, underscores should be converted to blanks for display purposes: "Big_Beetle" should appear in a printed tree as "Big Beetle." Two adjacent single quotes within quoted tokens should be converted to one single quote.

Some programs will display text in various styles (italics, bold, underlined) if the appropriate formatting information is included in the NEXUS file. This is done with a special set of comments:

```
[\i]: subsequent text is to be
 displayed in italics
[\b]: subsequent text is to be
 displayed in bold
[\u]: subsequent text is to be
 displayed underlined
[\p]: subsequent text is to be
 displayed in a plain style
  (i.e., not italics, bold, or
  underlined)
```

For example, [\i]Homo sapiens[\p] Linneaus would tell a program to display the text as *Homo sapiens* Linnaeus; [\i]Homo [\b]sapiens[\p] Linneaus would tell a program to display the text as *Homo* ***sapiens*** Linnaeus.

Descriptions of Public Blocks

The following conventions are used in syntax descriptions of public blocks that follow. Uppercase items should not be abbreviated. They can be entered in uppercase or lowercase. Italicized items (e.g., *x*) represent items to be substituted with the relevant text characters. Items inside square brackets (e.g., [X]) are optional. If the contents within square brackets end in an ellipsis (. . .), then the contents can be repeated one or more times. Items inside curly braces and separated by vertical bars (e.g., {X|Y|Z}) are mutually exclusive options. Underlined options are the default values.

TAXA Block

The TAXA block specifies information about taxa.

```
BEGIN TAXA;
 DIMENSIONS NTAX=number-of-taxa;
 TAXLABELS taxon-name [taxon-name
 . . .];
END;
```

DIMENSIONS must appear before TAXLABELS. Only one of each command is allowed per block.

DIMENSIONS.—The NTAX subcommand of the DIMENSIONS command indicates the number of taxa. The NEXUS standard does not impose limits on number of taxa; a limit may be imposed by particular computer programs.

TAXLABELS.—This command defines taxa, specifies their names, and determines their order:

```
TAXLABELS Fungus Insect Mammal;
```

Taxon names are single NEXUS words. They must not correspond to another taxon name or number; thus, 1 is not a valid name for the second taxon listed. The standard defines no limit to their length, although individual programs might impose restrictions.

Taxa may also be defined in the CHARACTERS, UNALIGNED, and DISTANCES blocks if the NEWTAXA token is included in the DIMENSIONS command; see the descriptions of those blocks for details.

CHARACTERS Block

A CHARACTERS block defines characters and includes character data. Taxa are usually not defined in a CHARACTERS block; if they are not, the CHARACTERS block must be preceded by a block that defines taxon labels and ordering (e.g., TAXA). Details about different types of character data are given at the end of this section.

Syntax of the CHARACTERS block is as follows:

```
BEGIN CHARACTERS;
 DIMENSIONS [NEWTAXA NTAX=number-
   of-taxa] NCHAR=number-of-
   characters;
 [FORMAT
  [DATATYPE={STANDARD| DNA | RNA |
    NUCLEOTIDE | PROTEIN | CONTINUOUS}]
  [RESPECTCASE]
  [MISSING=symbol]
  [GAP=symbol]
  [SYMBOLS="symbol [symbol...]"]
  [EQUATE="symbol=entry
    [symbol=entry...]"]
  [MATCHCHAR=symbol]
  [[NO]LABELS]
  [TRANSPOSE]
  [INTERLEAVE]
  [ITEMS=([MIN] [MAX] [MEDIAN]
    [AVERAGE] [VARIANCE] [STDERROR]
    [SAMPLESIZE] [STATES])]
  [STATESFORMAT={STATESPRESENT |
    INDIVIDUALS | COUNT | FREQUENCY}]
  [[NO]TOKENS]
  ;]
 [ELIMINATE character-set;]
 [TAXLABELS taxon-name [taxon-name
   ...];]
 [CHARSTATELABELS
  character-number
   [character-name]
   [/state-name [state-name...]]
  [, character-number
   [character-name]
   [/state-name [state-name...]]
   ...]
  ;]
 [CHARLABELS character-name
   [character-name...];]
 [STATELABELS
  character-number
   [state-name [state-name...]]
  [, character-number
   [state-name [state-name...]]
   ...]
  ;]
 MATRIX data-matrix;
END;
```

DIMENSIONS, FORMAT, and ELIMINATE must all precede CHARLABELS, CHARSTATELABELS, STATELABELS, and MATRIX. DIMENSIONS must precede ELIMINATE. Only one of each command is allowed per block.

DIMENSIONS.—The DIMENSIONS command specifies the number of characters. The number following NCHAR is the number of characters in the data matrix. The NEXUS standard does not impose limits on the number of characters; a limit may be imposed by particular computer programs.

It is strongly advised that new taxa not be defined in a CHARACTERS block, for the reasons discussed in the description of the

DATA block. If new taxa are to be defined, this must be indicated by the NEWTAXA subcommand, specifying that new taxa are to be defined (this allows the computer program to prepare for creation of new taxa). NEWTAXA, if present, must appear before the NTAX subcommand.

The NTAX subcommand, indicating the number of taxa in the MATRIX command in the block, is optional, unless NEWTAXA is specified, in which case it is required.

FORMAT.—The FORMAT command specifies the format of the data MATRIX. This is a crucial command because misinterpretation of the format of the data matrix could lead to anything from incorrect results to spectacular crashes. The DATATYPE subcommand must appear first in the command. The RESPECTCASE subcommand must appear before the MISSING, GAP, SYMBOLS, and MATCHCHAR subcommands. The following are possible formatting subcommands.

1. DATATYPE={STANDARD | DNA | RNA | NUCLEOTIDE | PROTEIN | CONTINUOUS}. This subcommand specifies the class of data. If present, it must be the first subcommand in the FORMAT command. Standard data consist of any general sort of discrete character data, and this class is typically used for morphological data, restriction site data, and so on. DNA, RNA, NUCLEOTIDE, and PROTEIN designate molecular sequence data. Meristic morphometric data and other information with continuous values can be housed in matrices of DATATYPE=CONTINUOUS. These DATATYPES are described in detail, with examples, at the end of the description of the CHARACTERS block.

2. RESPECTCASE. By default, information in a MATRIX may be entered in uppercase, lowercase, or a mixture of uppercase and lowercase. If RESPECTCASE is requested, case is considered significant in SYMBOLS, MISSING, GAP, and MATCHCHAR subcommands and in subsequent references to states. For example, if RESPECTCASE is invoked, then SYMBOLS="A a B b" designates four states whose symbols are A, a, B, and b, which can then each be used in the MATRIX command and elsewhere. If RESPECTCASE is not invoked, then A and a are considered homonymous state symbols. This subcommand must appear before the SYMBOLS subcommand. This subcommand is not applicable to DATATYPE=DNA, RNA, NUCLEOTIDE, PROTEIN, and CONTINUOUS.

3. MISSING. This subcommand declares the symbol that designates missing data. The default is "?". For example, MISSING=X defines an X to represent missing data. Whitespace is illegal as a missing data symbol, as are the following symbols: ( ) [ ] { } / \ , ; : = * ' " ` < > ^

4. GAP. This subcommand declares the symbol that designates a data gap (e.g., base absent in DNA sequence because of deletion or an inapplicable character in morphological data). There is no default gap symbol; a gap symbol must be defined by the GAP subcommand before any gaps can be entered into the matrix. For example, GAP=- defines a hyphen to represent a gap. Whitespace is illegal as a gap symbol, as are the following symbols: ( ) [ ] { } / \ , ; : = * ' " ` < > ^

5. SYMBOLS. This subcommand specifies the symbols and their order for character states used in the file (including in the MATRIX command). For example, SYMBOLS="0 1 2 3 4 5 6 7" designates numbers 0 through 7 as acceptable symbols in a matrix. The SYMBOLS subcommand is not allowed for DATATYPE=CONTINUOUS. The default symbols list differs from one DATATYPE to another, as described under *state symbol* in the Appendix. Whitespace is not needed between elements: SYMBOLS="012" is equivalent to SYMBOLS="0 1 2".

For STANDARD DATATYPES, a SYMBOLS subcommand will replace the default symbols list of "0 1". For DNA, RNA, NUCLEOTIDE, and PROTEIN DATATYPES, a SYMBOLS subcommand will not replace the default symbols list but will add character-state symbols to the SYMBOLS list. The NEXUS standard does not define the position of these additional symbols within the SYMBOLS list. (These additional symbols will be inserted at the beginning of the SYMBOLS list in PAUP and at the end in MacClade. MacClade will accept additional symbols

for PROTEIN but not DNA, RNA, and NUCLEOTIDE matrices.)

6. EQUATE. This subcommand allows one to define symbols to represent one matrix entry. For example, EQUATE="E=(012)" means that each occurrence of E in the MATRIX command will be interpreted as meaning states 0, 1, and 2. The equate symbols cannot be ( ) [ ] { } / \ , ; : = * ' " ` < > ^ or any of the currently defined MISSING, GAP, MATCHCHAR, or state SYMBOLS. Case is significant in equate symbols. That is, MISSING=? EQUATE="E=(012) e=?" means that E will be interpreted as 0, 1, and 2 and e will be interpreted as missing data.

7. MATCHCHAR. This subcommand defines a matching character symbol. If this subcommand is included, then a matching character symbol in the MATRIX indicates that the states are equivalent to the states possessed by the first taxon listed in the matrix for that character. In the following matrix, the sequence for taxon 2 is GACTTTC:

```
BEGIN DATA;
 DIMENSION NCHAR=7;
 FORMAT DATATYPE=DNA MATCHCHAR=.;
 MATRIX
  taxon_1 GACCTTA
  taxon_2 ...T..C
  taxon_3 ..T.C..;
END;
```

Whitespace is illegal as a matching character symbol, as are the following symbols: ( ) [ ] { } / \ , ; : = * ' " ` < > ^

8. [NO]LABELS. This subcommand declares whether taxon or character labels are to appear on the left side of the matrix. By default, they should appear. If NOLABELS is used, then no labels appear, but then all currently defined taxa must be included in the MATRIX in the order in which they were originally defined.

