Richard Gonzalez
Psych 614
Version 3.2 (Feb 2024)

LECTURE NOTES #10: MDS & Tree Structures

Reading Assignment

On Canvas: Davison's book on MDS; Corter's book on trees; various other readings.

For those using Johnson & Wichern you can read their chapter on clustering and MDS.

1. Multidimensional Scaling (MDS): an exploratory technique to recover an underlying metric space for data in the form of a distance or similarity matrix

   Motivating examples will be given throughout the first few lectures: geographical location of cities (see Figure 10-1), morse code data on 150 novices hearing the morse code signals in succession and judging "same" or "different", examples from previous class projects (e.g., cookie data, social judgments)

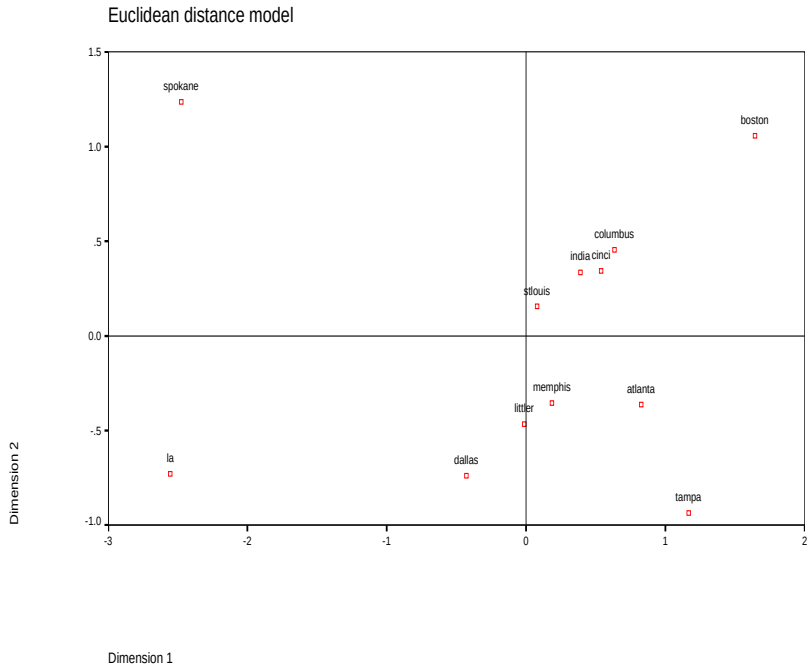
   We know it is possible to compute distances between points in a space (e.g., the airline magazine charts showing air miles between major cities). This presumes that the points are already placed in some space and one measures the distance between those points—distances are computed between the positions of those points.

   It turns out that it is possible to go in the other direction. From something that looks like a distance matrix, it is possible to estimate a "geographical" layout of the space and position the points in a manner that is consistent with the distance matrix. This idea is applicable to behavioral science because, as you will see below, there are frequent examples of data that look like distances and it is possible to infer an underlying space on those data. The dimensions of that space can then be interpreted and given substantive meaning.

   Even more surprising is that the geographic information can be extracted from a distance matrix in a "nonmetric" manner. In other words, all one needs are the ranks of the observed distances and there are enough constraints present, even though one is merely working with ranks, to estimate a dimensional space. This brilliant insight is due to Roger Shepard. Figure 10-1 provides an example using distances from 12 US cities.

   I will give the intuition with 4 points. Take the four points A, B, C, and D that have the following distances in a two-dimensional space: AB=5, AC=3, AD=1.8, BC=4, BD=3.2, and

Figure 10-1: Distances between 12 US cities using only the rank order information of the distances.




```
data list free
 /Atlanta Boston Cinci Columbus Dallas India LittleR LA Memphis
  StLouis Spokane Tampa.
begin data
    0 1068  461  549  805  508  505 2197  366  558 2467  467
 1068    0  867  769 1819  941 1494 3052 1355 1178 2747 1379
  461  867    0  107  943  108  618 2186  502  338 2067  928
  549  769  107    0 1050  172  725 2245  586  409 2131  985
  805 1819  943 1050    0  882  325 1403  464  645 1891 1077
  508  941  108  172  882    0  562 2080  436  234 1959  975
  505 1494  618  725  325  562    0 1701  137  353 1988  912
 2197 3052 2186 2245 1403 2080 1701    0 1831 1848 1227 2480
  366 1355  502  586  464  436  137 1831    0  294 2042  779
  558 1178  338  409  645  234  353 1848  294    0 1820 1016
 2467 2747 2067 2131 1891 1959 1988 1227 2042 1820    0 2821
  467 1379  928  985 1077  975  912 2480  779 1016 2821    0
end data.
alscal variables = all
 /level = ordinal
 /shape=symmetric
 /model=euclid
 /criteria=dimens(2,2)
 /plot=all.
```

Figure 10-2: R example: Distances between 12 US cities using only the rank order information of the distances.

```
# see appendix for explanation of syntax
library(MASS)
d <- as.matrix(read.table("city.dat"))
city <- c("Atlanta", "Boston", "Cinci", "Columbus", "Dallas", "India",
    "LittleR", "LA", "Memphis", "StLouis", "Spokane", "Tampa")
dimnames(d) <- list(city, city)
output <- isoMDS(d, k = 2)
par(pty = "s")
flip <- -1
plot(flip * output$points, xlim = c(-1800, 1200), ylim = c(-800, 1000),
    xlab = "", ylab = "", main = "Cities Example in R")
text(flip * output$points, labels = city, pos = 3, cex = 0.75)
```

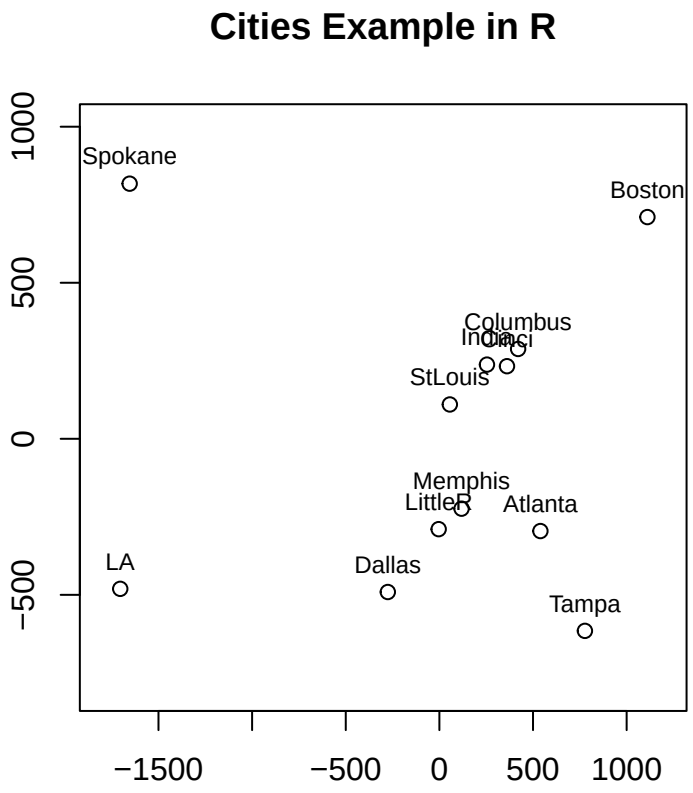
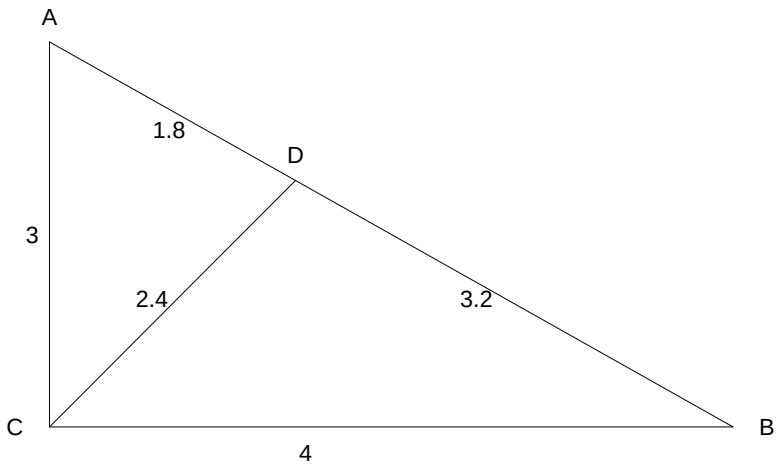


Figure 10-3: Simple example with 4 points.



