Cluster Analysis Notes

Introduction

The objective of cluster analysis (CA) is to uncover structure in a dataset by aggregating similar rows into groups called clusters. Therefore, CA contrasts with PCA (principal components analysis) in finding simplification among the rows instead of among the columns of the dataset. CA and PCA both attempt to simplify a dataset and do so without supervision. However, PCA actually can reduce the number of columns, whereas CA does not reduce the number of rows. Moreover, PCA changes the values of the variables, whereas CA does not. But for some purposes, each cluster may be treated as if each were a single entity (as in a marketing campaign, for example) and thus CA effectively reduces the number of entities to understand. Grouping in CA is based on various measures of similarity or dissimilarity among the rows.

The two major problems of CA are to determine the number of clusters and to assign the rows to clusters.

Some examples of applications of CA:

- Marketing: Market segmentation
- Politics: Analysis of voter blocs
- Psychology, psychiatry: Definition of personality types, mental diseases
- Biology: Reconstructing the evolutionary "tree of life"
- Linguistics: Reconstructing ancestral languages
- Anthropology: Mapping patterns of human migration and settlement
- Environment: Identifying weather patterns and their propensity for air pollution

You should understand clearly that there is *no supervision* in CA. What does that mean? It means that there is no training set of rows that have been correctly assigned to clusters, from which one can infer a rule for assigning unclassified rows to clusters. You do not know how many clusters there are, nor do you know *any* members of *any* cluster. Yet you must still figure out how many clusters there are and assign each row to a cluster.

There are parallel *supervised* statistical techniques that do infer rules to assign rows to clusters. If you know how many groups there are and you have a training set of rows that have been correctly assigned to groups, you have supervision and you can infer rules for future assignment of rows to groups. Supervised assignment of rows to groups is called *classification*. Two examples of classification methodologies are discriminant analysis and logistic regression.

> Example: Suppose that an insurance company would like to develop a rule to determine which accident claims are likely to be fraudulent and which non-fraudulent. Claims classified as potentially fraudulent can receive closer scrutiny than other claims. If the insurer has a set of claims that it has investigated and found to be fraudulent and a set of non-fraudulent claims, the insurer can use the two groups as a training set to infer a rule for assigning new claims to the two groups. Essentially, the insurer would take the training set and "regress" whether or not the claims are fraudulent on a set of predictor variables and produce an equation to score new claims for fraudulent potential. New claims that receive high scores would be assigned to the fraudulent group; new claims that receive low scores to the non-fraudulent group. The insurer could then

spend more effort checking out the putatively fraudulent claims. All this would be supervised learning.
By contrast, in CA, there would be no set of correctly assigned fraudulent and non-fraudulent claims. The insurer would just have a set of claims. The number of groups would be unknown. It would not even be known if any of the groups would correspond to fraudulent claims. Fraudulent claims could be dispersed randomly among whatever natural groups there may be. What a cluster analysis would do is to find clusters of similar claim characteristics so that the insurer could start to understand the different types of claims that it experiences. Some of those clusters may contain more fraudulent claims than other clusters. That may help with the specific problem of identifying likely fraudulent claims.

Example. An online content provider is often compensated by advertisers every time a viewer clicks on an ad displayed on a page seen by the viewer – whether or not the clicker subsequently buys anything. Clicking on an ad is called a “click-through”, or CT. Content providers entice viewers to click through by bribing them with articles thought to be of interest (“click bait”). A bribe article is then surrounded or imbedded with related ads. For example, a viewer who clicks on an article about reducing car expenses may be presented an ad for a new car. The content provider has two predictions to make: Will the viewer click on the article? If the viewer clicks on the article, will the viewer click through to the ad?
Supervised learning can help by analyzing the training set of past data of articles and ads that viewers have or have not clicked on. There are two groups: CTs and non-CTs. The goal is to use the characteristics of clickers and non-clickers to predict the CT rate for articles and ads.
Unsupervised learning can help by discovering naturally occurring clusters among content viewers. E.g., finding a cluster of young males interested in sports and cars and another cluster of older females interested in fine clothes and beauty products suggests different articles and ads that might be presented to each cluster in order to increase the probability of a CT. If there is no cluster of viewers interested in movie stars, then that suggests de-emphasizing articles and ads that feature them.

Because of the lack of supervision, CA is a more difficult challenge than classification. There is not a great deal of theoretical foundation for the clustering procedures that have been proposed. Many clustering algorithms are *ad hoc* procedures that have been proposed because they merely *seem* intuitively plausible. There has been a fair amount of simulation and empirical work to try to learn how competing clustering methodologies are likely to work. These clustering methodologies have originated in many different disciplines, as the list of examples on page 1 of these notes suggests. CA became a hot topic after the publication of *Principles of Numerical Taxonomy* (1963) by biologists Robert Sokal and Peter Sneath. They introduced numerical algorithms for grouping species by common descent rather than by subjective judgment of anatomical similarities.

There are five basic steps in all CA studies:
1) Select the observations to be clustered.
2) Define the variables to use in clustering the observations.
3) Compute similarities (or dissimilarities) among the observations.
4) Create groups (clusters) of similar observations.
5) Validate the resulting clusters.

Cautions about CA:

1) Most clustering techniques are simple procedures without much underlying statistical theory.
2) Clustering techniques evolved from many disciplines and bear the biases of those disciplines.
3) Different clustering techniques yield different solutions for the same data set.
4) The strategy of CA is structure-seeking, but its operation is structure-imposing.

Terminology and Notation

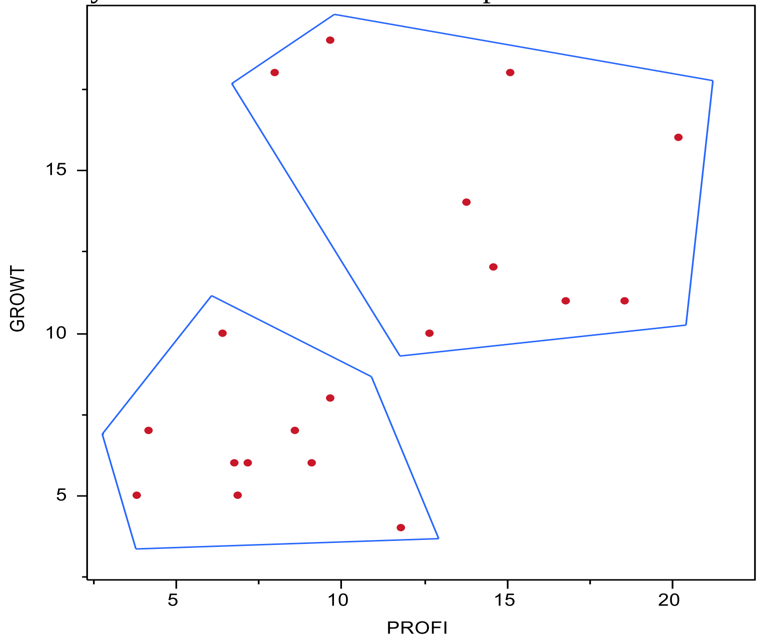
*Definition: **Clusters** are continuous regions of variable space containing a relatively high density of data points separated from other high density regions by areas containing a relatively low density of data points.*

The things being classified are variously called **cases**, **observations**, **objects**, **entities**, **patterns**.