9. TRANSPOSE. This subcommand indicates that the MATRIX is in transposed format, with each row of the matrix representing the information from one character and each column representing the information from one taxon. The following is an example of a TRANSPOSED MATRIX:

```
MATRIX
 character_1  1 0 1 1 0 1
 character_2  0 1 1 1 0 0
 character_3  0 1 1 1 1 0;
```

10. INTERLEAVE. This subcommand indicates that the MATRIX is in interleaved format, i.e., it is broken up into sections. If the data are not transposed, then each section contains the information for some of the characters for all taxa. For example, the first section might contain data for characters 1–50 for all taxa, the second section contains data for characters 51–100, etc. Taxa in each section must occur in the same order. This format is especially useful for molecular sequence data, where the number of characters can be large. A small interleaved matrix follows:

```
MATRIX
 taxon_1  A C C T C G G C
 taxon_2  A C C T C G G C
 taxon_3  A C G T C G C T
 taxon_4  A C G T C G C T

 taxon_1  T T A A C G A
 taxon_2  T T A A C C A
 taxon_3  C T C A C C A
 taxon_4  T T C A C C A
 ;
```

The interleaved sections need not all be of the same length. In an interleaved matrix, newline characters are significant: they indicate that the next character information encountered applies to a different taxon (for nontransposed matrices).

11. ITEMS. Each entry in the matrix gives information about a character's condition in a taxon. The ITEMS subcommand indicates what items of information are listed at each entry of the matrix. With discrete character data, the entry typically consists of the states observed in the taxon (either the single state observed or several states if the taxon is polymorphic or of uncertain state). This can be specified by the statement ITEMS=STATES, but because it is the default and the only option allowed by most current programs for discrete data, an ITEMS statement is usually unnecessary. For continuous data, however, the wealth of alternatives (average, median, variance,

minimum, maximum, sample size) often requires an explicit ITEMS statement to indicate what information is represented in each data matrix entry. Some ITEMS (such as VARIANCE) would be appropriate to only some DATATYPES; other ITEMS such as SAMPLESIZE and STATES would be appropriate to most or all DATATYPES. If more than one item is indicated, parentheses must be used to surround the list of items, e.g., ITEMS=(AVERAGE VARIANCE); otherwise the parentheses are unnecessary, e.g., ITEMS=AVERAGE. More information about ITEMS options can be found in the discussion of the different DATATYPES under MATRIX; information specifically about the STATES option is given under STATESFORMAT.

12. STATESFORMAT. The entry in a matrix usually lists (for discrete data) or may list (for continuous data) the states observed in the taxon. The STATESFORMAT subcommand specifies what information is conveyed in that list of STATES. In most current programs for discrete data, when a taxon is polymorphic the entry of the matrix lists only what distinct states were observed, without any indication of the number or frequency of individuals sampled with each of the states. Thus, if all individuals sampled within the taxon have state A, the matrix entry would be A, whereas if some have state A and others have state B, the entry would be (AB), which corresponds to the option STATESFORMAT=STATESPRESENT. Because it is the default for discrete data, this statement is typically unnecessary with current programs. The other STATESFORMAT options can be illustrated with an example, in which two individuals of a taxon were observed to have state A and three were observed to have state B. When STATESFORMAT=INDIVIDUALS, the state of each of the individuals (or other appropriate sampling subunit) is listed exhaustively, (A A B B B); when STATESFORMAT=COUNT, the number of individuals with each observed state is indicated, e.g., (A:2 B:3); when STATESFORMAT=FREQUENCY, the frequencies of various observed states are indicated, e.g., (A:0.40 B:0.60). The STATESFORMAT command may also be used for continuous data, for which the default is STATESFORMAT=INDIVIDUALS.

13. [NO]TOKENS. This subcommand specifies whether data matrix entries are single symbols or whether they can be tokens. If TOKENS, then the data values must be full NEXUS tokens, separated by whitespace or punctuation as appropriate, as in the following example:

```
BEGIN CHARACTERS;
 DIMENSIONS NCHAR=3;
 CHARSTATELABELS 1 hair/absent
  present, 2 color/red blue,
  3 size/small big;
 FORMAT TOKENS;
 MATRIX
  taxon_1 absent red big
  taxon_2 absent blue small
  taxon_3 present blue small;
END;
```

TOKENS is the default (and the only allowed option) for DATATYPE=CONTINUOUS; NOTOKENS is the default for all other DATATYPES. TOKENS is not allowed for DATATYPES DNA, RNA, and NUCLEOTIDE. If TOKENS is invoked, the standard three-letter amino acid abbreviations can be used with DATATYPE=PROTEIN and defined state names can be used for DATATYPE=STANDARD.

ELIMINATE.—This command allows specification of a list of characters that are to be excluded from consideration. Programs are expected to ignore ELIMINATED characters completely during reading. In avoiding allocation of memory to store character information, the programs can save a considerable amount of computer memory. (This subcommand is similar to ZAP in version 3.1.1 of PAUP.) For example,

```
ELIMINATE 4-100;
```

tells the program to skip over characters 4 through 100 in reading the matrix. Character-set names are not allowed in the character list. This command does not affect character numbers.

TAXLABELS.—This command allows specification of the names of the taxa. It serves to define taxa and is only allowed in a CHARACTERS block if the NEWTAXA to-

ken is included in the DIMENSIONS statement.

CHARSTATELABELS.—This command allows specification of both the names of the characters and the names of the states. This command was developed as an alternative to the older commands CHARLABELS and STATELABELS. For example,

```
CHARSTATELABELS
  1 eye_color/red blue green,
  3 head_shape/round square,
  5 pronotum_size/small medium
    large
  ;
```

A forward slash (/) separates the character name and the state names, with a comma separating the information for different characters. If no state names are to be specified, the slash may be omitted; if no character names are to be specified, the slash must be included, but no token needs to be included between the character number and the slash. If state x is the last state to be named, then subsequent states need not be named, but states 1 through x must be. If no name is to be applied to a state, enter a single underscore for its name. Character and state names are single NEXUS words. Character names must not correspond to another character name or number; thus, 1 is not a valid name for the second character listed. There is no restriction on the length of a character or state name imposed by the NEXUS standard; however, particular programs may limit the length. State names cannot be applied if DATATYPE=CONTINUOUS.

CHARLABELS.—This command allows specification of names of characters:

```
CHARLABELS
 flange microsculpture
 body_length
 hind_angles #_spines
 spine_size _ _ head_size
 pubescent_ intervals head_color
 clypeal_margin;
```

Character labels are listed consecutively. If character x is the last character to be named, then subsequent characters need not be named, but characters 1 through x need to be. If no name is to be applied to a character, a single underscore can be used for its name. Character names are single NEXUS words. They must not correspond to another character name or number; thus, 1 is not a valid name for the second character listed.

There is no restriction on the length of a character name imposed by the NEXUS standard; however, particular programs may limit the length. The command should be used only for nontransposed matrices (in transposed matrices, the character labels are defined in the MATRIX command).

We recommend that programs abandon this command in place of the more flexible CHARSTATELABELS command when writing NEXUS files, although programs should continue to read CHARLABELS because many existing NEXUS files use CHARLABELS.

STATELABELS.—This command allows specification of the names of states:

```
STATELABELS
 1 absent present,
 2 isodiametric transverse,
 3 '4.5-6.2mm' '6.3-7.0mm' '7.7-
  11.0mm' '>12.0mm',
 4 rounded subangulate angulate,
 10 0 '1-4' '6-9' '7-9' '8-9' 7 8 9,
 11 black rufous metallic flavous,
 12 straight concave,
 ;
```

(The single quotes that surround some of the state labels in this example are needed to properly define the boundaries of the words; see the definition of *word* in the Appendix.) State labels need not be specified for all characters. A comma must separate state labels for each character. State labels are listed consecutively within a character. If state x is the last state to be named, then subsequent states need not be named, but states 1 through x must be. If no name is to be applied to a state, enter a single underscore for its name. State names are single NEXUS words. The standard defines no limit to their length, although individual programs might impose restrictions.

This command is not valid for DATATYPE=CONTINUOUS.

We recommend that programs abandon this command in place of the more flexible CHARSTATELABELS command when writing NEXUS files, although programs should continue to read STATELABELS because many existing NEXUS files use STATELABELS.

MATRIX.—In its standard format, the MATRIX command contains a sequence of taxon names and state information for that taxon. The MATRIX itself is of the form

```
MATRIX
  taxon-name entry entry...entry
  taxon-name entry entry...entry
  ...
  taxon-name entry entry...entry;
```

Each entry in the matrix is the information about a particular character for a particular taxon. For example, it might be the assignment of state 0 to taxon 1 for character 1. Thus, the entry would consist of one state symbol, 0. If the taxon were polymorphic, the entry would consiste of multiple state symbols, e.g., (0 1), indicating the taxon has both states 0 and 1. More details about the nature of each entry of the matrix are given under ITEMS and under each DATATYPE.

Each entry needs to be enclosed in parentheses or braces whenever more than one state symbol is given, e.g. (01) with standard data and the default NOTOKENS option, or if the information is conveyed by more than one NEXUS token, e.g., (0:100) or (2.3 4.5 6.7). Otherwise, the parentheses or braces are optional. No whitespace is needed between entries in the matrix unless the TOKENS subcommand of the FORMAT command is invoked or implied and parentheses or braces do not surround an entry.

Taxa need not be in the same order as in the TAXA block, and the matrix need not contain all taxa. For interleaved matrices, all sections must have the same taxa in the same order.

Examples of matrices of different DATATYPES are described below.

