THE POWER TO KNOW.

SAS/STAT® 9.3 User's Guide

The TREE Procedure

(Chapter)

SAS® Documentation

This document is an individual chapter from *SAS/STAT® 9.3 User's Guide*.

The correct bibliographic citation for the complete manual is as follows: SAS Institute Inc. 2011. *SAS/STAT® 9.3 User's Guide*. Cary, NC: SAS Institute Inc.

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government Restricted Rights Notice: Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987).

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

1st electronic book, July 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Chapter 94

The TREE Procedure

Contents

Overview: TREE Procedure **7960**
Getting Started: TREE Procedure **7960**
Syntax: TREE Procedure **7967**
PROC TREE Statement 7967
BY Statement 7973
COPY Statement 7973
FREQ Statement 7973
HEIGHT Statement 7974
ID Statement 7974
NAME Statement 7974
PARENT Statement 7974
Details: TREE Procedure **7975**
Missing Values 7975
Output Data Set 7975
Displayed Output 7976
ODS Table Names 7976
Examples: TREE Procedure **7977**
Example 94.1: Mammals' Teeth 7977
Example 94.2: Iris Data 7987
References **7994**

Overview: TREE Procedure

The TREE procedure reads a data set created by the CLUSTER or VARCLUS procedure and produces a tree diagram (also known as a *dendrogram* or *phenogram*), which displays the results of a hierarchical clustering analysis as a tree structure. The TREE procedure uses the 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 that contains 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 that contains all objects is the *root*. A cluster that contains 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: TREE Procedure

The TREE procedure creates tree diagrams from a SAS data set that contains the tree structure. You can create this type of data set with the CLUSTER or VARCLUS procedure. See Chapter 30, "The CLUSTER Procedure," and Chapter 96, "The VARCLUS Procedure," for more information.

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 that contains 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
;
```

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

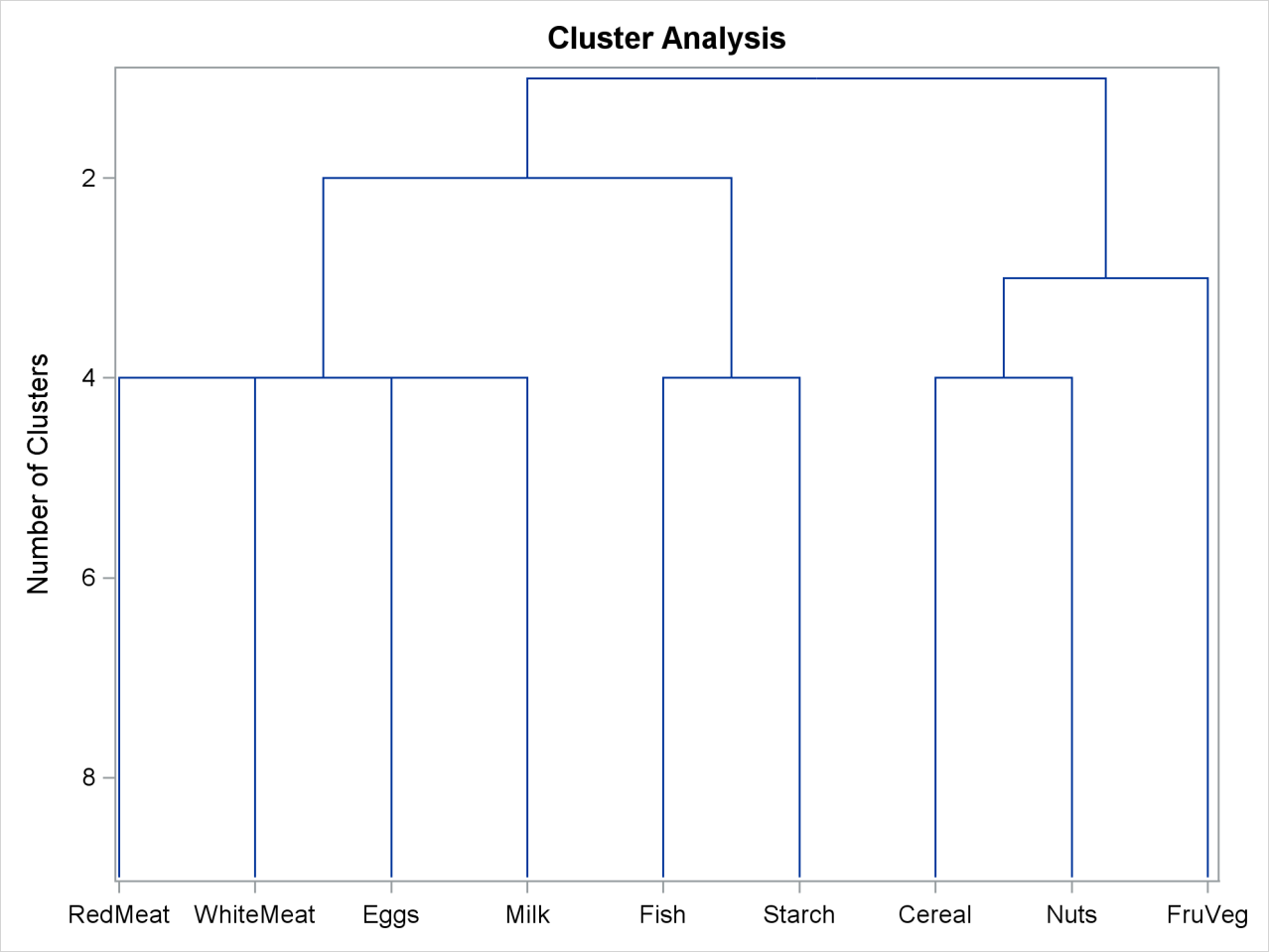
The following statements cluster the variables in the data set Protein:

```
ods graphics on;