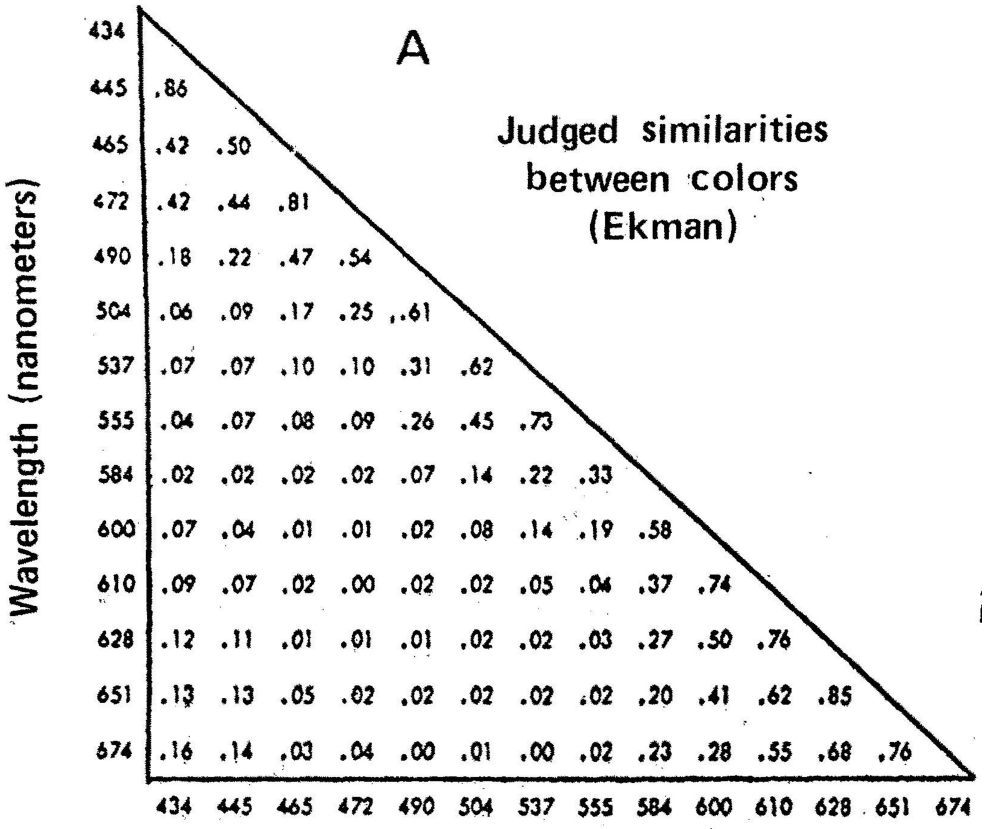
CD=2.4. Start by placing points A and B somewhere on a page five inches (or units) apart. That is, once you place A down, then B can be placed anywhere along the circumference of a circle with radius 5 (because it is a distance of 5 away from A). Once A and B are placed turn to point C. C is constrained because it must be 3 units from A and 4 units from B. There are only two locations that C can be placed under these constraints. Once a location for C is chosen, then there is only one place for D to be placed so that all interpoint distances are satisfied. It is in this sense that one can infer a configuration of points from the interpoint distances. See Figure 10-3. This example used the actual distances but it is possible to approximate the original spatial configuration quite well merely by using the ordinal information of the distances such as points A and B are maximally apart, points B and C have second largest distance, points B and D have third largest distance, etc.

From the interpoint distances it is possible to approximate the original space up to rotation and reflection. From the ranks of the interpoint distances it is possible to approximate the

space up to rotation, reflection, expansion, and contraction.

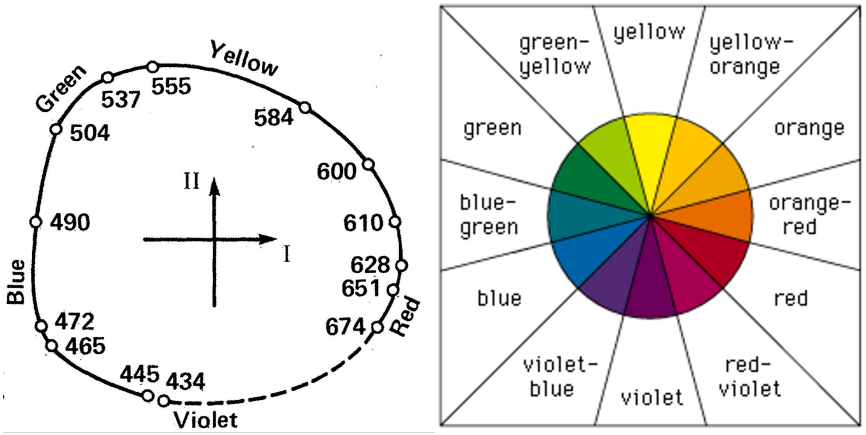
Let's look at a few more classic applications of this idea that we can recover a spatial arrangement from data that have a distance matrix structure.

In early work on color perception, participants were shown several color patches and asked to rate the similarity of those colors (e.g., how similar is color patch i to color patch j without using verbal labels such as "red"). This produces a matrix of data that for every possible pair of color patches there is a number reflecting similarity or distance, such as the percentage of participants who rated those two color patches similar. Here is an example data matrix arrange by the wavelengths of each color patch



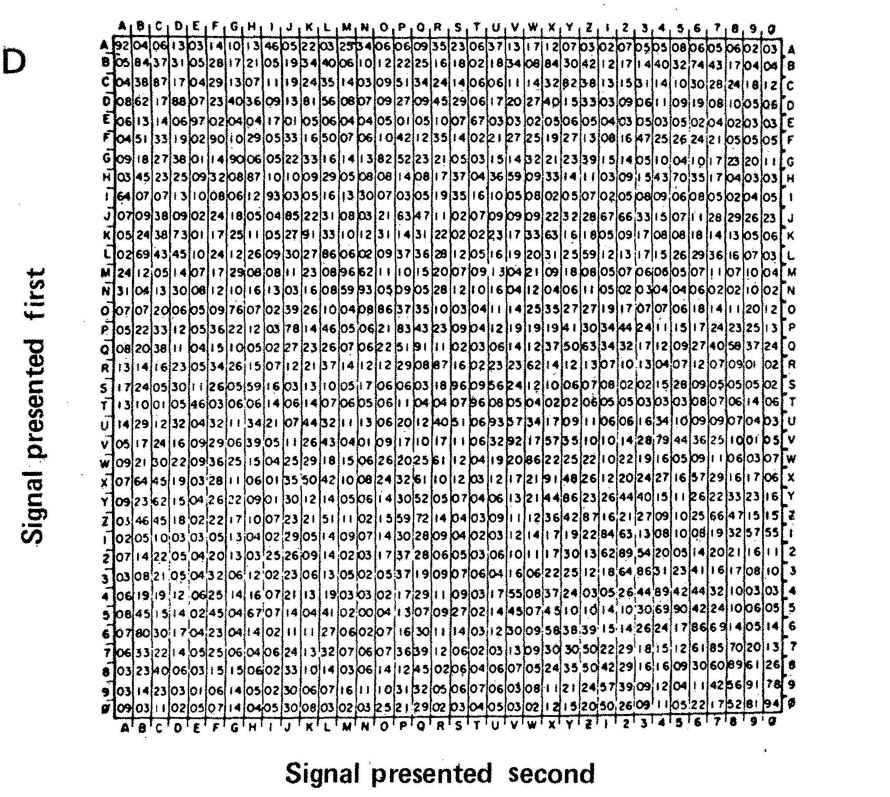
That's a lot of numbers and it isn't obvious that there is much structure to these numbers. The MDS procedure we will develop in these lecture notes provides an interesting two dimensional solution that is consistent with the opponent process theory of color vision. The information processing the visual system does is interesting in that the retinal cells process

color in three dimensions. That information is processed in the thalamus and reduced to two dimensions (blue-yellow and green-red). The information is then passed on to the visual cortex for additional processing.