1. STANDARD data. For DATATYPE=STANDARD, each entry of the matrix consists of a single state-set. Under the defaults (ITEMS=STATES and STATESFORMAT=STATESPRESENT), each entry of the matrix consists of a single state-set; if there are multiple states, then the entry must be enclosed in parentheses (indicating polymorphism) or braces (indicating uncertainty in state). For example, in the following matrix,

```
BEGIN CHARACTERS;
 DIMENSIONS NCHAR=9;
 FORMAT SYMBOLS="- + x";
 MATRIX
  taxon_1 (-+){-+}+---+--
  taxon_2 +x-++--+x
  taxon_3 -++++--+x;
END;
```

taxon_1 is polymorphic for the first character and has either state - or state + for the second character. If STATESFORMAT=COUNT or FREQUENCY, then each entry must be enclosed in parentheses because more than one token is required to convey information for even one state:

```
BEGIN CHARACTERS;
 DIMENSIONS NCHAR=3;
 FORMAT STATESFORMAT=FREQUENCY
  SYMBOLS="0 1 2";
 MATRIX
  taxon_1 (0:0.25 1:0.75)
   (0:0.3 1:0.7)
   (0:0.5 1:0.3 2:0.2)
  taxon_2 (0:0.4 1:0.6)
   (0:0.8 1:0.2) (1:0.15 2:0.85)
  taxon_3 (0:0.0 1:1.0)
   (0:0.55 1:0.45)(0:0.1 1:0.9);
END;
```

2. DNA, RNA, NUCLEOTIDE, and PROTEIN data. For DATATYPE=DNA, RNA, NUCLEOTIDE, or PROTEIN, each entry of the matrix consists of one or more state symbols describing the state(s) at one site in a molecular sequence. If STATESFORMAT=STATESPRESENT and if an entry represents a single state, then it is represented as a single state symbol (or if DATATYPE=PROTEIN and TOKENS, as a three-letter amino acid name). If an entry represents multiple states, then it must be

enclosed in parentheses (indicating polymorphism) or braces (indicating uncertainty in state). Following is a matrix of DATATYPE=DNA:

```
BEGIN CHARACTERS;
 DIMENSIONS NCHAR=12;
 FORMAT DATATYPE=DNA;
 MATRIX
  taxon_1 ACCATGGTACGT
  taxon_2 TCCATGCTACCC
  taxon_3 TCCATGGAACCC;
END;
```

3. CONTINUOUS data. For DATATYPE=CONTINUOUS, each entry in the matrix must be enclosed by parentheses if more than one item is specified in the ITEMS subcommand. Parentheses must also be used whenever multiple tokens are needed for an entry in the matrix. If an entry consists of a single token (e.g., 0.231), it may be written without parentheses but must then be separated from other entries by whitespace.

```
MATRIX
 A 0.453 1.43 78.6
 B 0.34  1.02 55.7
 C 0.22  1.79 69.1;
```

A matrix entry can include average, minimum, maximum, variance, standard error, sample size, and a listing of states observed in the taxon, as specified in the ITEMS subcommand. The sample size, if included, must be in the form of an integer; the other numbers can be either in English decimal (e.g., 0.00452) or in exponential form (e.g., 4.52E-3).

The information listed for each taxon for a continuous character is specified in the ITEMS subcommand of the FORMAT command. For example, if the matrix contains only information about the minimum and maximum value for each taxon, the ITEMS subcommand would be

```
ITEMS=(MIN MAX)
```

and a small matrix might look something like this:

```
MATRIX
 taxon_1 (0.21 0.45) (0.34 0.36)
 taxon_2 (0.13 0.22) (0.45 0.55);
```

If the ITEMS include the raw measurements (states), e.g., to list a sample of measurements from individuals, then the other items must precede the listing of states. There is no restriction on the number of elements in the listing of states. This example has only one continuous character:

```
FORMAT DATATYPE=CONTINUOUS
 ITEMS=(AVERAGE STATES)
 STATESFORMAT=INDIVIDUALS;
MATRIX
 taxon_1 (1.2 2.1 1.6 0.8 1.8 0.3
  0.6)
 taxon_2 (1.6 2.2 1.7 1.0 2.0 1.6
  1.9 0.8);
```

in which the first value is the sample average and the subsequent values comprise the sample of observed states.

Possible ITEMS to be included are MIN (minimum), MAX (maximum), AVERAGE (sample average), VARIANCE (sample variance), STDERROR (standard error), MEDIAN (sample median), SAMPLESIZE, and STATES. The manner of presentations of states can be indicated using the STATESFORMAT command. The default ITEMS for continuous data is AVERAGE.

UNALIGNED Block

The UNALIGNED block includes data that are not aligned. Its primary intent is to house unaligned DNA, RNA, NUCLEOTIDE, and PROTEIN sequence data. Taxa are usually not defined in an UNALIGNED block; if not, this block must be preceded by a block that defines taxon labels and ordering (e.g., TAXA).

Syntax of the UNALIGNED block is as follows:

```
BEGIN UNALIGNED;
 [DIMENSIONS NEWTAXA NTAX=number-
   of-taxa;]
 [FORMAT
  [DATATYPE={STANDARD | DNA | RNA |
   NUCLEOTIDE | PROTEIN}]
  [RESPECTCASE]
  [MISSING=symbol]
  [SYMBOLS="symbol [symbol...]"]
  [EQUATE="symbol=entry
```

```
    [symbol=entry...]"]
  [[NO]LABELS];]
 [TAXLABELS taxon-name [taxon-name
  ...];]
 MATRIX data-matrix;
END;
```

Commands must appear in the order listed. Only one of each command is allowed per block.

The DIMENSIONS command should only be included if new taxa are being defined in this block, which is discouraged (see discussion under DATA block). The format for the DIMENSIONS command is as in the CHARACTERS block, except NCHAR is not allowed.

Subcommands of the FORMAT command are described in the CHARACTERS block.

The TAXLABELS command serves to define taxa and is only allowed if the NEWTAXA token is included in the DIMENSIONS statement. It follows the same form as described in the TAXA block.

Here is an example of an UNALIGNED block:

```
BEGIN UNALIGNED;
 FORMAT DATATYPE=DNA;
 MATRIX
  taxon_1 ACTAGGACTAGATCAAGTT,
  taxon_2 ACCAGGACTAGCGGATCAAG,
  taxon_3 ACCAGGACTAGATCAAG,
  taxon_4 AGCCAGGACTAGTTC,
  taxon_5 ATCAGGACTAGATCAAGTTC;
END;
```

A comma must be placed at the end of each sequence (except the last, which requires a semicolon). Each sequence can occupy more than one line.

DISTANCES Block

This block contains distance matrices. Taxa are usually not defined in a DISTANCES block; if they are not, this block must be preceded by a block that defines taxon labels and ordering (e.g., TAXA).

The syntax of the block is as follows:

```
BEGIN DISTANCES;
 [DIMENSIONS [NEWTAXA]
  NTAX=number-of-taxa
  NCHAR=number-of-characters;]
 [FORMAT
  [TRIANGLE={LOWER | UPPER | BOTH}]
  [[NO]DIAGONAL]
  [[NO]LABELS]
  [MISSING=SYMBOL]
  [INTERLEAVE]
  ;]
 [TAXLABELS taxon-name [taxon-name
  ...];]
 MATRIX distance-matrix ;
END;
```

Commands must appear in the order listed. Only one of each command is allowed per block.

DIMENSIONS.—The NTAX subcommand of this command is needed to process the matrix when some defined taxa are omitted from the distance matrix. The NCHAR subcommand is optional and can be used to indicate the number of characters for those analyses that need to know how many characters (if any) were used to calculate the distances. NCHAR is not required for successful reading of the matrix.

As for the CHARACTERS and UNALIGNED block, taxa can be defined in a DISTANCES block if NEWTAXA precedes the NTAXA subcommand in the DIMENSIONS command. It is advised that new taxa not be defined in a DISTANCES block, for the reasons discussed in the description of the DATA block. NEWTAXA, if present, must be appear before the NTAX subcommand.

FORMAT.—This command specifies the formatting of the MATRIX. The [NO]LABELS and MISSING subcommands are as described in the CHARACTERS block.

1. TRIANGLE={LOWER | UPPER | BOTH}. This subcommand specifies whether only the lower left half of the matrix is present, or only the upper right, or both halves. Below is one example of an upper triangular matrix and one of a matrix with both halves included.

```
BEGIN DISTANCES;
 FORMAT TRIANGLE=UPPER;
 MATRIX
  taxon_1 0.0 1.0 2.0 4.0  7.0
  taxon_2     0.0 3.0 5.0  8.0
  taxon_3         0.0 6.0  9.0
  taxon_4             0.0 10.0
```

```
  taxon_5                    0.0;
END;

BEGIN DISTANCES;
 FORMAT TRIANGLE=BOTH;
 MATRIX
  taxon_1    0 1.0 2.0  4.0  7.0
  taxon_2  1.0   0 3.0  5.0  8.0
  taxon_3  2.0 3.0   0  6.0  9.0
  taxon_4  4.0 5.0 6.0    0 10.0
  taxon_5  7.0 8.0 9.0 10.0    0;
END;
```

2. DIAGONAL. If DIAGONAL is turned off, the diagonal elements are not included:

```
FORMAT NODIAGONAL;
MATRIX
 taxon_1
 taxon_2 1.0
 taxon_3 2.0 3.0
 taxon_4 4.0 5.0 6.0
 taxon_5 7.0 8.0 9.0 10.0;
```

If TRIANGLE is not BOTH and DIAGONAL is turned off, then there will be one row that contains only the name of a taxon. This row is required. If TRIANGLE=BOTH, then the diagonal must be included.

3. INTERLEAVE. As in the CHARACTERS block, this subcommand indicates sections in the matrix, although interleaved matrices take a slightly different form for distance matrices:

```
taxon_1   0
taxon_2   1  0
taxon_3   2  3  0
taxon_4   4  5  6
taxon_5   7  8  9
taxon_6  11 12 13

taxon_4   0
taxon_5 10  0
taxon_6 14 15  0;
```

As in the CHARACTERS block, newline characters in interleaved matrices are significant, in that they indicate a switch to a new taxon.

TAXLABELS.—This command allows specification of the names and ordering of the taxa. It serves to define taxa and is allowed only if the NEWTAXA token is included in the DIMENSIONS statement.

MATRIX.—This command contains the distance data.

DATA Block

This block is equivalent to a CHARACTERS block in which the NEWTAXA subcommand is included in the DIMENSIONS command. That is, the DATA block is a CHARACTERS block that includes not only the definition of characters but also the definition of taxa.

In versions 3.0–3.07 of MacClade and 3.0–3.1.1 of PAUP, data were generally saved in a DATA block, not in TAXA and CHARACTERS blocks. However, as the NEXUS format developed, it became apparent that a number of programs would be written that would not use any of the information in the DATA block except the taxon ordering and labels. For example, a program might be interested only in trees and need only the taxon labels and TREES block (this is the case for COMPONENT). Forcing such a program to know enough about the many formats of the MATRIX command of the DATA block to extract information about taxon labels is excessive when the information can be stored in a separate, easy-to-read TAXA block. Another program might only be interested in a distance matrix, making a file with DNA sequence data and distance data much easier to process if the taxon labels were not buried in a DATA block of DNA data.

The DATA block and the ability to define new taxa in CHARACTERS, UNALIGNED, and DISTANCES blocks have been retained in the NEXUS format because there are circumstances in which it will be much easier to manually edit and maintain files with taxa definitions and data housed in the same matrix. One disadvantage of the placement of taxon labels into a separate block is the increased chance that manual editing of the file will lead to conflicts between the taxon labels defined in the TAXA block and those used in other blocks (see TAXA block).

We strongly recommend against use of a DATA block or definition of taxa in a CHARACTERS, UNALIGNED, or DISTANCES block because this will prevent some pro-

grams from extracting taxon information from the file.