proc varclus data=Protein outtree=Tree centroid maxclusters=4
             plots=dendrogram(vertical height=ncl);
   var RedMeat--FruVeg;
run;
```

The OUTTREE= option creates an output 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 VAR statement specifies that all numeric variables (RedMeat—FruVeg) are used by the procedure. Since ODS Graphics is enabled, PROC VARCLUS creates a dendrogram, which is displayed in Figure 94.1. The option **plots=dendrogram(vertical height=ncl)** specifies a vertical dendrogram with the number of clusters on the vertical axis. The default is a horizontal dendrogram with, for this cluster analysis, the proportion of variance explained on the horizontal axis.

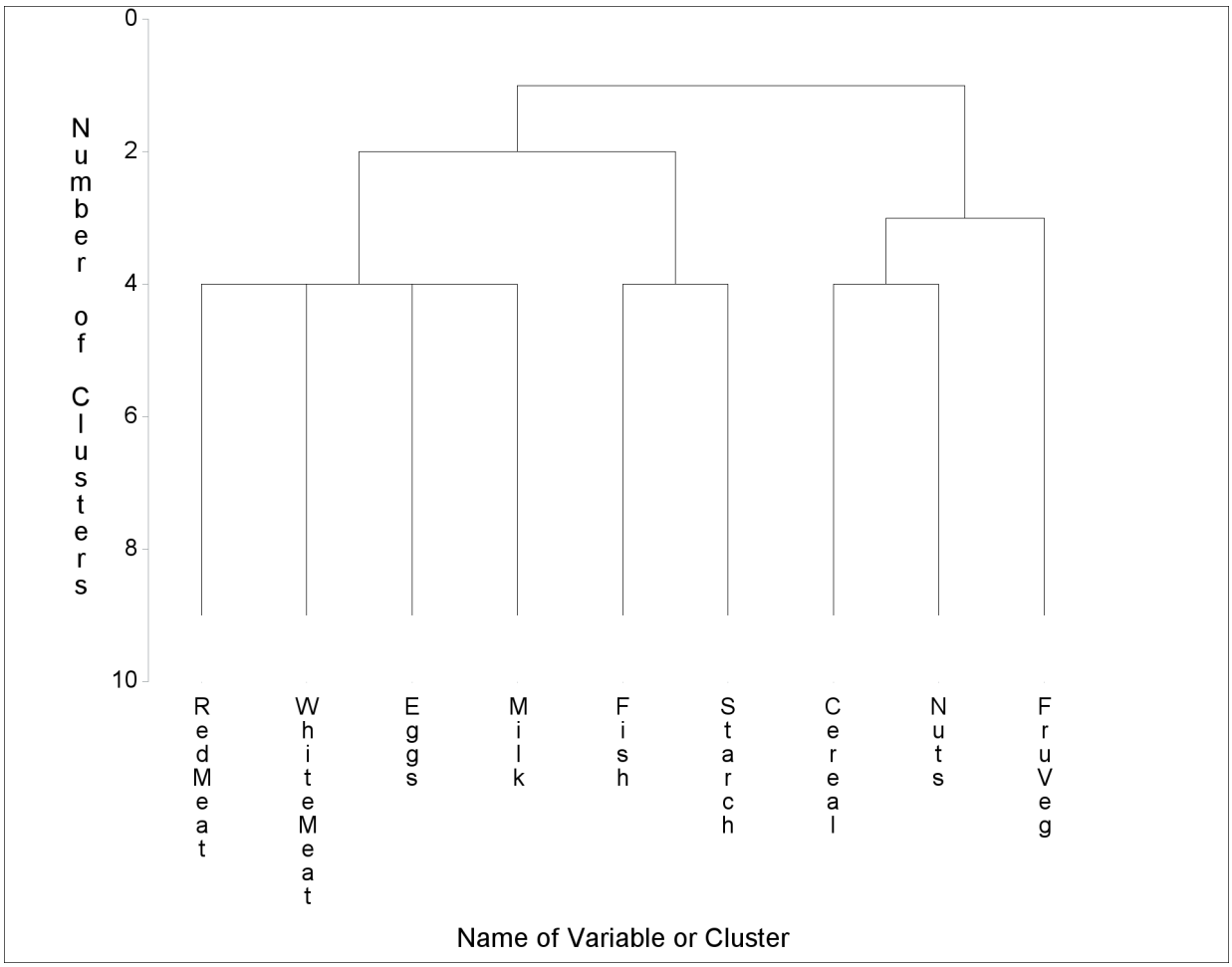
Figure 94.1 Dendrogram from PROC VARCLUS and ODS Graphics



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

Figure 94.2 Graphical Tree Diagram from PROC TREE



The following statements use PROC TREE to produce tree diagrams of the clusters created by PROC VARCLUS:

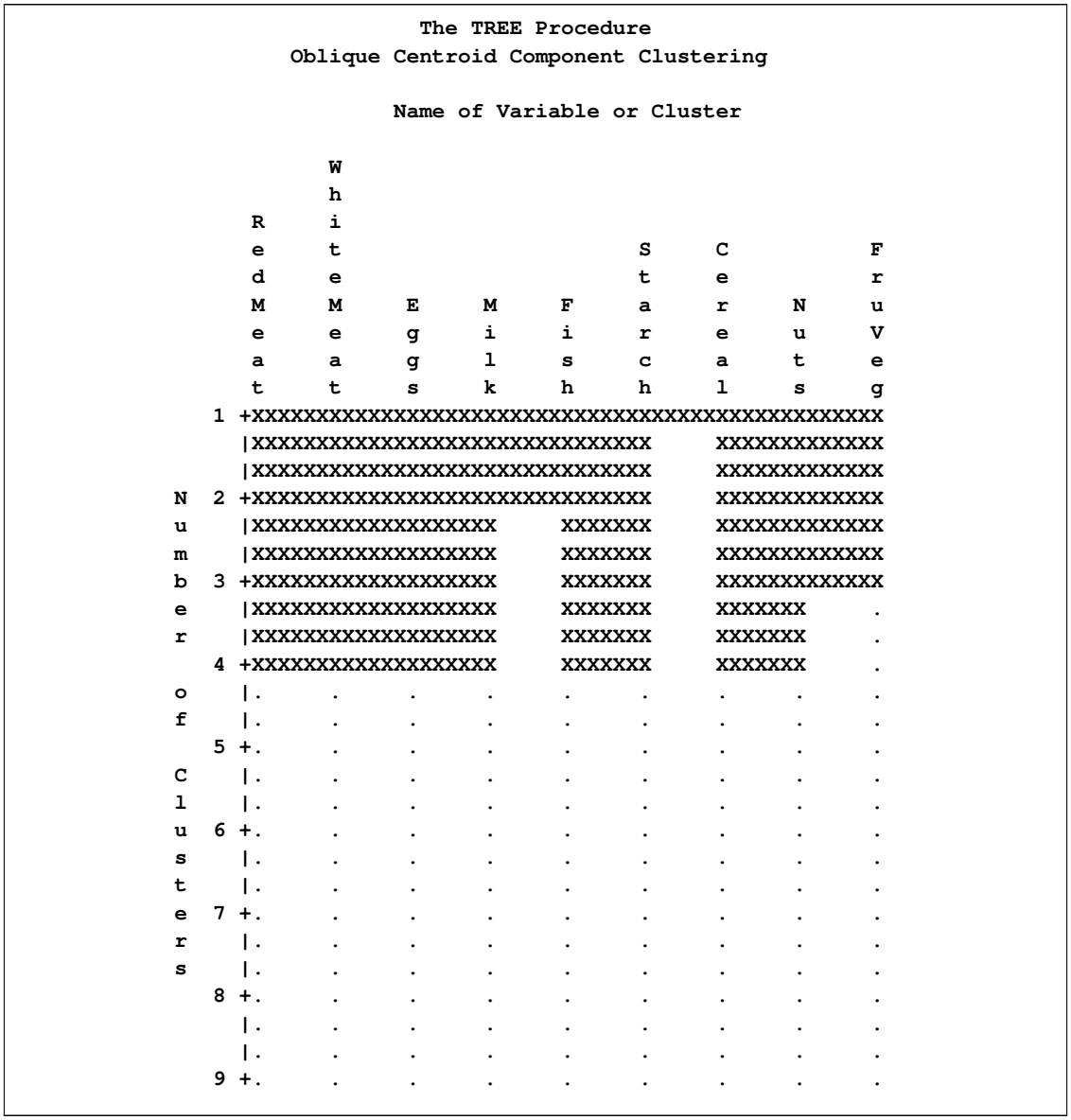
```
proc tree data=tree;
run;

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

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

Figure 94.2 displays the default graphical representation of the tree diagram. Figure 94.3 displays the same information as Figure 94.2 by using line printer output.

Figure 94.3 Line Printer Representation of the Tree Diagram



In each diagram the name of the cluster is displayed on the horizontal axis and the number of clusters is displayed on the vertical (height) axis.

As you look up from the bottom of either diagram, clusters are progressively joined until a single, all-encompassing cluster is formed at the top (or root) of the tree. 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:

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

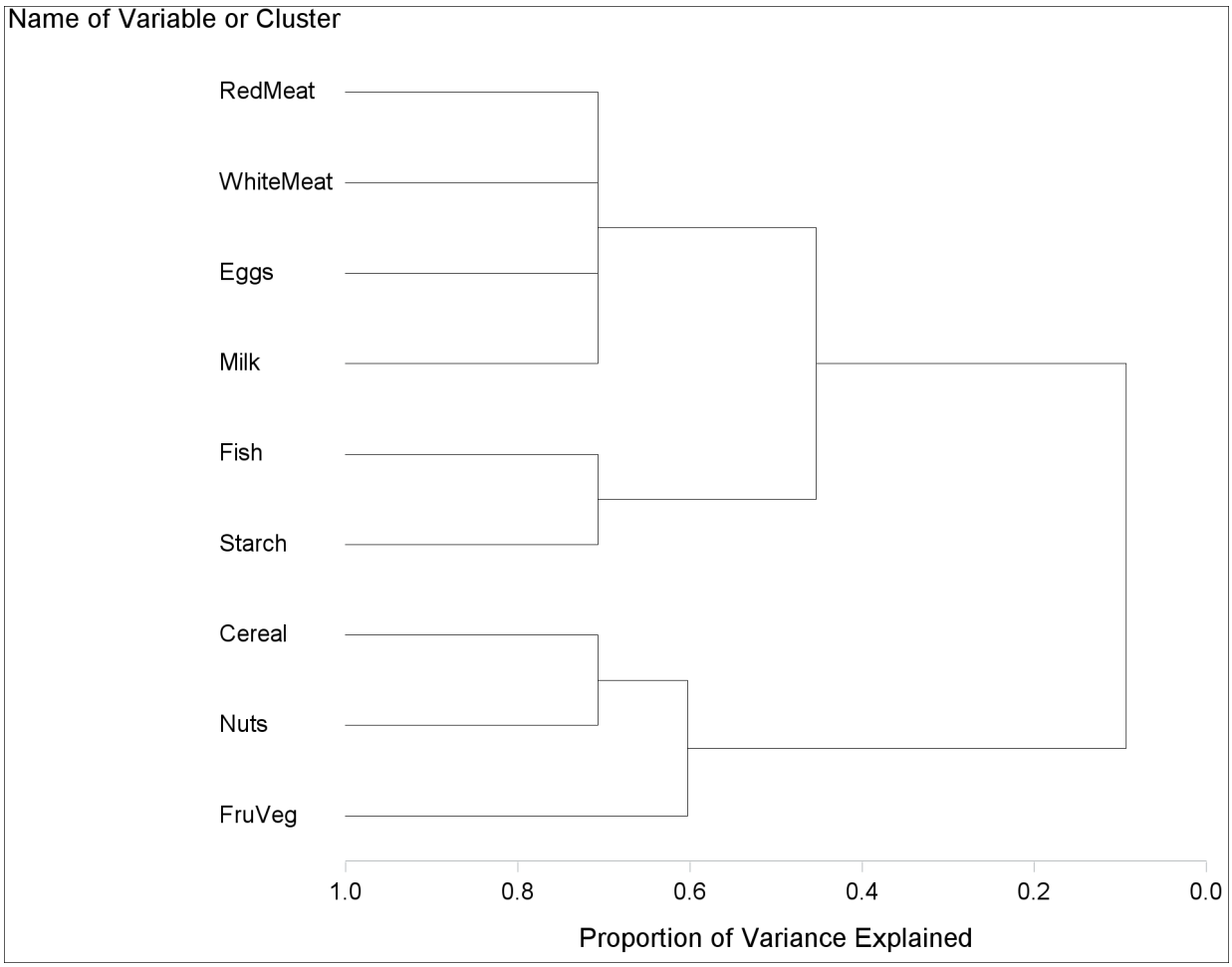
The following statements illustrate how you can specify the numeric variable that defines the height of each node (cluster) in the tree:

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

The ORDER= option in the AXIS1 statement (see *SAS/GRAPH: Reference*) specifies the data values in the order in which they are to appear on the axis. The HORIZONTAL option in the PROC TREE statement 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.