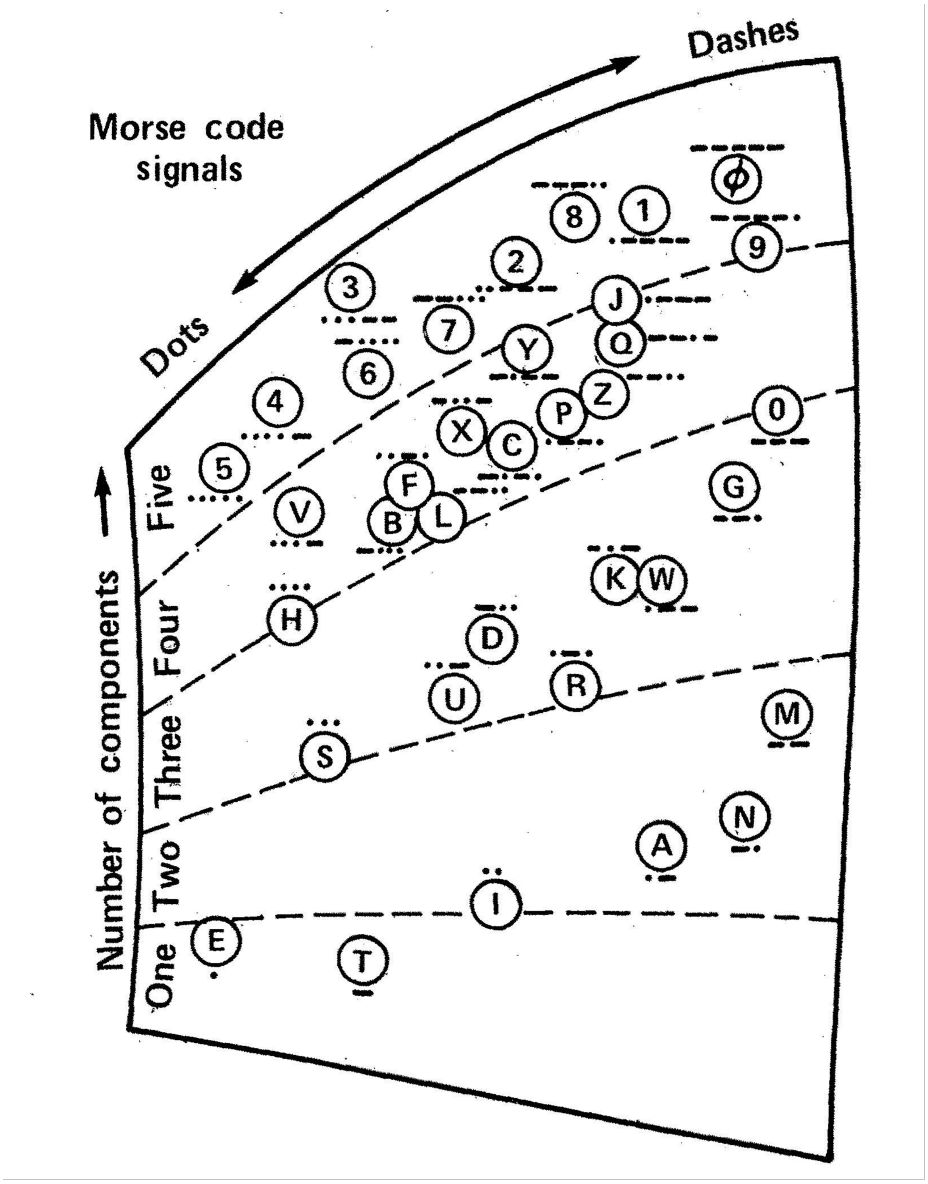


This two dimensional solution is similar to the traditional color wheel.

Another classic example uses confusion data from 150 novice morse code operators. Morse code was used decades ago to send messages in the form of short and long tones (dots and dashes, respectively). Letters and digits (0-9) are mapped onto those sequences of dots and dashes. For example, the letter E is one dot, the letter M is two dashes. Novices are likely to confuse similar letters than dissimilar letters, so confusion percentages provide information on the perceived similarity of these stimuli. The number 0 (5 dashes) is unlikely to be confused with the letter E (one dot). The observed matrix of confusion percentages is



The MDS procedure gave rise to this two dimensional solution of number of components (1 to 5) and whether the signal is predominantly dots or predominantly dashes.



(a) Examples of data suitable for MDS and related dimensional reduction techniques like PCA; each of these examples of data collection methods produce a data matrix that has the same shape as a proximity matrix so can be analysed with MDS and related techniques; a proximity matrix refers to either a distance matrix or a similarity matrix

proximity matrix

i. judgment of similarity or distance

ii. frequency of confusion

iii. discriminatory reaction time

iv. frequency one word is associated with another

v. sort cards into two stacks—count how often any two objects are put in same pile across many subjects

vi. frequency of people communicating with another group member (note that this measure need not be symmetric) or animals grooming each other

vii. for a group people, a "yes" or "no" whether person i and person j are "facebook friends"

viii. correlation matrix (original data are not "distances" but the data analyst converts original data into a matrix of correlations which is a type of proximity matrix)

ix. the heart of deep learning algorithms and "kernel methods," which work with distance matrices

These include a wide ranging number of applications. Some have used MDS with animals and insects, who obviously can't provide direct similarity comparisons. For instance, Backhaus et al (1987, *Biological Cybernetics, 56,* 293-304) examined color similarities in bees using a clever paradigm involving sucrose, 12 color patches, and videotaping where the bees "chose" to land. The pairwise choice proportions led to a matrix that has the same structure as a distance matrix. They recovered a two dimensional space that mimics the psychophysiological properties of bees' visual receptors in the blue-green color region. The experiments involved stimuli with natural properties that bees encounter (e.g., the reflectance of leaves and material like stones under natural light).

The MDS approach with extensions is used by satellite systems to locate sensors and devices, such as your smartphone's use of satellites to locate your coordinates on a map (see Costa et al, 2006, ACM Transactions on Sensor Networks).

The key feature underlying all of these elicitation procedures is that the investigator is not required to ask subjects particular dimensions in their judgment. This is an important difference from other techniques that we will cover later in this class where the dimensions are imposed on the subjects by the questions asked. For example, a questionnaire may ask people to rate different products on 8 dimensions (such as taste, texture, etc.) so the investigator is forcing the subject to use those specific dimensions. The inferences

you can make from this setup will be on those 8 dimensions, the ones you happened to ask of the subject. However, pure MDS-type procedures have the feature that you recover the dimensions subjects used without imposing any additional constraints from the way you ask the question. This makes MDS-style data collection very easy to do with unobtrusive measures, on animals, interconnectivity of different types including brain networks, etc. The color patch example and the morse code examples presented in class are good examples of the underlying dimensions that are recovered from MDS. For example, the morse code examples shows that the confusion proportions on the 36 letters and digits can be represented as two dimensions, number of components and predominance of dots/dashes; the subjects may not have been aware those were the dimensions nor where they asked directly—the confusion matrix of proportions pointed to a simplification of 36 stimuli down to two dimensions.

(b) Examples will be given in class from previous group projects as well as published literature

(c) Distance & similarity axioms (assumptions used in MDS). The axioms for distance and similarity are related but must be stated differently.

The general term used in this literature to encompass both "distance" data and "similarity" data is "proximity".

For an elementary introduction to the concept of distance from a mathematical point of view see the monograph by Shreider, *What is Distance?*, 1974, translated from Russian by Cohn and Edelberg. It is an excellent book that is both readable and deep.

distance axioms

Metric axioms for distance:

positivity: $d_{ij} > d_{ii} = 0$, for $i \neq j$
symmetry: $d_{ij} = d_{ji}$
triangle inequality: $d_{ij} \leq d_{ik} + d_{kj}$

The triangle inequality says that the direct distance between two points i and j never exceeds the indirect distance via a third point. Equality occurs when point k in the above notation lies in the shortest path between points i and j (e.g., in Euclidean distance this means that k falls on the straight line between i and j).[1]

[1] A tangential note for those who undertake additional reading in this area.... Published articles and books may use slightly different notation or list of axioms, but they are equivalent to what I present above. For instance, some authors in the mathematics literature define distance with these two axioms (it is considered macho to define concepts with as few axioms as possible):

restricted positivity: $d_{ij} > 0$ if and only if points i and j are different

It is possible for data to violate these seemingly basic axioms. For example, symmetry was shown to be violated by Tversky in similarity judgments comparing a general to a specific (e.g., people's judgments of how similar North Korea is to China are not equal to judgments of how similar China is to North Korea). The morse code confusion data also showed some violation of symmetry (d(U,A) = .63 but d(A,U) = .86). A more intuitive, if not pedestrian, analogy of a violation of symmetry is that hiking uphill is a very different experience than hiking downhill even on the same route.

similarity axioms

Metric axioms restated for similarity:

reflexive extremality: $s_{ii} > s_{ij}$ (each object must be at least as similar to itself)
symmetry: $s_{ij} = s_{ji}$
triangle inequality: s_{ij} large implies $s_{ik} \approx s_{jk}$

The intuition for the similarity triangle inequality is that if two objects i and j are highly similar to each other, then two objects similar to any third object must be equally good.

For simplicity, I will use the term *proximity* to refer to both distance and similarity.