The following DATA block serves to define and order five taxa and contains descriptions of sequence data for the taxa:

```
BEGIN DATA;
 DIMENSIONS NTAX=5 NCHAR=20;
 FORMAT DATATYPE=DNA GAP=-;
 MATRIX
  taxon_1 A-CTAGGACTA---GATCAA
  taxon_2 A-CCAGGACTAGCGGATCAA
  taxon_3 A-CCAGGACTA---GATCAA
  taxon_4 AGCCAGGACTA---GTTCAA
  taxon_5 ATC-AGGACTA---GATCAA;
END;
```

CODONS Block

The CODONS block contains information about the genetic code, the regions of DNA and RNA sequences that are protein coding, and the location of triplets coding for amino acids in nucleotide sequences.

```
BEGIN CODONS;
 [CODONPOSSET [*] name [({STANDARD |
  VECTOR})]=
  N: character-set,
  1: character-set,
  2: character-set,
  3: character-set;]
 [GENETICCODE code-name
  [([CODEORDER=132 | other]
  [NUCORDER=TCAG | other]
  [[NO]TOKENS]
  [EXTENSIONS="symbols-list"])]
  =genetic code description];]
 [CODESET [*] codeset-name
  {(CHARACTERS | UNALIGNED | TAXA)}=
  code-name:character-set or
    taxon-set
  [,code-name:character-set or
    taxon-set...];]
END;
```

GENETICCODE must precede any CODESET that refers to it. There are several predefined genetic codes:

```
UNIVERSAL        [universal]
UNIVERSAL.EXT    [universal,
                  extended]
MTDNA.DROS       [Drosophila mtDNA]
MTDNA.DROS.EXT   [Drosophila
                  mtDNA,extended]
MTDNA.MAM        [Mammalian mtDNA]
MTDNA.MAM.EXT    [Mammalian mtDNA,
                  extended]
MTDNA.YEAST      [Yeast mtDNA]
```

For a summary of the genetic codes, see Osawa et al. (1992). Extended codes are those in which "extra" amino acids have been added to avoid disjunct amino acids (see the EXTENSIONS subcommand under GENETICCODE).

CODONPOSSET.—This command stores information about protein-coding regions and the codon positions of nucleotide bases in protein-coding regions and follows the format of a standard object definition command.

Those characters designated as 1, 2, or 3 are coding bases specified as being of positions 1, 2, and 3, respectively. Those characters designated as N are considered non-protein-coding. Those characters designated as ? are of unknown nature. Any unspecified bases are considered of unknown nature (equivalent to ?). If no CODONPOSSET statement is present, all bases are presumed of unknown nature.

For example, the following command

```
CODONPOSSET * coding=
 N:1-10,
 1:11-.\3,
 2:12-.\3,
 3:13-.\3;
```

designates bases 1–10 as noncoding and positions the remaining bases in the order 123123123. . . .

GENETICCODE.—GENETICCODE stores information about a user-defined genetic code. Multiple GENETICCODES may be defined in the block. This is a standard object definition command except that the default genetic code is not indicated by an asterisk after GENETICCODE.

The genetic code description is a listing of amino acids. By default, the first amino acid listed is that coded for by the triplet TTT, and the last amino acid listed is that coded for by GGG. In between, the order of triplets follows a pattern controlled by

the subcommands NUCORDER and CODEORDER. By default, the amino acids are listed in the following order: TTT, TCT, TAT, TGT, TTC, TCC, TAC, TGC, and so on. The universal genetic code can thus be written

```
GENETICCODE UNTITLED=
 F S Y C
 F S Y C
 L S * *
 L S * W

 L P H R
 L P H R
 L P Q R
 L P Q R

 I T N S
 I T N S
 I T K R
 M T K R

 V A D G
 V A D G
 V A E G
 V A E G
 ;
```

This assigns TTT to phenylalanine, TCT to serine, TAT to tyrosine, TGT to cysteine, and so on.

The following subcommands are included.

1. CODEORDER. The default CODEORDER is 231, indicating that the second codon nucleotide changes most quickly (i.e., codons represented by adjacent amino acids in the listing always differ at second positions), the third nucleotide changes next most quickly, and the first nucleotide changes most slowly in the list (such that the codons representing the first 16 listed amino acids all have the same first nucleotide).

2. NUCORDER. The default NUCORDER is TCAG, indicating that the codons with T at a given position are listed first, C next, etc. For example, if CODEORDER is 123 and the NUCORDER is ACGT, then the amino acids would be listed in order to correspond to codons in the order AAA CAA GAA TAA ACA CCA GCA TCA, etc.

3. [NO]TOKENS. If TOKENS, then amino acids are to be listed by their standard three-letter abbreviations for amino acids (e.g., Leu, Glu). A termination codon is designated by Ter or Stp. If NOTOKENS (the default), then the IUPAC symbols are used in the listing. A termination codon is designated by an asterisk.

4. EXTENSIONS. This command lists the symbols for "extra" amino acids that are added to avoid disjunct amino acids. For example, serines in the universal code are coded for by two distinct groups of codons; one cannot change between these groups without going through a different amino acid. Serine is therefore disjunct. In the UNIVERSAL code, serine is kept disjunct, with all serines symbolized by S. In the UNIVERSAL.EXT code, however, one serine group is identified by the extra amino acid symbol 1 and the other group is identified by 2. To indicate this, the EXTENSIONS subcommand would read EXTENSIONS="S S", indicating that "extra" amino acids 1 and 2 are both serines. In the genetic code description, the symbols 1 and 2 would then both stand for serine. For example, the universal extended genetic code could be represented by

```
GENETICCODE * universal
 (NUCORDER=TCAG CODEORDER=213
 EXTENSIONS="S S")=
 F 1 Y C L P H R I T N 2 V A D G
 F 1 Y C L P H R I T N 2 V A D G
 L 1 * * L P Q R I T K R V A E G
 L 1 * W L P Q R M T K R V A E G;
```

(Note that the CODEORDER has been changed in this example.)

CODESET.—This object definition assigns genetic codes to various characters and taxa. All nucleotide sites are designated as coding, and all amino acid sites have a genetic code assigned to them. If the CHARACTERS format is used, then character-sets are to be used in the description, and the genetic code is thus applied to the characters listed. For example,

```
CODESET oddcodeset=customcode:
 4-99;
```

designates the genetic code "custom code" as applying to characters 4–99 for all taxa.

If UNALIGNED, then genetic code is pre-

sumed to apply to all sites in an UNALIGNED block. Such a CODESET command might look like this:

```
CODESET mycodeset
 (UNALIGNED)=universal:ALL;
```

For UNALIGNED, the only character-set that can be used is ALL.

If the TAXA format is used, then taxon-sets are used in the description. Thus, different genetic codes can be assigned to different taxa for all characters. Current programs do not accept any character-set other than ALL nor any taxon-set other than ALL.

SETS Block

This block stores sets of objects (characters, states, taxa, etc.). The general structure of the SETS block is as follows.

```
BEGIN SETS;
 [CHARSET charset-name [({STANDARD |
   VECTOR})]=character-set;]
 [STATESET stateset-name
   [({STANDARD | VECTOR})]=
   state-set;]
 [CHANGESET changeset-name=
   state-set<->state-set
   [state-set<->state-set...]
   ;]
 [TAXSET taxset-name [({STANDARD |
   VECTOR})]=taxon-set;]
 [TREESET treeset-name
   [({STANDARD | VECTOR})]=
   tree-set;]
 [CHARPARTITION partition-name
   [([{[NO]TOKENS}]
   [{STANDARD | VECTOR}])]
   =subset-name:character-set
   [, subset-name:character-set
    ...]
   ;]
 [TAXPARTITION partition-name
   [([{[NO]TOKENS}]
   [{STANDARD | VECTOR}])]
   =subset-name:taxon-set
   [, subset-name:taxon-set...]
   ;]
 [TREEPARTITION partition-name
   [([{[NO]TOKENS}]
   [{STANDARD | VECTOR}])]
   =subset-name:tree-set
   [, subset-name:tree-set...]
   ;]
END;
```

(See object definition command in the Appendix for information about STANDARD versus VECTOR formats.)

An example SETS block is

```
BEGIN SETS;
 CHARSET larval=1-3 5-8;
 STATESET eyeless=0;
 STATESET eyed=1 2 3;
 CHANGESET eyeloss=eyed ->
   eyeless;
 TAXSET outgroup=1-4;
 TREESET AfrNZVicariance=3 5 9-12;
 CHARPARTITION bodyparts=head:1-4
   7, body:5 6, legs:8-10;
END;
```

CHARSET.—This command specifies and names a set of characters; this name can then be used in subsequent CHARSET definitions or wherever a character-set is required. The VECTOR format consists of 0's and 1's: a 1 indicates that the character is to be included in the CHARSET; whitespace is not necessary between 0's and 1's. The name of a CHARSET cannot be equivalent to a character name or character number.

The character-set CONSTANT is predefined for all DATATYPES; it specifies all invariant characters. The character-set REMAINDER is predefined for all DATATYPES; it specifies all characters not previously referenced in the command. The character-set GAPPED is predefined for all DATATYPES; it specifies all characters with a gap for at least one taxon.

There are four additional predefined character-sets for characters of DATATYPE=DNA, RNA, and NUCLEOTIDE:

1. POS1. All characters defined by current CODONPOSSET as first positions.
2. POS2. All characters defined by current CODONPOSSET as second positions.
3. POS3. All characters defined by current CODONPOSSET as third positions.
4. NONCODING. All characters defined by current CODONPOSSET as non-protein-coding sites.

STATESET.—This command allows one to name a set of states; it is not currently supported by any program. It is not available for DATATYPE=CONTINUOUS.

For STANDARD format, the state-set is described by a list of state symbols, except that it should not be enclosed in parentheses or braces. Any current state-set symbols are valid in the state-set description. The following STATESET

```
STATESET theSet=2 3 4 5;
```

defines the set composed of states 2, 3, 4, and 5.

The VECTOR format consists of 0's and 1's: a 1 indicates that the state is to be included in the STATESET; whitespace is not necessary between 0's and 1's. For example, the state-set

```
STATESET theSet (VECTOR)=1001000;
```

designates theSet to be the set containing first and fourth states.

CHANGESET.—This command allows naming of a set of state changes; it is not currently supported by any program. It is not available for DATATYPE=CONTINUOUS.