The resulting tree diagram is shown in Figure 94.4.

Figure 94.4 Horizontal Tree Diagram with _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 tree.

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 leftmost 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: TREE Procedure

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 Table 94.1.

Table 94.1 PROC TREE Statement Options

Option	Description
Data Sets	
DATA=	specifies input data set
DOCK=	specifies that small clusters not be counted in OUT= data set
LEVEL=	defines disjoint cluster in OUT= data set
NCLUSTERS=	specifies number of clusters in OUT= data set
OUT=	specifies output data set
ROOT=	displays root of a subtree
Cluster Heights	
HEIGHT=	specifies variable for the height axis
DISSIMILAR	specifies that height values indicate dissimilarity
SIMILAR	specifies that height values indicate similarity
Horizontal Trees	
HORIZONTAL	specifies that height axis be horizontal
Sort Order	
DESCENDING	reverses sort order
SORT	sorts children by HEIGHT variable

Table 94.1 *continued*

Option	Description
Displayed Output	
INC=	specifies increment between tick values
LINEPRINTER	displays tree by using line printer graphics
LIST	displays all nodes in tree
MAXHEIGHT=	specifies maximum value on axis
MINHEIGHT=	specifies minimum value on axis
NOPRINT	suppresses display of tree
NTICK=	specifies number of tick intervals
Graphics	
CFRAME=	specifies color of the frame
DESCRIPTION=	specifies catalog description
GOUT=	specifies catalog name
HAXIS=	customizes horizontal axis
HORDISPLAY=	displays horizontal tree with leaves on right
HPAGES=	specifies number of pages to expand tree horizontally
LINES=	specifies line color and thickness, dots at nodes
NAME=	specifies name of graph in catalog
VAXIS=	customizes vertical axis
VPAGES=	specifies number of pages to expand tree vertically
Line Printer Graphics	
PAGES=	specifies number of pages
POS=	specifies number of column positions
SPACES=	specifies number of spaces between objects
TICKPOS=	specifies number of column positions between ticks
FILLCHAR=	specifies fill character between unjoined leaves
JOINCHAR=	specifies character displayed between joined leaves
LEAFCHAR=	specifies character representing clusters with no children
TREECHAR=	specifies character representing 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 that defines 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

specifies 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 that have 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 displayed 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 that the AXIS*n* statement be 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 section "HEIGHT Statement" on page 7974. The HEIGHT statement can specify any variable in the input data set to be used for the height axis. In rare cases, you might need to specify either the DISSIMILAR option or the SIMILAR option.

HORDISPLAY=RIGHT

specifies that the graph 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

displays the tree diagram with the height axis oriented horizontally. 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, horizontal orientation can make the tree diagram considerably easier to read.

HPAGES=*n1*

specifies that the original graph 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 displayed 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 the character used to represent clusters that have 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 that defines 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 square) and LEVEL=0.9 is specified, then the OUT= data set contains the smallest number of clusters that yields an R square of at least 0.9.

LINEPRINTER

specifies that the tree diagram be displayed using line printer graphics.

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

specifies the color and the thickness of the lines of the tree, and whether a dot is drawn at each leaf node. 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. 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 might 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 multiway 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.

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 that show 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 (see "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 statement 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 that belong to the subtree specified by the ROOT= option.

SIMILAR

SIM

specifies that the values of the HEIGHT variable represent 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 for details.

SPACES=*s*

S=*s*

specifies the number of spaces between objects in 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 the character used 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 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 that are 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 you specify more than one BY statement, only the last one specified is used.

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

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the NOTSORTED or DESCENDING option 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 by using the DATASETS procedure (in Base SAS software).

For more information about BY-group processing, see the discussion in *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base 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 *variable* **;**

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 uses the variable _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 uses the variable _HEIGHT_. If the data set does not contain _HEIGHT_, PROC TREE uses the variable _NCL_. If _NCL_ is not present either, the height of each node is defined to be its path length from the root.

ID Statement

ID *variable* **;**

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 uses the variable _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 *variable* **;**

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

PARENT Statement

PARENT *variable* **;**

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

Details: TREE Procedure

Missing Values

An observation with a missing value for the NAME statement variable is omitted from processing. If the PARENT statement variable has a missing value but the NAME statement 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 statement variable if the ID statement is not used
- the COPY variables
- a numeric variable CLUSTER that takes 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 that gives the value of the NAME statement 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.

The leaves of the tree diagram 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 that contains 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 table names to refer to tables when using the Output Delivery System (ODS) to select tables and create output data sets. The name of PROC TREE's only table is listed in Table 94.2. For more information about ODS, see Chapter 20, "Using the Output Delivery System."

Table 94.2 ODS Tables Produced by PROC TREE

ODS Table Name	Description	Statement	Option
TreeListing	Listing of all nodes in the tree	PROC	LIST

Examples: TREE Procedure

Example 94.1: Mammals' Teeth

The following statements produce a data set that contains the numbers of different kinds of teeth for a variety of mammals:

```
data teeth;
   title 'Mammals'' Teeth';
   input mammal & $16. v1-v8 @@;
   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         2 3 1 1 3 3 3 3   Mole            3 2 1 0 3 3 3 3
Silver Hair Bat   2 3 1 1 2 3 3 3   Pigmy Bat       2 3 1 1 2 2 3 3
House Bat         2 3 1 1 1 2 3 3   Red Bat         1 3 1 1 2 2 3 3
Pika              2 1 0 0 2 2 3 3   Rabbit          2 1 0 0 3 2 3 3
Beaver            1 1 0 0 2 1 3 3   Groundhog       1 1 0 0 2 1 3 3
Gray Squirrel     1 1 0 0 1 1 3 3   House Mouse     1 1 0 0 0 0 3 3
Porcupine         1 1 0 0 1 1 3 3   Wolf            3 3 1 1 4 4 2 3
Bear              3 3 1 1 4 4 2 3   Raccoon         3 3 1 1 4 4 3 2
Marten            3 3 1 1 4 4 1 2   Weasel          3 3 1 1 3 3 1 2
Wolverine         3 3 1 1 4 4 1 2   Badger          3 3 1 1 3 3 1 2
River Otter       3 3 1 1 4 3 1 2   Sea Otter       3 2 1 1 3 3 1 2
Jaguar            3 3 1 1 3 2 1 1   Cougar          3 3 1 1 3 2 1 1
Fur Seal          3 2 1 1 4 4 1 1   Sea Lion        3 2 1 1 4 4 1 1
Grey Seal         3 2 1 1 3 3 2 2   Elephant Seal   2 1 1 1 4 4 1 1
Reindeer          0 4 1 0 3 3 3 3   Elk             0 4 1 0 3 3 3 3
Deer              0 4 0 0 3 3 3 3   Moose           0 4 0 0 3 3 3 3
;
```

The following statements use the CLUSTER procedure to cluster the mammals by average linkage and use ODS Graphics and the TREE procedure to produce a horizontal tree diagram that uses the average-linkage distance as its height axis:

```
ods graphics on;

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

proc tree horizontal;
run;
```

Output 94.1.1 displays the information about how the clusters are joined. For example, the cluster history shows that the mammals `Wolf` and `Bear` form cluster 29, which is merged with `Raccoon` to form cluster 11.