It is possible to satisfy the triangle inequality artificially by adding a constant to all distances. Davison's MDS book shows that one can create a new dissimilarity measure

$$\gamma_{ij} = \begin{pmatrix} 0 & \text{if} \quad i = j \\ \delta_{ij} + c & \text{if} \quad i \neq j \end{pmatrix} \tag{10-1}$$

where δ_{ij} is the original dissimilarity and c = max($\delta_{hj} - \delta_{hi} - \delta_{ij}$) over all h,i,j. In other words, if you are willing to add a constant to all your data, you can always find a large enough constant so that the triangle inequality will be satisfied. I'll discuss issues surrounding the testing of the metric axioms later.

metric space

A "metric space" refers to both a set of objects and a distance metric that satisfies the above axioms. A metric space is usually denoted as $< S, d >$ where S represents the set of objects and d represents the particular distance metric used.

variation on triangle inequality: $d_{ij} \leq d_{ik} + d_{jk}$ for all i,j,k

The only difference between the previous statement of the triangle inequality and the present is that the last term has the order of points j and k reversed. Note that symmetry is not listed, nor is there any mention that $d_{ij} > d_{ii}$ or $d_{ii} = 0$. It turns out that these additional properties are implied by the above two axioms. For instance, to prove that $d_{ii} = 0$ apply the variation on the triangle inequality condition to the points i=j=k which yields $d_{ii} \leq d_{ii} + d_{ii}$ and implies $d_{ii} = 0$. To prove symmetry take the "variation on the triangle inequality" condition and let point k be the same as point i. This yields $d_{ij} \leq d_{ii} + d_{ji}$. But we already showed that $d_{ii} = 0$ so we have $d_{ij} \leq d_{ji}$. Repeat this argument with the positions of i and j interchanged and you get the conclusion that $d_{ji} \leq d_{ij}$, which together with the previous result implies symmetry (i.e., $d_{ij} = d_{ji}$). The mathematician who can define a concept with two axioms is considered more macho than the mathematician who needs three axioms to state the same thing; unfortunately, no macho points are given if the longer list of axioms is easier to understand.

Many different types of measures satisfy the distance axioms. For instance, the trivial binary measure that takes the value 0 if two points coincide and the value 1 if they do not satisfies the distance axioms. Of course, the usual Euclidean measure satisfies these measures as does measuring distance in terms of "city blocks." Two examples of a more interesting distance measure are using the number of inversions needed to make one rank order identical to another rank order (the greater number of inversions, the less similar the two rank orders) and the concept of a dominance metric, which will be discussed in class. The number of inversions as a distance metric is related to a class of rank order correlation measures such as Kendall's τ, Goodman and Kruskal's γ, Somer's d, and others.

Distance is also used in regression. The idea of minimizing the sum of squared residuals (SS residual) can be given a distance interpretation—the βs in the regression are selected to make SS residual as small as possible (i.e., minimizing the sum of the discrepancies between the observed data Y and the model-implied $\hat{Y}$).

(d) General Minkowski Distance Metric

The Euclidean lengths a, b, and c of a right triangle are given by the Pythagorean Theorem

$$\sqrt{a^2 + b^2} \quad = \quad c$$

In a two-dimensional plane a point is characterized by two numbers. For instance, the point X is the vector (x_1, x_2) and the point Y is the vector (y_1, y_2). The distance between the two points X and Y is given by c.

$$\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \quad = \quad c \tag{10-2}$$

This measure can be generalized to any number of dimensions by

$$c \quad = \quad \sqrt{\sum (x_i - y_i)^2} \tag{10-3}$$

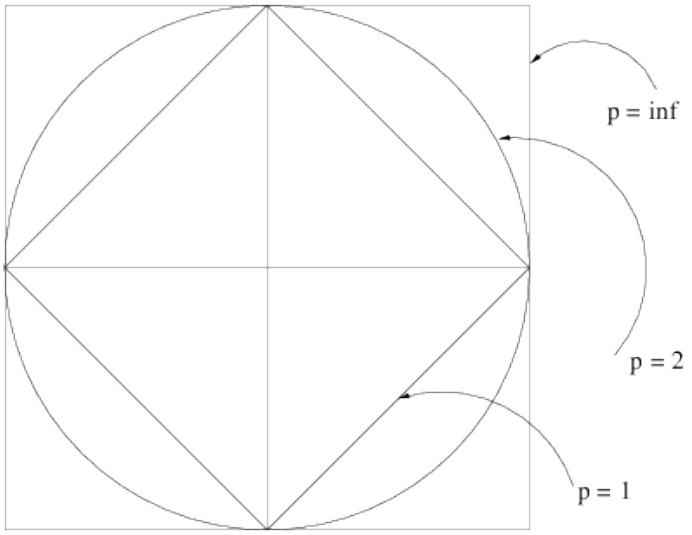
Minkowski metric

This type of distance can be generalized further. Minkowski proposed the general distance metric that produces a family of distance metrics

$$c \quad = \quad (\sum |x_i - y_i|^p)^{\frac{1}{p}} \tag{10-4}$$

When p = 1, you have the city block metric, when p = 2 you have the Euclidean metric ("as the crow flies"), and when p = infinity you have the supremum metric (aka the

Figure 10-4: The geometry of equidistant points in Minkowski metrics.



dominance metric) because this becomes d(x,y) = max(abs(x_i - y_i)), i.e., the total distance equals the maximum distance on a single dimension. When $p<1$, then the triangle inequality will not be satisfied, so it is not possible to have a Minkowski measure with $p<1$.

A good way to understand the family of Minkowski distance metrics is to look at what gets defined as "equally distant". In a Euclidean metric, once a point is located we know that a circle around that point defines all points that are equidistant to the original point. This intuition can be extended to other distance metrics. For instance, in the city block distance metric, equidistant points form a diamond shape around the original point. Walking two blocks to the east is the same distance as walking two blocks to the north, and they are both the same distance as walking one block east and then one block north. In each of these cases a person walked a total of 2 blocks. For a dominance metric, the set of equidistant points form a square around the original point. See Figure 10-4 for an illustration of the set of equidistant points for three examples from the Minkowski family. Note that the Minkowski metrics are continuous on this graph. That is, all exponents p between 1 and 2 fall continuously between the diamond and the circle, and all exponents p between 2 and ∞ fall continuously between the circle and the square.

The Minkowski distance metric has many features similar to the Euclidean metric. For instance, adding a constant to all points (i.e., shifting the origin) does not change the interpoint distances. In symbols, d(x,y) = d(x+c, y+c), where x, y, and c are vectors.

Similarly, multiplying all points by a vector (shrinking or expanding the dimensions) will simply change the interpoint distance by that multiplier. In symbols,

$$\text{d(kx, ky)} = \text{k d(x,y)}$$

We saw in Lecture Notes #6 that the covariance follows these uniqueness patterns related to addition and multiplication by constants, and, as we will see later, a covariance matrix can be treated as a similarity matrix.

What won't work in general across all metrics is rotation. In a Euclidean metric, a space can be rotated and all interpoint distances remain the same. However, not all Minkowski metrics satisfy rotational invariance.

An alternative term for the Minkowski metric is the L^p norm. This terminology is commonly used in machine learning and other subareas. The Euclidean metric is referred to as the L^2 (pronounced "l two") norm and the city block metric is referred to as the L^1 (pronounced "l one") norm. We saw the former used in a previous lecture notes for ridge regression and we will see the latter used in a subsequent lecture notes for the lasso regression.

(e) Minimize Stress

We need to define the analog to residuals so that we can apply these metric ideas to data.

One proposal is to find a set of points such that the squared residuals from the observed distances to the model-implied distances (as defined by the Minkowski metric) are minimized. Here we use a measure analogous to sum of squares, which in this literature is called *stress*.

original stress definition

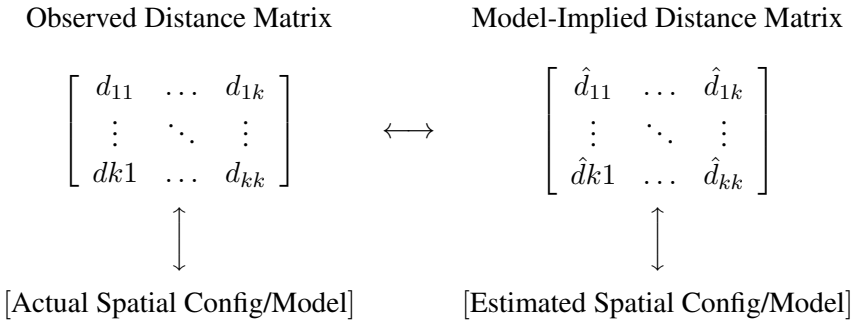
$$\text{stress} \quad = \quad \sqrt{\frac{\sum (d - \hat{d})^2}{\sum \hat{d}^2}} \tag{10-5}$$

where the denominator is a normalization[2]. The letter d represents observed data and $\hat{d}$ represents the "model-implied" distances. The model-implied distances are the derived distances according the configuration of points the model inferred from the data, somewhat analogous to how regression infers $\hat{\text{Y}}$ from its estimated β parameters.