The aspects of the entities used to assess their similarities/dissimilarities are variously called **variables**, **attributes**, **characters**, **features**.

For example, in Figure 1, 19 companies are plotted by their profitability and growth rate. It is easy to see at least two clusters of companies. In the upper right are companies of high profitability and growth rate. In the lower left is a denser set of companies of low profitability and growth rate.

Figure 1. Profitability vs Growth Rates for 19 Companies in the Forbes 500



There are some characteristics of clusters that can be quantified when cases are plotted as points in p-dimensional variable space:

1) **Density** -- the concentration of cases is high within a cluster
2) **Variance** -- the dispersion from the center of a cluster is small for the cases in the cluster
3) **Dimension** -- the size or radius of the cluster is small
4) **Shape** -- clusters are typically ellipsoidal
5) **Separation** -- the degree to which clusters overlap or are disconnected.

These characteristics can be used to help define algorithms to group objects into clusters.

We can think of the entities as the rows in a data matrix, with the attributes as the columns. Thus the data matrix corresponds to the standard SAS data set, or Excel spreadsheet, with observations as rows and variables as columns. Let *n* denote the number of rows (entities) and *p* denote the number of columns (variables). The data matrix is *n* x *p*.

From the data matrix, a **similarity** (or **dissimilarity**) **matrix** will be computed. The entry in row *i* and column *j* of the similarity matrix shows how similar (dissimilar) entity *i* and entity *j* are. Thus, the similarity matrix must be *n* x *n*. Often, correlation is used to measure similarity. If correlation is used, the resulting similarity matrix is very different from the ordinary correlation matrix for the same *n* x *p* data matrix. The ordinary correlation matrix is *p* x *p*. The entry in row *i* and column *j* of the ordinary correlation matrix shows how similar *variable i* and *variable j* are. An example is the correlation between the rent and the area of a set of apartments. The set of all rents would be correlated with the set of all areas. By contrast, the entry in row *i* and column *j* of a similarity matrix shows how similar *entity i* and *entity j* are. An example would be the correlation between apartment 1 and apartment 2. The rent, area, age, number of rooms, etc. of apartment 1 would be correlated with the values of the same variables for apartment 2. This is an unusual and controversial type of correlation because the numbers being correlated are of very different units. An *n* x *n* similarity matrix shows how similar the rows of the *n* x *p* data matrix are, whereas the ordinary *p* x *p* correlation matrix shows how similar the columns of the *n* x *p* data matrix are. The similarity matrix is also called a **proximity matrix**. If desired, a dissimilarity matrix often can be created from a similarity matrix by simple means, such as subtracting correlations from ±1.

Similarities and Dissimilarities

Two concepts underlie most techniques for calculating (dis)similarities among cases that are to be clustered: (A) correlation and (B) distance.

(A) **CORRELATION COEFFICIENTS** for similarities.[1] This is a strange kind of correlation coefficient. It is a correlation between observations instead of between variables. It uses the same formula as the ordinary correlation coefficient, but the formula is applied to two rows of the *n* x *p* data matrix instead of two columns. This results in mixing together numbers measured in possibly very different units to find means, standard deviations, and covariances.

[1] SAS does not use correlations as similarity measures in its Q-mode CA procedures CLUSTER, FASTCLUS, TREE.

Ex: Jim's (age, salary, family size, home value) data vector of (40 years, $35000, 3 people, $80000) would be correlated with Jane's (age, salary, family size, home value) of (30 years, $50000, 4 people, $60000) to yield 0.9295166 as the similarity between Jim and Jane.

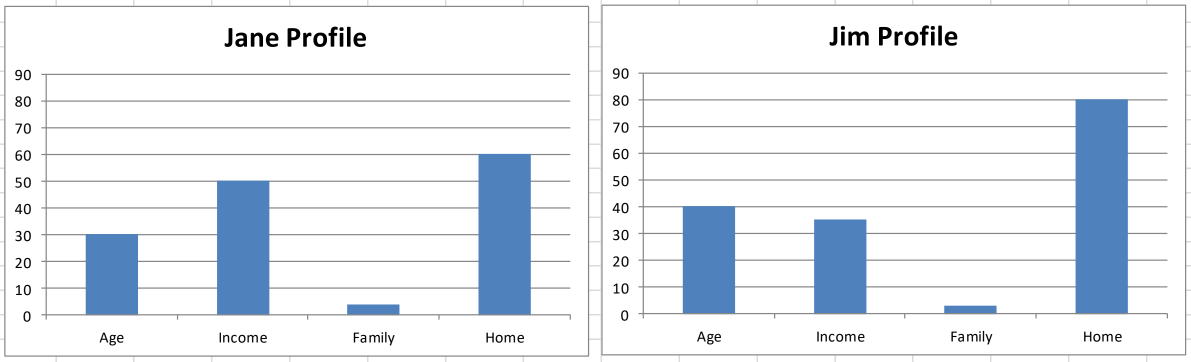
Besides the issue of potentially meaningless units resulting from composites across unrelated variables, correlations as similarities have the difficulty that small variations in large numbers (such as salary) may overwhelm proportionately much larger variations in smaller numbers (such as family size) in the similarity measure; therefore, the clustering may be driven by, and cater to, variables with large values, such as salary.

Ex: In the preceding example, if Jane's family size increases by 25%, her correlation with Jim increases microscopically to 0.9295170. But if her salary increases by 25%, her correlation increases to 0.9736926.

An obvious possible remedy is to standardize all variables before clustering, but this suggestion is controversial because it implies that all variables are equally important in assessing similarity. Another potential problem results from the selection of highly correlated variables as the basis for clustering. For example, if X_1, X_2, X_3 are three highly correlated variables used with an unrelated X_4 to compute a similarity measure, the effect is almost the same as using just X_1 and X_4, but giving X_1 three times the weight of X_4. So selecting the variables to use for CA could bias results. Principal Components Analysis and Factor Analysis can be used prior to CA to reduce the number of variables and produce uncorrelated "variables", but there is controversy over this, as well, because PCA and FA may tend to blur together two distinct clusters that are actually widely separated.[2] The problems discussed in this paragraph are also faced by distance (dissimilarity) measures discussed in (B), below.

Profile is a useful concept for comparing entities. A profile of a case is simply a vector of attribute values for a case -- in other words, the profile of a case is its row in the *n* x *p* data matrix for that case. A profile is usually displayed as a graph with the variable names listed on the horizontal axis and the scores (values) of the variables shown vertically as bars and/or connected dots. CA can be thought of as a way to group cases with similar profiles.

Figure 2. Two illustrative profiles (income and home value are in $1000s)



[2] Using the complete set of principal components preserves the geometry and therefore the clusters. However, using a subset of principal components changes the geometry.

The similarity between profiles can be decomposed into three parts:

1) **Shape** -- the pattern of dips and rises of the profile as you look across the variables
2) **Scatter** -- the dispersion of the scores around their average
3) **Elevation** -- the mean score of the case over all of the variables.