Output 94.1.1 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

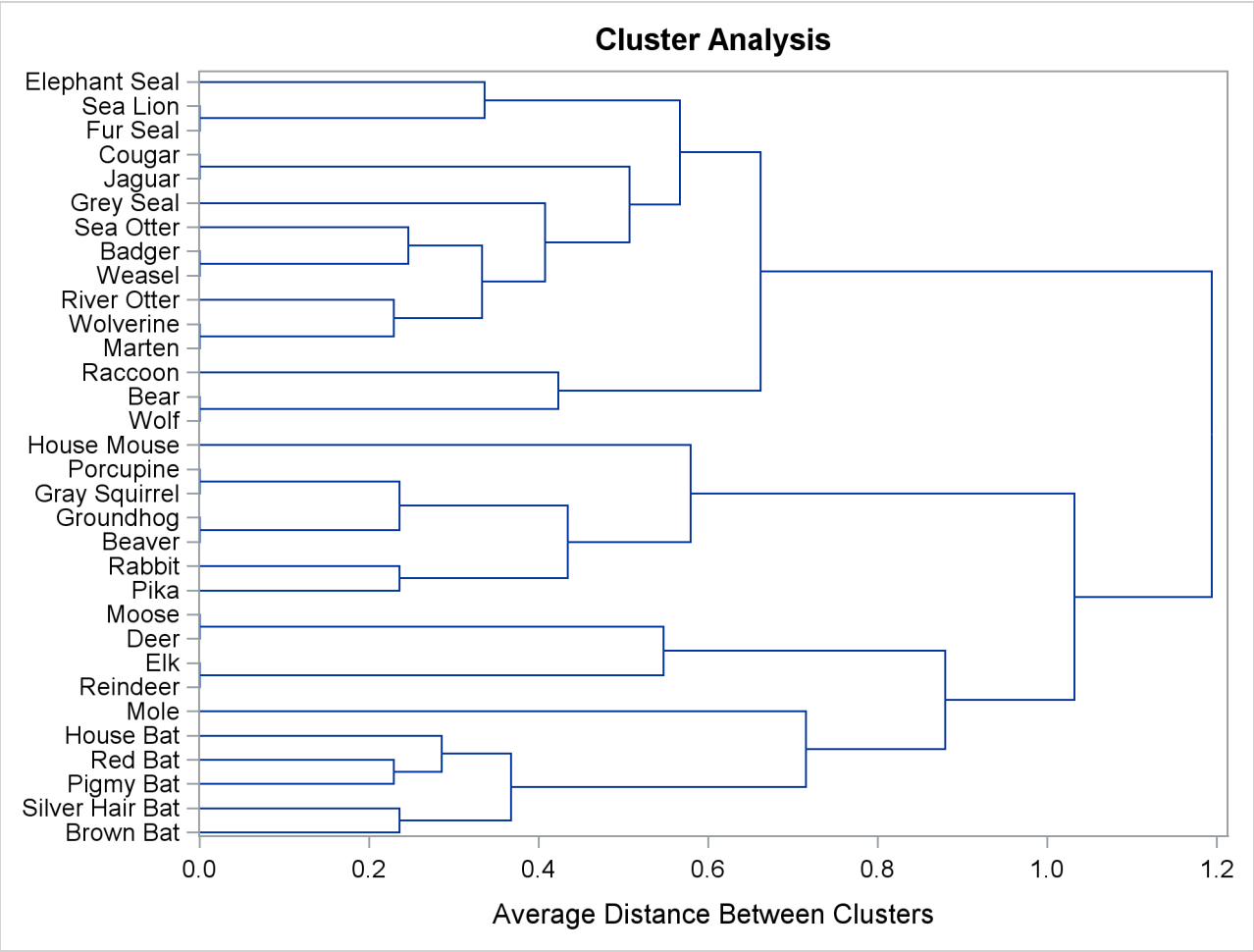
Root-Mean-Square Distance Between Observations 4

Cluster History

NCL	Clusters Joined		Freq	Ps F	PsT2	Norm RMS Dist	T i e
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	

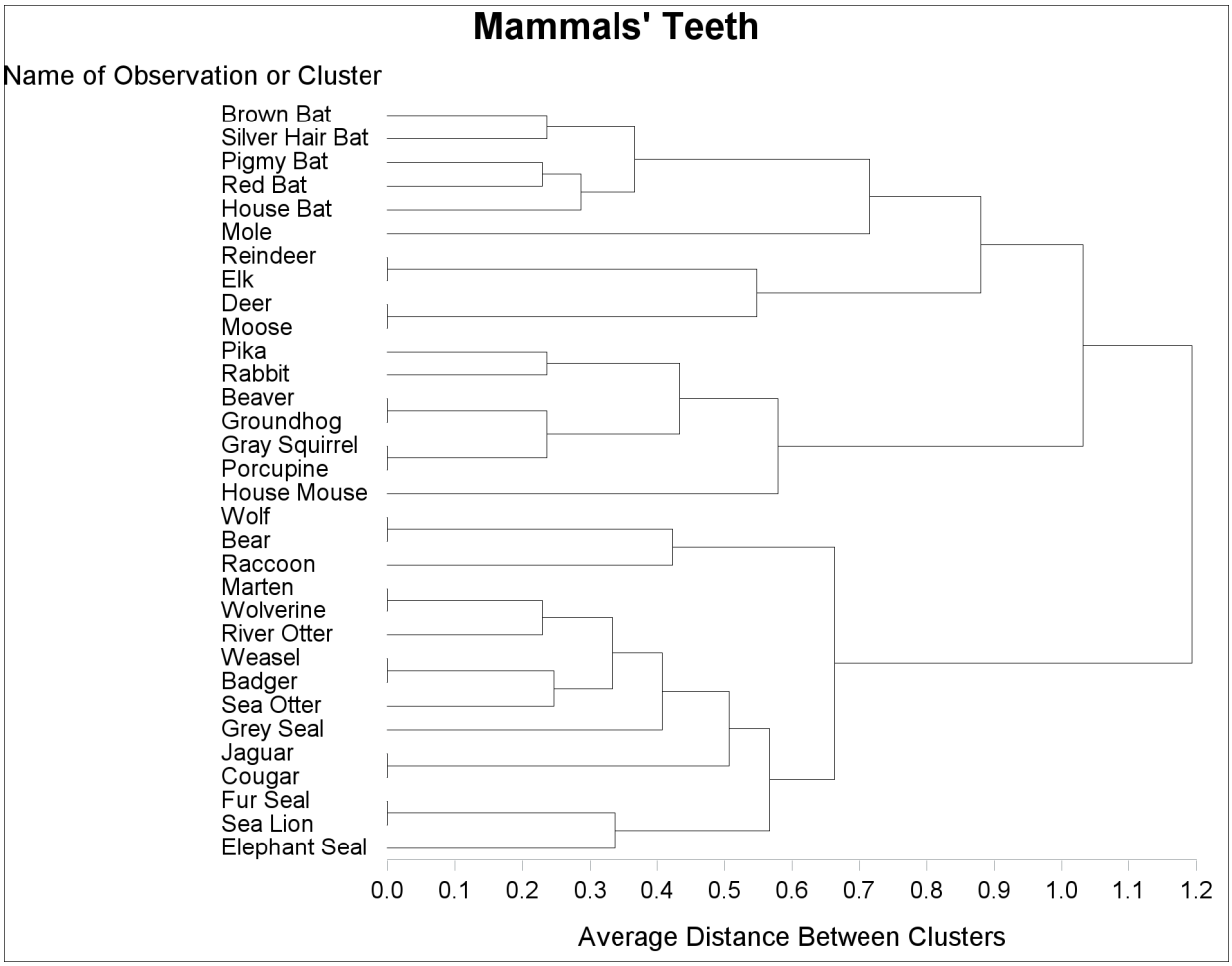
Output 94.1.2 shows the tree diagram produced by PROC CLUSTER.

Output 94.1.2 Dendrogram from PROC CLUSTER



Output 94.1.3 shows the corresponding tree diagram produced by PROC TREE.

Output 94.1.3 Tree Diagram of Mammal Teeth Clusters



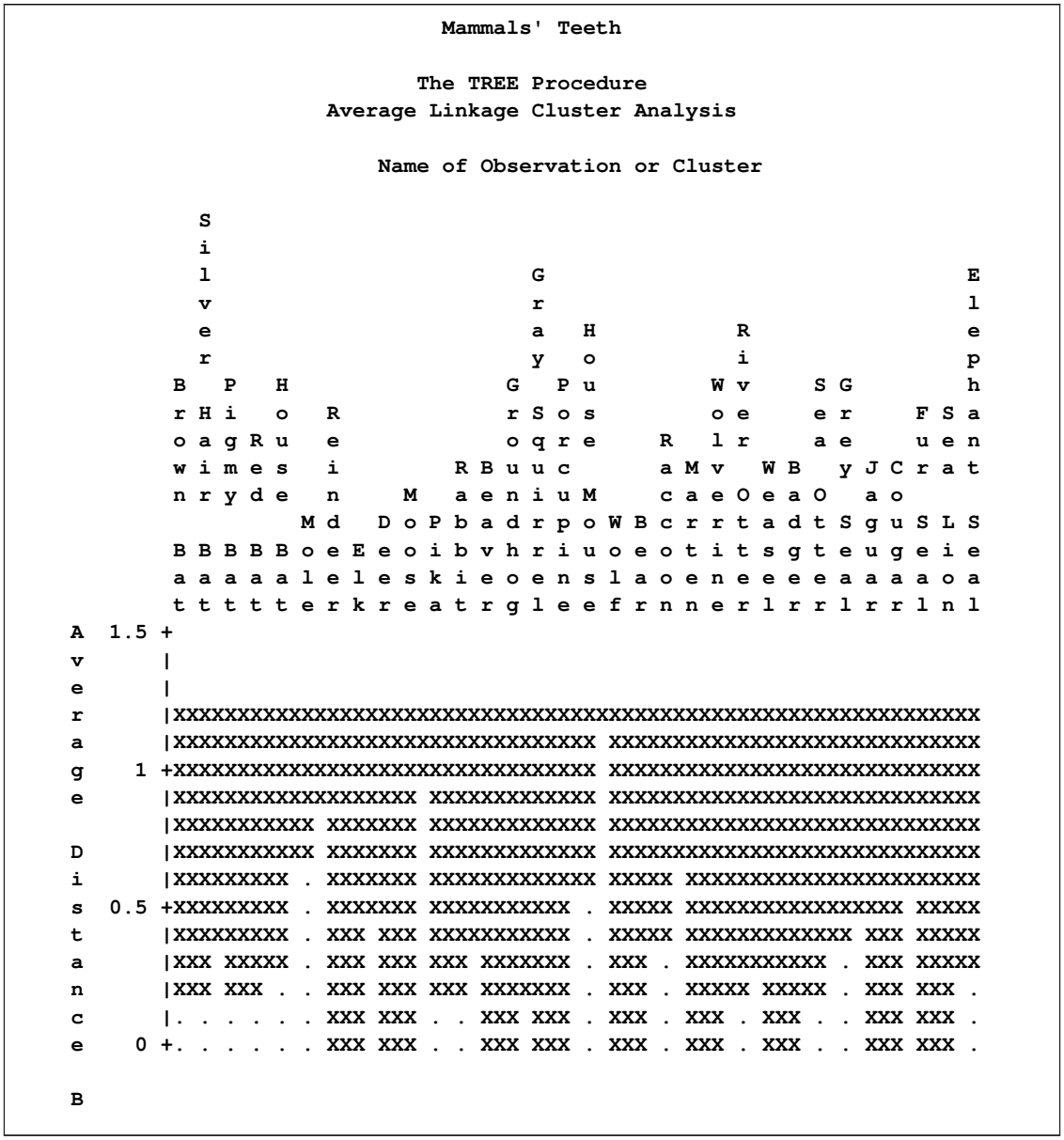
As you view the diagram in Output 94.1.3 from left to right, objects and clusters are progressively joined until a single, all-encompassing cluster is formed at the right (or root) of the tree. Clusters exist at each level of the diagram, and every vertical line connects leaves and branches into progressively 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 mammals `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 mammals `Wolf`, `Bear`, and `Raccoon` form the fifth cluster, and the last cluster contains the mammals `Marten` through `Elephant Seal`.

