Chapter 66
# The TREE Procedure

## Chapter Table of Contents

# Chapter 66
# The TREE Procedure

## Overview

The TREE procedure produces a tree diagram, also known as a *dendrogram* or *phenogram*, using a data set created by the CLUSTER or VARCLUS procedure. The CLUSTER and VARCLUS procedures create output data sets that contain the results of hierarchical clustering as a tree structure. The TREE procedure uses the output data set to produce a diagram of the tree structure in the style of Johnson(1967), with the root at the top. Alternatively, the diagram can be oriented horizontally, with the root at the left. Any numeric variable in the output data set can be used to specify the heights of the clusters. PROC TREE can also create an output data set containing a variable to indicate the disjoint clusters at a specified level in the tree.

Tree diagrams are discussed in the context of cluster analysis by Duran and Odell (1974), Hartigan (1975), and Everitt (1980). Knuth (1973) provides a general treatment of tree diagrams in computer programming.

The literature on tree diagrams contains a mixture of botanical and genealogical terminology. The objects that are clustered are *leaves*. The cluster containing all objects is the *root*. A cluster containing at least two objects but not all of them is a *branch*. The general term for leaves, branches, and roots is *node*. If a cluster A is the union of clusters B and C, then A is the *parent* of B and C, and B and C are *children* of A. A leaf is thus a node with no children, and a root is a node with no parent. If every cluster has at most two children, the tree diagram is a *binary tree*. The CLUSTER procedure always produces binary trees. The VARCLUS procedure can produce tree diagrams with clusters that have many children.

## Getting Started

The TREE procedure creates tree diagrams from a SAS data set containing the tree structure. You can create this type of data set with the CLUSTER or VARCLUS procedure.

In the following example, the VARCLUS procedure is used to divide a set of variables into hierarchical clusters and to create the SAS data set containing the tree structure. The TREE procedure then generates the tree diagrams.

The following data, from Hand, et al. (1994), represent the amount of protein consumed from nine food groups for each of 25 European countries. The nine food groups are red meat (RedMeat), white meat (WhiteMeat), eggs (Eggs), milk (Milk), fish (Fish), cereal (Cereal), starch (Starch), nuts (Nuts), and fruits and vegetables (FruVeg).

The following SAS statements create the data set Protein:

```
data Protein;
   input Country $15. RedMeat WhiteMeat Eggs Milk
      Fish Cereal Starch Nuts FruVeg;
   datalines;
Albania        10.1  1.4  0.5   8.9  0.2  42.3  0.6  5.5  1.7
Austria         8.9 14.0  4.3  19.9  2.1  28.0  3.6  1.3  4.3
Belgium        13.5  9.3  4.1  17.5  4.5  26.6  5.7  2.1  4.0
Bulgaria        7.8  6.0  1.6   8.3  1.2  56.7  1.1  3.7  4.2
Czechoslovakia  9.7 11.4  2.8  12.5  2.0  34.3  5.0  1.1  4.0
Denmark        10.6 10.8  3.7  25.0  9.9  21.9  4.8  0.7  2.4
E Germany       8.4 11.6  3.7  11.1  5.4  24.6  6.5  0.8  3.6
Finland         9.5  4.9  2.7  33.7  5.8  26.3  5.1  1.0  1.4
France         18.0  9.9  3.3  19.5  5.7  28.1  4.8  2.4  6.5
Greece         10.2  3.0  2.8  17.6  5.9  41.7  2.2  7.8  6.5
Hungary         5.3 12.4  2.9   9.7  0.3  40.1  4.0  5.4  4.2
Ireland        13.9 10.0  4.7  25.8  2.2  24.0  6.2  1.6  2.9
Italy           9.0  5.1  2.9  13.7  3.4  36.8  2.1  4.3  6.7
Netherlands     9.5 13.6  3.6  23.4  2.5  22.4  4.2  1.8  3.7
Norway          9.4  4.7  2.7  23.3  9.7  23.0  4.6  1.6  2.7
Poland          6.9 10.2  2.7  19.3  3.0  36.1  5.9  2.0  6.6
Portugal        6.2  3.7  1.1   4.9 14.2  27.0  5.9  4.7  7.9
Romania         6.2  6.3  1.5  11.1  1.0  49.6  3.1  5.3  2.8
Spain           7.1  3.4  3.1   8.6  7.0  29.2  5.7  5.9  7.2
Sweden          9.9  7.8  3.5   4.7  7.5  19.5  3.7  1.4  2.0
Switzerland    13.1 10.1  3.1  23.8  2.3  25.6  2.8  2.4  4.9
UK             17.4  5.7  4.7  20.6  4.3  24.3  4.7  3.4  3.3
USSR            9.3  4.6  2.1  16.6  3.0  43.6  6.4  3.4  2.9
W Germany      11.4 12.5  4.1  18.8  3.4  18.6  5.2  1.5  3.8
Yugoslavia      4.4  5.0  1.2   9.5  0.6  55.9  3.0  5.7  3.2
;
run;
```

The data set Protein contains the character variable Country and the nine numeric variables representing the food groups. The $15. in the INPUT statement specifies that the variable Country is a character variable with a length of 15.

The following statements cluster the variables in the data set Protein. The OUTTREE= option creates an output SAS data set named Tree to contain the tree structure. The CENTROID option specifies the centroid clustering method, and the MAXCLUSTERS= option specifies that the largest number of clusters desired is four. The NOPRINT option suppresses the display of the output. The VAR statement specifies that all numeric variables (RedMeat—FruVeg) are used by the procedure.

```
proc varclus data=Protein outtree=Tree
             centroid maxclusters=4 noprint;
   var RedMeat--FruVeg;
run;
```

The output data set Tree, created by the OUTTREE= option in the previous statements, contains the following variables:

| | |
|---|---|
| _NAME_ | the name of the cluster |
| _PARENT_ | the parent of the cluster |
| _NCL_ | the number of clusters |
| _VAREXP_ | the amount of variance explained by the cluster |
| _PROPOR_ | the proportion of variance explained by the clusters at the current level of the tree diagram |
| _MINPRO_ | the minimum proportion of variance explained by a cluster |
| _MAXEIGEN_ | the maximum second eigenvalue of a cluster |

The following statements produce a tree diagram of the clusters created by PROC VARCLUS:

```
proc tree data=tree ;
proc tree data=tree lineprinter;
```

PROC TREE is invoked twice. In the first invocation, the tree diagram is presented using the default high resolution graphical output. In the second invocation, the LINEPRINTER option specifies line printer output.

Figure 66.1 displays the default high resolution graphics version of the tree diagram.

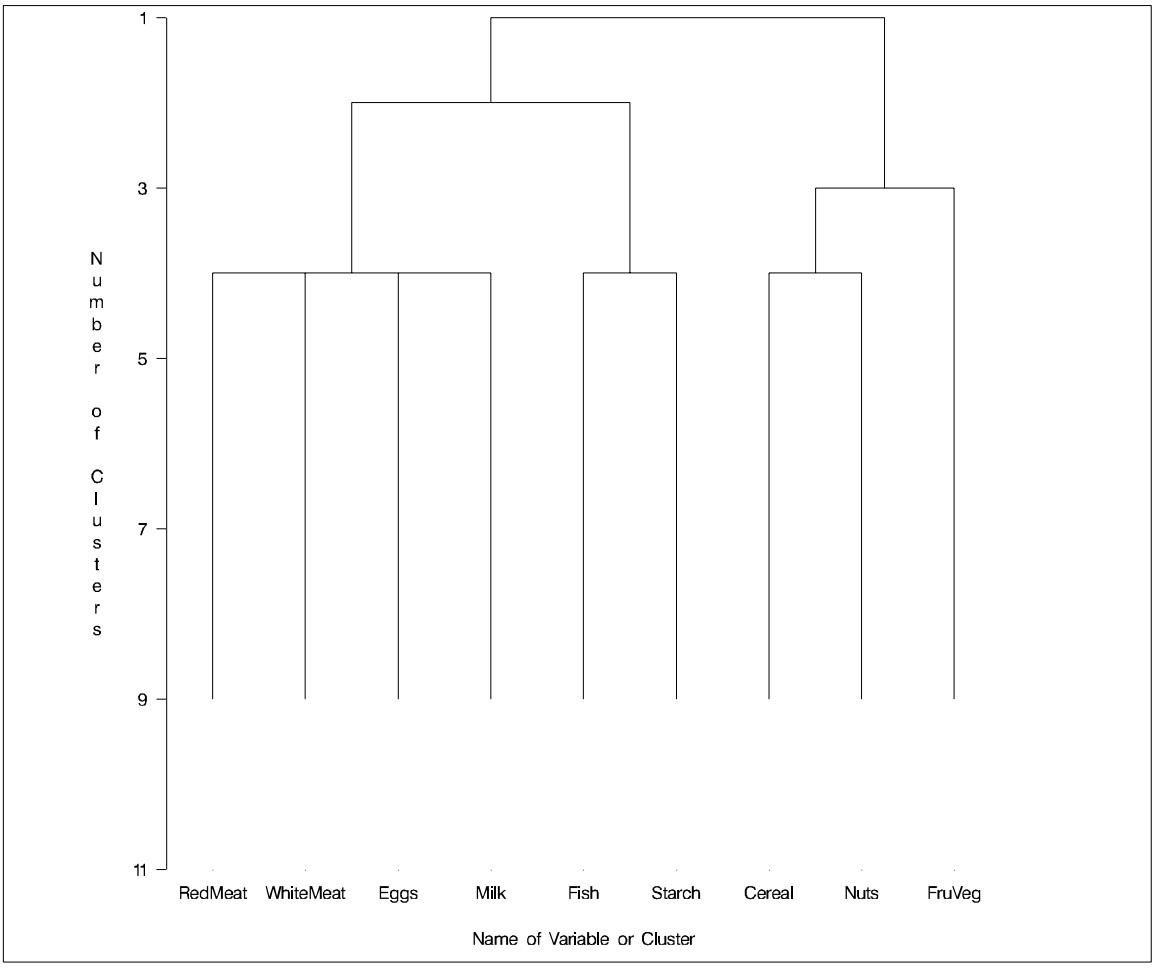



**Figure 66.1.** High Resolution Tree Diagram from PROC TREE

Figure 66.2 displays the same information as Figure 66.1, using line printer output.