The correlation coefficient can be enlisted to measure the similarity between two profiles. But it is sensitive only to *shape*. It is unaffected by differences between two profiles in terms of *scatter* and/or *elevation*. Two profiles with the same shape but different scatter and/or elevation will have correlation coefficient = 1.00. If you add the same number to every component of a profile, and/or if you multiply every component of a profile by the same positive number, the correlation with any other profile remains unaffected.

Correlation measures for similarities are seldom used in practice.

(B) **DISTANCE MEASURES** for dissimilarities. These are the most common (dis)similarity measures. Common measures of distance between cases are:

1) Euclidean distance (L^2 distance) = $\sqrt{\sum_{k=1}^{p}(x_{ik}-x_{jk})^2}$
2) "Manhattan" (city-block – L^1) distance = $\sum_{k=1}^{p}|x_{ik}-x_{jk}|$
3) Mahalanobis distance = $(\mathbf{x}_i-\mathbf{x}_j)'\mathbf{S}^{-1}(\mathbf{x}_i-\mathbf{x}_j)$

where x_{ik} is the score of case *i* on variable *k*, x_{jk} is the score of case *j* on variable *k*, $(\mathbf{x}_i-\mathbf{x}_j)'$ is the 1 x *p* row vector of differences between row *i* and row *j* of the data matrix, and $\mathbf{S}^{-1}$ is the inverse of the *p* x *p* covariance matrix of the *variables* (not the cases).[3]

Ex: Given Jim's (age, salary, family size, home value) data vector of (40 years, $35000, 3 people, $80000) and Jane's (age, salary, family size, home value) of (30 years, $50000, 4 people, $60000):

1) The Euclidean distance between Jim and Jane = sqrt((40-30)^2 + (35000-50000)^2 + (3-4)^2 + (80000-60000)^2) = 25000;
2) The "Manhattan" distance = abs(40-30) + abs(35000-50000) + abs(3-4) + abs(80000-60000) = 35011;
3) Calculation of the Mahalanobis distance between Jim and Jane needs data on the other cases in order to populate the cells in $\mathbf{S}^{-1}$.

Generally, when Euclidean distance is used as a (dis)similarity measure, the square root is not taken because of the relative ease of analyzing sums of squares and the difficulty of analyzing square roots. Mahalanobis distance is a generalization of Euclidean distance that adjusts for different variances and correlations between the variables. Euclidean distance and Mahalanobis distance are identical if the variables are uncorrelated and have the same variances. [4]

In addition to the problems in common to correlation and distance measures mentioned in (A) above, distance measures are strongly affected by the scatter (scale) of variables used in

[3] Note that $(\mathbf{x}_i-\mathbf{x}_j)'\mathbf{S}^{-1}(\mathbf{x}_i-\mathbf{x}_j)$ is conformable. It is (1 x p) x (p x p) x (p x 1) = 1 x 1.

[4]In that case, **S** = **I**. If Euclidean distance may be thought of as distance between nested spheres, Mahalanobis distance may be thought of as distance between nested ellipsoids.

profiles. Therefore, it is standard practice to standardize variables prior to calculating distance measures used as (dis)similarities.

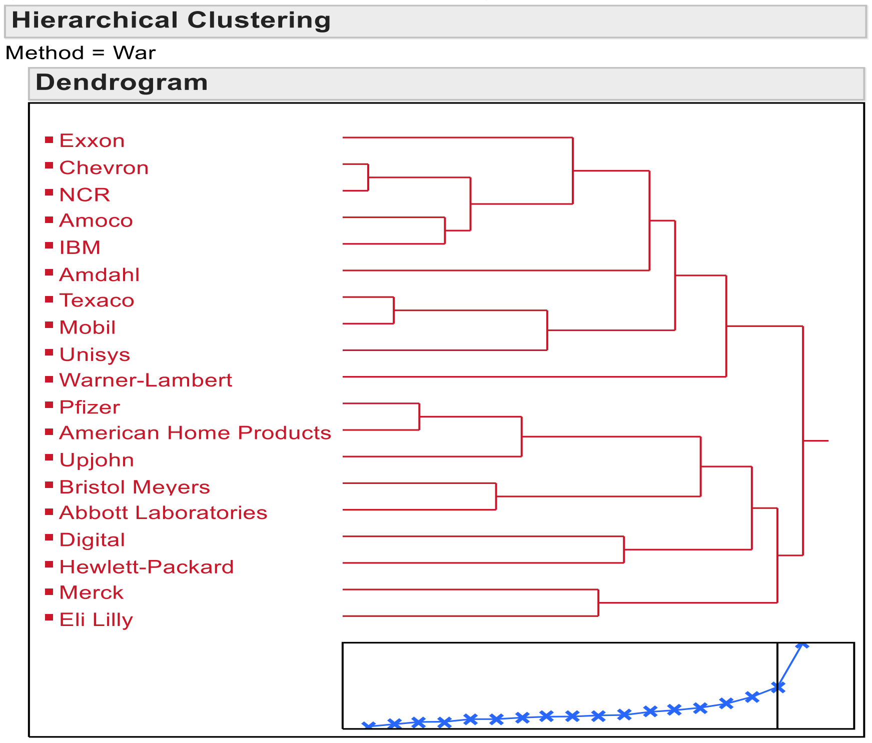
SAS uses dissimilarity measures in its CA procedures, and by default they are squared Euclidean distances calculated from the input DATA=<SAS DATASET>. Similarity measures must be converted into dissimilarities in order for the SAS procedures to work properly. By default SAS does not standardize the variables; that can be done with the **STANDARD** option in **PROC CLUSTER**, or by running **PROC STDIZE** prior to **PROC FASTCLUS** (which lacks the **STANDARD** option). You can construct your own matrix of distances for inputting into **PROC CLUSTER** (but not **PROC FASTCLUS**) by creating a **TYPE=DISTANCE** dataset (see “Example 30.1 Cluster Analysis of Flying Mileages between 10 American Cities” in the Examples section of the SAS help file for **PROC CLUSTER** for an example).

Clustering Methods

HIERARCHICAL AGGLOMERATIVE. These are the most-used clustering methods. They all begin with each case in its own cluster and proceed to join cases together to form larger clusters, eventually ending with all cases in one super-cluster. So there are *n* steps. At the end, the intermediate history of the joining (agglomeration) can be displayed as a **TREE**, with the cases listed on the horizontal axis at the bottom (or left side), with a vertical line leading up from each case until it joins with the line of another case or cluster, the height of the tree indicating the degree of (dis)similarity when the join occurs. [See Figure 3 for an example.] So the "tree" is upside-down (or sideways) with leaves at the bottom (or left side) and root at the top (or right side). In fact, the arboreal vocabulary is retained and mixed with genealogical vocabulary: A **LEAF** is a case; a **ROOT** is the super-cluster at the top (right side) consisting of all the leaves; a **BRANCH** is a cluster containing more than one, but not all, cases; a **NODE** is a leaf, a branch, or a root; if cluster A is formed by joining clusters B and C, then A is the **PARENT** of B and C, which are, of course, the **CHILDREN** of A. So a leaf is a node with no children; a root is a node with no parent. If every cluster has at most two children, the tree is **BINARY**.