We can represent these terms conceptually as follows. On the left is the matrix of ob-

[2] ALSCAL uses a different formula for stress, one developed by Young, to do the iteration. ALSCAL also reports the stress value according to Equation 10-5 in the output, but the computation/minimization is performed with respect to the Young formula. Interested readers can see the Davison book for details.

served distances; on the right is the matrix of model-implied distances that arise from the spatial configuration (e.g., placement of computed points on the resulting dimensional space).



The "spatial configuration/model" can also be a set of regression equations. Those regression equations give rise to a model-implied covariance matrix (which is a similarity matrix) that we compare to the observed covariance matrix (also a similarity matrix), which emerged from actual data generating mechanisms. This way of thinking of regression and covariances is called structural equation modeling, which we will cover in Lecture Notes 13. This provides a hint for how the material in these lecture notes will be used in later lecture notes and will connect to all the material from Lecture Notes 1 - 9.

observed ds are transformed

The minimization routine developed by Kruskal uses a steepest descent algorithm. Actually, the observed proximities d don't enter into the computation of stress, rather a transformation f of the data, i.e., f(d), is used. Different MDS programs and procedures take different f's. For example, if f is linear, then we have a solution that is identical to principal components analysis (covered in LN11 and is a special case of a deep learning neural network). If f is a monotonic transformation, that is, a transformation that merely preserves the order of the original data such as taking ranks, then we have what is called "nonmetric MDS". In the program ALSCAL, when you specify "interval" you are asking for f to be linear, and when you specify "ordinal" you are asking f to be a monotonic transformation (described below in the section on monotonic regression).

The algorithm ALSCAL uses is iterative. It starts by putting a random configuration of points in the space of specified dimensions (some MDS programs, like ALSCAL, perform a pre-analysis of sorts to get a better initial configuration than merely "random") and then "moves" the points around until stress is minimized. That is, points are moved a little, stress is reassessed, points move again, stress reassessed, etc., until convergence is reached. The algorithm uses information from the current configuration to decide which direction to move the points to have an optimal opportunity to improve the fit at the next iteration (i.e., reduce stress at the next iteration). The algorithm may include varying step sizes to minimize the chance of getting stuck in a local minimum. One of

the problem set questions will help you develop the intuition for this algorithm, which will be helpful for more complicated models we will tackle later this semester.

Paul Simon, the singer-songwriter, described the process of getting stuck while writing/composing a song as "everywhere I went led me to where I didn't want to be, so I was stuck." I think that is a good description of what can go wrong inside an algorithm when it is trying to optimize a measure like stress: if the fit of the model gets worse with any change that is made to the parameters so the algorithm has no where to go, we say the algorithm is stuck in a local minima. This could mean the optimal fit has been found (i.e,. that local minimum is a global minimum), so any other set of parameter values will produce a worse fit. Or it could mean that one is stuck in a local minimum, where small perturbations to the parameter values do not improve the fit, but there actually exists another set of parameter values that leads to a better fit but unfortunately that set of parameters isn't "visible" from your current vantage point. We'll develop this intuition more as we learn how to debug error messages in these types of algorithms.

A wonderful feature of nonmetric multidimensional scaling is that it uses only ordinal information from the data and yields solutions that are invariant under monotone transformations of the given proximity data. Thus, one can go a long way by only working with distance axioms and not much more needs be assumed about the quantitative or numerical properties of the data.

(f) Monotonic Regression (aka isotonic regression)

A technique to compute a regression that satisfies monotonicity (i.e., the computed regression is not restricted to be linear).

Animations will be presented in class to illustrate this algorithm.

When the data are very noisy and there is no pattern in the data, the monotonic regression fit will simply lead to a straight horizontal line.

Similarity produces a descending monotonic regression plot whereas distance produces an ascending plot. This is an important fact to remember as it can help you spot a common error. One needs to tell the program whether the data are similarity or distances, and among other things that sets whether the monotonic regression is fitting an ascending or descending pattern. A key distinguishing feature of distance data is that the diagonal contains 0s (distance between a point and itself), whereas, for similarity data the diagonal contains the maximal data points (a point is maximally similar to itself).

The nonmetric MDS program uses monotonic regression to relate the observed distances

to the computed distances at each iteration. That is, (1) take a configuration, (2) compute the monotonic regression of the model-implied distances for that iteration against the observed data, (3) compute stress using the results of the monotonic regression (recall what I mentioned on page 10-15 where the observed ds are transformed) and the model-implied distances, (4) use the results from the monotonic regression to find a new configuration, and (5) go back to step (1) repeating until stress is minimized.

(g) Design Considerations for MDS Studies

i. selecting stimuli and subject population

A. need good range of stimuli so dimensions can emerge

ii. presentation of stimuli

A. what scale to use (how many points, labels used on the points); keep in mind that it would be silly to use a 3-point scale with 15 stimuli because you'd have too many ties in your judgments; usually a 15 or 21 points scale is good because it permits fine grained discrimination, which the subject will want to make after seeing lots of stimuli pairs

B. tests of symmetry; may want to present stimuli in different orders to check whether you get the same distance estimate; Hubert & Baker, 1979, *Quality and Quantity, 13,* 77-84, propose a one way to measure violation of symmetry

C. tests of positivity (not always possible); can do it with auditory items like the morse code data

D. instructions to subject

E. cues you want subject to use or not use

F. will the subject have some practice trials to calibrate the scale and understand the procedure

(h) Program Issues (compiled from several papers by Shepard and also several papers by Kruskal)

i. Indications of entrapment in a local minimum

A. large stress

B. one-dimensional solution

C. if the plot of stress by number of dimensions is nonmonotonic, that could be an indication of being trapped in a local minimum for the solution that is out of order

ii. Avoiding entrapment

A. use several random starting configurations

B. use a rationally constructed starting configuration (such as a metric solution).

iii. Indications of Degeneracy

A. tied distances in configuration

B. step-like monotonic function (with only a few "steps")

C. zero stress (i.e., spuriously low)

There is the possibility that objects cluster into two or more separate groups. MDS will give the groups but lose the structure within the group.

iv. Avoiding Degeneracy (possible remedial measures)

A. apply nonmetric MDS separately to each subset

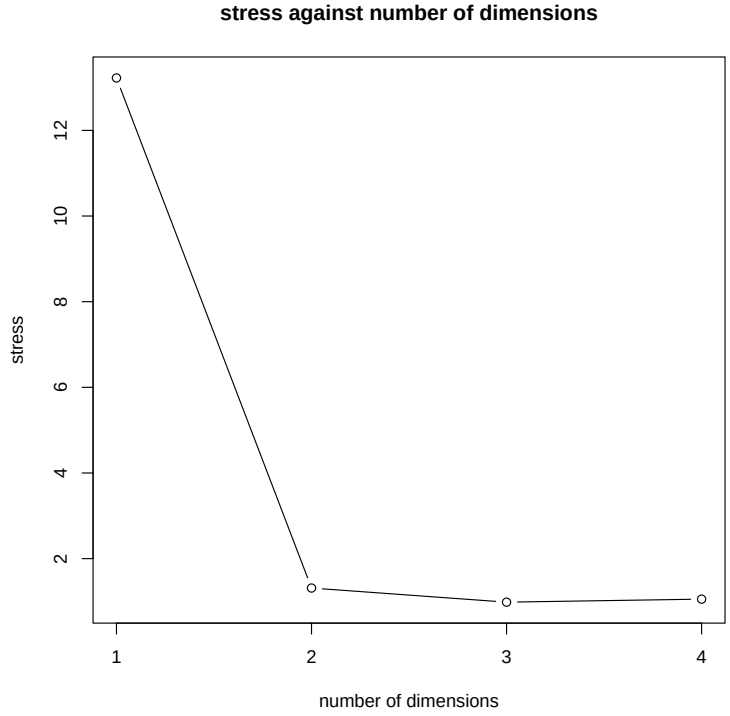
B. apply metric MDS to entire data (this gives more constraint but one loses the invariance under ordinal transformation)

C. avoid selecting objects for study that are highly clustered (problem is that an investigator may not know this in advance of the study)

v. Determining the Number of Dimensions

A. a plot of stress and number of dimensions (look for an obvious elbow and use the number of dimensions where the elbow occurs); some people call this a "scree plot"; some people (e.g., Davison) recommend using number of dimensions where the elbow occurs plus one; the scree plot is simply a heuristic so maybe consider number of dimensions near the elbow (i.e., n - 1, n, n + 1).