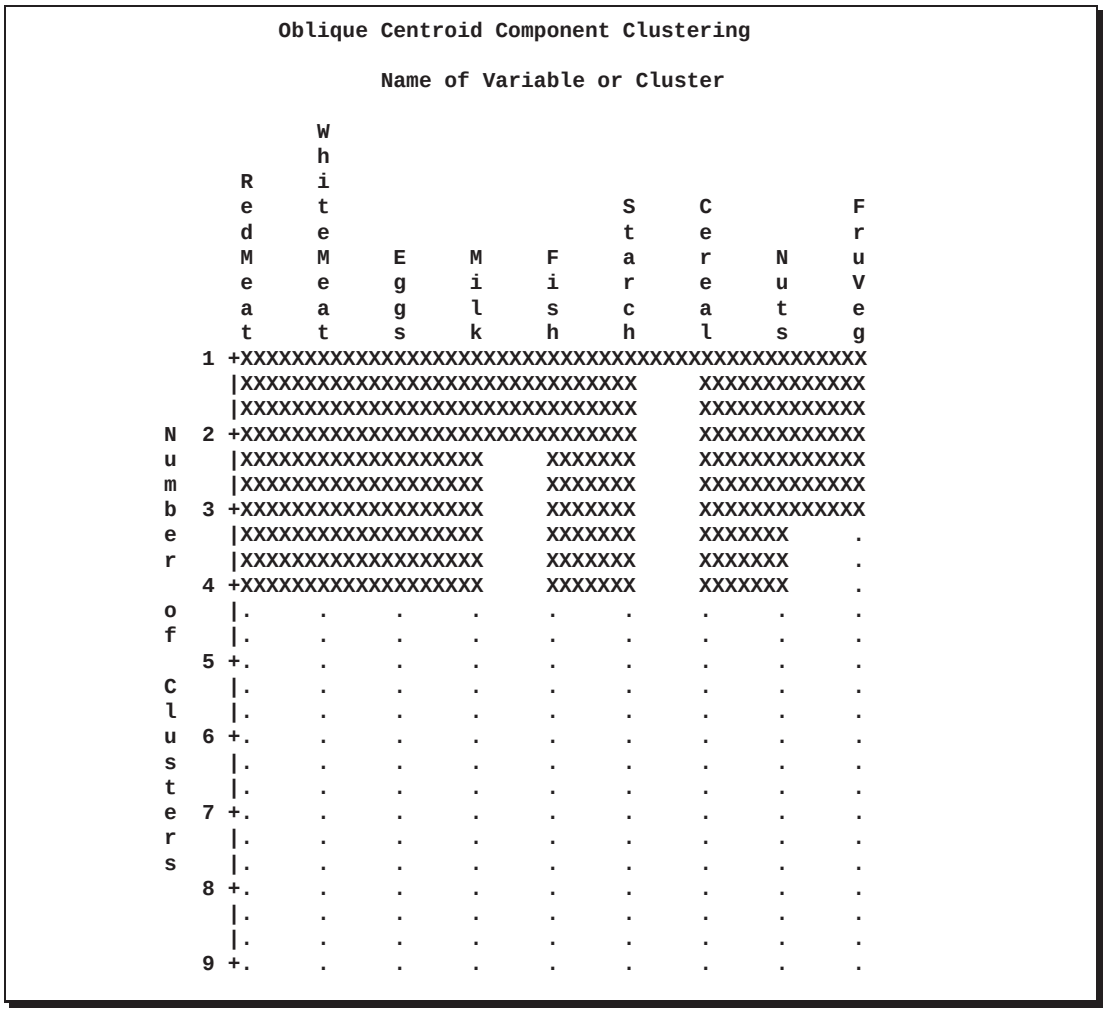



**Figure 66.2.** Line Printer Graphics Version of the Tree Diagram

In both figures, the name of the cluster is displayed on the horizontal axis and the number of clusters is displayed on the vertical or height axis.

As you look up from the bottom of the figures, clusters are progressively joined until a single, all-encompassing cluster is formed at the top (or root) of the diagram. Clusters exist at each level of the diagram. For example, at the level where the diagram indicates three clusters, the clusters are as follows:

- Cluster 1: RedMeat WhiteMeat Eggs Milk
- Cluster 2: Fish Starch
- Cluster 3: Cereal Nuts FruVeg

As you proceed up the diagram one level, the number of clusters is two. The clusters are

- Cluster 1: RedMeat WhiteMeat Eggs Milk Fish Starch
- Cluster 2: Cereal Nuts FruVeg

The following statements illustrate how you can specify the numeric variable defining the height of each node (cluster) in the tree. First, the AXIS1 statement is defined. The ORDER= option specifies the data values in the order in which they are to appear on the axis.

Next, the TREE procedure is invoked. The HORIZONTAL option orients the tree diagram horizontally. The HAXIS option specifies that the AXIS1 statement be used to customize the appearance of the horizontal axis. The HEIGHT statement specifies the variable _PROPOR_ (the proportion of variance explained) as the height variable.

```
axis1 order=(0 to 1 by 0.2);
proc tree data=Tree horizontal haxis=axis1;
   height _PROPOR_;
run;
```

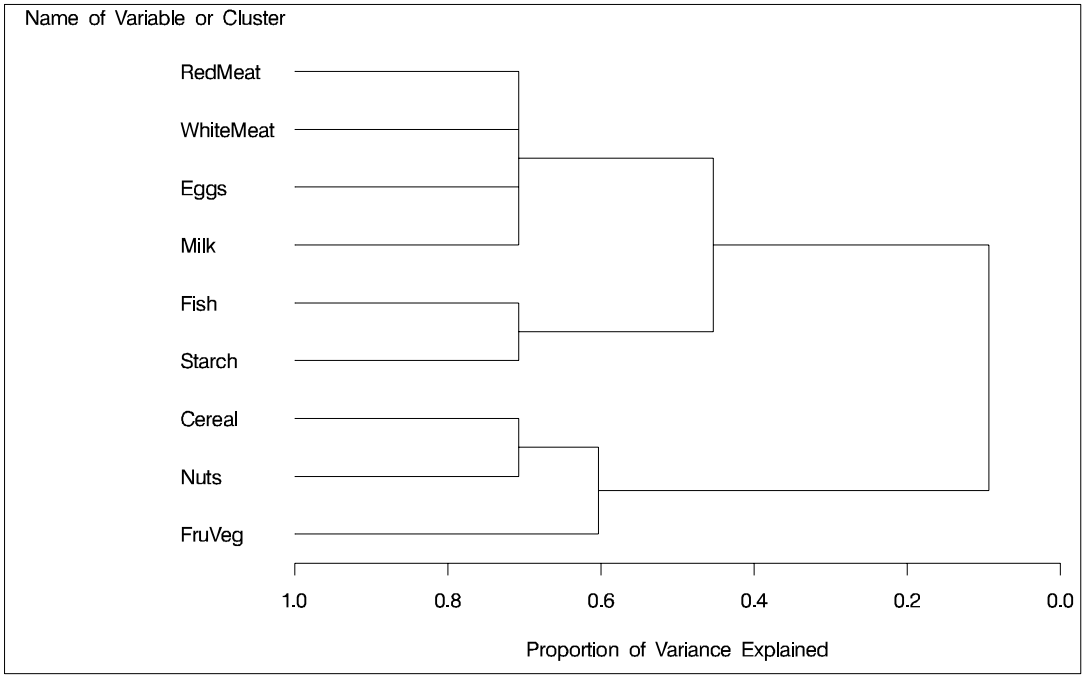



**Figure 66.3.** Horizontal Tree Diagram Using _PROPOR_ as the HEIGHT Variable

Figure 66.3 displays the tree diagram oriented horizontally, using the variable _PROPOR_ as the height variable. As you look from left to right in the diagram, objects and clusters are progressively joined until a single, all-encompassing cluster is formed at the right (or root) of the diagram.

Clusters exist at each level of the diagram, represented by horizontal line segments. Each vertical line segment represents a point where leaves and branches are connected into progressively larger clusters.

For example, three clusters are formed at the left-most point along the axis where three horizontal line segments exist. At that point, where a vertical line segment connects the Cereal-Nuts and FruVeg clusters, the proportion of variance explained is about 0.6 (_PROPOR_ = 0.6). At the next clustering level the variables Fish and Starch are clustered with variables RedMeat through Milk, resulting in a total of two clusters. The proportion of variance explained is about 0.45 at that point.

# Syntax

The TREE procedure is invoked by the following statements:

**PROC TREE** < *options* > **;**
**NAME** *variables* **;**
**HEIGHT** *variable* **;**
**PARENT** *variables* **;**
**BY** *variables* **;**
**COPY** *variables* **;**
**FREQ** *variable* **;**
**ID** *variable* **;**

If the input data set has been created by CLUSTER or VARCLUS, the only statement required is the PROC TREE statement. The BY, COPY, FREQ, HEIGHT, ID, NAME, and PARENT statements are described after the PROC TREE statement.

## PROC TREE Statement

**PROC TREE** < *options* > **;**

The PROC TREE statement starts the TREE procedure.

The options that can appear in the PROC TREE statement are summarized in the following table.

**Table 66.1.** PROC TREE Statement Options

| Task | Options | Effect |
|---|---|---|
| Specify data sets | DATA= | specifies the input data set |
| | DOCK= | does not count small clusters in OUT= data set |
| | LEVEL= | defines disjoint cluster in OUT= data set |
| | NCLUSTERS= | specifies the number of clusters in OUT= data set |
| | OUT= | specifies the output data set |
| | ROOT= | displays the root of a subtree |
| Specify cluster heights | HEIGHT= | specifies the variable for the height axis |
| | DISSIMILAR | specifies that large values are far apart |
| | SIMILAR | specifies that small values are close together |
| Display horizontal trees | HORIZONTAL | specifies that the height axis is horizontal |
| Control sort order | DESCENDING | reverses SORT order |
| | SORT | sorts children by HEIGHT variable |
| Control displayed output | LIST | displays all nodes in the tree |
| | NOPRINT | suppresses display of the tree |
| | LINEPRINTER | displays tree using line printer style graphics |
| High resolution graphics | INC= | specifies the increment between tick values |
| | MAXHEIGHT= | specifies the maximum value on axis |
| | MINHEIGHT= | specifies the minimum value on axis |
| | NTICK= | specifies the number of tick intervals |
| | CFRAME= | specifies the color of the frame |
| | DESCRIPTION= | specifies the catalog description |
| | GOUT= | specifies the catalog name |
| | HAXIS= | customizes horizontal axis |
| | HORDISPLAY= | displays a horizontal tree with leaves on the right |
| | HPAGES= | specifies the number of pages to expand tree horizontally |
| | LINES= | specifies the line color and thickness, dots at the nodes |
| | NAME= | specifies the name of graph in the catalog |
| | VAXIS= | customizes vertical axis |
| | VPAGES= | specifies the number of pages to expand tree vertically |
| Line printer graphics | INC= | specifies the increment between tick values |
| | MAXHEIGHT= | specifies the maximum value on axis |
| | MINHEIGHT= | specifies the minimum value on axis |
| | NTICK= | specifies the number of tick intervals |
| | PAGES= | specifies the number of pages |
| | POS= | specifies the number of column positions |

**Table 66.1.** (continued)

| Task | Options | Effect |
|---|---|---|
| | SPACES= | specifies the number of spaces between objects |
| | TICKPOS= | specifies the number of column positions between ticks |
| | FILLCHAR= | specifies the fill character between unjoined leaves |
| | JOINCHAR= | specifies the character to display between joined leaves |
| | LEAFCHAR= | specifies the character to represent clusters with no children |
| | TREECHAR= | specifies the character to represent clusters with children |

**CFRAME=***color*

specifies a color for the frame, which is the rectangle bounded by the axes.