The description of the CHANGESET consists of pairs of state-sets joined by an operator. State-sets that consist of more than one token must be contained in parentheses. There are two allowed operators: -> and <-> (<- is not allowed). These operators can best be explained by example.

```
CHANGESET changes1=(1 2 3)->(4 6);
CHANGESET changes2=1<->4;
CHANGESET transversions=(A G)<->
 (C T);
```

The first CHANGESET represents any change from 1 to 4, 1 to 6, 2 to 4, 2 to 6, 3 to 4, or 3 to 6, and the second set represents changes from 1 to 4 and 4 to 1. The CHANGESET "transversions" defines the set of all changes between purines and pyrimidines as transversions.

TAXSET.—This command defines a set of taxa. A TAXSET name can be used in subsequent TAXSET definitions or wherever a taxon-set is required. The name of a TAXSET cannot be equivalent to a taxon name or taxon number. The taxa to be included are described in a taxon-set. For example, the following command

```
TAXSET beetles=Omma-.;
```

defines the TAXSET "beetles" to include all taxa from the taxon Omma to the last defined taxon.

The VECTOR format consists of 0's and 1's: a 1 indicates that the taxon is to be included in the TAXSET; whitespace is not necessary between 0's and 1's.

TREESET.—This command defines a set of trees. A TREESET name can be used in subsequent TREESET definitions or wherever a tree-set is required. It is not currently supported by any program. It follows the same general format as a TAXSET command.

CHARPARTITION, TAXPARTITION, TREEPARTITION.—These commands define partitions of characters, taxa, and trees, respectively. The partition divides the objects into several (mutually exclusive) subsets. They all follow the same format.

There are several formatting options. The VECTOR format consists of a list of partition names. By default, the name of each subset is a NEXUS word (this is the TOKENS option). The NOTOKENS option is only available in the VECTOR format; this allows use of single symbols for the subset names. Each value in a definition in VECTOR format must be separated by whitespace if the names are tokens but not if they are NOTOKENS. The following two examples are equivalent:

```
TAXPARTITION populations=1:1-3,
 2:4-6, 3:7 8;
TAXPARTITION populations (VECTOR
 NOTOKENS)=11122233;
```

The following two examples are equivalent:

```
TAXPARTITION mypartition=
 Chiricahua: 1-3,
 Huachuca: 4-6, Galiuro: 7 8;
TAXPARTITION mypartition (VECTOR)=
 Chiricahua Chiricahua
 Chiricahua Huachuca Huachuca
 Huachuca Galiuro Galiuro;
```

ASSUMPTIONS Block

The ASSUMPTIONS block houses assumptions about the data or gives general directions as to how to treat them (e.g., which characters are to be excluded from consideration). The commands currently placed in this block were primarily designed for parsimony analysis. More commands, embodying assumptions useful in distance, maximum likelihood, and other sorts of analyses, will be developed in the future. For example, matrices specifying relative rates of character state change, useful for both distance and likelihood analyses, will eventually be included here.

The general structure of the assumptions block is

```
BEGIN ASSUMPTIONS;
 [OPTIONS [DEFTYPE=type-name]
  [POLYTCOUNT={MINSTEPS | MAXSTEPS}]
  [GAPMODE={MISSING | NEWSTATE};]
 [USERTYPE type-name[({STEPMATRIX |
  CSTREE})]=USERTYPE-description;]
 [TYPESET [*] typeset-name
  [({STANDARD | VECTOR})]=TYPESET-
  definition;]
 [WTSET [*] wtset-name [({STANDARD |
  VECTOR} {TOKENS | NOTOKENS})]=
  WTSET-definition;]
 [EXSET [*] exset-name [({STANDARD |
  VECTOR})]=character-set;]
 [ANCSTATES [*] ancstates-name
  [({STANDARD | VECTOR}{TOKENS | NO
  TOKENS})]=ANCSTATES-definition;]
END;
```

An example ASSUMPTIONS block follows:

```
BEGIN ASSUMPTIONS;
 OPTIONS DEFTYPE=ORD;
 USERTYPE myOrd=4
   0 1 2 3
   . 1 2 3
   1 . 1 2
   2 1 . 1
   3 2 1 .;
 USERTYPE myTree (CSTREE)=((0,1)a,
   (2,3)b)c;
 TYPESET * mixed=IRREV: 1 3 10,
   UNORD 5-7;
 WTSET * one=2: 1-3 6 11-15, 3: 7 8;
 WTSET two=2:4 9, 3:1-3 5;
 EXSET nolarval=1-9;
 ANCSTATES mixed=0: 1 3 5-8 11,
   1: 2 4 9-15;
END;
```

USERTYPES must be defined before they are referred to in any TYPESET.

In earlier versions of MacClade and PAUP, TAXSET and CHARSET also appeared in the ASSUMPTIONS block. These now appear in the SETS block. There are a number of other commands in the ASSUMPTIONS block that also have SET in their name (e.g., WTSET, EXSET), but these commands assign values to objects, they do not define sets of objects, and therefore they do not belong in the SETS block. (Commands such as WTSET and EXSET are so named for historical reasons; although they might ideally be renamed CHARACTERWEIGHTS and EXCLUDEDCHARACTERS, doing so would cause existing programs to be incompatible with the file format.) We recommend that programs also accept TAXSET and CHARSET commands in the ASSUMPTIONS block so that older files can be read. In addition, the GAPMODE subcommand of the OPTIONS command of this block was originally housed in an OPTIONS command in the DATA block. Because this subcommand dictates how data are to be treated rather than providing details about the data themselves, it was moved into the ASSUMPTIONS block.

OPTIONS.—This command houses a number of disparate subcommands. They are all of the form subcommand=option.

1. DEFTYPE. This subcommand specifies the default character type for parsimony analyses. Whenever a character's type is not explicitly stated, its type is taken to be the default type. Default DEFTYPE is UNORD (see the Appendix, character transformation type, for a definition of UNORD).

2. POLYTCOUNT. Setting POLYTCOUNT to MINSTEPS specifies that trees with polytomies are to be counted (in parsimony analyses) in such a way that the number of steps for each character is the minimum number of steps for that character over any resolution of the polytomy. A tree length that is the sum of these minimum numbers

of steps may be below the tree length of the most-parsimonious dichotomous resolution. Setting POLYTCOUNT to MAXSTEPS specifies that trees with polytomies are to be counted in such a way that occurrence of derived states on elements of a polytomy are to be counted as independent derivations. Such a tree length may be above the tree length of any fully dichotomous resolution. The NEXUS format does not specify a default value for POLYTCOUNT; the default value may differ from program to program.

3. GAPMODE. This subcommand specifies how gaps are to be treated. GAPMODE=MISSING specifies that gaps are to be treated in the same way as missing data; GAPMODE=NEWSTATE specifies that gaps are to be treated as an additional state (for DNA/RNA/NUCLEOTIDE data, as a fifth base).

USERTYPE.—This command defines a character transformation type, as used in parsimony analysis to designate the cost of changes between states. There are several predefined character types (see character transformation type in the Appendix); USERTYPE allows additional character types to be created.

USERTYPE is an object definition command with the exception that an asterisk cannot be used to indicate the default type (default type is stated in the OPTIONS command of the ASSUMPTIONS block). The standard defines no limit to the length of the type name, although individual programs might impose restrictions.

STEPMATRIX format is

```
USERTYPE myMatrix (STEPMATRIX)=n
 s s s s
 . k k k
 k . k k
 k k . k
 k k k .;
```

where n is the number of rows and columns in the step matrix, the s's are state symbols, and the k's are the cost for going between states. The n can take any value ≥2. Diagonal elements may be listed as periods. If a change is to be prohibited, then one enters an "i" for infinity. Typically, the state symbols will be in sequence, but they need not be. The following matrices assign values identically:

```
USERTYPE myMatrix (STEPMATRIX)=4
 0 1 2 3
 . 1 5 1
 1 . 5 1
 5 5 . 5
 1 1 5 .;

USERTYPE myMatrix2 (STEPMATRIX)=4
 2 0 3 1
 . 5 5 5
 5 . 1 1
 5 1 . 1
 5 1 1 .;
```

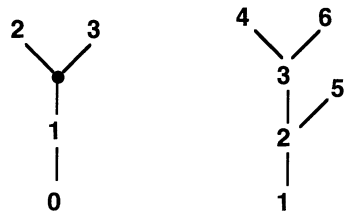
The number of steps may be either integers or real numbers. The range of possible values will differ from program to program. Versions 3.0–3.04 of MacClade use the format name REALMATRIX rather than STEPMATRIX if the matrix contains real numbers. Future programs should treat REALMATRIX as a synonym of STEPMATRIX.

CSTREE format is very similar to the TREE format in a TREES block. That is, character state trees are described in the parenthesis notation following the rules given for TREES of taxa. Instead of taxon labels, character state symbols are used. Thus,

```
USERTYPE cstree-name (CSTREE) =
   [(list-of-subtrees)]
    [state-symbol]];
```

where each subtree has the same format as the overall tree and the subtrees are separated by commas. Two examples are shown in Figure 1.

TYPESET.—This command specifies the type assigned to each character as used in parsimony analysis. This is a standard object definition command. Any characters not listed in the character-set have the default character type. See "object definition command" in the Appendix for information about STANDARD versus VECTOR formats. The type names to be used are either the predefined ones or those defined in a USERTYPE command. Each value in a definition in VECTOR format must be separated




```
USERTYPE  first (CSTREE)=(((2,3))1)0;
USERTYPE  second (CSTREE)=(((4,6)3,5)2)1;
```

FIGURE 1. Two character state trees and the NEXUS commands that define them. The first tree has an unnamed state.

by whitespace. The following are equivalent type sets:

```
TYPESET mytypes=ORD: 1 4 6, UNORD:
 2 3 5;
TYPESET mytypes (VECTOR)=ORD UNORD
 UNORD ORD UNORD ORD;
```

WTSET.—This command specifies the weights of each character. This is a standard object definition command. Any characters not listed in the character-set have weight 1. The weights may be either integers or real numbers. The minimum and maximum weight value will differ from program to program. Each value in a definition in VECTOR format must be separated by whitespace unless the NOTOKENS option is invoked, in which case no whitespace is needed and all weights must be integers in the range 0–9. The following are equivalent weight sets:

```
WTSET mywts=3: 1 4 6, 1: 2 3 5;
WTSET mywts (VECTOR)=3 1 1 3 1 3;
```

In earlier versions of MacClade, the formatting subcommand REAL was used to indicate that real-valued weights were included in the WTSET. This subcommand is no longer in use; programs are expected to detect the presence of integral or real-value weights while reading the WTSET command.