SPSS and R do not provide this plot in ALSCAL, so you need to do the scree plot manually. For example, for the cities example I ran MDS four times varying the number of dimensions from 1 to 4. For each run I computed stress and then plot the number of dimensions against stress. You can see an elbow appears at 2 dimensions. The stress values were 13.2, 1.3, 0.98 and 1.05, respectively. Note that the stress value increased in moving from 3 to 4 dimensions (small increase so not easy to see on the plot) suggesting 4 dimensions is overfitting the data.



B. reliability of different data sets for K dimensions but not for K+1 dimensions; that is, if you get similar solutions for K dimensions across several data sets but different solutions with K+1 dimensions, then that's one signal that maybe K+1 dimensions is too many

C. interpretability (the criterion of overriding importance)—if nothing interpretable is added by an additional dimension, then don't include it.

D. critical pattern for one dimensional solutions: one dimension has a similarity data matrix that can be permuted to be decreasing away from diagonal

For example, consider this hypothetical data matrix below. The matrix has the characteristic pattern of a one dimensional solution. Even when the syntax asks for two dimensions, in the final solution the points are aligned in a straight line (see Figure 10-5).

```
data list free
 /a b c d e.

begin data
10 8 6 4 2
8 10 8 6 4
6 8 10 8 6
4 6 8 10 8
2 4 6 8 10
end data.

alscal variables = a b c d e
 /level = ordinal(similar)
 /shape=symmetric
 /model=euclid
 /criteria=dimens(2)
 /plot.
```

E. Michael Lee (2001, Journal of Mathematical Psychology, 45, 149-166) developed a Bayesian approach to determine the number of dimensions in MDS.

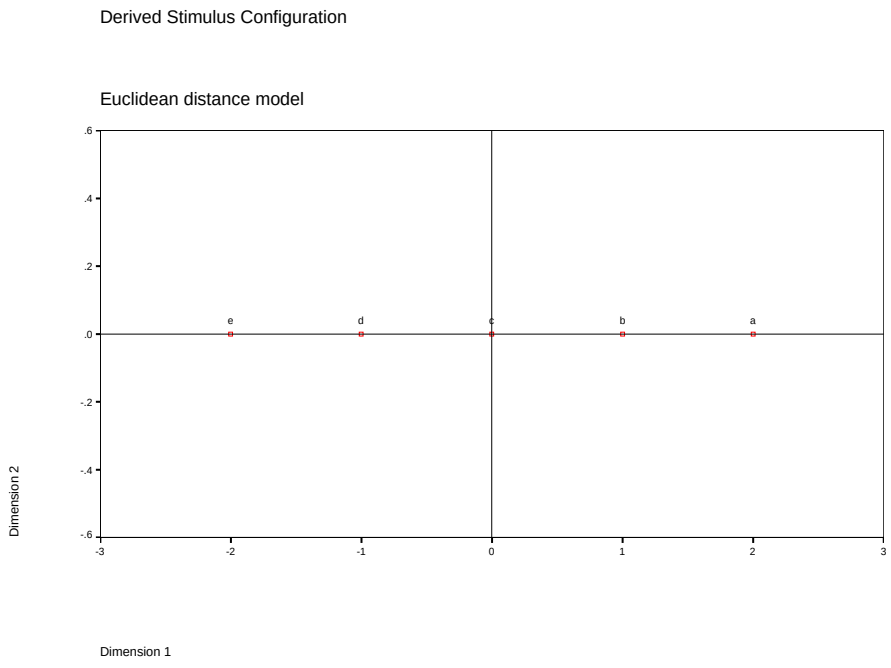
vi. Indications of Too Many Dimensions

A. stress is zero (or very close to zero)

B. points fall on a curved subspace (a C-shaped configuration is troubling unless it is implied by the theory such as in color or the critical pattern is present in the data matrix)

C. critical pattern for two dimensional, C-shaped solution: if you really have a

Figure 10-5: Solution of matrix with characteristic of a one-dimensional solution.



circle, then the data matrix should exhibit a pattern where similarities decrease and then increase away from the diagonal (more on this with PCA).

Here is a hypothetical example of a similarity matrix that exhibits a "C" shaped pattern that we can trust as "real" (i.e., not spurious because we are fitting too many dimensions). The solution appears in Figure 10-6.

```
data list free
 /a b c d e.

begin data
10 8 6 8 10
8 10 8 6 8
6 8 10 8 6
8 6 8 10 8
10 8 6 8 10
end data.

alscal variables = a b c d e
 /level = ordinal(similar)
 /shape=symmetric
 /model=euclid
 /criteria=dimens(2)
 /plot.
```

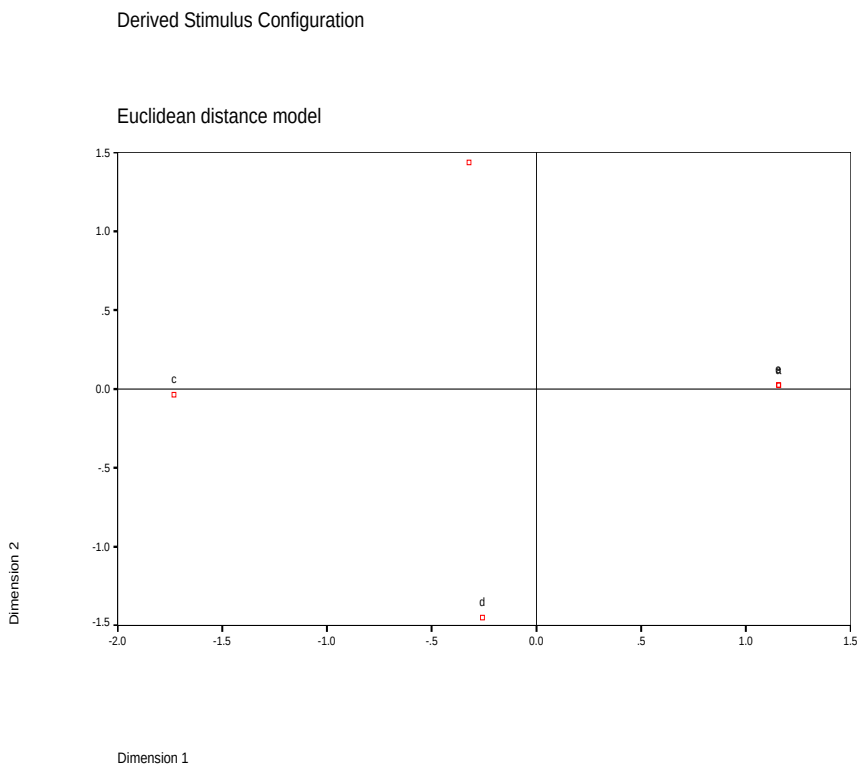
vii. Interpretation of the Spatial Configuration

A. Distance is invariant up to rotation for commonly used metrics, so you are free to rotate the final solution. There is no correct orientation; even on maps, north-south pointing up/down is a convention. This is where the exploration comes in because you are trying to find the dimensions used by the subject.

B. One could collect new data to aid in the interpretation of the dimension. One could then regress these new data on the computed dimensions (e.g., the set of dimensions as predictors and the average rating on the scale as the dependent variable). More on this below.

C. The goal of MDS is exploratory, so the result should give ideas for future research and manipulations. A feature of MDS is that you let the subject define the dimensions he or she uses, so you might discover new dimensions that do not fall out of theory.

Figure 10-6: Solution of matrix with characteristic of “C” shape in two dimensions.



viii. Comments on the interpretation of the stimulus configuration

A. Rotation

The following discussion assumes that a Euclidean solution was requested as opposed to, say, a city block metric. For a Euclidean solution, rotations are possible, hence the discussion in this section. For nonEuclidean solutions, not all rotations are possible and one needs to be mindful of which type of rotations are permissible by the particular metric one is using.

Sometimes rotation of axes help facilitate interpretation. The ALSCAL program chooses one particular orientation but you are free to rotate (assuming you run the option method=euclid).

Orthogonal rotations preserve the initial angle of the axes (which in this case is 90 degrees)[3]. When are dimensions orthogonal? Orthogonality in this general context is defined as follows: a matrix M is orthogonal if $MM' = M'M = I$, where the prime denotes transpose and I is the identity matrix.

B. Additional measured variables

Additional measures can be either "subjective" or "objective." For instance, in a class project on the representation of pizza, the students hypothesized that the thickness of the crust would be one dimension that people judge various pizzas. They predicted that one of the dimensions in the MDS solution would be crust thickness[4]. Their study involved having eight different pizza houses deliver a plain cheese pizza at a preset time and asking subjects (people over for a pizza party) to do all possible comparisons of pizza on similarity. Take a bite of A, then of B, rate their similarity. Take a sip of a "liquid refreshment", and repeat with two other pizzas, etc. Of course, there are many factors not controlled for in this study.