**DATA=***SAS-data-set*

specifies the input data set defining the tree. If you omit the DATA= option, the most recently created SAS data set is used.

**DESCENDING**
**DES**

reverses the sorting order for the SORT option.

**DESCRIPTION=***entry-description*

specifies a description for the graph in the GOUT= catalog. The default is "Proc Tree Graph Output."

**DISSIMILAR**
**DIS**

implies that the values of the HEIGHT variable are dissimilarities; that is, a large height value means that the clusters are very dissimilar or far apart.

If neither the SIMILAR nor the DISSIMILAR option is specified, PROC TREE attempts to infer from the data whether the height values are similarities or dissimilarities. If PROC TREE cannot tell this from the data, it issues an error message and does not display a tree diagram.

**DOCK=***n*

causes observations in the OUT= data set assigned to output clusters with a frequency of $n$ or less to be given missing values for the output variables CLUSTER and CLUSNAME. If the NCLUSTERS= option is also specified, DOCK= also prevents clusters with a frequency of $n$ or less from being counted toward the number of clusters requested by the NCLUSTERS= option. By default, DOCK=0.

**FILLCHAR=***'c'*
**FC=***'c'*
specifies the character to display between leaves that are not joined into a cluster. The character should be enclosed in single quotes. The default is a blank. The LINEPRINTER option must also be specified.

**GOUT=***<libref.>member-name*
specifies the catalog in which the generated graph is stored. The default is WORK.GSEG.

**HAXIS=AXIS***n*
specifies the AXIS$n$ statement used to customize the appearance of the horizontal axis.

**HEIGHT=***name*
**H=***name*
specifies certain conventional variables to be used for the height axis of the tree diagram. For many situations, the only option you need is the HEIGHT= option. Valid values for *name* and their meanings are as follows:

| | |
|---|---|
| HEIGHT \| H | specifies the _HEIGHT_ variable. |
| LENGTH \| L | defines the height of each node as its path length from the root. This can also be interpreted as the number of ancestors of the node. |
| MODE \| M | specifies the _MODE_ variable. |
| NCL \| N | specifies the _NCL_ (number of clusters) variable. |
| RSQ \| R | specifies the _RSQ_ variable. |

See also the "HEIGHT Statement" section on page 3547, which can specify any variable in the input data set to be used for the height axis. In rare cases, you may need to specify either the DISSIMILAR option or the SIMILAR option.

**HORDISPLAY=RIGHT**
specifies that the graph is to be oriented horizontally, with the leaf nodes on the right side, when the HORIZONTAL option is also specified. By default, the leaf nodes are on the left side.

**HORIZONTAL**
**HOR**
orients the tree diagram with the height axis horizontal and the root at the left. The leaf nodes are on the side specified in the HORDISPLAY= option. If you do not specify the HORIZONTAL option, the height axis is vertical, with the root at the top. When the tree takes up more than one page and is viewed on a screen, horizontal orientation can make the tree diagram considerably easier to read.

**HPAGES=***n1*
specifies that the original graph is to be enlarged to cover $n1$ pages. If you also specify the VPAGES=$n2$ option, the original graph is enlarged to cover $n1 \times n2$ graphs. For example, if HPAGES=2 and VPAGES=3, then the original graph is generated followed by $2 \times 3 = 6$ more graphs. In these six graphs, the original is enlarged by

a factor of 2 in the horizontal direction and by a factor of 3 in the vertical direction. The graphs are generated in left-to-right and top-to-bottom order.

**INC=***n*

specifies the increment between tick values on the height axis. If the HEIGHT variable is _NCL_, the default is usually 1, although a different value can be specified for consistency with other options. For any other HEIGHT variable, the default is some power of 10 times 1, 2, 2.5, or 5.

**JOINCHAR=***'c'*
**JC=***'c'*

specifies the character to display between leaves that are joined into a cluster. The character should be enclosed in single quotes. The default is X. The LINEPRINTER option must also be specified.

**LEAFCHAR=***'c'*
**LC=***'c'*

specifies a character to represent clusters having no children. The character should be enclosed in single quotes. The default is a period. The LINEPRINTER option must also be specified.

**LEVEL=***n*

specifies the level of the tree defining disjoint clusters for the OUT= data set. The LEVEL= option also causes only clusters between the root and a height of $n$ to be displayed. The clusters in the output data set are those that exist at a height of $n$ on the tree diagram. For example, if the HEIGHT variable is _NCL_ (number of clusters) and LEVEL=5 is specified, then the OUT= data set contains five disjoint clusters. If the HEIGHT variable is _RSQ_ ($R^2$) and LEVEL=0.9 is specified, then the OUT= data set contains the smallest number of clusters that yields an $R^2$ of at least 0.9.

**LINEPRINTER**

specifies that the generated report is to be displayed using line printer graphics.

**LINES=(**<**COLOR=***color*><**WIDTH=***n*><**DOTS**>**)**

enables you to specify both the color and the thickness of the lines. In addition, a dot can be drawn at each leaf node. Note that if the frame and the lines are specified to be the same color, PROC TREE selects a different color for the lines.

**LIST**

lists all the nodes in the tree, displaying the height, parent, and children of each node.

**MAXHEIGHT=***n*
**MAXH=***n*

specifies the maximum value displayed on the height axis.

**MINHEIGHT=***n*
**MINH=***n*

specifies the minimum value displayed on the height axis.

**NAME=***name*
specifies the entry name for the generated graph in the GOUT= catalog. Note that each time another graph is generated with the same name, the name is modified by appending a number to make it unique.

**NCLUSTERS=***n*
**NCL=***n*
**N=***n*
specifies the number of clusters desired in the OUT= data set. The number of clusters obtained may not equal the number specified if (1) there are fewer than $n$ leaves in the tree, (2) there are more than $n$ unconnected trees in the data set, (3) a multi-way tree does not contain a level with the specified number of clusters, or (4) the DOCK= option eliminates too many clusters.

The NCLUSTERS= option uses the _NCL_ variable to determine the order in which the clusters are formed. If there is no _NCL_ variable, the height variable (as determined by the HEIGHT statement or HEIGHT= option) is used instead.

**NTICK=***n*
specifies the number of tick intervals on the height axis. The default depends on the values of other options.

**NOPRINT**
suppresses the display of the tree. Specify the NOPRINT option if you want only to create an OUT= data set. Note that this option temporarily disables the Output Delivery System (ODS). For more information, see Chapter 15, "Using the Output Delivery System."

**OUT=***SAS-data-set*
creates an output data set that contains one observation for each object in the tree or subtree being processed and variables called CLUSTER and CLUSNAME showing cluster membership at any specified level in the tree. If you specify the OUT= option, you must also specify either the NCLUSTERS= or LEVEL= option in order to define the output partition level. If you want to create a permanent SAS data set, you must specify a two-level name (refer to "SAS Data Files" in *SAS Language Reference: Concepts*).

**PAGES=***n*
specifies the number of pages over which the tree diagram (from root to leaves) is to extend. The default is 1. The LINEPRINTER option must also be specified.

**POS=***n*
specifies the number of column positions on the height axis. The default depends on the value of the PAGES= option, the orientation of the tree diagram, and the values specified by the PAGESIZE= and LINESIZE= options. The LINEPRINTER option must also be specified.

**ROOT=***'name'*
specifies the value of the NAME variable for the root of a subtree to be displayed if you do not want to display the entire tree. If you also specify the OUT= option, the output data set contains only objects belonging to the subtree specified by the ROOT= option.

**SIMILAR**
**SIM**
implies that the values of the HEIGHT variable are similarities; that is, a large height value means that the clusters are very similar or close together.

If neither the SIMILAR nor the DISSIMILAR option is specified, PROC TREE attempts to infer from the data whether the height values are similarities or dissimilarities. If PROC TREE cannot tell this from the data, it issues an error message and does not display a tree diagram.

**SORT**
sorts the children of each node by the HEIGHT variable, in the order of cluster formation. See the DESCENDING option on page 3541.

**SPACES=***s*
**S=***s*
specifies the number of spaces between objects on the output. The default depends on the number of objects, the orientation of the tree diagram, and the values specified by the PAGESIZE= and LINESIZE= options. The LINEPRINTER option must also be specified.

**TICKPOS=***n*
specifies the number of column positions per tick interval on the height axis. The default value is usually between 5 and 10, although a different value can be specified for consistency with other options.

**TREECHAR=***'c'*
**TC=***'c'*
specifies a character to represent clusters with children. The character should be enclosed in single quotes. The default is X. The LINEPRINTER option must also be specified.

**VAXIS=AXIS***n*
specifies that the AXIS$n$ statement be used to customize the appearance of the vertical axis.

**VPAGES=***n2*
specifies that the original graph is to be enlarged to cover $n2$ pages. If you also specify the HPAGES=$n1$ option, the original graph is enlarged to cover $n1 \times n2$ pages. For example, if HPAGES=2 and VPAGES=3, then the original graph is generated followed by $2 \times 3 = 6$ more graphs. In these six graphs, the original is enlarged by a factor of 2 in the horizontal direction and by a factor of 3 in the vertical direction. The graphs are generated in left-to-right and top-to-bottom order.

## BY Statement

**BY** *variables* **;**

You can specify a BY statement with PROC TREE to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the TREE procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables using the DATASETS procedure.

For more information on the BY statement, refer to the discussion in *SAS Language Reference: Concepts*. For more information on the DATASETS procedure, refer to the discussion in the *SAS Procedures Guide.*

## COPY Statement

**COPY** *variables* **;**

The COPY statement specifies one or more character or numeric variables to be copied to the OUT= data set.

## FREQ Statement

**FREQ** *variables* **;**

The FREQ statement specifies one numeric variable that tells how many clustering observations belong to the cluster. If the FREQ statement is omitted, PROC TREE looks for a variable called _FREQ_ to specify the number of observations per cluster. If neither the FREQ statement nor the _FREQ_ variable is present, each leaf is assumed to represent one clustering observation, and the frequency for each internal node is found by summing the frequencies of its children.

## HEIGHT Statement

**HEIGHT** *variable* **;**

The HEIGHT statement specifies the name of a numeric variable to define the height of each node (cluster) in the tree. The height variable can also be specified by the HEIGHT= option in the PROC TREE statement. If both the HEIGHT statement and the HEIGHT= option are omitted, PROC TREE looks for a variable called _HEIGHT_. If the data set does not contain _HEIGHT_, PROC TREE looks for a variable called _NCL_. If _NCL_ is not found either, the height of each node is defined to be its path length from the root.

## ID Statement

**ID** *variables* **;**

The ID variable is used to identify the objects (leaves) in the tree on the output. The ID variable can be a character or numeric variable of any length. If the ID statement is omitted, the variable in the NAME statement is used instead. If both the ID and NAME statements are omitted, PROC TREE looks for a variable called _NAME_. If the _NAME_ variable is not found in the data set, PROC TREE issues an error message and stops. The ID variable is copied to the OUT= data set.