EXSET.—This command specifies which characters are to be excluded from consideration. This is a standard object definition command. Any characters not listed in the character-set are included. The VECTOR format consists of 0's and 1's: a 1 indicates that the character is to be excluded; whitespace is not necessary between 0's and 1's. The following commands are equivalent and serve to exclude characters 5, 6, 7, 8, and 12.

```
EXSET * toExclude=5-8 12;
EXSET * toExclude (VECTOR)=
 000011110001;
```

ANCSTATES.—This command allows specification of ancestral states. This is a standard object definition command. Any valid state symbol can be used in the description for discrete data, and any valid value can be used for continuous data. TOKENS is the default for DATATYPE=CONTINUOUS; NOTOKENS is the default for all other DATATYPES. TOKENS is not allowed for DATATYPES DNA, RNA, and NUCLEOTIDE. If TOKENS is invoked, the standard three-letter amino acid abbreviations can be used with DATATYPE=PROTEIN and defined state names can be used for DATATYPE=STANDARD. NOTOKENS is not allowed for DATATYPE=CONTINUOUS.

The following commands are equivalent:

```
ANCSTATES ancestor=0:1-3 5-7 12,
 1:4 8-10, 2:11;
ANCSTATES ancestor (VECTOR)=
 000100011120;
```

TREES Block

This block stores information about trees. The syntax for the TREES block is

```
BEGIN TREES;
 [TRANSLATE arbitrary-token-used-
  in-tree-description valid-
  taxon-name [, arbitrary-token-
  used-in-tree-description
  valid-taxon-name...];]
 [TREE [*] tree-name=tree-
  specification;]
END;
```

A TRANSLATE command, if present, must precede any TREE command.

TRANSLATE.—The tree description requires references to the taxa defined in a

TAXA, DATA, CHARACTERS, UNALIGNED, or DISTANCES block. These references can be made using the label assigned to them in the TAXA or DATA blocks, their numbers, or a token specified in the TRANSLATE command. The TRANSLATE statement maps arbitrary labels in the tree specification to valid taxon names. If the arbitrary labels are integers, they are mapped onto the valid taxon names as dictated by the TRANSLATE command without any consideration of the order of the taxa in the matrix. Thus, if an integer is encountered in the tree description, a program first checks to see if it matches one of the arbitrary labels defined in the TRANSLATE command; only if no matching label is found will the integer be presumed to refer to the taxon in that position in the matrix (e.g., if the label in the description is 15, but this is not a label defined in the TRANSLATE command, a program should take this to refer to the 15th taxon).

In the following example,

```
BEGIN TAXA;
 TAXLABELS Scarabaeus Drosophila
   Aranaeus;
END;

BEGIN TREES;
 TRANSLATE beetle Scarabaeus, fly
   Drosophila, spider Aranaeus;
 TREE tree1=((1,2),3);
 TREE tree2=((beetle,fly),
   spider);
 TREE tree3=((Scarabaeus,
   Drosophila),Aranaeus);
END;
```

the TRANSLATE command specifies that the label "beetle" can be used in the tree description to refer to *Scarabaeus,* "fly" to *Drosophila,* and "spider" to *Aranaeus.* This means that *Scarabaeus* can be referred to in a tree description as 1, *Scarabaeus,* or beetle. Thus, the three trees are identical.

TREE.—This command describes a tree. Tree descriptions are standard object definition commands. They use the familiar parenthesis notation, with node names, branch lengths, and comments following the established Newick tree standard (see Felsenstein, 1993).

A tree description consists of nested descriptions of clades. Thus, ((A, B),(C, D)) indicates that the tree includes two clades (or subtrees), (A, B) and (C, D), each of which consists of two terminal taxa. Formally, a description of a clade consists of a label for the node that is its most recent common ancestor (if that node has a label) preceded by a list of descriptions of the clade's subclades (if the clade has subclades) and followed by the length of the branch below the node (optional) and any extra information about the node. Thus, a clade is described as follows:

```
[(list of subclades)][label of
node][:length of branch below
node][other node information]
```

The *list of subclades* is enclosed in parentheses, and the subclades are separated by commas:

```
(description of first subclade [,
..., description of last sub-
clade])
```

Because the descriptions of subclades would include lists of their subclades, the familar nested parenthesis notation results. (An alternative way to describe this notation is to say that the parentheses and label describe a node, and within the parentheses are all the nodes immediately descendant from it.) Terminal nodes, of course, do not have subclade lists. Thus, the description of a terminal node consists only of its label and any additional information such as the length of the branch below the node.

The *label of the node* is a NEXUS token that is a taxon's defined name, a taxon's number, a taxon's label from the translation table, or a clade's defined name. The label is optional for internal nodes that are not observed taxa; it is not optional for terminal nodes. Internal nodes that have no label are represented implicitly by the parentheses containing the list of subclades. If the name of a TAXSET is used, it is interpreted as a list of the terminal taxa de-

fined to be in the TAXSET (with commas implicitly inserted between the taxa).

The *length of the branch* below the node is a number, positive or negative.

Rooted and unrooted trees can be specified using the [&R] and [&U] comments at the start of the tree description. For example,

```
TREE mytree=[&R] ((1,2),(3,4));
```

is a rooted tree, whereas

```
TREE mytree=[&U] ((1,2),(3,4));
```

is an unrooted tree. The NEXUS standard does not specify whether rooted or unrooted is default.

An example tree with branch lengths is

```
TREE tree4=((beetle:4.3,fly:1.1)
 :1.8,spider:2.5);
```

If a file (and its data matrix) has four defined taxa, Crocodile, Bluebird, *Archaeopteryx*, and Rattlesnake, the following tree,

```
TREE tree4=(((Bluebird)
 Archaeopteryx,Crocodile)
 Archosauria,Rattlesnake);
```

would indicate that the taxon *Archaeopteryx* is ancestral to Bluebird and that Crocodile is their sister. Archosauria, because it does not refer to a taxon that has been defined in a TAXA or DATA block, is interpreted as the name of the clade including *Archaeopteryx*, Bluebird, and Crocodile.

Any additional information about a clade, its ancestral node, or the branch below it is to be placed in NEXUS comment commands associated with the node. Although different programs may choose their own conventions for how to embed information in comments, the comments that begin with &N are reserved for future NEXUS comment commands.

The NEXUS standard places no restrictions on the number of taxa contained in each tree.

The observance of the Newick tree standard for the TREE command causes some elements of this command to differ from that of more standard NEXUS commands. For example, the use of [&U] and [&R] to designate unrooted and rooted trees is not the natural way of specifying this information in a NEXUS command because this critical information should not be relegated to a command comment. However, in the interests of compatibility with other computer programs that read Newick format trees but do not support other aspects of the NEXUS format, the NEXUS standard follows the Newick format rather than creating a new tree format. In future versions of NEXUS, a more flexible GRAPH command will be present, allowing clearer designation of root position and expression of other aspects not currently available in Newick trees (including reticulations).

NOTES Block

The NOTES block stores notes about various objects in a NEXUS file, including taxa, characters, states, and trees:

```
BEGIN NOTES;
  [TEXT [TAXON=taxon-set] [CHARACTER=
   character-set] [STATE=state-set]
   [TREE=tree-set] SOURCE={INLINE |
   FILE | RESOURCE} TEXT=text-or-
   source-descriptor;]
  [PICTURE [TAXON=taxon-set] [CHARACTER=
   character-set] [STATE=state-set]
   [TREE=tree-set] [FORMAT={PICT |
   TIFF | EPS | JPEG | GIF}] [ENCODE=
   {NONE | UUENCODE | BINHEX}] SOURCE=
   {INLINE | FILE | RESOURCE} PICTURE=
   picture-or-source-
   descriptor;]
END;
```

There are no restrictions on the order of commands.

If the written description of the taxon-set, character-set, state-set, or tree-set contains more than one token, it must be enclosed in parentheses, as in the following example:

```
TEXT TAXON=(1-3) TEXT='these taxa
 from the far north';
```

If both a taxon-set and a character-set are specified, then the text or picture applies to those characters for those particular taxa. If both a character-set and a state-

set are specified, then the text or picture applies to those states for those particular characters.

TEXT.—This command allows text to be attached to various objects.

The SOURCE subcommand indicates the location of the text. The INLINE option indicates that the text is present at the end of the TEXT command; the FILE option indicates that it is in a separate file (the name of which is then specified in the TEXT subcommand); the RESOURCE option indicates that it is in the resource fork of the current file, in a resource of type TEXT (the numerical ID of which is then specified in the TEXT subcommand).

For example, in the following

```
TEXT TAXON=5 CHARACTER=2 TEXT='4
 specimens observed';
TEXT TAXON=Pan TEXT='This genus
 lives in Africa';
TEXT CHARACTER=2 TEXT='Perhaps this
 character should be deleted';
TEXT CHARACTER=1 STATE=0 TEXT='This
 state is hard to detect';
```

the first command assigns the note "4 specimens observed" to the data entry for taxon 5, character 2; the second command assigns the note "Perhaps this character should be deleted" to character 2; the third command assigns the note "This genus lives in Africa" to the taxon *Pan*; and the last command assigns the note "This state is hard to detect" to state 0 of character 1.

The text or source descriptor must be a single NEXUS word. If the text contains NEXUS whitespace or punctuation, it needs to be surrounded by single quotes, with any contained single quotes converted to a pair of single quotes.

PICTURE.— This command allows a picture to be attached to an object.

The FORMAT subcommand allows specification of the graphics format of the image.

The SOURCE subcommand indicates the location of the picture. The INLINE option indicates that the picture is present at the end of the PICTURE command; the FILE option indicates that it is in a separate file (the name of which is then specified in the PICTURE subcommand); the RESOURCE option indicates that it is in the resource fork of the current file, in a resource of type PICT (the numerical ID of which is then specified in the PICTURE command). The RESOURCE option is designed for Apple Macintosh™ text files.

For example, the following command

```
PICTURE TAXON=5 CHARACTER=2
 FORMAT=GIF SOURCE=file
 PICTURE=wide.thorax.gif;
```

assigns the image in the GIF-formatted file wide.thorax.gif to the data entry for taxon 5, character 2.

The picture or source descriptor must be a single NEXUS word. If the picture contains NEXUS whitespace or punctuation, it needs to be surrounded by single quotes, with any contained single quotes converted to a pair of single quotes.

Most graphics formats do not describe pictures using standard text characters. For this reason many images cannot be included INLINE in a NEXUS command unless they are converted into text characters. The ENCODE subcommand specifies the conversion mechanism used for inline images; an example is shown in Figure 2.