The degree of relatedness of two cases is indicated by their most recent common "ancestor" (cluster in which both are members) in the resulting hierarchy. Each of the *n* steps in the agglomeration represents a possible solution to the two major clustering problems (how many clusters and how to assign cluster membership). Presumably, the solution that puts all cases in separate clusters (at the bottom [left] of the tree) and the solution with all cases in one super-cluster (at the top [right] of the tree) are not acceptable. Solutions that are in-between the top and the bottom can be examined and tested for their reasonableness.

Figure 3. An example of a tree (dendrogram): 19 companies in Forbes 500



The tree in Figure 3 is read from left to right. At the first step, Chevron and NCR are found to be the two most similar companies and are joined together in a cluster. At the second step, Texaco and Mobil are the most similar and are joined. Eventually, clusters start to join in larger clusters. For example, in step 5 the Chevron/NCR cluster that was formed in step 1 joins the Amoco/IBM cluster that was formed in step 4. The joins continue until in step 18, two large clusters of 10 and 9 companies join to form the root cluster of all 19 companies. The tree can be cut by a vertical line at any point between the root and the leaves to yield a solution. The branches that remain intact at the cut point become the clusters for the solution. For example, if the tree is cut vertically just to the left of the last joining, then there will be two large intact branches of 9 and 10 companies, respectively.

In SAS, **PROC CLUSTER** implements hierarchical agglomerative clustering (with 11 different methods). **PROC CLUSTER** produces binary trees only. **PROC TREE** displays the tree and permits outputting a dataset containing the cases' cluster memberships from any specified cut point on the tree. The following are some common hierarchical agglomerative methods implemented in SAS:

(1) **Single Linkage** ("**Nearest Neighbor**"). This hierarchical agglomerative method begins with each case in its own cluster and joins the two most similar cases into a new cluster, replacing the two former cases. After the join, the similarity between the new cluster and remaining ones must be redefined and calculated. In single linkage, the similarity between the new cluster and a remaining entity is defined to be the similarity between the *nearest* members of the entity and the cluster – i.e., the greatest pairwise similarity (least dissimilarity) between the entity and any member of the new cluster. This means that the algorithm can continue to use the initial similarity matrix – no need to recompute its values – one just needs to keep track of which rows have been clustered. At the next step, the two most similar entities from the new set of

clusters are joined, and the similarities between remaining entities and the new entity are recalculated. The process continues until all entities have been joined in the super-cluster at the root. This method tends to do the worst in simulation studies. It tends to form elongated clusters of unequal numbers of cases (e.g., two clusters of 89 cases and one cluster of one case).

(2) **Complete Linkage** ("**Farthest Neighbor**"). Like single linkage, except that similarity is the distance between the *farthest* members of two entities. So a joining entity must be close to ALL members of the entity to be joined. In complete linkage, the similarity between the new cluster and a remaining entity is defined to be the least pairwise similarity (greatest dissimilarity) between the entity and any member of the new cluster. Again, like single linkage, this means that the algorithm can continue to use the initial similarity matrix – no need to recompute its values – one just needs to keep track of which rows have been clustered. The complete linkage method tends to find compact hyper-spherical clusters of highly similar cases.

(3) **Average Linkage** ("**Average Neighbor**"). Like complete linkage, except that the AVERAGE level of similarity between any candidate entity for inclusion into an existing cluster and the individual members of that existing cluster must be within a specified level. In average linkage, the similarity between the new cluster and a remaining entity is defined to be the mean pairwise similarity (mean dissimilarity) between the entity and the members of the new cluster. Unlike single linkage and complete linkage, the similarity matrix must be recomputed after each step. This method was proposed as an antidote to the extremes of single and complete linkage. It tends to find clusters of roughly equal variance. Average linkage and Ward's method tend to do best in simulation studies.

(4) **Ward's Method**. Like single linkage, except that at each step, clusters are joined that result in the least increase in the within-cluster variance.[5] To gain an intuitive understanding of what Ward's method tries to do, look at Figure 1, which shows two clusters. Ward's method tries to choose the clusters to make the points within each cluster as close to each other as possible and as far from the other cluster as possible. In Figure 1, the points within the lower left cluster are closer to each other, on average, than they are to the points in the upper right cluster, and vice-versa. The lower left cluster is clearly separated from the upper right cluster. One can measure how close the points in each cluster are to each other by their variance: The smaller the variance within a cluster, the closer its points are to each other. In the most extreme case, the variance within a cluster could be zero if the points were all equal to the mean within the cluster. The bigger the variance within a cluster, the farther apart its points are. In Figure 1, the points within the lower left cluster clearly have less within-cluster variance than the points in the upper right cluster.

How can the variance within a cluster be calculated? Suppose that X_{ikl} is the value of case *l* on variable *k* in cluster *i*. Suppose also that n_i is the number of cases in cluster *i*, and that there are *c* clusters. Then the within-cluster sum-of-squares for variable *k* in cluster *i* is $\sum_{l=1}^{n_i}(X_{ikl} - \bar{X}_{ik\cdot})^2$, where $\bar{X}_{ik\cdot}$ is the mean of all of the kth variable values that are in cluster *i*. So $\sum_{l=1}^{n_i}(X_{ikl} - \bar{X}_{ik\cdot})^2$ is the total squared deviation between this mean and the kth variable's values of the cases that are in cluster *i*. So $\sum_{l=1}^{n_i}(X_{ikl} - \bar{X}_{ik\cdot})^2$ is a natural measure of how big cluster *i* is in terms of variable *k* – i.e., how close these points are to each other. To convert this into a

[5] Actually, the within-cluster sum of squares – the variance is the sum of squares divided by the degrees of freedom.

variance, we would simply divide by the appropriate degrees of freedom. That is not a necessary step since it is just adjustment by a constant. We can use the sum of squared deviations itself as the criterion. But we are not finished. $\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{ik\cdot})^2$ is a measure that can be calculated for each of the c clusters. Let us add them up over all clusters to get an overall measure of closeness for variable k for all c clusters = $\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{ik\cdot})^2$. But we are still not finished. We can calculate this for each of the variables. If there are p variables, then the total within-cluster variance (sum-of-squares) is $\sum_{k=1}^{p}\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{ik\cdot})^2$, which is called the **within-cluster variance** (more precisely: sum of squares). This is the criterion that Ward's method seeks to minimize from step to step in the clustering algorithm. Ward's method tends to create hyper-spherically-shaped clusters having roughly equal numbers of cases. Ward's method and average linkage tend to do best in simulation studies.