## NAME Statement

**NAME** *variables* **;**

The NAME statement specifies a character or numeric variable identifying the node represented by each observation. The NAME variable and the PARENT variable jointly define the tree structure. If the NAME statement is omitted, PROC TREE looks for a variable called _NAME_. If the _NAME_ variable is not found in the data set, PROC TREE issues an error message and stops.

## PARENT Statement

**PARENT** *variables* **;**

The PARENT statement specifies a character or numeric variable identifying the node in the tree that is the parent of each observation. The PARENT variable must have the same formatted length as the NAME variable. If the PARENT statement is omitted, PROC TREE looks for a variable called _PARENT_. If the _PARENT_ variable is not found in the data set, PROC TREE issues an error message and stops.

# Details

## Missing Values

An observation with a missing value for the NAME variable is omitted from processing. If the PARENT variable has a missing value but the NAME variable is present, the observation is treated as the root of a tree. A data set can contain several roots and, hence, several trees.

Missing values of the HEIGHT variable are set to upper or lower bounds determined from the nonmissing values under the assumption that the heights are monotonic with respect to the tree structure.

Missing values of the FREQ variable are inferred from nonmissing values where possible; otherwise, they are treated as zero.

## Output Data Set

The OUT= data set contains one observation for each leaf in the tree or subtree being processed. The variables are as follows:

- the BY variables, if any
- the ID variable, or the NAME variable if the ID statement is not used
- the COPY variables
- a numeric variable CLUSTER taking values from 1 to $c$, where $c$ is the number of disjoint clusters. The cluster to which the first observation belongs is given the number 1, the cluster to which the next observation belongs that does not belong to cluster 1 is given the number 2, and so on.
- a character variable CLUSNAME giving the value of the NAME variable of the cluster to which the observation belongs

The CLUSTER and CLUSNAME variables are missing if the corresponding leaf has a nonpositive frequency.

## Displayed Output

The displayed output from the TREE procedure includes the following:

- the names of the objects in the tree
- the height axis
- the tree diagram. A high-resolution graphics tree diagram is produced on the graphics device. The leaves are displayed at the bottom of the graph. Horizontal lines connect the leaves into branches, while the topmost horizontal line indicates the root.

  If the LINEPRINTER option is specified, the root (the cluster containing all the objects) is indicated by a solid line of the character specified by the

TREECHAR= option (the default character is 'X'). At each level of the tree, clusters are shown by unbroken lines of the TREECHAR= symbol with the FILLCHAR= symbol (the default is a blank) separating the clusters. The LEAFCHAR= symbol (the default character is a period) represents single-member clusters.

By default, the tree diagram is oriented with the height axis vertical and the object names at the top of the diagram. If the HORIZONTAL option is specified, then the height axis is horizontal and the object names are on the left.

## ODS Table Names

PROC TREE assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 15, "Using the Output Delivery System."

**Table 66.2.** ODS Tables Produced in PROC TREE

| ODS Table Name | Description | Statement | Option |
|---|---|---|---|
| Tree | Line-printer plot of the tree | PROC | LINEPRINTER |
| TreeListing | Line-printer listing of all nodes in the tree | PROC | LIST |

# Examples

## Example 66.1. Mammals' Teeth

The following data give the numbers of different kinds of teeth for a variety of mammals. The mammals are clustered by average linkage using the CLUSTER procedure (Output 66.1.1). The PROC TREE statement uses the average-linkage distance as the height axis, which is the default, and creates a horizontal high-resolution graphics tree (Output 66.1.2).

```
data teeth;
   title 'Mammals'' Teeth';
   input mammal $ 1-16 @21 (v1-v8) (1.);
   label V1='Right Top Incisors'
         V2='Right Bottom Incisors'
         V3='Right Top Canines'
         V4='Right Bottom Canines'
         V5='Right Top Premolars'
         V6='Right Bottom Premolars'
         V7='Right Top Molars'
         V8='Right Bottom Molars';
   datalines;
```

```
Brown Bat           23113333
Mole                32103333
Silver Hair Bat     23112333
Pigmy Bat           23112233
House Bat           23111233
Red Bat             13112233
Pika                21002233
Rabbit              21003233
Beaver              11002133
Groundhog           11002133
Gray Squirrel       11001133
House Mouse         11000033
Porcupine           11001133
Wolf                33114423
Bear                33114423
Raccoon             33114432
Marten              33114412
Weasel              33113312
Wolverine           33114412
Badger              33113312
River Otter         33114312
Sea Otter           32113312
Jaguar              33113211
Cougar              33113211
Fur Seal            32114411
Sea Lion            32114411
Grey Seal           32113322
Elephant Seal       21114411
Reindeer            04103333
Elk                 04103333
Deer                04003333
Moose               04003333
;
options pagesize=60 linesize=110;

proc cluster method=average std pseudo noeigen outtree=tree;
   id mammal;
   var v1-v8;
run;

proc tree graphics horizontal;
run;
```

Output 66.1.1 displays the information on how the clusters are joined. For example, the cluster history shows that the observations Wolf and Bear form cluster 29, which is merged with Raccoon to form cluster 11.

**Output 66.1.1.** Output from PROC CLUSTER

Mammals' Teeth

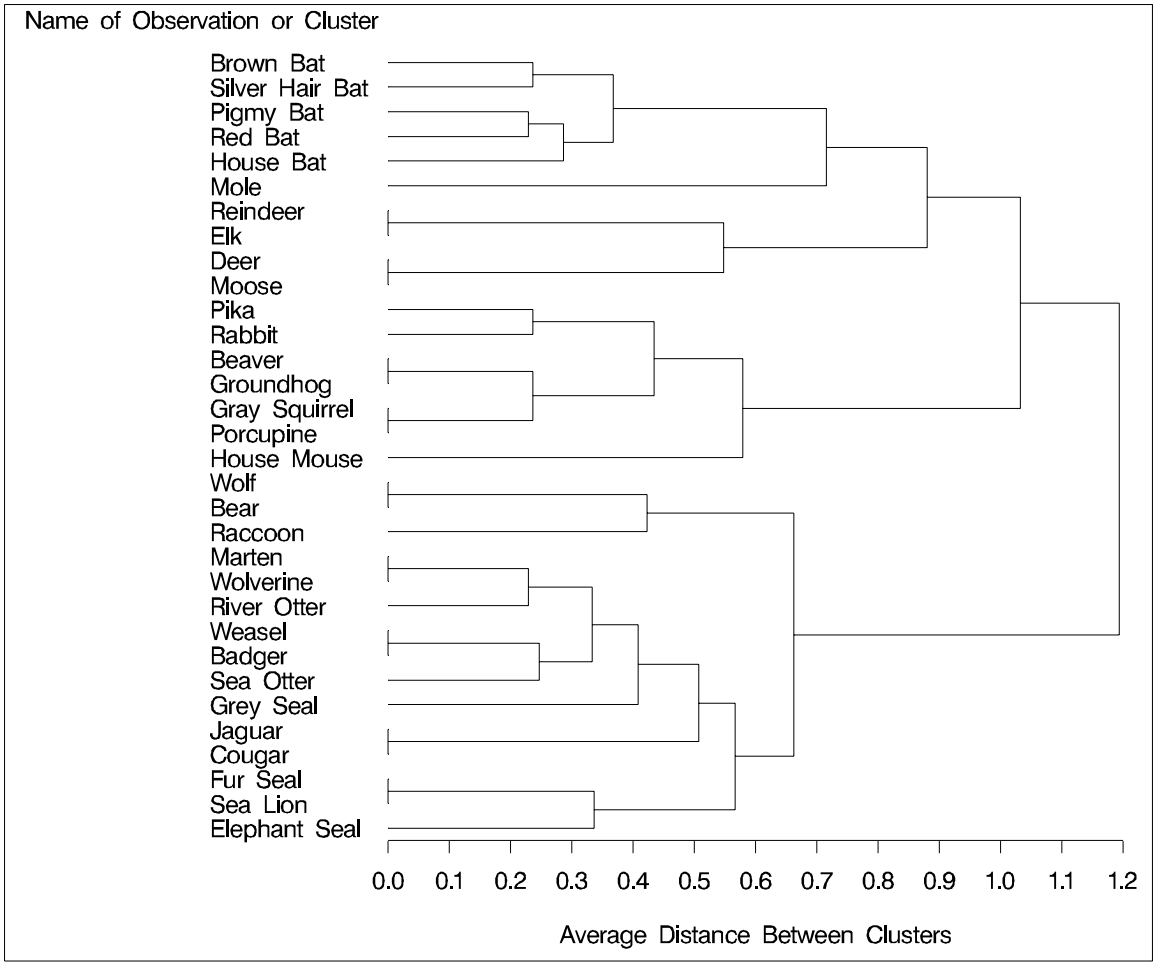
The CLUSTER Procedure
Average Linkage Cluster Analysis

The data have been standardized to mean 0 and variance 1
Root-Mean-Square Total-Sample Standard Deviation = 1
Root-Mean-Square Distance Between Observations = 4

Cluster History

| NCL | Clusters Joined | | FREQ | PSF | PST2 | Norm RMS Dist | Tie |
|---|---|---|---|---|---|---|---|
| 31 | Beaver | Groundhog | 2 | . | . | 0 | T |
| 30 | Gray Squirrel | Porcupine | 2 | . | . | 0 | T |
| 29 | Wolf | Bear | 2 | . | . | 0 | T |
| 28 | Marten | Wolverine | 2 | . | . | 0 | T |
| 27 | Weasel | Badger | 2 | . | . | 0 | T |
| 26 | Jaguar | Cougar | 2 | . | . | 0 | T |
| 25 | Fur Seal | Sea Lion | 2 | . | . | 0 | T |
| 24 | Reindeer | Elk | 2 | . | . | 0 | T |
| 23 | Deer | Moose | 2 | . | . | 0 | |
| 22 | Pigmy Bat | Red Bat | 2 | 281 | . | 0.2289 | |
| 21 | CL28 | River Otter | 3 | 139 | . | 0.2292 | |
| 20 | CL31 | CL30 | 4 | 83.2 | . | 0.2357 | T |
| 19 | Brown Bat | Silver Hair Bat | 2 | 76.7 | . | 0.2357 | T |
| 18 | Pika | Rabbit | 2 | 73.2 | . | 0.2357 | |
| 17 | CL27 | Sea Otter | 3 | 67.4 | . | 0.2462 | |
| 16 | CL22 | House Bat | 3 | 62.9 | 1.7 | 0.2859 | |
| 15 | CL21 | CL17 | 6 | 47.4 | 6.8 | 0.3328 | |
| 14 | CL25 | Elephant Seal | 3 | 45.0 | . | 0.3362 | |
| 13 | CL19 | CL16 | 5 | 40.8 | 3.5 | 0.3672 | |
| 12 | CL15 | Grey Seal | 7 | 38.9 | 2.8 | 0.4078 | |
| 11 | CL29 | Raccoon | 3 | 38.0 | . | 0.423 | |
| 10 | CL18 | CL20 | 6 | 34.5 | 10.3 | 0.4339 | |
| 9 | CL12 | CL26 | 9 | 30.0 | 7.3 | 0.5071 | |
| 8 | CL24 | CL23 | 4 | 28.7 | . | 0.5473 | |
| 7 | CL9 | CL14 | 12 | 25.7 | 7.0 | 0.5668 | |
| 6 | CL10 | House Mouse | 7 | 28.3 | 4.1 | 0.5792 | |
| 5 | CL11 | CL7 | 15 | 26.8 | 6.9 | 0.6621 | |
| 4 | CL13 | Mole | 6 | 31.9 | 7.2 | 0.7156 | |
| 3 | CL4 | CL8 | 10 | 31.0 | 12.7 | 0.8799 | |
| 2 | CL3 | CL6 | 17 | 27.8 | 16.1 | 1.0316 | |
| 1 | CL2 | CL5 | 32 | . | 27.8 | 1.1938 | |