The following statements create the same tree with line printer graphics in a vertical orientation:

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

The tree is displayed in Output 94.1.4.

Output 94.1.4 PROC TREE with the LINEPRINTER Option



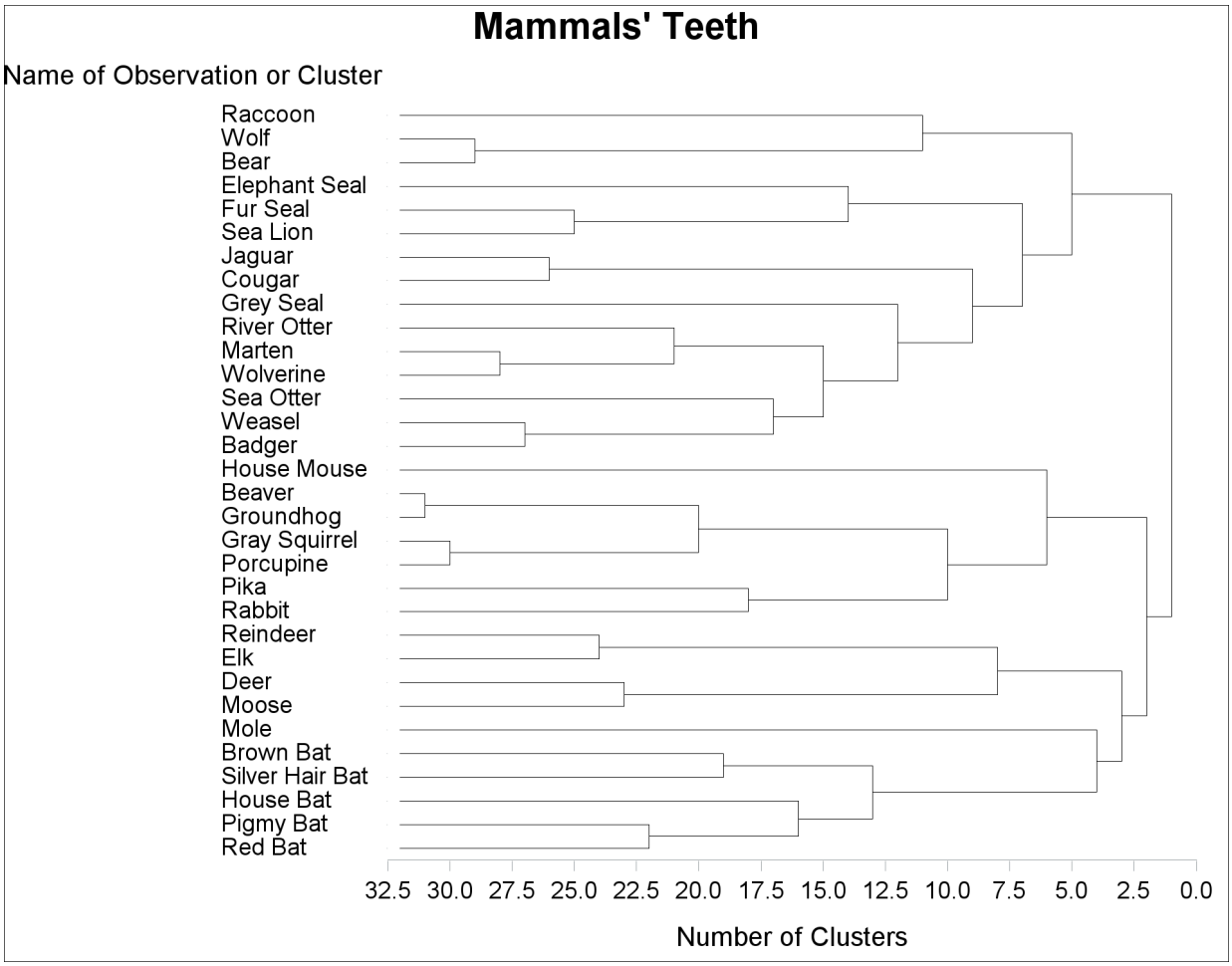
As you look up from the bottom of the diagram, objects and clusters are progressively joined until a single, all-encompassing cluster is formed at the top (or root) of the root. Clusters exist at each level of the diagram. For example, the unbroken line of Xs at the leftmost side of the 0.6 level indicates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

The next statements sort the clusters at each branch in order of formation and use the number of clusters as the height axis:

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

The resulting tree is displayed in Output 94.1.5.

Output 94.1.5 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 94.1.1.

The same clusters as in Output 94.1.3 and Output 94.1.4 can be seen at the six-cluster level of the tree diagram in Output 94.1.5, although the SORT and HEIGHT= options make them appear in a different order.

The following statements create these six clusters and save the result in the output data set part:

```
proc tree noprint out=part nclusters=6;
   id mammal;
   copy v1-v8;
run;

proc sort;
   by cluster;
run;
```

PROC TREE with the NOPRINT option displays no output but creates an output data set that indicates the cluster to which each observation belongs at the six-cluster level in the tree. The following statements print the data set part, with the results shown in Output 94.1.6:

```
proc print label uniform;
   id mammal;
   var v1-v8;
   format v1-v8 1.;
   by cluster;
run;
```

Output 94.1.6 PROC TREE OUT= Data Set

Mammals' Teeth

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

Output 94.1.6 *continued*

Mammals' Teeth

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

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

Output 94.1.6 *continued*

Mammals' Teeth

CLUSTER=3

(continued)

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

Output 94.1.6 *continued*

Mammals' Teeth

CLUSTER=4

mammal	Right Top Incisors	Right Bottom Incisors	Right Top Canines	Right Bottom Canines
Reindeer	0	4	1	0
Elk	0	4	1	0
Deer	0	4	0	0
Moose	0	4	0	0

mammal	Right Top Premolars	Right Bottom Premolars	Right Top Molars	Right Bottom Molars
Reindeer	3	3	3	3
Elk	3	3	3	3
Deer	3	3	3	3
Moose	3	3	3	3

CLUSTER=5

mammal	Right Top Incisors	Right Bottom Incisors	Right Top Canines	Right Bottom Canines
Pigmy Bat	2	3	1	1
Red Bat	1	3	1	1

mammal	Right Top Premolars	Right Bottom Premolars	Right Top Molars	Right Bottom Molars
Pigmy Bat	2	2	3	3
Red Bat	2	2	3	3

Output 94.1.6 *continued*

Mammals' Teeth

CLUSTER=5

(continued)

mammal	Right Top Incisors	Right Bottom Incisors	Right Top Canines	Right Bottom Canines
Brown Bat	2	3	1	1
Silver Hair Bat	2	3	1	1
House Bat	2	3	1	1

mammal	Right Top Premolars	Right Bottom Premolars	Right Top Molars	Right Bottom Molars
Brown Bat	3	3	3	3
Silver Hair Bat	2	3	3	3
House Bat	1	2	3	3

CLUSTER=6

mammal	Right Top Incisors	Right Bottom Incisors	Right Top Canines	Right Bottom Canines
Mole	3	2	1	0

mammal	Right Top Premolars	Right Bottom Premolars	Right Top Molars	Right Bottom Molars
Mole	3	3	3	3