Mathematical note: At the same time that Ward's method minimizes the within-cluster sum-of-squares, it maximizes the between-cluster sum-of-squares, thus obtaining maximal spread between the clusters. There is a mathematical identify that illuminates this relationship:

$$\sum_{k=1}^{p}\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{\cdot k\cdot})^2=\sum_{k=1}^{p}\sum_{i=1}^{c}n_i(\bar{X}_{ik\cdot}-\bar{X}_{\cdot k\cdot})^2+\sum_{k=1}^{p}\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{ik\cdot})^2$$

Total Sum-of-Squares= **Between-cluster Sum-of-Squares** + **Within-cluster Sum-of Squares**

This can be written simply as **TSS** = **BSS** + **WSS**. $\bar{X}_{\cdot k\cdot}$ is the grand mean of all of the data for variable k. The total variability of all of the data for variable k can be broken down into a measure of the deviations of the cluster means from the grand mean + the deviations of the data from the cluster means. That is, for variable k,

$$\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{\cdot k\cdot})^2=\sum_{i=1}^{c}n_i(\bar{X}_{ik\cdot}-\bar{X}_{\cdot k\cdot})^2+\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{ik\cdot})^2$$

Then add up these parts over all p variables to get the above identity:

$$\sum_{k=1}^{p}\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{\cdot k\cdot})^2=\sum_{k=1}^{p}\sum_{i=1}^{c}n_i(\bar{X}_{ik\cdot}-\bar{X}_{\cdot k\cdot})^2+\sum_{k=1}^{p}\sum_{i=1}^{c}\sum_{l=1}^{n_i}(X_{ikl}-\bar{X}_{ik\cdot})^2$$

Once the data are in hand, **TSS** is fixed. Therefore, if **WSS** increases, **BSS** must decrease. At the beginning of the hierarchical process for Ward's method, each point is in its own singleton cluster; WSS = 0, and BSS is at its maximum and equals TSS. As the agglomeration proceeds, WSS increases (but slowly, since Ward minimizes its growth) and BSS decreases. At the end of the agglomeration, all points are in one super-cluster, with BSS = 0 and WSS is at its maximum and equals TSS.

From the identity **TSS** = **BSS** + **WSS**, further insight can be gained into Ward's method. Suppose that c indicator variables are created at a given step to signal cluster membership for each case: $I_1=1$ if a case is in cluster 1; $I_1=0$ if the case is not in cluster 1; …; $I_c=1$ if a case is in cluster c; $I_c=0$ if the case is not in cluster c. How much of the X variables can be explained by the clustering of the cases? The question is answered by regressing the X's on c – 1 of the indicators $I_1,\ldots,I_c$. These regressions amount to **analysis of variance** (**ANOVA**),[6] which is the statistical technique for regression with categorical predictors. The R-square for the regressions is BSS/TSS. So BSS acts like the sum-of-squares for regression, and the WSS acts like the error sum-of-squares. So Ward's method tries to maximize the R-square – the explanatory power – of

[6] When analysis of variance is done for each of p response variables and the results of the ANOVAs are combined, as they are in the formula TSS = BSS + WSS, the procedure is called **multiple analysis of variance** (**MANOVA**).

the clustering as clusters are combined during agglomeration. In selecting a solution (a cut point on the dendrogram), the user must decide on an appropriate trade-off between explanatory power (R-square) and parsimony (number of clusters). This recalls similar tradeoffs in statistical model building in general, as the gain in explanatory power from adding a predictor must be weighed against the loss of parsimonious explanation; one wants a parsimonious model with high explanatory power. In CA, the ideal situation is to have a few clusters that still have high explanatory power. Such considerations can be applied to all hierarchical agglomerative clustering methods.

(5) **Centroid Method**. The **centroid** of a cluster is the vector of means of the *p* variables of the cases in the cluster. The centroid method is like single linkage, except that the (dis)similarity between two clusters is defined to be the squared Euclidean distance between the two centroids of the two clusters. When two clusters are joined, the centroid of the new cluster becomes the reference point for recalculating (dis)similarities.

(6) **Density Linkage**. Like single linkage, except that the (dis)similarity between two clusters is based on nonparametric density estimation, which estimates the density of points in the vicinity of the clusters. No assumptions are made about the shape of clusters. This method is relatively unbiased and makes few assumptions of the data, but does not work quite as well as other methods in situations where the biases of other methods are met.

NON-HIERARCHICAL METHODS: ITERATIVE PARTITIONING. Most of these methods begin by dividing the cases into a number of preliminary clusters. The number of preliminary clusters is user-specified. The centroids of the preliminary clusters are computed and become the "seeds" for growing subsequent clusters iteratively. Each case is then assigned to the nearest seed. The cases assigned to the same seed constitute a new cluster, and the set of all such new clusters thus determine a new partitioning of the cases into a second set of clusters. The centroids of the second set of clusters are then calculated, and cases are (re)assigned to the nearest of the second set of centroids. This process continues until no case changes cluster membership. The most well-known and most commonly used of these methods is ***k*-means clustering**.

In SAS, `PROC FASTCLUS` implements iterative partitioning. One method is used: *K*-means clustering. **`PROC FASTCLUS`** can handle much larger datasets than **`PROC CLUSTER`**. Therefore, **`PROC FASTCLUS`** is often used to reduce the dataset to a smaller number of clusters (< 100) before running them through **`PROC CLUSTER`** for final clustering. By default, **`PROC FASTCLUS`** does only one pass through the data.

There are three major decisions to make in iterative partitioning:

1) Initial partition. This can be determined by pre-specifying a set of seeds, or by determining the preliminary partition. Both of these can be done randomly or by a method. SAS offers both options.
2) Type of pass. This is the way to reassign cases during each pass. A common method is "*K*-means" (reassignment to nearest centroid), implemented in SAS, which offers various options on the technical details of carrying it out. Another method is "hill-climbing", which reassigns cases on the basis of some statistical criterion other than distance.
3) Statistical criterion (mentioned in the preceding point). Distance is the simplest, but other methods use properties of covariance matrices.

Simulation studies show that the major cause of inferior performance in iterative partitioning is a poor initial partition.

***K*-means clustering**. The objective of *K*-means clustering is to partition the data rows into a given number (*K*) of clusters in such a way that the within-cluster sum-of-squares (WSS – see earlier discussion of Ward's method) is minimized. In its objective, *K*-means clustering resembles Ward's method. Both methods seek to minimize the WSS. However, Ward's method operates hierarchically and agglomeratively, whereas *K*-means operates iteratively. Ward's method produces a solution for each possible number of clusters, whereas *K*-means produces a solution for only one given number of clusters. *K*-means begins with an initial set of K cluster "seeds" spread more or less uniformly across the data points. These seeds can be user specified, picked at random, or selected by some algorithm. They are often data points, but need not be. Then, in the first pass, each data point is assigned to the closest seed. This results in K preliminary clusters. Then the centroid (mean) of each cluster is calculated. The K centroids become the new seeds. Another pass is made through the data points. Each data point is (re)assigned to the closest centroid/new seed/mean. A revised set of K clusters results.[7] Centroids are calculated yet again, and the process is iterated until no data points change cluster membership – or until the limit on number of passes is reached.