**Output 66.1.2.** PROC TREE High-Resolution Graphics

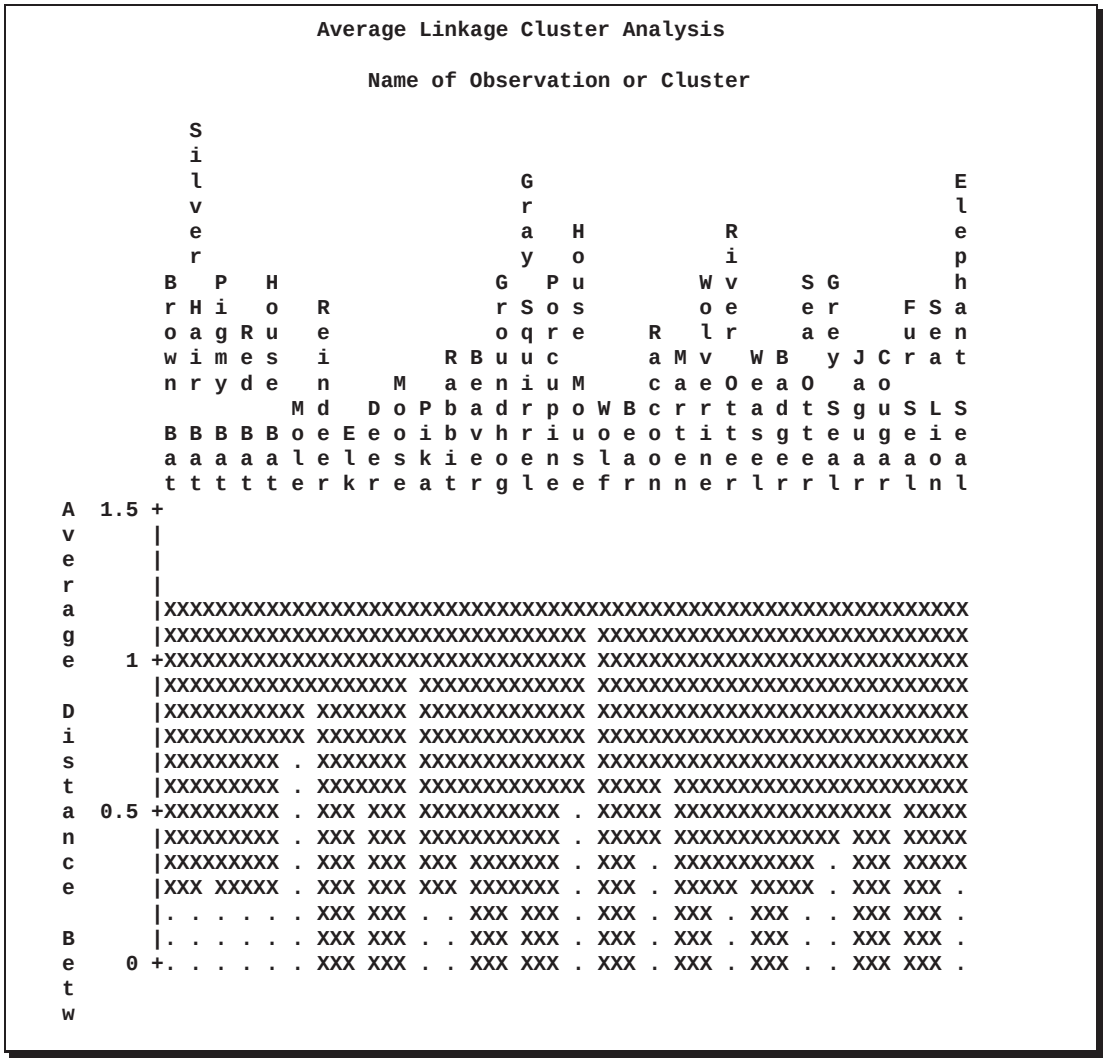



As you look from left-to-right in the diagram in Output 66.1.2, objects and clusters are progressively joined until a single, all-encompassing cluster is formed at the right (or root) of the diagram. Clusters exist at each level of the diagram, and every vertical line connects leaves and branches into progressively larger clusters. For example, the five bats form a cluster at the 0.6 level, while the next cluster consists only of the mole. The observations Reindeer, Elk, Deer, and Moose form the next cluster at the 0.6 level, the mammals Pika through House Mouse are in the fourth cluster, The observations Wolf, Bear, and Raccoon form the fifth cluster, while the last cluster contains the observations Marten through Elephant Seal.

The following statements create the same tree with line printer graphics in a vertical orientation; the tree is displayed in Output 66.1.3.

```
proc tree lineprinter;
run;
```

**Output 66.1.3.** PROC TREE with the LINEPRINTER Option

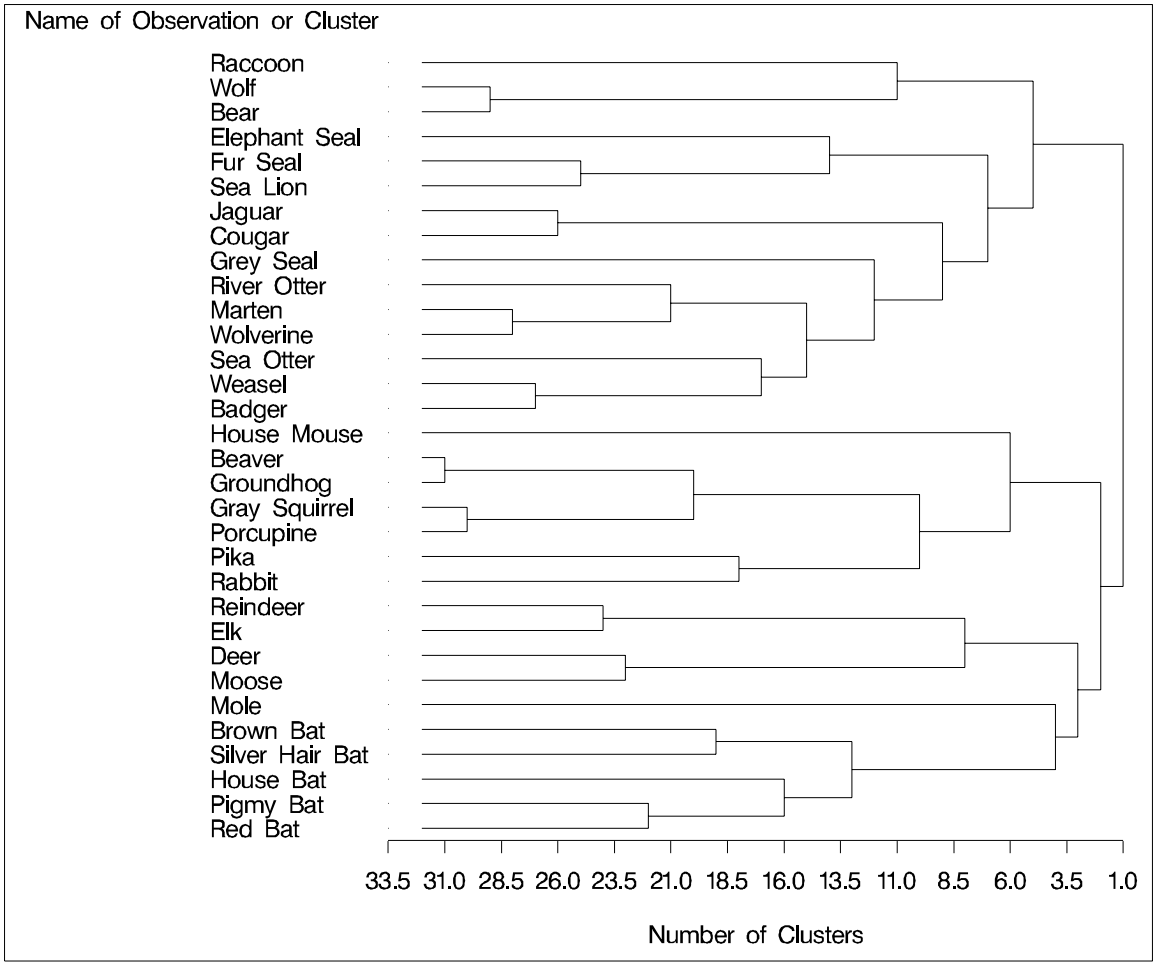



As you look up from the bottom of the diagram, objects and clusters are progressively joined until a single, all-encompassing cluster is formed at the top (or root) of the diagram. Clusters exist at each level of the diagram. For example, the unbroken line of Xs at the left-most side of the 0.6 level indicates that the five bats have formed a cluster. The next cluster is represented by a period because it contains only one mammal, Mole. Reindeer, Elk, Deer, and Moose form the next cluster, indicated by Xs again. The mammals Pika through House Mouse are in the fourth cluster. The observations Wolf, Bear, and Raccoon form the fifth cluster, while the last cluster contains the observations Marten through Elephant Seal.

The next statement sorts the clusters at each branch in order of formation and uses the number of clusters as the height axis. The resulting tree is displayed in Output 66.1.4.

```
proc tree sort height=n horizontal;
run;
```

**Output 66.1.4.** PROC TREE with SORT and HEIGHT= Options




Because the CLUSTER procedure always produces binary trees, the number of internal (root and branch) nodes in the tree is one less than the number of leaves. Therefore 31 clusters are formed from the 32 mammals in the input data set. These are represented by the 31 vertical line segments in the tree diagram, each at a different value along the horizontal axis.

As you examine the tree from left to right, the first vertical line segment is where Beaver and Groundhog are clustered and the number of clusters is 31. The next cluster is formed from Gray Squirrel and Porcupine. The third contains Wolf and Bear. Note how the tree graphically displays the clustering order information that was presented in tabular form by the CLUSTER procedure in Output 66.1.1.

The same clusters as in Output 66.1.2 and Output 66.1.3 can be seen at the six-cluster level of the tree diagram in Output 66.1.4, although the SORT and HEIGHT= options make them appear in a different order.