FUTURE OF THE NEXUS FORMAT

The NEXUS format will evolve as required by the evolution of computer programs. A number of new elements are being developed. For example, the USERTYPE command in the ASSUMPTIONS block, currently containing assumptions relevant for parsimony analyses, will likely be replaced by a more general command that will include cost matrices, character state trees, probabilistic rate matrices, etc., useful for parsimony, maximum likelihood, and distance methods. There will be new set operators added, a GRAPH format for describing reticulate phylogenies, and the ability to name and refer to separate blocks of characters, among other features. The most recent version of the file format, including additions and proposed additions to the format, is available on the World Wide Web at http://phylogeny.arizona.edu/nexus/nexus.html. This site should be checked by

```
PICTURE TAXON=1 FORMAT=GIF ENCODE=UUENCODE SOURCE=INLINE
      PICTURE=
'begin 644 tree.GIF
M1TE&.#EA3@!*`(```/___P```"P`````3@!*```"_X2/J<OM#Z.<M-J+,PJ\
M!_,I(3.*''J<M)9BLD2NFC;M^*#C:^0$#M7_K;&ZX7:L5,@)9/J8*N>$YD]+F
M,3JM6FV.''W9;I<6^2B.1=.5YBD2A4YN-@WMO*[Q^%9.U9OS="\:D-,97*''CW
M9U+B5H0H\P@9*3E)66EYB9FIN<G9Z?FI20<I&KF822JC=XF:T<.J\5HA&HM!
M^P)A:Y$[([%+X9MVBPGL]T`L?.%[C)L,>RJ;^MP[*FTLN6S=18FMC5;)S4OX
M75U\70V>O6JWB?[TV?[U/@[JZ$P?GP)?^ZC?FG_?K1G`@+\&,BMH\""RA.D(
M,FSH[N$T;Q(GP.C''"5!%:"SE-C+#2`\D*)''R//HSN0^E0)4<62)T:1''F2YD*
M:4*T21%G.)T[>4;T*4YB`0`A_PM-04-''0V]N(`0#$#`````!5W)I=''1E;B!B
E>2!''249#;VYV97)T97(@,BXS+C(@;V8@36%R(#8L(#$Y.3,`.R`@

end';
```

FIGURE 2. A PICTURE command with a UUencoded Compuserve® Graphics Interchange Format (GIF) image. Although it may not appear so, all of the text from "begin" to "end" is a single NEXUS word.

those who are considering adopting or extending the NEXUS format.

ACKNOWLEDGMENTS

We thank Rod Page, Paul Lewis, and Dmitri Zaykin, who contributed to the refinement of the NEXUS file format through discussions about their extensions to the format and who, with David Penny, Torsten Eriksson, Gary Olsen, Sudhir Kumar, and Robert Colwell, provided valuable comments about this manuscript. We also thank Kimberlee Wollter and Allen Press for their careful work in formatting this paper.

REFERENCES

AMERICAN NATIONAL STANDARDS INSTITUTE. 1986. 7-Bit American Standard Code for Information Interchange, Standard ANSI X3.4-1986. American National Standards Institute, New York.

BANDELT, H.-J., AND A. W. M. DRESS. 1992. Split decomposition: A new and useful approach to phylogenetic analysis of distance data. Mol. Phylogenet. Evol. 1:242–252.

DONOGHUE, M. J., T. ERIKSSON, W. PIEL, K. RICE, AND M. SANDERSON. 1996. TreeBASE. A database of phylogenetic knowledge. World Wide Web: http://herbaria.harvard.edu/treebase/

FARRIS, J. S. 1970. Methods for computing Wagner trees. Syst. Zool. 19:83–92.

FELSENSTEIN, J. 1993. PHYLIP: Phylogeny inference package, version 3.5c. Distributed by the author, Department of Genetics, Univ. Washington, Seattle.

FISHER, D. C. 1992. Stratigraphic parsimony. Pages 124–129 *in* MacClade: Analysis of phylogeny and character evolution (W. P. Maddison and D. R. Maddison, eds.). Sinauer, Sunderland, Massachusetts.

FITCH, W. M. 1971. Toward defining the course of evolution: Minimal change for a specific tree topology. Syst. Zool. 20:406–416.

HARTIGAN, J. A. 1973. Minimum mutation fits to a given tree. Biometrics 29:53–65.

HUSON, D. H., AND R. WETZEL. 1994. SplitsTree, version 1.01. Distributed by the authors, Univ. Bielefeld, Bielefeld, Germany.

LEWIS, P. O., AND D. ZAYKIN. 1996. Genetic Data Analysis: Software for the analysis of discrete genetic data. World Wide Web: http://biology.unm.edu/~lewisp/gda.html

MADDISON, W. P., AND D. R. MADDISON. 1992. MacClade: Analysis of phylogeny and character evolution, version 3. Sinauer, Sunderland, Massachusetts.

MADDISON, W. P., AND D. R. MADDISON. 1997. MacClade: Analysis of phylogeny and character evolution, version 3.07. Sinauer, Sunderland, Massachusetts.

OSAWA, S., T. H. JUKES, K. WATANABE, AND A. MUTO. 1992. Recent evidence for the evolution of the genetic code. Microbiol. Rev. 56:229–264.

PAGE, R. D. M. 1993. COMPONENT, version 2.0. The Natural History Museum, London.

SWOFFORD, D. L. 1989. PAUP: Phylogenetic analysis using parsimony, version 3.0. Illinois Natural History Survey, Champaign.

SWOFFORD, D. L. 1993. PAUP: Phylogenetic analysis using parsimony, version 3.1.1. Illinois Natural History Survey, Champaign.

SWOFFORD, D. L., AND W. P. MADDISON. 1987. Reconstructing ancestral character states under Wagner parsimony. Math. Biosci. 87:199–229.

WETZEL, R. 1994. The split decomposition algorithm. Ph.D. Thesis, Univ. Bielefeld, Bielefeld, Germany.

Received 13 June 1995; accepted 1 August 1995
Associate Editor: David Cannatella

APPENDIX
GLOSSARY

Block.—A block is a sequence of NEXUS commands, starting with a BEGIN command and ending with the

next END command. (In MacClade, PAUP, and COMPONENT, the ENDBLOCK command has been used as a synonym of the END command.)

Character number.—This is the number of a character, as defined by its position in a CHARLABELS command or MATRIX in a CHARACTERS or DATA block. For example, the third character listed in a CHARACTERS block is character number 3.

Character-set.—This is a set of characters, described in a *list* (see below). A character-set can be given a formal name using the CHARSET command (see SETS block description).

Character transformation type.—A character type is a specification of the costs and rules imposed on specific state-to-state changes in a parsimony analysis. There are several predefined types: UNORD (unordered), ORD (linearly ordered reversible), IRREV, IRREV.UP, and IRREV.DN (all linearly ordered irreversible), and DOLLO, DOLLO.UP, and DOLLO.DN (variants of Dollo; Swofford, 1989; Maddison and Maddison, 1992). In addition, MacClade has the predefined type STRAT (Fisher's stratigraphic type [Fisher, 1992; Maddison and Maddison, 1992]). UNORD specifies a cost matrix in which all state-to-state changes cost one step; this is the assumption embodied by Fitch parsimony (Fitch, 1971; Hartigan, 1973). ORD specifies a cost of change between two states to be the absolute value of the difference in their state numbers (e.g., a change from state 3 to state 7 costs four steps); this is the assumption in Wagner parsimony (Farris, 1970; Swofford and Maddison, 1987). IRREV and IRREV.UP are equivalent and specify the same cost of change as ORD except that all changes that would involve a decrease in state number are disallowed. IRREV.DN is identical to IRREV except that the changes that are disallowed are those that involve an increase in state number. DOLLO and DOLLO.UP are equivalent and specify the same cost of change as ORD except that each increase in state number can occur only once. DOLLO.DN is identical to DOLLO except that the changes that can only occur once are decreases in state number.

Command.—A command is a collection of tokens terminated by a semicolon. Commands cannot contain semicolons, except as terminators, unless the semicolons are contained within a comment or within a quoted token consisting of more than one text character. If a delimiter between command sections is needed, it should be a comma (for examples, see STATELABELS in the CHARACTERS block and WTSET in the ASSUMPTIONS block).

Command comment.—A comment containing a command is a command comment. Comments containing commands are indicated by a special symbol (\ or &) as the first text character within the comment. Comments whose first text character is \ house commands relating to display of tokens (see Display of Text). Comments whose first text character is & are of the form [&*word* . . .], where *word* is an unquoted NEXUS word. The word contains the command (e.g., [&U] indicating unrooted trees in the TREE command). If the word begins with & (so that the comment itself is of the form "[&&. . .]", then the rest of the word indicates to which processing programs the command applies (e.g., [&&PAUP *command*]). The comments "&&MC" and "&&PAUP" are reserved for MacClade and PAUP, respectively.

In general, the command contained in such a command comment should not affect the nature or interpretation of the NEXUS file elements contained outside of comments. For example, one should not add a command comment to a WTSET that will tell some program to set the weights to be the reciprocal of the weights listed because this information would be ignored by most programs, leading to different interpretations of the same file by different programs. Examples of the information that could be housed in these command comments are the status of a tree search or the positions of windows on screen.

There is one exception to this rule, in the TREE command. For this command, comments are allowed that can actually change the interpretation of the tree. The [&U] and [&R] commands are of this nature (see the TREES block), and similar commands might be invented in the future. This exception is allowed in the TREE command because adding information to a tree description outside of comments would cause the file to be noncompliant with the Newick tree standard, and support of this broader standard is more desirable than strict adherence to the NEXUS standard.

Comment.—Any comments in the data file are delimited by brackets:

```
[please ignore me]
```

Comments are ignored by all programs, with few exceptions. These exceptions can occur if the first text character inside the comment is any of !, &, %, /, \, or @. Comments that begin with an exclamation point are output comments; those beginning with any of these other symbols are command comments.

Comments may contain any combination of dark character and whitespace (which need not be organized into NEXUS tokens). They can extend over multiple lines (i.e., they can contain newline characters). Quotes are ignored within these comments. All text characters are treated equally within such comments, with the exception of open and close brackets. A close bracket will terminate a comment. Comments can be nested:

```
[this is a comment [nested inside]]
```

If a program encounters an open bracket within a comment, it will expect two close brackets to close both the main comment and the nested comment.

Comments do not break tokens. Thus, ASSUMP[comment]TIONS is processed as ASSUMPTIONS.

Left and right brackets are not treated as symbols that delimit a comment if they are found within a quoted token.

Dark characters.—All text characters that are not whitespace are dark characters.

Data block.—A data block is any of the following blocks: CHARACTERS, UNALIGNED, DISTANCES, DATA.