To evaluate their interpretation of the MDS dimensions, the investigators took two additional measurements on each pizza. After all pairwise similarity judgments were collected, subjects then rated each pizza individually on the thickness of the crust (How thick is the crust? on a 7pt scale). Note that during

[3]A rotation can be expressed in linear algebra terms. For original matrix X and rotation T, we have a new matrix Y=XT where each row of T corresponds to a dimension of X and each column corresponds to a dimension of Y. The element t_{rc} is the cosine of the angle between dimension r in X and dimension c in Y. Beware: in a nonEuclidean space, the matrix Y may not be a permissible solution in that metric.

[4]Other hypothesized dimensions in addition to crust thickness were the texture of the crust, the amount of sauce, the flavor of the pizza sauce, the amount of cheese, and the consistency of the cheese.

the similarity ratings subjects were not instructed to think about any particular dimension, only after all similarity data were collected were subjects asked to judge each pizza on the thickness of the crust.

In addition to asking subjects to rate their perceptions of crust thickness, the researchers also measured the average crust thickness of each pizza by taking 10 measurements at random along the outer crust and averaging them (the outer crust on a pizza usually has "bubbles" making the objective measurement of crust thickness difficult).

These two additional measures of crust thickness were then correlated with the dimensions from the MDS solution. If one, or both, of these variables correlate with an MDS dimension, then that gives confidence that the particular MDS dimension can be interpreted as crust thickness (either as defined by subject's perception or actual measurement). This is one way that the MDS dimensions can be validated.

Note that in this example the additional data (the objective measurement of crust thickness and the subjective ratings of crust thickness) are somewhat different in how they are collected. The objective measure of crust thickness is computed pizza by pizza, so each pizza gets a single score of crust thickness on this measure. The correlation between that and the MDS dimensions is straightforward because MDS assigns each pizza a number on a dimension, e.g., if there were eight pizzas there would be eight values on dimension 1 and eight measurements of pizza thickness. Here's an example of what I mean using MDS dimension 1 and objective crust thickness:

Pizza	MDS Dimension 1	objective crust thickness (inches)
Round Table	-1.01	0.50
Pizza Hut	0.43	.75
⋮	⋮	⋮

If MDS dimension 1 and this objective measure of crust thickness correlate highly, then that validates our interpretation that MDS dimension 1 is related to objective crust thickness.

However, the subjective measure of crust thickness was collected subject by subject so it was necessary to take a measure of central tendency (like the median) for each pizza in order to "average" over subjects and get numbers for each pizza. That is, the table of subjective measurements look like this (each subject was asked to rate each pizza on how thick they perceived the crust using a 7pt scale ranging from very thick to not at all):

	Subjective Rating Thickness			
Subject	Pizza 1	Pizza 2	Pizza 3	...
1	5	7		
2	4	3		
⋮	⋮	⋮	⋮	⋮
median	4	3		

Note that there is a similar table for each subjective rating asked. For example, if the investigators had also asked each subject to rate each pizza on how salty the sauce tasted, then there would be an analogous table for the saltiness rating. The goal is to aggregate the subjective ratings each subject gave into data that can be related to the MDS dimensions. One uses the medians above when correlating the subjective ratings with the MDS dimensions, as in:

Pizza	MDS Dimension 1	median subjective rating
Round Table	-1.01	4
Pizza Hut	0.43	3
⋮	⋮	⋮

Later, when we cover "individual MDS" I'll show you how to use each subject's rating on these additional measures in a more efficient way.

Actually, a better way to "correlate" the additional variables with the coordinates of the stimulus configuration is to run multiple regressions. Let each additional variable be a dependent variable and the dimensions of the stimulus configuration be the predictor variables. For example, if you ask for a two dimensional solution and had subject rate 6 different additional variables, you'll run a total of six regression (one for each additional variable) each with the same two predictors (the configuration from the two dimensional solution, i.e., the output from ALSCAL). You'll find that these R^2's will tend to be high. What you look for is cases where one slope is high and the other (close to) zero. That is interpreted as one dimension "correlating" with the additional variable independent from the other dimension(s). A result such as this supplies evidence in favor of interpreting that dimension in terms the additional variable for which its beta was high (but low for the other dimensions).

There is a general point here. If you hypothesize particular dimensions in an MDS solution, then it is a good idea to collect additional data so that you will be in a position to evaluate the appropriateness of the interpretations you place on the MDS dimensions.

C. Using the additional variables to inform your rotation

The additional variables collected above can also provide information about a possible rotation you may try. Run the multiple regressions I described above. For a given regression, transform the βs (raw slope coefficients) using this formula:

$$c_i = \frac{b_i}{\sqrt{\sum b_i^2}} \tag{10-6}$$

where b_i are betas. It can be shown that c_i is related to the angle of rotation needed to make dimension i coincide with the dependent variable in the regression. More specifically, c_i is the cosine of the angle θ with respect to the ith dimension. In a two-dimensional solution, start at the "east end" and angle θ moves counter clockwise. In other words, take the arccos of the c_i and you have the angle, e.g., if c for dim 1 is .99, then the angle of rotation to bring dimension 1 in line with that dependent variable is 8 degrees.

It is for this reason that above I interpreted the case of a multiple regression result where one beta is high and all others are near zero as suggesting that the one dimension with a high beta can be interpreted as being consistent with that dependent variable. The case when one beta is high and all others are near zero, corresponds to a c close to 1. This means that little or no rotation is needed to bring that dimension "in line" with the dependent variable, so the direction is consistent with the dependent variable.

No MDS program that I'm aware of does this rotation automatically. I'm just glossing over the details here and merely want to give you a flavor that a more rigorous way of interpreting dimensions and rotations is possible. If you are interested in learning more about using additional variables to decide on a sensible rotation, see the appendix of Kruskal & Wish (1978, *Multidimensional Scaling*). In order for you to implement this procedure in a general way, you will need a good understanding of linear algebra and a flexible graphing program such as the one available in the statistics package Splus or R. The good understanding of linear algebra is important not only because of the rotation involved but because the betas from the regressions are used in a complicated way, involving a projection in "predictor space."

D. Dealing with many dimensions.

It is not easy to "see" in more than two dimensions. On a computer screen, three dimensions can be shown as long as it is possible to spin the graph.

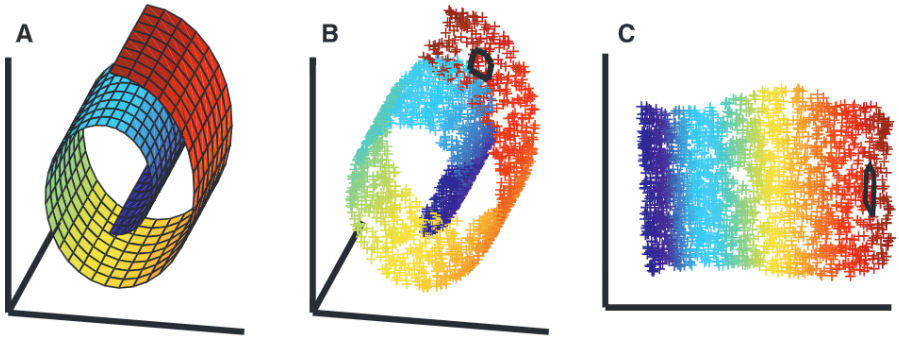
A simple strategy to understanding many dimensions is to select the one or two most extreme stimuli (in each direction) on each dimension. You can circle those values on the stimulus configuration matrix of the output. This heuristic will at least get you started on trying to interpret many dimensions. After you come up with an interpretation of that dimension by looking at the extremes, then look at the middle stimuli to see if they fall along the dimension in an orderly way according to your interpretation.

(i) Comparing multiple spatial solutions

Sometimes you will have MDS solutions from multiple groups of subjects. It is tempting to think of ways to compare the multiple solutions (for example, is the spatial solution for women different than the spatial solution for men?). Several approaches have been suggested to permit comparison of MDS solutions, but none work well across the board. In the case of two groups, one common approach is to transform one solution to be "as close as possible" to the other solution, then correlate the two resulting distance matrices. The first part of transforming one solution involves a series of rotations, translations, reflections and contractions to get one space to resemble the other. This is called a Procrustean transformation after the figure in Greek mythology who would make travelers fit their bed by stretching the short travelers and chopping off the feet of the tall travelers—so all travelers end up having the height. The second part of correlating the resulting transformed spaces across groups is itself problematic because correlations are invariant to linear transformation but a linear transformation of distances is not invariant. Further, the correlation is sensitive to the number of dimensions and number of points. See Borg and Groenen (2005) for a review of these issues.