The following statements create these six clusters and display them in Output 66.1.5. The PROC TREE statement produces no output but creates an output data set indicating the cluster to which each observation belongs at the six-cluster level in the tree.

```
proc tree noprint out=part nclusters=6;
   id mammal;
   copy v1-v8;
proc sort;
   by cluster;
proc print label uniform;
   id mammal;
   var v1-v8;
   format v1-v8 1.;
   by cluster;
run;
```

**Output 66.1.5.** PROC TREE OUT= Data Set

**CLUSTER=1**

| mammal | Right Top Incisors | Right Bottom Incisors | Right Top Canines | Right Bottom Canines |
|---|---|---|---|---|
| Beaver | 1 | 1 | 0 | 0 |
| Groundhog | 1 | 1 | 0 | 0 |
| Gray Squirrel | 1 | 1 | 0 | 0 |
| Porcupine | 1 | 1 | 0 | 0 |
| Pika | 2 | 1 | 0 | 0 |
| Rabbit | 2 | 1 | 0 | 0 |
| House Mouse | 1 | 1 | 0 | 0 |

| mammal | Right Top Premolars | Right Bottom Premolars | Right Top Molars | Right Bottom Molars |
|---|---|---|---|---|
| Beaver | 2 | 1 | 3 | 3 |
| Groundhog | 2 | 1 | 3 | 3 |
| Gray Squirrel | 1 | 1 | 3 | 3 |
| Porcupine | 1 | 1 | 3 | 3 |
| Pika | 2 | 2 | 3 | 3 |
| Rabbit | 3 | 2 | 3 | 3 |
| House Mouse | 0 | 0 | 3 | 3 |

**CLUSTER=2**

| mammal | Right Top Incisors | Right Bottom Incisors | Right Top Canines | Right Bottom Canines |
|---|---|---|---|---|
| Wolf | 3 | 3 | 1 | 1 |
| Bear | 3 | 3 | 1 | 1 |
| Raccoon | 3 | 3 | 1 | 1 |

| mammal | Right Top Premolars | Right Bottom Premolars | Right Top Molars | Right Bottom Molars |
|---|---|---|---|---|
| Wolf | 4 | 4 | 2 | 3 |
| Bear | 4 | 4 | 2 | 3 |
| Raccoon | 4 | 4 | 3 | 2 |

**CLUSTER=3**

| mammal | Right Top Incisors | Right Bottom Incisors | Right Top Canines | Right Bottom Canines |
|---|---|---|---|---|
| Marten | 3 | 3 | 1 | 1 |
| Wolverine | 3 | 3 | 1 | 1 |
| Weasel | 3 | 3 | 1 | 1 |
| Badger | 3 | 3 | 1 | 1 |
| Jaguar | 3 | 3 | 1 | 1 |
| Cougar | 3 | 3 | 1 | 1 |
| Fur Seal | 3 | 2 | 1 | 1 |
| Sea Lion | 3 | 2 | 1 | 1 |
| River Otter | 3 | 3 | 1 | 1 |
| Sea Otter | 3 | 2 | 1 | 1 |
| Elephant Seal | 2 | 1 | 1 | 1 |
| Grey Seal | 3 | 2 | 1 | 1 |

| mammal | Right Top Premolars | Right Bottom Premolars | Right Top Molars | Right Bottom Molars |
|---|---|---|---|---|
| Marten | 4 | 4 | 1 | 2 |
| Wolverine | 4 | 4 | 1 | 2 |
| Weasel | 3 | 3 | 1 | 2 |
| Badger | 3 | 3 | 1 | 2 |
| Jaguar | 3 | 2 | 1 | 1 |
| Cougar | 3 | 2 | 1 | 1 |
| Fur Seal | 4 | 4 | 1 | 1 |
| Sea Lion | 4 | 4 | 1 | 1 |
| River Otter | 4 | 3 | 1 | 2 |
| Sea Otter | 3 | 3 | 1 | 2 |
| Elephant Seal | 4 | 4 | 1 | 1 |
| Grey Seal | 3 | 3 | 2 | 2 |

```
---------------------------------- CLUSTER=4 ----------------------------------

                        Right       Right       Right       Right
                         Top        Bottom       Top        Bottom
mammal                Incisors    Incisors    Canines     Canines

Reindeer                 0           4           1           0
Elk                      0           4           1           0
Deer                     0           4           0           0
Moose                    0           4           0           0

                                     Right       Right       Right
                      Right Top      Bottom       Top        Bottom
mammal                Premolars    Premolars    Molars      Molars

Reindeer                 3           3           3           3
Elk                      3           3           3           3
Deer                     3           3           3           3
Moose                    3           3           3           3


---------------------------------- CLUSTER=5 ----------------------------------

                        Right       Right       Right       Right
                         Top        Bottom       Top        Bottom
mammal                Incisors    Incisors    Canines     Canines

Pigmy Bat                2           3           1           1
Red Bat                  1           3           1           1
Brown Bat                2           3           1           1
Silver Hair Bat          2           3           1           1
House Bat                2           3           1           1

                                     Right       Right       Right
                      Right Top      Bottom       Top        Bottom
mammal                Premolars    Premolars    Molars      Molars

Pigmy Bat                2           2           3           3
Red Bat                  2           2           3           3
Brown Bat                3           3           3           3
Silver Hair Bat          2           3           3           3
House Bat                1           2           3           3
```

```
--------------------------------- CLUSTER=6 ---------------------------------

                          Right       Right       Right       Right
                           Top        Bottom       Top        Bottom
        mammal           Incisors    Incisors    Canines     Canines

        Mole                3           2           1           0

                                       Right       Right       Right
                         Right Top     Bottom       Top        Bottom
        mammal           Premolars   Premolars    Molars      Molars

        Mole                3           3           3           3
```

## Example 66.2. Iris Data

Fisher's (1936) iris data gives sepal and petal dimensions for three different species of iris. The data are clustered by *k*th-nearest-neighbor density linkage using the CLUSTER procedure with K=8. Observations are identified by species (Setosa, Versicolor or Virginica) in the tree diagram, which is oriented with the height axis horizontal. The following statements produce Output 66.2.1 and Output 66.2.2.

```
proc format;
   value specname
      1='Setosa    '
      2='Versicolor'
      3='Virginica ';
run;

data iris;
   title 'Fisher (1936) Iris Data';
   input SepalLength SepalWidth PetalLength PetalWidth
         Species @@;
   format Species specname.;
   label SepalLength='Sepal Length in mm.'
         SepalWidth ='Sepal Width in mm.'
         PetalLength='Petal Length in mm.'
         PetalWidth ='Petal Width in mm.';
   symbol = put(species, specname10.);
   datalines;
50 33 14 02 1 64 28 56 22 3 65 28 46 15 2 67 31 56 24 3
63 28 51 15 3 46 34 14 03 1 69 31 51 23 3 62 22 45 15 2
59 32 48 18 2 46 36 10 02 1 61 30 46 14 2 60 27 51 16 2
65 30 52 20 3 56 25 39 11 2 65 30 55 18 3 58 27 51 19 3
68 32 59 23 3 51 33 17 05 1 57 28 45 13 2 62 34 54 23 3
77 38 67 22 3 63 33 47 16 2 67 33 57 25 3 76 30 66 21 3
49 25 45 17 3 55 35 13 02 1 67 30 52 23 3 70 32 47 14 2
64 32 45 15 2 61 28 40 13 2 48 31 16 02 1 59 30 51 18 3
55 24 38 11 2 63 25 50 19 3 64 32 53 23 3 52 34 14 02 1
49 36 14 01 1 54 30 45 15 2 79 38 64 20 3 44 32 13 02 1
67 33 57 21 3 50 35 16 06 1 58 26 40 12 2 44 30 13 02 1
77 28 67 20 3 63 27 49 18 3 47 32 16 02 1 55 26 44 12 2
50 23 33 10 2 72 32 60 18 3 48 30 14 03 1 51 38 16 02 1
61 30 49 18 3 48 34 19 02 1 50 30 16 02 1 50 32 12 02 1
```

```
61 26 56 14 3 64 28 56 21 3 43 30 11 01 1 58 40 12 02 1
51 38 19 04 1 67 31 44 14 2 62 28 48 18 3 49 30 14 02 1
51 35 14 02 1 56 30 45 15 2 58 27 41 10 2 50 34 16 04 1
46 32 14 02 1 60 29 45 15 2 57 26 35 10 2 57 44 15 04 1
50 36 14 02 1 77 30 61 23 3 63 34 56 24 3 58 27 51 19 3
57 29 42 13 2 72 30 58 16 3 54 34 15 04 1 52 41 15 01 1
71 30 59 21 3 64 31 55 18 3 60 30 48 18 3 63 29 56 18 3
49 24 33 10 2 56 27 42 13 2 57 30 42 12 2 55 42 14 02 1
49 31 15 02 1 77 26 69 23 3 60 22 50 15 3 54 39 17 04 1
66 29 46 13 2 52 27 39 14 2 60 34 45 16 2 50 34 15 02 1
44 29 14 02 1 50 20 35 10 2 55 24 37 10 2 58 27 39 12 2
47 32 13 02 1 46 31 15 02 1 69 32 57 23 3 62 29 43 13 2
74 28 61 19 3 59 30 42 15 2 51 34 15 02 1 50 35 13 03 1
56 28 49 20 3 60 22 40 10 2 73 29 63 18 3 67 25 58 18 3
49 31 15 01 1 67 31 47 15 2 63 23 44 13 2 54 37 15 02 1
56 30 41 13 2 63 25 49 15 2 61 28 47 12 2 64 29 43 13 2
51 25 30 11 2 57 28 41 13 2 65 30 58 22 3 69 31 54 21 3
54 39 13 04 1 51 35 14 03 1 72 36 61 25 3 65 32 51 20 3
61 29 47 14 2 56 29 36 13 2 69 31 49 15 2 64 27 53 19 3
68 30 55 21 3 55 25 40 13 2 48 34 16 02 1 48 30 14 01 1
45 23 13 03 1 57 25 50 20 3 57 38 17 03 1 51 38 15 03 1
55 23 40 13 2 66 30 44 14 2 68 28 48 14 2 54 34 17 02 1
51 37 15 04 1 52 35 15 02 1 58 28 51 24 3 67 30 50 17 2
63 33 60 25 3 53 37 15 02 1
;
proc cluster data=iris method=twostage print=10
             outtree=tree k=8 noeigen;
   var SepalLength SepalWidth PetalLength PetalWidth;
   copy Species;
   id Species;
run;

options pagesize=60 linesize=110;

proc tree data=tree horizontal lineprinter pages=1 maxh=10;
   id species;
run;
```

The PAGES=1 option specifies that the tree diagram extends over one page from tree to root. Since the HORIZONTAL option is also specified, the horizontal extent of the diagram is one page. The number of vertical pages required for the diagram is dictated by the number of leaves in the tree.

The MAXH=10 limits the values displayed on the height axis to a maximum of 10. This prunes the tree diagram so that only the portion from the leaves to level 10 is displayed. You can see this pruning effect in Output 66.2.2.