Example 94.2: Iris Data

Fisher (1936)'s iris data give sepal and petal dimensions for three different species of iris. The data, which are available in the Sashelp library, are clustered by *k*th-nearest-neighbor density linkage by 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 the results shown in Output 94.2.1 and Output 94.2.2:

```
title 'Fisher (1936) Iris Data';
ods graphics on;

proc cluster data=sashelp.iris method=twostage print=10
             outtree=tree k=8 noeigen;
   var SepalLength SepalWidth PetalLength PetalWidth;
   copy Species;
run;

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

The PAGES=1 option specifies that the tree diagram extend 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 94.2.2.

Output 94.2.1 Clustering of Fisher's Iris Data

Fisher (1936) Iris Data

The CLUSTER Procedure
Two-Stage Density Linkage Clustering

K = 8

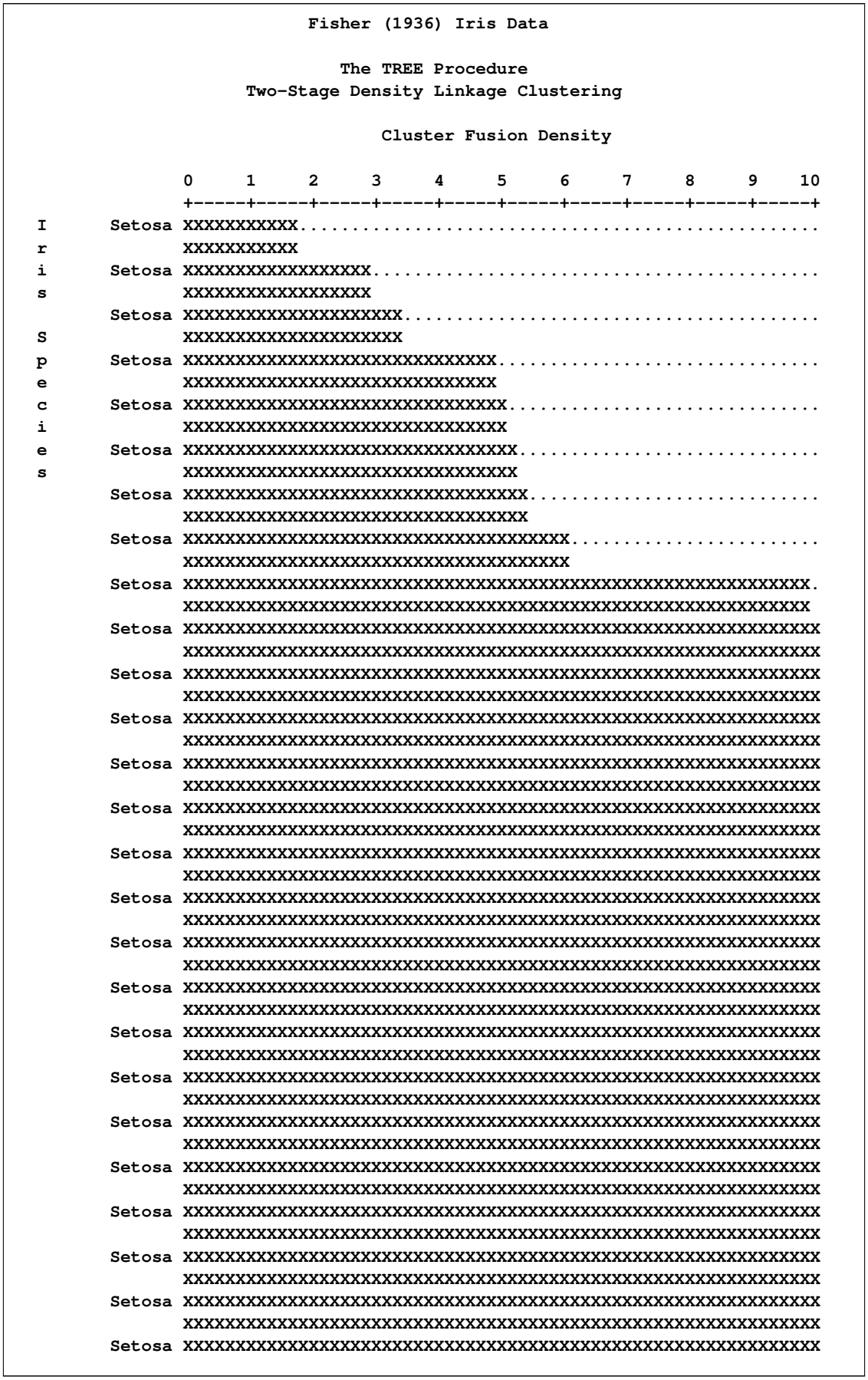
Root-Mean-Square Total-Sample Standard Deviation 10.69224

Cluster History