FACTOR ANALYSIS VARIANTS.[8] The application of factor analysis (FA) ideas to the clustering of cases is based on the intuitive observation that factor analysis, in a sense, groups similar variables together, whereas cluster analysis groups similar cases together. Since FA, in this sense, operates on the columns of the data matrix, just rotate the data matrix 90 degrees and apply FA to the rows of the data matrix. Ordinarily, FA is performed on the *p* x *p* matrix of correlations between variables; this is called **R-mode** FA. But here, FA is proposed for the *n* x *n* correlation matrix *of cases*, rather than the more familiar *p* x *p* correlation matrix *of variables*. When FA is performed on the *n* x *n* matrix of correlations between cases, it is called **Q-mode** FA. In Q-mode FA, cases are assigned to clusters based on the factor loadings of the cases. Note that FA-based cluster analysis implies that cases are standardized to mean 0 and stdev 1 because Q-mode FA analyzes the case correlation matrix. Thus the Q-mode approach to cluster analysis ignores elevation and scatter in the profiles of cases and groups cases only by their shapes.

In SAS, **`PROC FACTOR`** can be used to cluster cases if the usual *n* x *p* SAS data set is transposed, say by **`PROC TRANSPOSE`**. (Another SAS procedure, **`PROC VARCLUS`**, uses FA to cluster variables; it is primarily a dimensionality reduction technique.)

HIERARCHICAL DIVISIVE. These methods are the logical opposites of hierarchical agglomerative methods. They begin with all cases in one cluster and divide it into successively smaller groups.

SAS does not implement these methods.

[7] This explains the origin of the name "*K*-means": data points are assigned to the closest of K means.
[8] Principal components analysis can be considered a variant or special case of factor analysis.

CLUMPING METHODS. These methods permit overlapping clusters, with cases sharing membership in more than one cluster. They calculate a similarity matrix between cases and iteratively reallocate cases until a "cohesion function" is stabilized. The methods originated in linguistics in order to group words, which may have more than one meaning.
SAS does not implement these methods.

GRAPH THEORETIC METHODS. These methods draw on the well-developed mathematical theory of graphs (a tree is a graph, e.g.). This link with mathematics may in time provide testable hypotheses and insights on cluster formation.
SAS does not implement these methods.

Determining the Number of Clusters

This is an unsolved problem in statistics. None of the proposed solutions is widely accepted. The reasons for this:

1) A suitable formulation of a null hypothesis to capture the meaning of "there are no clusters" is elusive.
2) A suitable formulation of alternative hypotheses is elusive.
3) Those hypotheses that can be tested are uninformative.
4) Those hypotheses that are most meaningful have test statistics with intractable distributions.
5) F tests based on ANOVA (such as those found in **PROC DISCRIM** or **PROC GLM**) to determine the explanatory power of the clusters produced by CA are invalid for testing differences between clusters. The reason is that CA is *structure-imposing*: CA will find clusters even if none exist. Consequently, such F tests are almost always misleadingly significant.

Some advice and observations on determining the number of clusters:

1) Always pay attention to the underlying theory of your subject. If theory suggests a certain number of clusters or if there is some "natural" number of clusters, then be sure to include that number among the possibilities that you check out.
2) The three top performing heuristic methods in 1985 & 1988 studies by Milligan and Cooper are implemented in SAS:
   a) The SAS-developed **cubic clustering criterion** (**CCC**). Look for local peaks in CCC when plotted against the number of clusters.
   b) The **pseudo-F statistic**, which measures the separation among all clusters. Look for local peaks in pseudo-F when plotted against the number of clusters.
   c) The **pseudo-T2 statistic**, which measures the separation between the two clusters most recently joined. Plot pseudo-T2 against the number of clusters and choose the number to be one more than the peak (or end of a run of large values) of pseudo-T2. **PROC CLUSTER** can output variables _NCL_, _CCC_, _PSF_, and _PST_ for certain clustering methods. These variables are the number of clusters in a step, cubic clustering criterion, pseudo-F, and pseudo-T2 statistics, respectively.
3) In a similar vein as (2), use **PROC CLUSTER** with **`METHOD=DENSITY`** and vary the K parameter (number of nearest neighbors). With this method, the variable _MODE_ (number of modes,

i.e., clusters) can be output and plotted against K. If _MODE_ is constant on a range of K's, then there are at least that many clusters.

4) (Generalization of (3)) Try a number of different solutions and make a qualitative assessment of which number of clusters is most satisfactory.
5) Plot the data, or their canonical discriminant functions (**PROC CANDISC**), on two dimensions with plotting character equal to cluster number. Look at the clusters and see if they make sense.

Validating the Cluster Analysis

Like the determination of the number of clusters, validating the grouping of cases into clusters is an unsolved problem. Two methods in common use are rejected by theoreticians and/or fail simulation studies:

(1) **Cophenetic correlation** (only for hierarchical agglomerative methods). This is the ordinary correlation between the entries in the original similarity matrix and an artificial similarity matrix constructed from the tree: The latter matrix uses the height (similarity) of the tree where two entities have their earliest common ancestor as the constructed similarity between the two entities. Thus cophenetic correlation measures how well the tree actually represents the similarities between entities. But there are only n - 1 possible distinct values for the constructed similarity matrix, whereas the original similarity matrix has n (n - 1)/2. And it assumes normality. And at least one simulation found it wanting.

(2) **M/ANOVA F-statistics** are commonly used but theoretically invalid for the reasons cited above under "Determining the Number of Clusters".

Three potentially useful validation techniques:

(3) **Cross-validation**. Split the cases and run CA on each set. If similar clusters develop, then the CA may be OK; but if different clusters appear, then the CA is not generalizable. (Note that this method is a negative method in that it can disconfirm but not confirm the CA.) A variation on this method is to run the CA on the first set of cases, then use its cluster centroids as seeds to cluster the second set. This forces the same number of clusters in the cross-validation. If the cluster centroids from the first set reproduce similar case assignments and the clusters in the second set of cases have small within variation and big between variation, then the CA looks good.

(4) If the clusters are separating the cases on the variables used in the CA, then they should separate the cases on other variables NOT used to develop the CA. Try it with new variables. Since such variables could have been used to develop the CA, this suggestion is somewhat like cross-validation by splitting the variables instead of the cases.

(5) **Simulation**. Create a simulated data set with major characteristics like those of the real data. Run CA on each and compare results.

Cluster Analysis in SAS

SAS offers six procedures related to CA:
(1) **PROC CLUSTER** -- for hierarchical agglomerative methods
(2) **PROC FASTCLUS** -- for iterative partitioning methods like *k*-means clustering
(3) **PROC TREE** -- works with output of PROC CLUSTER to print dendrogram of case relationships and to output the *k*-cluster solution for any *k*
(4) **PROC FACTOR** -- for factor analysis variants (first transpose the dataset)
(5) **PROC VARCLUS** -- for clustering the variables instead of the cases
(6) **PROC ACECLUS** -- Approximate Covariance Estimation for CA: estimates within-cluster covariance (assuming they are the same for all clusters) without knowing how many clusters there are.

I will discuss the first three:

(1) **PROC CLUSTER**

(A) **Type of Data**: May be raw *n* x *p* SAS dataset [default] or *n* x *n* matrix of distances.