**Output 66.2.1.** Clustering of Fisher's Iris Data

Fisher (1936) Iris Data

The CLUSTER Procedure
Two-Stage Density Linkage Clustering

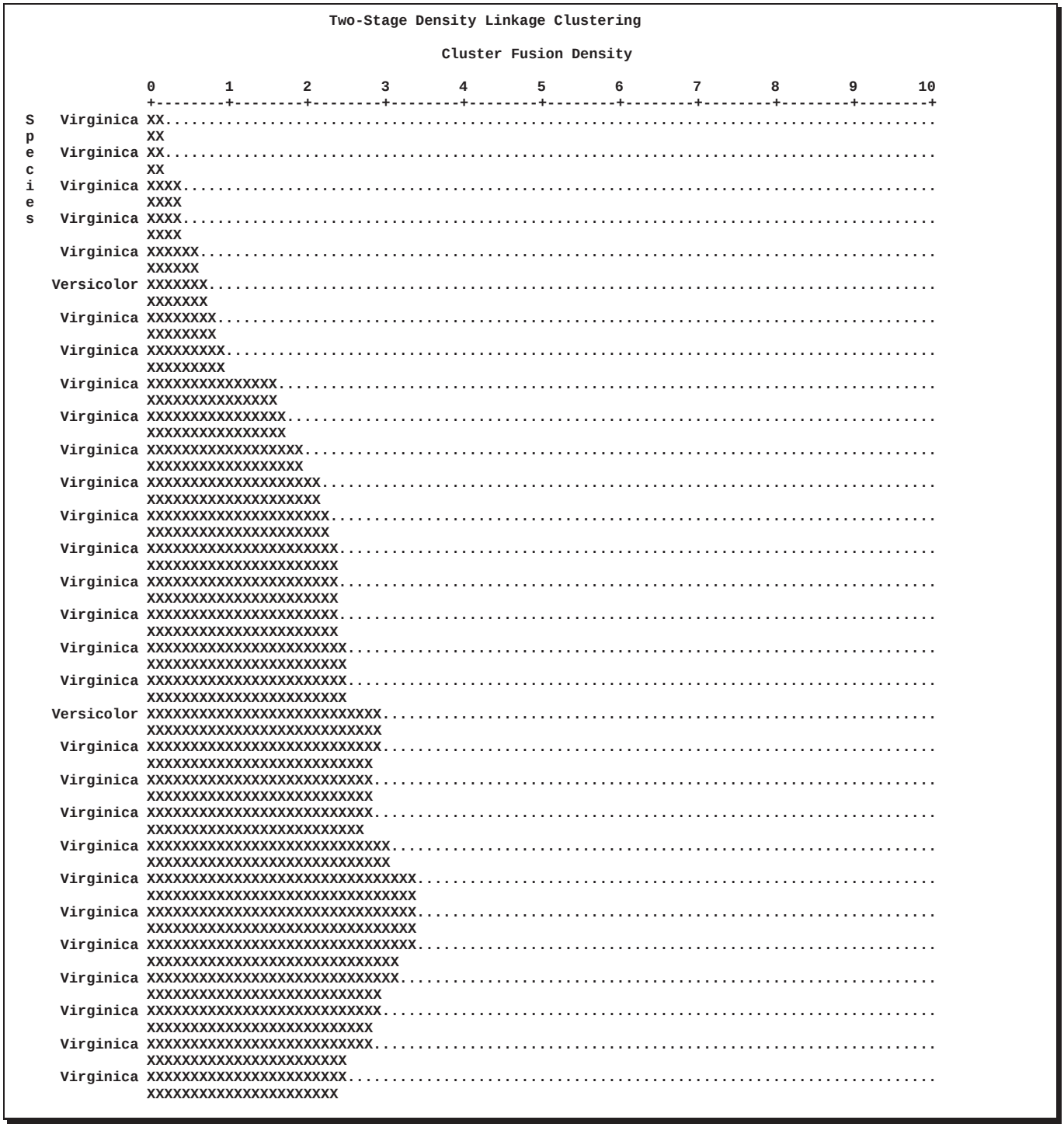
K = 8
Root-Mean-Square Total-Sample Standard Deviation = 10.69224

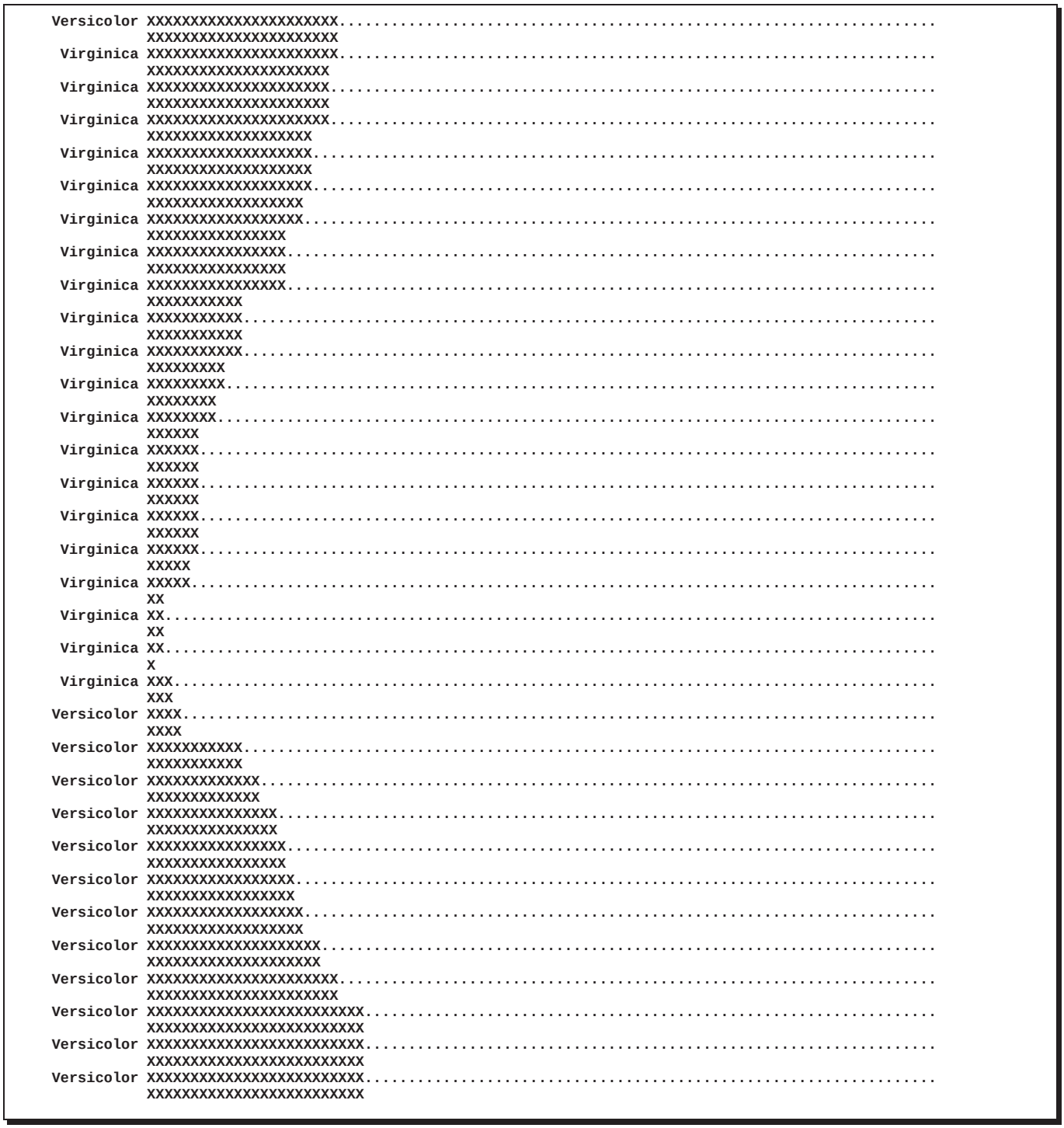
Cluster History

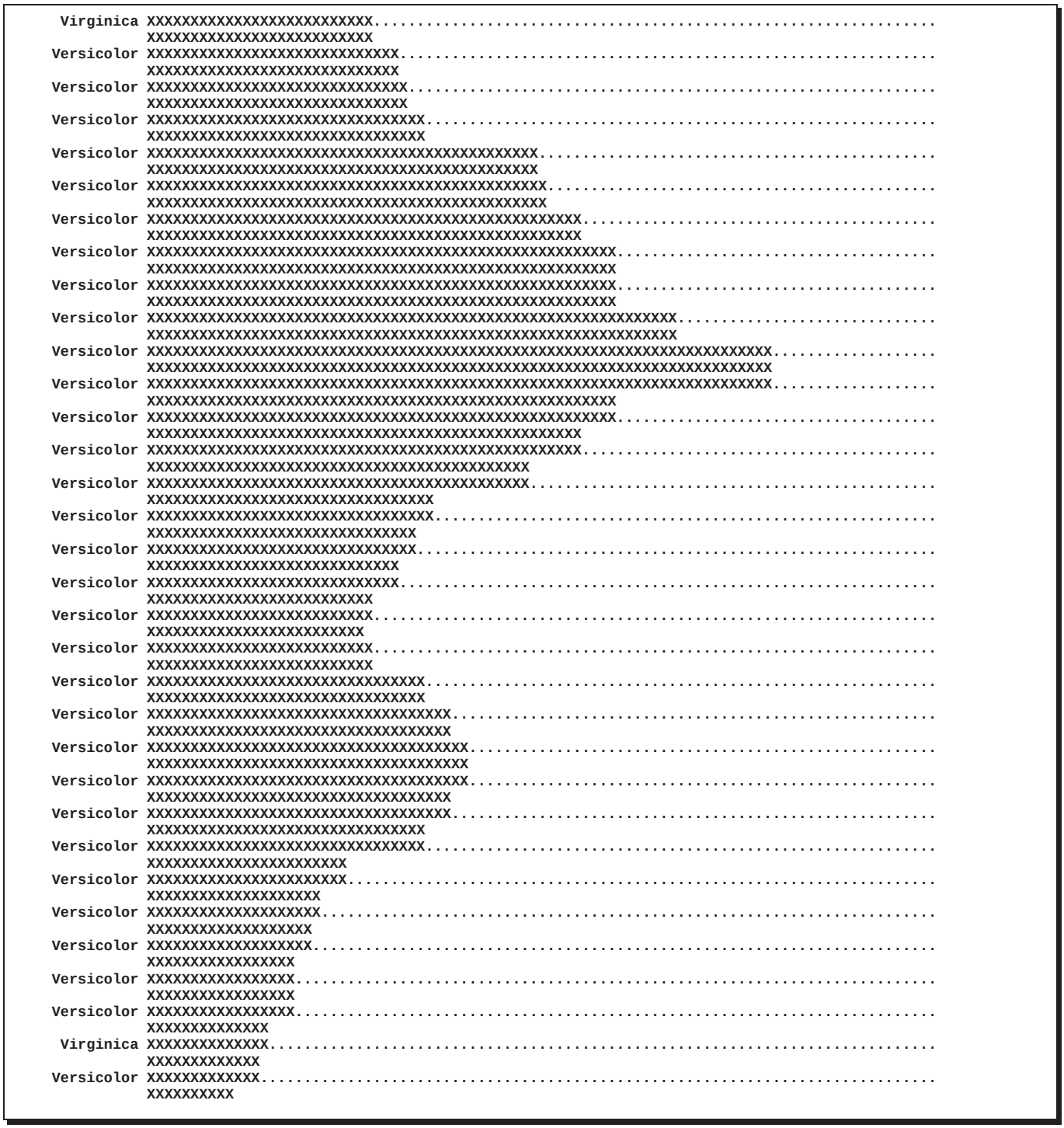
| NCL | ----Clusters Joined----- | | FREQ | Normalized Fusion Density | Maximum Density in Each Cluster Lesser | Greater | T i e |
|---|---|---|---|---|---|---|---|
| 10 | CL11 | Versicolor | 48 | 0.2879 | 0.1479 | 8.3678 | |
| 9 | CL13 | Virginica | 46 | 0.2802 | 0.2005 | 3.5156 | |
| 8 | CL10 | Virginica | 49 | 0.2699 | 0.1372 | 8.3678 | |
| 7 | CL8 | Versicolor | 50 | 0.2586 | 0.1372 | 8.3678 | |
| 6 | CL9 | Virginica | 47 | 0.1412 | 0.0832 | 3.5156 | |
| 5 | CL6 | Virginica | 48 | 0.107 | 0.0605 | 3.5156 | |
| 4 | CL5 | Virginica | 49 | 0.0969 | 0.0541 | 3.5156 | |
| 3 | CL4 | Virginica | 50 | 0.0715 | 0.0370 | 3.5156 | |
| 2 | CL3 | CL7 | 100 | 2.6277 | 3.5156 | 8.3678 | |