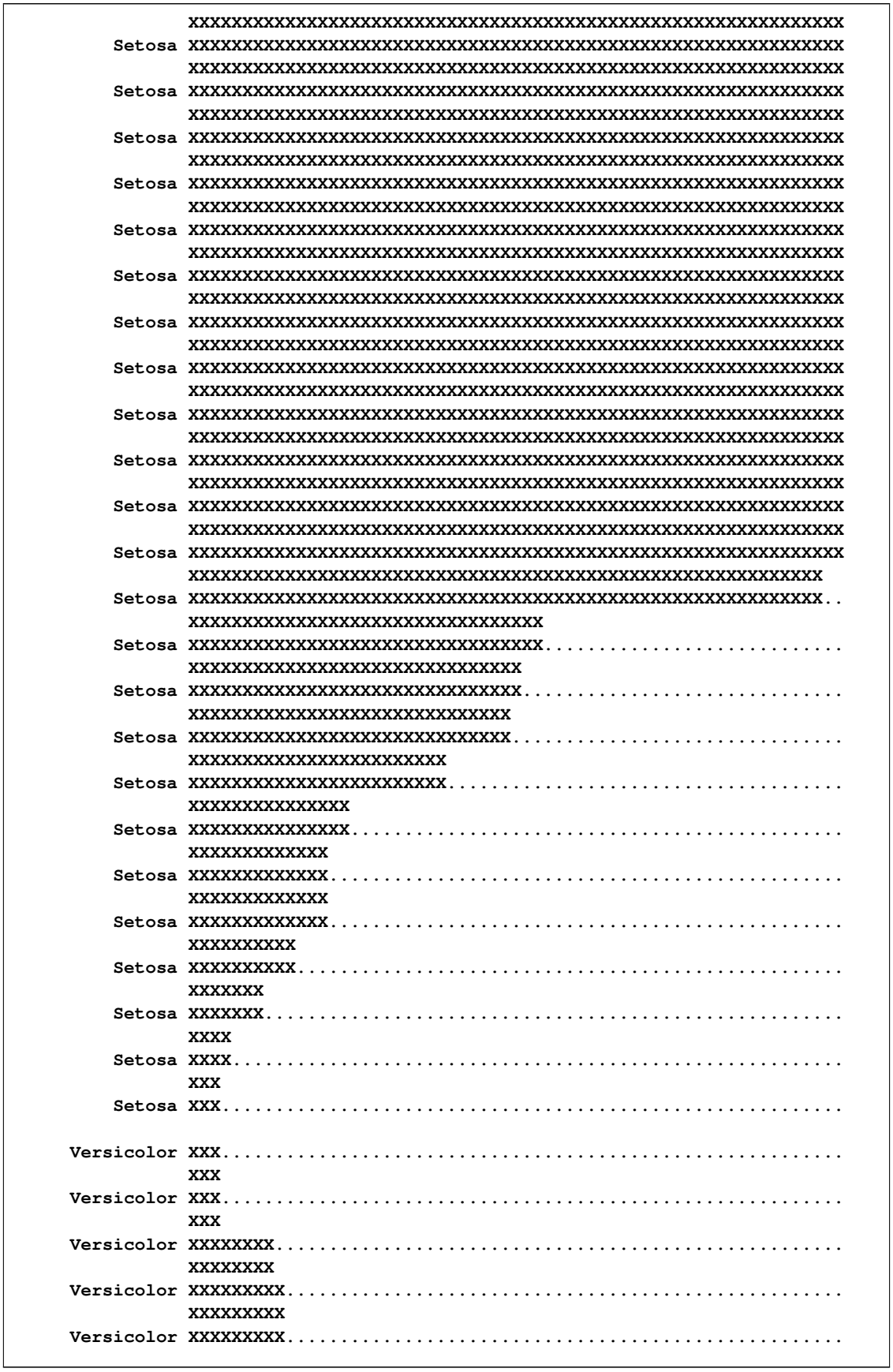
Number of Clusters	Clusters Joined		Freq	Normalized Fusion Density	Maximum Density in Each Cluster Lesser	Greater	Tie
10	CL11	OB79	48	0.2879	0.1479	8.3678	
9	CL13	OB112	46	0.2802	0.2005	3.5156	
8	CL10	OB113	49	0.2699	0.1372	8.3678	
7	CL8	OB91	50	0.2586	0.1372	8.3678	
6	CL9	OB120	47	0.1412	0.0832	3.5156	
5	CL6	OB118	48	0.107	0.0605	3.5156	
4	CL5	OB110	49	0.0969	0.0541	3.5156	
3	CL4	OB135	50	0.0715	0.0370	3.5156	
2	CL7	CL3	100	2.6277	3.5156	8.3678	

3 modal clusters have been formed.

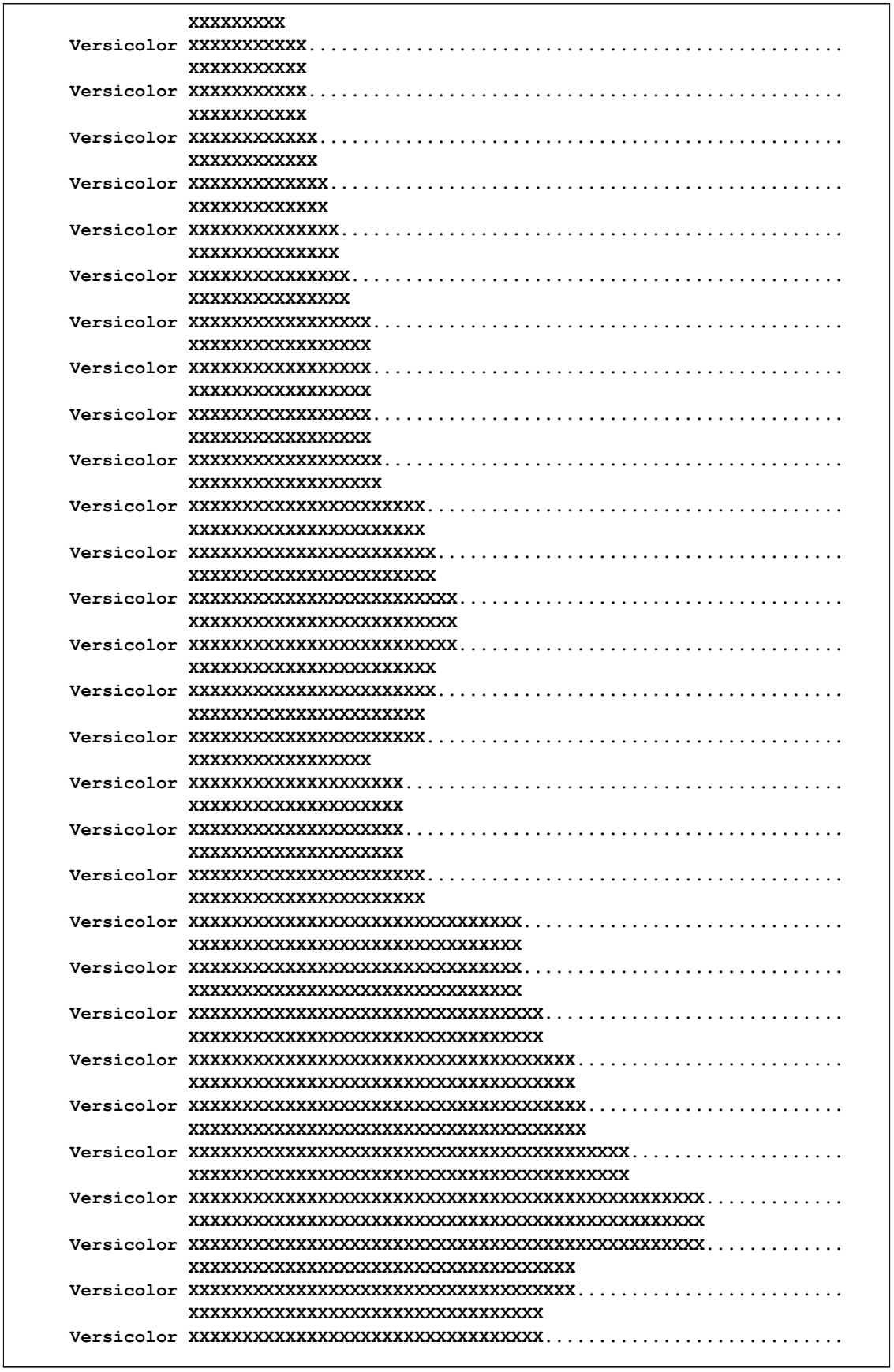
Output 94.2.2 Horizontal Tree for Fisher's Iris Data


Fisher (1936) Iris Data
The TREE Procedure
Two-Stage Density Linkage Clustering
Cluster Fusion Density
0 1 2 3 4 5 6 7 8 9 10
Iris Species
Setosa
Setosa
Setosa

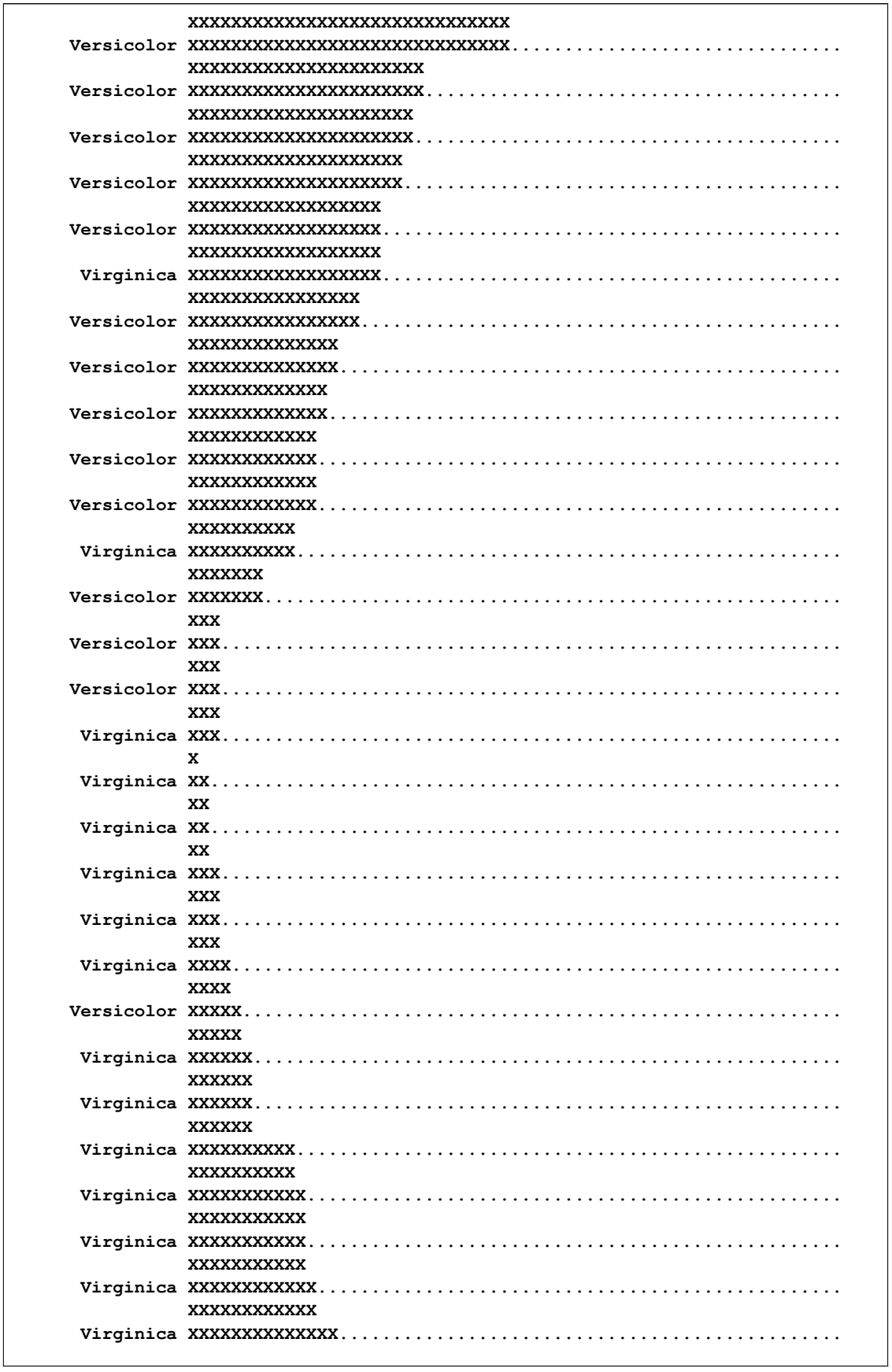
Output 94.2.2 *continued*



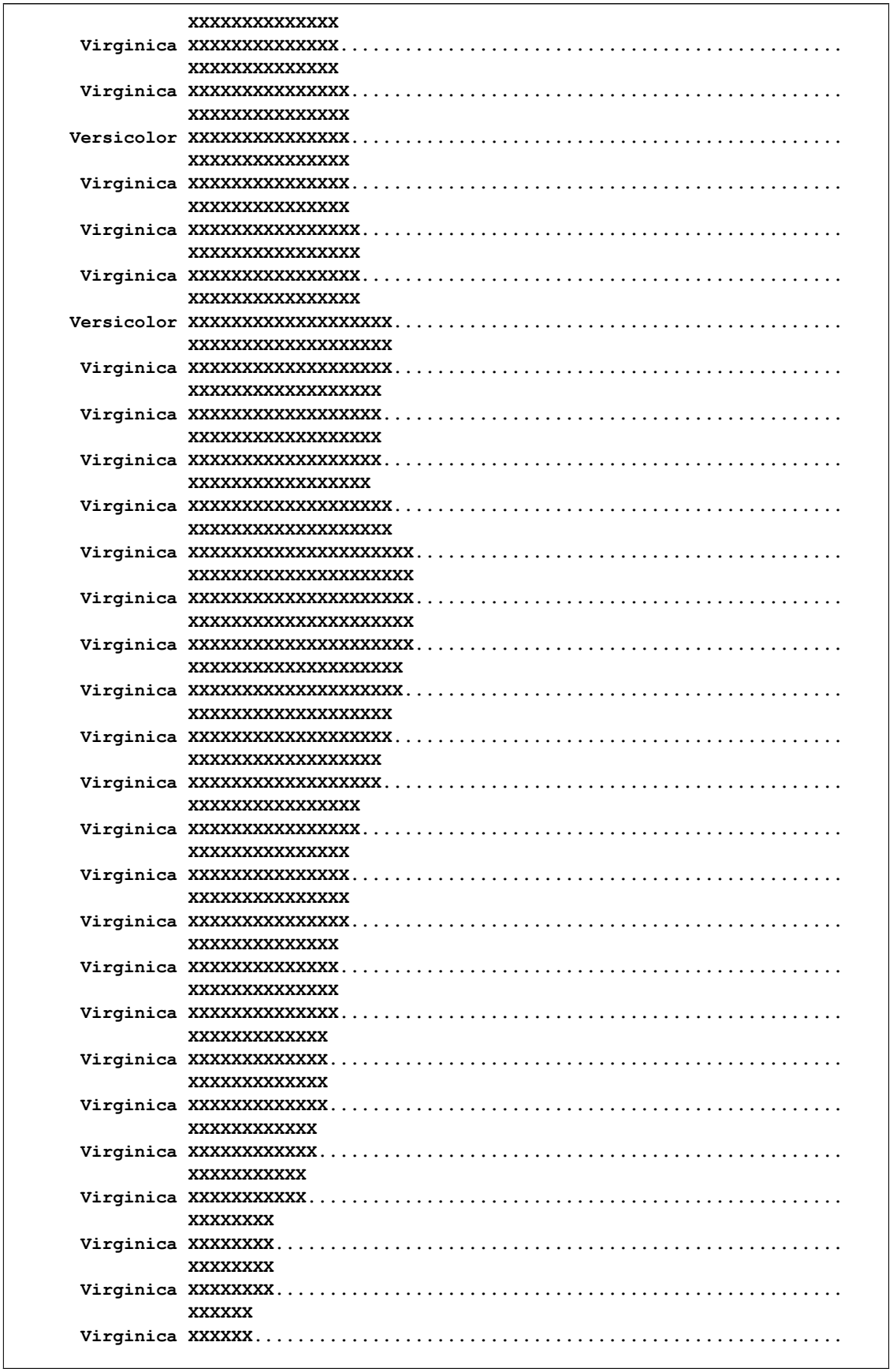
Output 94.2.2 *continued*



Output 94.2.2 *continued*



Output 94.2.2 *continued*



Output 94.2.2 *continued*