(B) **Similarity Measures**: SAS does not use similarities. Similarities such as correlations must be converted into dissimilarities before running PROC CLUSTER (e.g. by subtracting from +/- 1). Usually, distances are used. If raw *n* x *p* SAS dataset is input into PROC CLUSTER, it will be converted into a *n* x *n* matrix of squared [unless **NOSQUARE** option is selected, for **METHOD=CENTROID**, **MEDIAN**, or **WARD**] Euclidean distances between cases. If *n* x *n* matrix of TYPE=DISTANCE is input, it will be treated as distances between cases [and then squared for METHOD=AVERAGE, CENTROID, MEDIAN, or WARD, unless NOSQUARE option is selected]. In both cases, a dissimilarity matrix is obtained.

(C) **Standardization of Variables**: The default is to calculate the dissimilarity matrix of the raw *n* x *p* SAS dataset *without* standardizing the variables. SAS will standardize the data (standardize each variable [column] to mean 0 and stdev 1) prior to calculating the dissimilarities if the **STANDARD** (or **STD**) option is selected. The decision to standardize or not is nontrivial. Standardization treats all variables the same. On the other hand, if several variables represent about the same thing, then standardization will cause the clustering to favor that thing because the multiple variables effectively give that thing multiple weights.

(D) **Setting Number of Clusters and Cluster Memberships**: Do this in **PROC TREE**. PROC CLUSTER produces the whole history of the agglomerative clustering process, from beginning *n* clusters to final single super-cluster. So its output contains the solution for any specified number of clusters. You cannot tell PROC CLUSTER to assign cases to clusters for a specific number (say) 3-cluster solution. To do this you must run PROC CLUSTER with output option **OUTTREE=<SAS DS>**, then input the OUTTREE dataset into PROC TREE with **NCLUSTERS=3** and then output and print **PROC TREE**'s **OUT=<SAS DS**>. See example programs below for SAS code to do this.

(E) **Diagnostic Options**: **CCC**, **PSEUDO**, **RSQUARE** give some guidance to deciding how many clusters are present. They are not valid for all clustering methods. **CCC** prints the cubic clustering criterion for each of the *n* - 1 hierarchical steps of the agglomerative process for which the CCC makes sense. Look for the number of clusters where CCC makes local maxima.

PSEUDO prints the pseudo-F and pseudo-T2 statistics for each step for which the statistics make sense. Look for local peaks in pseudo-F, and add one to the number of clusters where pseudo-T2 peaks. **RSQUARE** prints the explanatory power (ANOVA R-square) of the clusters at each step of the process [9]; this decreases as the number of clusters decreases, so you need to balance the number of clusters versus explanatory power in using this diagnostic in deciding the number of clusters.

(F) **Output Datasets**: **OUTTREE=<SAS DS>** contains a summary of the steps of the clustering process for input into PROC TREE. Used for printing dendrograms and selecting solutions with a specified number of clusters. This dataset can contain variables **_CCC_**, **_PSF_** (pseudo-F), **_PST2_**, **_RSQ_**, **_NCL_** (number of clusters at each step), **_FREQ_** (number of cases in each cluster at each step), **_HEIGHT_** (height of tree for each case), which can all be analyzed by other PROCs.

(G) **Clustering Methods in PROC CLUSTER**: Notes:

AVERAGE linkage

CENTROID linkage

COMPLETE linkage Use **TRIM=<P>** to reduce distortion by outliers

DENSITY linkage Based on nonparametric density estimation; input data must be raw SAS dataset, not distances; must use one of **K=<K>**, **R=<R>**, or **HYBRID** options with DENSITY method; K is number of nearest neighbors.

EML Maximum likelihood based on model for mixtures of spherical multinormal data; input data must be raw SAS ds, not distances;

FLEXIBLE beta Use with **BETA=<N>** option [default BETA=-.25]

MCQUITTY

MEDIAN

SINGLE linkage Use **TRIM=<P>** to reduce chaining

TWOSTAGE density See above for DENSITY linkage

WARD Use **TRIM=<P>** to reduce distortion by outliers

(H) **Speed**: Speed of most methods is proportional to cube of number of cases (**EML** much slower), other things beings being equal. So large datasets may not be feasible to cluster in PROC CLUSTER.

(2) **PROC FASTCLUS**

(A) **Type of Data**: Must be raw *n* x *p* SAS dataset, not *n* x *n* matrix of distances, similarities, or correlations. Along with the *n* x *p* data matrix, you may use **SEED=<SAS DS>** containing coordinates of initial seeds [default is to calculate seeds from DATA=<SAS ds>]; the SEED=<SAS ds> may also be the **MEAN=<SAS DS>** output from a previous run of PROC FASTCLUS.

(B) **Similarity Measures**: The raw *n* x *p* SAS dataset is converted into a *n* x *n* matrix of squared Euclidean distances between cases. This is a dissimilarity matrix. Neither similarities nor correlations can be used in PROC FASTCLUS.

[9] See discussion of ANOVA and R-square in earlier comments on Ward's method.

(C) **Standardization of Variables**: PROC FASTCLUS has no option to standardize the variables. The raw *n* x *p* input SAS ds is taken as it is. So if you want to standardize the variables, do it before running PROC FASTCLUS. [May use **PROC STDIZE** to do this.]

(D) **Setting Number of Clusters**: Two options to do this: **MAXCLUSTERS=<N>** [default=100] or **RADIUS=<T>**. This determines the upper limit on the number of clusters produced. Because PROC FASTCLUS is not hierarchical, only one *k*-cluster solution can be produced with each run.

(E) **Diagnostic Options**: CCC, pseudo-F, and R-square are all printed but not available in output datasets. They are less useful in PROC FASTCLUS than in PROC CLUSTER, because PROC FASTCLUS solves for *k*-clusters for only one *k* per run. With PROC FASTCLUS you do not get a range of CCC, pseudo-F, or pseud-T2 for a range of *k*'s that you can examine for peaks, unless you run PROC FASTCLUS repeatedly. The option **DISTANCE** prints distances between cluster means, and the DISTANCE variable is in the **OUT=<SAS DS>**. The **MEAN=<SAS DS**> contains **_FREQ_** (number of cases in each cluster), **_GAP_** (centroid distance to nearest cluster mean), and **_RADIUS_** (distance from cluster centroid to most distant case in that cluster), which can be used to diagnose outliers as well as "good seeds".

(F) **Output Datasets**: **MEAN=<SAS DS>** has one observation per cluster with info about each cluster, variables named **CLUSTER**, **_FREQ_**, **_GAP_**, **_RADIUS_**, and a few others; can be used as input dataset into PROC CLUSTER. **OUT=<SAS DS>** contains original *n* x *p* input SAS dataset plus the variables **CLUSTER** (showing cluster memberships of the cases) and **DISTANCE**.

(G) **Clustering Methods in PROC FASTCLUS**: One method is used: K-means iterative partitioning, with a number of options (**RANDOM=**, **REPLACE=**, **CONVERGE**=, **DELETE=**, **DRIFT**, **MAXITER=**, **STRICT**) to vary the technical details of how it works. (See earlier discussion of iterative partitioning.) By default, the seeds for the initial partition are determined by making the first case a seed, then finding the next case that is sufficiently distant from the first seed and making it a seed, and so on. Random seeds can also be chosen with **RANDOM=**. The default number of iterations is **MAXITER=1**.