Entry.—An element in a data matrix consisting of the information for one character for one taxon or an element in a distance matrix consisting of the information for one taxon-by-taxon comparison is a matrix

entry. It may consist of a single symbol, e.g., A, if the data are discrete and the taxon shows a single character state. However, it may include a large amount of information, such as the average value from the taxon, the variance, and the sample size for continuous data.

List.—The description of a set of objects (taxa, characters, trees) is a list. Elements in the list are separated by whitespace. If a series of contiguous elements are to be listed, a hyphen can be used in place of the intermediate values. If all elements are to be listed, then the word ALL can be used in place of listing the numbers. The word REMAINDER designates all elements not previously referred to in a command. On each of the following lines is a legal list:

```
1 3
1-4
2 4-8 20-45
ALL
```

A period represents the number of the last element; thus, if there are 320 characters in the matrix, the character-set described by "56-." means characters 56 through 320 inclusive.

If, immediately following a contiguous range of numbers, one places a backslash followed by an integer *n*, this indicates that only every *n*th element, beginning at the first element in the range, is to be included in the set. For example, the character-set 2-12\3 is equivalent to 2 5 8 11, and ALL\2 means every second character, beginning with character 1. This is especially useful for protein-coding nucleotide sequence data for which one might wish to specify, say, every third position.

Names of elements can also be used in the list. For example, if character 3 has been given the name "eye color" and character 5 is "leg length" then

```
'eye color'-'leg length'
```

specifies characters 3, 4, and 5.

For unaligned data, the only valid list of characters is ALL.

If sets of elements have been formally defined by a CHARSET, TAXSET, or TREESET command in the SETS block, then the names of those sets can also be used in a subsequent list. For example, if the CHARSET "larval" has been defined in the SETS block, a subsequent character-set could be "2-5 larval," which would mean characters 2–5 plus all those contained within the CHARSET "larval."

Newline character.—A newline character consists of any of the following: (1) single text character for carriage return (ASCII 13), (2) linefeed (ASCII 10), (3) carriage return followed immediately by a linefeed, or (4) other text characters used on the current operating system to designate a new line in a text file. Options 1–3 are valid for all programs that are reading ASCII text files, no matter what operating system they are running under.

Object.—An object is any of the many entities defined in a NEXUS file that are or could be given a name. The only type of objects that need not be given a name are characters and states; for all others, names are given when they are defined. Objects include taxa, characters, trees, genetic codes, taxon-sets, etc. The NEXUS standard allows for creation of new kinds of objects.

Object definition command.—A number of commands consist of definitions of objects; these definitions follow a standard format:

```
category [*] object-name [(format)]=
 description;
```

where the kind of object is given by *category* and the name of the object is given by the token *object name,* optionally followed by tokens, enclosed in parentheses, describing the format of the following description, an equal sign, and then the description. The standard defines no limit to the length of the name, although individual programs might impose restrictions. If the object is to be considered the default object of its category, then this can be indicated by an asterisk between the category and object name. (This convention is not used for categories in which there are predefined objects, e.g., USERTYPE and GENETICCODE.) If more than one object is so designated, then the last one encountered is taken to be the default.

The following is an example of an object definition command:

```
WTSET * myweights=2:1 3 5;
```

This is equivalent to

```
WTSET * myweights (VECTOR)=2 1 2 1 2;
```

A number of object definitions can be written in standard or vector formats. STANDARD format is a listing of values (weights, types, and so on), each followed by a character-set or taxon-set including those characters or taxa that are to be of that value, or if no such value is needed (e.g., EXSETS), then the format is just a list of taxa or characters.

In the following,

```
WTSET myweights (STANDARD)=2:1 3 5;
```

the weight value 2 is assigned to characters 1, 3, and 5. Different value assignments are separated by commas:

```
WTSET myweights (STANDARD)=2:1 3 5, 0:4;
```

This assigns to the first five characters weights of 2, 1, 2, 0, and 2.

VECTOR definitions consist of a string of values:

```
WTSET myweights (VECTOR)=2 1 2 0 2;
```

The first value in a vector format is assigned to the first object (character, taxon, etc.), as appropriate, the second value to the second object, etc. Unless specific exceptions are made (i.e., for the commands EXSET, CHARSET, STATESET, TAXSET, TREESET) or unless the NOTOKENS formatting option is used, a VECTORed object definition requires whitespace between the values.

Output comments.—If the first character within the brackets is an exclamation mark, then the words that follow are considered output comments. These may optionally be displayed, in some fashion, to the user of the program. For example,

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

PAUP deals with these by rewriting them to the output; MacClade displays them in the About *datafile* or About Trees windows if they are within the TAXA, CHARACTERS, DATA, or TREES blocks, respectively (if they are not in these blocks, MacClade ignores them).

Punctuation.—Any of the following text characters are considered punctuation at some times: ( ) [ ] { } / \ , ; : = * ' " ` + - < > The following punctuation marks have special properties: [ ] do not break a word; + and - are allowed as state symbols, but none of the rest are allowed; - is considered punctuation except were it is the minus sign in a negative number.

State range.—A description of the range of values for a continuously valued character is the state range. If there is a single state, the state range is a single real value, e.g., 0.86. If there are multiple values, the state range is indicated by placing a tilde between the boundary values, e.g., 0.86~1.12.

State-set.—A description of the set of states of a discrete character is the state-set. If there is a single state in the set, the state-set is the state symbol, e.g., 0. In some areas of the file, if there is more than one state in the set, then the state-set consists of the state symbols inside of parentheses or braces, e.g., (012) or (0~3) or {013}. Parentheses indicate that the taxon is polymorphic, whereas braces indicate that the states of the taxon are uncertain. For example, (012) indicates 0 and 1 and 2, whereas {012} indicates 0 or 1 or 2. A tilde indicates all intermediate states are included, according to the ordering of the states as determined in the list of state symbols (see *state symbol*). For molecular sequence data, there are some symbols that are defined to represent state-sets with multiple states. For example, R is the state-set that is equivalent to the state-set {AG} (see *state symbol*).

State symbol.—A state symbol is any valid symbol representing the state of a discrete character. The predefined state symbols for standard data matrices are 0 and 1, those for DNA data are A, C, G, and T, and so on. Whitespace and the following text characters are disallowed as state symbols: ( ) [ ] { } / \ , ; : = * ' " ` < > ~

DATATYPE=STANDARD sets the default acceptable symbols to 0 and 1 (these symbols may be altered using the SYMBOLS subcommand, as described in the specification of the FORMAT command of the CHARACTERS block).

If DATATYPE is set to DNA, the SYMBOLS list is set to ACGT, and the following equate macros (see EQUATES subcommand), corresponding to the IUPAC (IUB) ambiguity codes, are defined:

```
R = {AG}    [puRine]
Y = {CT}    [pYrimidine]
M = {AC}    [aMino]
K = {GT}    [Keto]
S = {CG}    [Strong]
W = {AT}    [Weak]
H = {ACT}   [not G]
B = {CGT}   [not A]
V = {ACG}   [not T]
D = {AGT}   [not C]
N = {ACGT}  [unkNown]
X = {ACGT}  [unknown]
```

Lowercase symbols have the same meaning as their uppercase equivalents.

If DATATYPE=RNA, the SYMBOLS list is set to ACGU and the same set of equate macros are defined, except that U is substituted for T.

If DATATYPE=NUCLEOTIDE, the SYMBOLS list is set to ACGT, with U treated as synonymous with T.

If DATATYPE=PROTEIN, the symbols list is set to ACDEFGHIKLMNPQRSTVWY*, corresponding to the standard IUB single-letter amino acid codes:

```
A = ala  [alanine]
C = cys  [cysteine]
D = asp  [aspartic acid]
E = glu  [glutamic acid]
F = phe  [phenylalanine]
G = gly  [glycine]
H = his  [histidine]
I = ile  [isoleucine]
K = lys  [lysine]
L = leu  [leucine]
M = met  [methionine]
N = asn  [asparagine]
P = pro  [proline]
Q = gln  [glutamine]
R = arg  [arginine]
S = ser  [serine]
T = thr  [threonine]
V = val  [valine]
W = trp  [tryptophan]
Y = tyr  [tyrosine]
* = stop [chain termination]
```

In addition, two equate macros are defined:

```
B = {DN} [asx = asp or asn]
Z = {EQ} [glx = glu or gln]
```

Lowercase symbols have the same meaning as their uppercase equivalents.

Symbol.—A single dark character is a symbol.

Taxon number.—The taxon number is the number of a taxon, as defined by its position in a TAXLABELS command, its position in a MATRIX in a CHARACTERS, UNALIGNED, or DISTANCE block (in which NEWTAXA is invoked in the DIMENSIONS command), or its position in a MATRIX in a DATA block. For example, the third taxon listed in TAXLABELS is taxon number 3.

Taxon-set.—A set of taxa, described in a *list*, is a taxon-set. A taxon-set can be given a formal name using the TAXSET command (see SETS block description).

Token.—A token is a NEXUS word or punctuation.

Tree number.—The number of a tree, as defined by its position in a TREES block, is a tree number.

Tree-set.—A tree-set is a set of TREES, described in a *list*. A tree-set can be given a formal name using the TREESET command (see SETS block description).

Whitespace.—Whitespace consists of any of the following text characters: tab (ASCII 9), newline character (as defined in this glossary), blank (ASCII 32), and ASCII 0–6.

Word.—Except for special cases involving quotes or comments, a NEXUS word is any string of text characters that is bounded by whitespace or punctuation

and that does not contain whitespace or punctuation. If the first character of a word is a single quote, then the word ends with the next single quote (unless that single quote is in a pair of consecutive single quotes; if so, then the word ends at the first unpaired single quote). Any character, including punctuation and whitespace, may be contained within a quoted word. A word cannot consist of only whitespace and punctuation. On each of the following lines is a single legal word:

```
Bembidion
B._zephyrum
'John''s sparrow (eastern)'
```

Underscores are considered equivalent to blank spaces, except that underscores are dark characters and blank spaces are whitespace. Thus, a program encountering B._zephyrum and 'B. zephyrum' should judge them to be identical.

Any doubled single quotes within a quoted word should be converted to single quotes within the program. Thus, 'John''s would be treated internally within a program as John's. Of course, if the program writes out such a word to a NEXUS file, the quote should be redoubled, and quotes should be written around the word: 'John''s.

A string of dark characters is broken into several words by whitespace and any punctuation other than brackets, unless the string is surrounded by quotes. Thus, [\i]Bembidion_velox[\p]_Linnaeus consists of one word, as does '- - A single word', whereas two()words consists of four tokens: the word "two," two punctuation characters, and the word "words."