```
          XXXXXX
Virginica XXXXXX.....................................................
          XXXXX
Virginica XXXXX.....................................................
          XXXXX
Virginica XXXXX.....................................................
          XXXX
Virginica XXXX......................................................
          XXXX
Virginica XXXX......................................................
          XXX
Virginica XXX.......................................................
          XX
Virginica XX........................................................
          X
Virginica X.........................................................
```

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.

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.

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

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.

Subject Index

C

cluster analysis
    tree diagrams, 7960
CLUSTER procedure, *see also* TREE procedure

D

dendrogram, 7960

M

missing values
    TREE procedure, 7975

O

OUT= data sets
    TREE procedure, 7975
output data sets
    TREE procedure, 7975
output table names
    TREE procedure, 7976

P

phenogram, 7960

T

tree diagram
    binary tree, 7960
    branch, 7960
    children, 7960
    definitions, 7960
    leaves, 7960
    node, 7960
    parent, 7960
    root, 7960
tree diagrams
    cluster analysis, 7960
TREE procedure, 7960
    missing values, 7975
    OUT= data sets, 7975
    output data sets, 7975
    output table names, 7976

V

VARCLUS procedure, *see also* TREE procedure

Syntax Index

B

BY statement
    TREE procedure, 7973

C

CFRAME= option
    PROC TREE statement, 7968
COPY statement
    TREE procedure, 7973

D

DATA= option
    PROC TREE statement, 7968
DESCENDING option
    PROC TREE statement, 7968
DESCRIPTION= option
    PROC TREE statement, 7968
DISSIMILAR option
    PROC TREE statement, 7969
DOCK= option
    PROC TREE statement, 7969

F

FILLCHAR= option
    PROC TREE statement, 7969
FREQ statement
    TREE procedure, 7973

G

GOUT= option
    PROC TREE statement, 7969

H

HAXIS= option
    PROC TREE statement, 7969
HEIGHT statement
    TREE procedure, 7974
HEIGHT= option
    PROC TREE statement, 7969
HORDISPLAY= option
    PROC TREE statement, 7969
HORIZONTAL option
    PROC TREE statement, 7970
HPAGES= option
    PROC TREE statement, 7970

I

ID statement
    TREE procedure, 7974
INC= option
    PROC TREE statement, 7970

J

JOINCHAR= option
    PROC TREE statement, 7970

L

LEAFCHAR= option
    PROC TREE statement, 7970
LEVEL= option
    PROC TREE statement, 7970
LINEPRINTER option
    PROC TREE statement, 7970
LINES= option
    PROC TREE statement, 7970
LIST option
    PROC TREE statement, 7970

M

MAXHEIGHT= option
    PROC TREE statement, 7971
MINHEIGHT= option
    PROC TREE statement, 7971

N

NAME statement
    TREE procedure, 7974
NAME= option
    PROC TREE statement, 7971
NCLUSTERS= option
    PROC TREE statement, 7971
NOPRINT option
    PROC TREE statement, 7971
NTICK= option
    PROC TREE statement, 7971

O

OUT= option
PROC TREE statement, 7971

P

PAGES= option
PROC TREE statement, 7971
PARENT statement
TREE procedure, 7974
POS= option
PROC TREE statement, 7971
PROC TREE statement, *see* TREE procedure

R

ROOT= option
PROC TREE statement, 7972

S

SIMILAR option
PROC TREE statement, 7972
SORT option
PROC TREE statement, 7972
SPACES= option
PROC TREE statement, 7972

T

TICKPOS= option
PROC TREE statement, 7972
TREE procedure
syntax, 7967
TREE procedure, BY statement, 7973
TREE procedure, COPY statement, 7973
TREE procedure, FREQ statement, 7973
TREE procedure, HEIGHT statement, 7974
TREE procedure, ID statement, 7974
TREE procedure, NAME statement, 7974
TREE procedure, PARENT statement, 7974
TREE procedure, PROC TREE statement, 7967
CFRAME= option, 7968
DATA= option, 7968
DESCENDING option, 7968
DESCRIPTION= option, 7968
DISSIMILAR option, 7969
DOCK= option, 7969
FILLCHAR= option, 7969
GOUT= option, 7969
HAXIS= option, 7969
HEIGHT= option, 7969
HORDISPLAY= option, 7969
HORIZONTAL option, 7970
HPAGES= option, 7970
INC= option, 7970
JOINCHAR= option, 7970
LEAFCHAR= option, 7970
LEVEL= option, 7970
LINEPRINTER option, 7970
LINES= option, 7970
LIST option, 7970
MAXHEIGHT= option, 7971
MINHEIGHT= option, 7971
NAME= option, 7971
NCLUSTERS= option, 7971
NOPRINT option, 7971
NTICK= option, 7971
OUT= option, 7971
PAGES= option, 7971
POS= option, 7971
ROOT= option, 7972
SIMILAR option, 7972
SORT option, 7972
SPACES= option, 7972
TICKPOS= option, 7972
TREECHAR= option, 7972
VAXIS= option, 7972
VPAGES= option, 7972
TREECHAR= option
PROC TREE statement, 7972

V

VAXIS= option
PROC TREE statement, 7972
VPAGES= option
PROC TREE statement, 7972

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.

SAS® Publishing Delivers!

Whether you are new to the work force or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart. Visit us online at support.sas.com/bookstore.

SAS® Press

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from SAS Press. Written by experienced SAS professionals from around the world, SAS Press books deliver real-world insights on a broad range of topics for all skill levels.

support.sas.com/saspress

SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information: SAS documentation. We currently produce the following types of reference documentation to improve your work experience:

- Online help that is built into the software.
- Tutorials that are integrated into the product.
- Reference documentation delivered in HTML and PDF – **free** on the Web.
- Hard-copy books.

support.sas.com/publishing

SAS® Publishing News

Subscribe to SAS Publishing News to receive up-to-date information about all new SAS titles, author podcasts, and new Web site features via e-mail. Complete instructions on how to subscribe, as well as access to past issues, are available at our Web site.

support.sas.com/spn

sas® THE POWER TO KNOW®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2009 SAS Institute Inc. All rights reserved. 518177_1US.0109