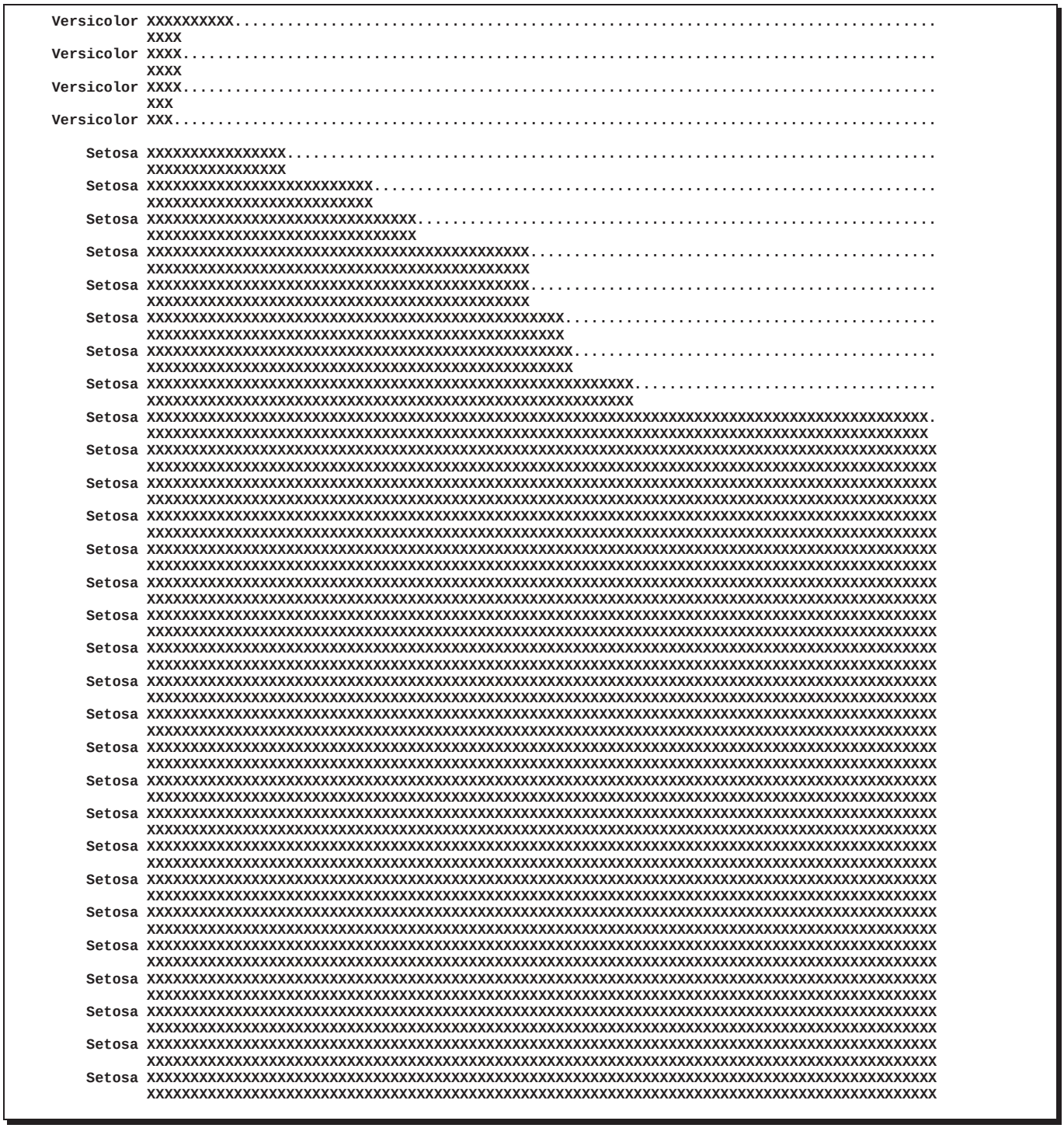
3 modal clusters have been formed.

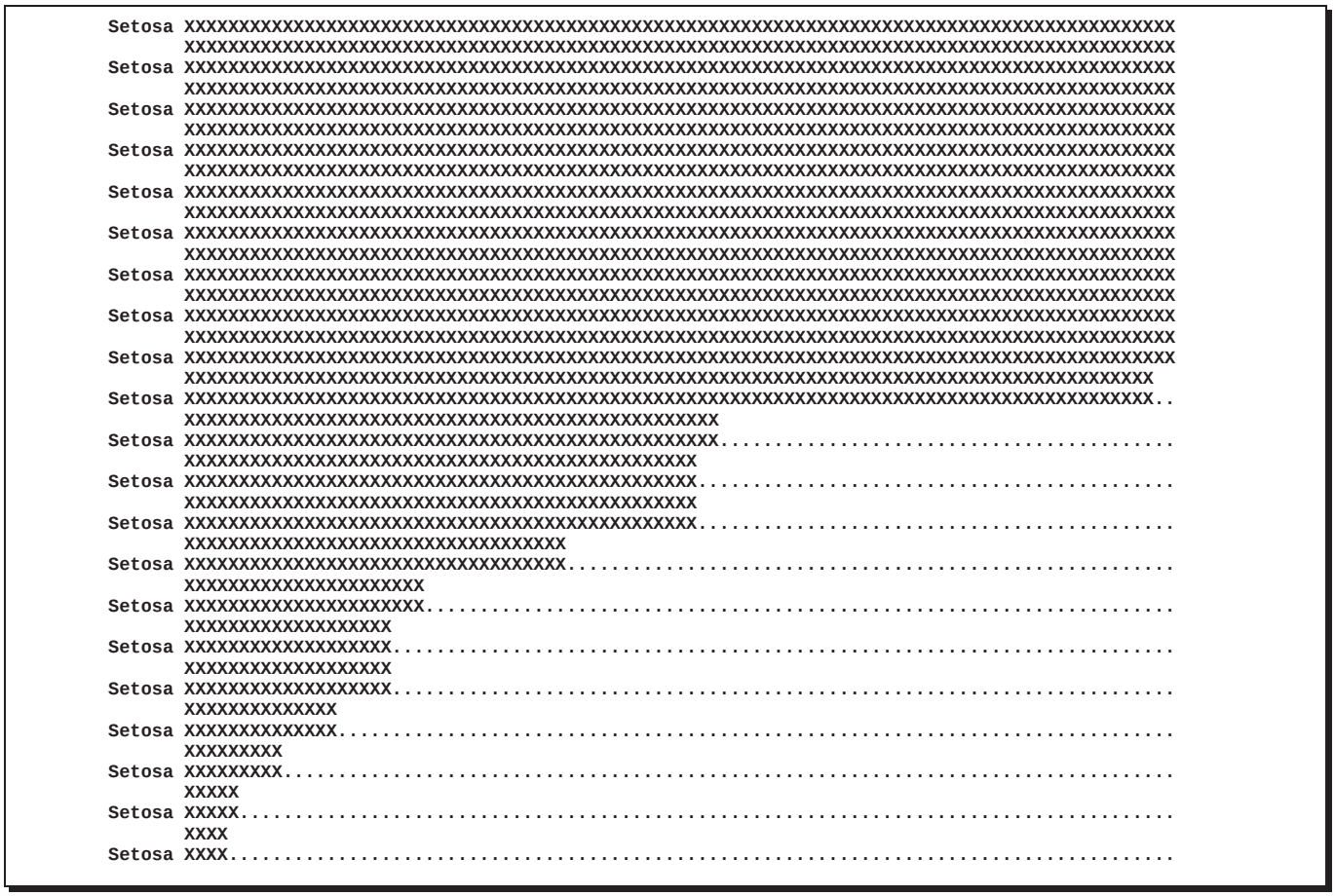
**Output 66.2.2.** Horizontal Tree for Fisher's Iris Data




Versicolor
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Virginica
Versicolor
Versicolor
Versicolor
Versicolor
Versicolor
Versicolor
Versicolor
Versicolor
Versicolor
Versicolor


Virginica
Versicolor
Virginica
Versicolor


Versicolor
Setosa




# References

Duran, B.S. and Odell, P.L. (1974), *Cluster Analysis*, New York: Springer-Verlag.

Everitt, B.S. (1980), *Cluster Analysis*, Second Edition, London: Heineman Educational Books Ltd.

Fisher, R.A. (1936), "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, 7, 179–188.

Hand, D.J.; Daly, F.; Lunn, A.D.; McConway, K.J.; and Ostrowski E. (1994), *A Handbook of Small Data Sets*, London: Chapman & Hall, 297–298.

Hartigan, J.A. (1975), *Clustering Algorithms*, New York: John Wiley & Sons, Inc.

Johnson, S.C. (1967), "Hierarchical Clustering Schemes," *Psychometrika*, 32, 241–254.

Knuth, D.E. (1973), *The Art of Computer Programming, Volume 1, Fundamental Algorithms*, Reading, MA: Addison-Wesley Publishing Co., Inc.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/STAT® User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999.

**SAS/STAT® User's Guide, Version 8**



ISBN 1–58025–494–2



SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999