(H) **Speed**: All other things being the same, the speed of PROC FASTCLUS is roughly proportional to the number of cases (*n*). Thus large datasets are feasible. PROC FASTCLUS may be used to reduce a large ds to a smaller number of clusters (output through MEAN=<SAS ds>) for inputting into PROC CLUSTER.

(3) **PROC TREE**

Not a cluster-creating procedure, PROC TREE takes as input a SAS ds output from PROC CLUSTER (through the latter's **OUTTREE=<SAS DS>**) and (a) prints the associated dendrogram and (b) can divide the cases into any specified number of disjoint clusters.

How to Read a Tree (dendrogram): By default the tree is printed vertically with the leaves (individual cases) at the top and the root (super-cluster) at the bottom. But the tree may also be printed horizontally (by the **HORIZONTAL** option) with the leaves at the left at the root at the right. The scale and units of the height of the tree are determined by the **HEIGHT=<VARIABLE>**. The default is **HEIGHT= _HEIGHT_** (a variable in the OUTTREE=<SAS ds> from PROC CLUSTER, and the default _HEIGHT_ in PROC CLUSTER is the (dis)similarity between the two clusters joining at that level). So as you move from the leaves toward the root of the tree, pairs of clusters are joining, and the dissimilarity value of the joining pairs determines the value of the height of the tree at the point where the pair join. However, the value of the HEIGHT=<variable> can be set explicitly in PROC TREE to override the default. Other values for <variable> include **LENGTH** (the path length from the root [number of ancestors]), **MODE** (number of modes), **NCL** (number of clusters), **RSQ** (R-square).

EXAMPLE PROGRAMS (SAS keywords are capitalized; user-supplied values are in small caps):

```
(1) PROC CLUSTER  DATA=MYDATA  STD  OUTTREE=MYTREE  METHOD=WARD;
       VAR X Y Z;
    PROC TREE  DATA=MYTREE  OUT=MYOUT  NCLUSTERS=3;
       COPY X Y;
    PROC PLOT  DATA=MYOUT;
       PLOT Y * X = CLUSTER;
```

The program uses Ward's minimum-variance method to cluster WORK.MYDATA after standardizing the clustering variables X, Y, and Z to mean 0 and stdev 1 (**STD** option), outputting the *n* steps of the hierarchical agglomeration to **WORK.MYTREE**. Then prints a dendrogram constructed from **WORK.MYTREE**, and divides the cases into 3 clusters, the memberships for which are stored in WORK.MYOUT along with variables X and Y (**COPY** statement), which were in **WORK.MYTREE** (and before that in **WORK.MYDATA**), the variables of which are not automatically carried over to the **OUT= MYOUT**. Then the cases are plotted on the Y*X plane, with their **CLUSTER** number as plotting character.

```
(2) PROC STDIZE  DATA=MYDATA  METHOD=STD  OUT=MYDATA;
    PROC FASTCLUS  DATA=MYDATA  OUT=MYOUT  MAXCLUSTERS=3;
      VAR X Y Z;
    PROC PLOT  DATA=MYOUT;
      PLOT Y * X = CLUSTER;
```

Same as (1) except that *k*-means iterative partitioning method is used to cluster (with 1 iteration [default]). If you want to use standardized variables, **PROC STDIZE** must be used to standardize variables (mean 0 and stdev 1 are the defaults: **METHOD=STD**). **WORK.MYOUT** contains **WORK.MYDATA**, so it is not necessary to **COPY** X and Y.

```
(3) PROC STDIZE  DATA=MYDATA  METHOD=STD  OUT=MYDATA;
    PROC FASTCLUS  DATA=MYDATA  OUT=MYFAST  MAXCLUSTERS=50 MAXITER=3;
    PROC CLUSTER  DATA=MYFAST  OUTTREE=MYTREE  METHOD=DENSITY R=5.24;
      VAR x1-x10;
    PROC TREE  DATA=MYTREE  OUT=MYOUT  NCLUSTERS=3;
    PROC PLOT  DATA=MYOUT;
      PLOT x1 * x2 = CLUSTER;
```

Suppose **WORK.MYDATA** has many cases and several (10) variables. After all variables are standardized, **PROC FASTCLUS** outputs to **WORK.MYFAST** a preliminary set of 50 clusters after at most 3 iterations. **WORK.MYFAST** will be input to **PROC CLUSTER** for clustering the 50 clusters. Density estimation is used to cluster the 50 clusters, with a radius for the uniform kernel of 5.24 (value recommended by a table in a help file). The 10 clustering variables are declared in the **VAR** statement. The dendrogram for the 50 clusters is printed and the 3-cluster solution for the

50 clusters is output to WORK.MYOUT. Finally the 3 clusters are plotted on the axes of the first two variables x1 and x2.

```
(4) PROC CLUSTER  DATA=MYDATA  STD  METHOD=AVERAGE  CCC  PSEUDO
                 OUTTREE=MYTREE;
    PROC PLOT  DATA=MYTREE;
      PLOT _CCC_*_NCL_  _PSF_*_NCL_='F'  _PST2_*_NCL_='T';
```

Uses average linkage to cluster WORK.MYDATA after standardizing all variables. Prints cubic clustering criterion (CCC), pseudo-F and pseudo-T2 statistics (PSEUDO) for each level of the clustering. Plots these diagnostic variables (_CCC_, _PSF_, _PST2_ in WORK.MYTREE) against the number of clusters (_NCL_), with mnemonic plotting characters.

```
(5) PROC FASTCLUS  DATA=MYDATA  MEAN=MYMEAN  MAXCLUSTERS=20
                  MAXITER=0;
    PROC PLOT  DATA=MYMEAN;
      PLOT _GAP_*_FREQ_='G'  _RADIUS_*_FREQ_='R' / OVERLAY;
    DATA MYSEED;
      SET MYMEAN;
      IF _FREQ_ <= 5 THEN DELETE;
    PROC FASTCLUS  DATA=MYDATA  SEED=MYSEED  MAXCLUSTERS=2
                  STRICT=3.0   OUT=MYOUT;
```

Without standardizing, WORK.MYDATA is grouped into 20 preliminary clusters. MAXITER=0 tells SAS to stop after finding the first set of 20 clusters and NOT replace the initial cluster seeds (which are cases) with the cluster centroids. The effect is that WORK.MYMEAN contains 20 cases that are widely separated. Next these 20 cases are plotted, with _GAP_ (distance to nearest other cluster [here, case]) and _RADIUS_ (distance to farthest case in the cluster) on the vertical axis versus the number of cases initially assigned to the cluster on the horizontal axis. In the _GAP_ plot, outliers will show in the upper left, good clusters (already!) in the upper right, good seeds in the lower right, and poor seeds in the lower left. Next, low-count seeds are eliminated (IF _FREQ_ <= 5 THEN DELETE; based on inspection of the plot). Then PROC FASTCLUS is rerun with the processed seeds to produce 2 clusters, with all cases more than 3.0 apart (from STRICT, based on examination of the plot) being prevented from joining a cluster (such cases receive negative CLUSTER numbers in WORK.MYOUT).