(j) New developments in MDS

There have been several extensions of MDS, mostly in the computer science literature, dealing with a high number of dimensions and nonlinear mappings. One example of such an extension occurs when the underlying phenomenon follows a curved surface, such as a ribbon. The next figure displays a curved surface in three dimensions (panel A) with sampled points along that curved surface (panel B). We can roll out that curved surface (panel C). Without careful consideration of the geometric properties of this structure, traditional MDS approaches could become fooled by putting points along the blue section of the ribbon close to points along the red section of the ribbon (as would be the case given the structure in panel A or B). But from the perspective of the surface of the ribbon, the points located in the blue region should be relative far from the points located in the red region. See Roweis & Saul, 2000, *Science*, for a computational extension of MDS to address these types of issues.



2. INDSCAL: Generalizing MDS to include individual subject information

So far we have discussed multidimensional scaling and haven't said anything about how to incorporate an individual difference approach (i.e., how do subjects vary from a common group space). This is the information that individual difference multidimensional scaling (INDSCAL) provides. The price we pay is that we can no longer rotate the space for interpretability (for an unusual exception see Davison, p 131) but it turns out that INDSCAL solutions are mostly interpretable "out of the box." That is, typically solutions are interpretable without a rotation and are consistent with an MDS that has been rotated. The gain in using INDSCAL, however, is impressive: we can characterize individual subjects by how much they "use" or "invoke" the various dimensions, and that will allow one to test hypotheses about individual subjects or groups of subjects.

(a) Details

INDSCAL's distance computation

The primary mathematical trick in this procedure is to add a weight for each subject in the standard distance formula used to compute the model-implied distances:

$$d_{mij} = \sqrt{\sum^{K} w_{mk}(x_{ik} - x_{jk})^2} \quad (10\text{-}7)$$

where K is the number of dimensions, subscript m is the subject, w is the weight given by subject m on dimension k, and i/j are the two items being compared.

You get two simultaneous solutions from INDSCAL: a configuration of points (analogous to the usual MDS) and a configuration of subject weights. The stimulus configuration matrix, as with regular MDS, is interpreted as the common group space. The subject weights tell you how much each subject "used" each dimension. The important aspect of the subject weight matrix is which points fall above or below the identity line. You will have to draw the identity line in by hand (if you use SPSS, be careful of the silly scaling differences across dimensions that appear in its graphs). If a subject falls below the identity line, then that subject gives primary weight to dimension i; if a subject falls above the identity line, then that subject gives primary weight to dimension j. Don't worry about how far the subject's point is from the origin of the plot (that information is secondary). What matters most is the position of the subject's weight relative to the identity line.

It is possible to interpret the length of the weight vector as an index of the goodness of fit for that subject. In SPSS and R the subject weights are plotted as points but they are actually vectors. You can think of them as arrows originating at the origin of the SPSS weight plot and extending out to the point. So, each subject is represented by a vector rather than a point. The length of the vector (how far the point is from the origin) is related to how well the common space can be distorted by shrinking and expanding the dimensions with scalars to match the subject's own data. Because of this feature, some people (notably Young and Lewycky) have suggested that the weight space be plotted not with raw weights but with normalized weights (so all subjects are on equal footing with respect to the length of the vector). The most obvious normalization is:

$$w_i^* = \frac{w_i}{\sum w_i^2} \quad (10\text{-}8)$$

where w_i^* is the normalized weight on the ith dimension. In two dimensions, this has the effect of putting all the subject weights along the arc of a circle in the northeast quadrant. Carroll, the original developer of INDSCAL, has complained about the normalization of subject weights used in ALSCAL; he claims it gives incorrect solutions. Thus, ignore the "flattened weights" printed in ALSCAL until the debate gets resolved. To my knowledge, the issue of normalization of subject weights has not received much research attention (see MacCallum, 1974, for an exception). These methods, however, are gaining renewed attention by the computer science community with a modern machine learning lens.

INDSCAL cannot be computed with one dimension. The minimum is two dimensions. This makes sense given that the technique is estimating how much each unit weights each dimension, and this cannot be accomplished when there is only one dimension. The maximum number of dimensions in INDSCAL is one minus the number of stimuli/objects in the task (e.g., one minus the number of color patches in the proximity task).

(b) INDSCAL assumptions and their consequences

i. differential weighting of dimensions but one common space

This model creates a new configuration matrix for each subject using this formula: $X_s = XW_s$ where X is a group stimulus configuration, W_s is the weight matrix (diagonal k x k matrix where k is the number of dimensions). Thus each subject shrinks or expands the dimensions of the common space.

ii. the original characterization of INDSCAL was only for "interval" data (linear f in the language developed above in the section on stress); however, developments by Young, the person who wrote ALSCAL, permit ordinal data too

iii. no rotational indeterminacy (as in factor analysis); differential shrinking and stretching per subject is permitted only along the orthogonal axes, more general models exist (see Davison)

iv. a larger number of dimensions can be interpreted because rotations are not permitted

(c) Example

For a published example using INDSCAL see Clark et al (1989, Pain, vol 37, 23-32). This study provided subjects with pain labels such as "burning", "shooting", "cramping", "annoying", etc. They had subjects judge similarity between nine such labels. There were 24 healthy subjects and 24 cancer pain patients. In brief, two dimensions came up. One was pain intensity and the other was emotional quality, which was "sickening" vs "shooting" (basically whether there was an affective component to the description or just "perceptual"). Healthy patients gave predominant weight to pain intensity in their similarity judgments, whereas cancer pain patients gave predominant weight to emotional quality.

Here is a silly example involving television shows. Six participants rated the similarity

of 10 tv shows (e.g., love boat, melrose place, er, charlie's angels, baywatch, la law). Here is the syntax for an ordinal level INDSCAL:

```
DATA LIST FREE
  /LOVE MEL FRI  ER THREE BRADY CHARLIE BAY LAW  REM.


BEGIN DATA
[subject 1 distance matrix]
[subject 2 distance matrix]
...
[subject n distance matrix]
END DATA.

ALSCAL VARIABLES = LOVE MEL FRI  ER THREE BRADY CHARLIE BAY LAW  REM
  /SHAPE = SYMMETRIC
  /LEVEL = ORDINAL (SIMILAR)
  /MODEL = INDSCAL
  /CRITERIA = DIMENS(2)
  /PLOT
/PRINT = DATA.
```

The first plot in Figure 10-7 shows the stimulus configuration matrix (you can try your hand at an interpretation). There is a "C" shape to the plot.[5]

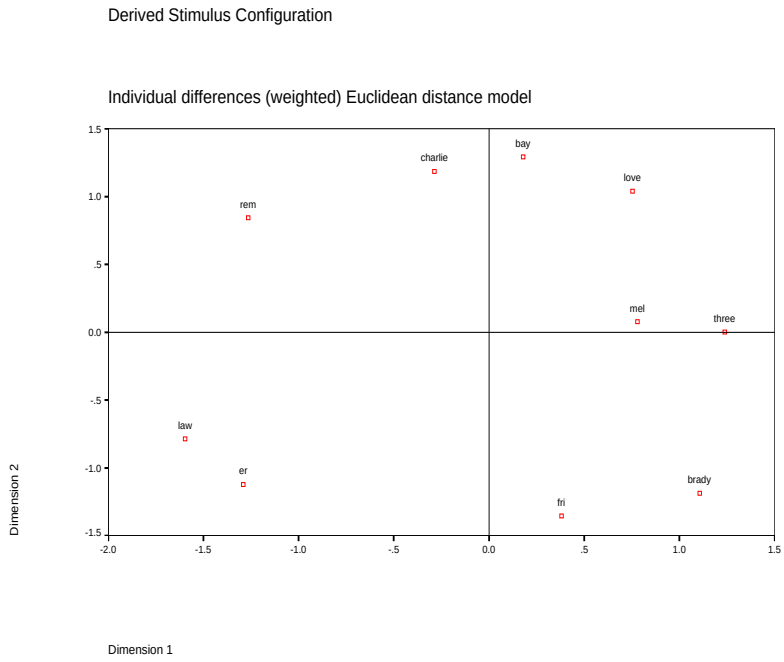
The second plot in Figure 10-7 shows the subject weight matrix. Note that SPSS graphics don't make life easy. I had to tinker with the range of the two axes but the plot is still difficult to interpret because the spacing along the vertical axis is not the same as the spacing along the horizontal axis. The best thing to do is draw, with a pencil, the identity line. What you most care about is which subjects are above or below the identity line. But here we see that most subjects are near or below the identity line (once that is drawn in by hand). This suggests that most subjects are primarily using dimension 1 or using both dimensions about equally. We could use this subject weight matrix to argue that the second dimension is not that important (unless for theoretical reasons it is an important finding to distinguish subjects who primarily use dimension 1 vs. use a 50/50 mixture of the two dimensions). For me, this example is a close call on whether one or two dimensions are needed. I'd want to see more data before making a conclusion either way.

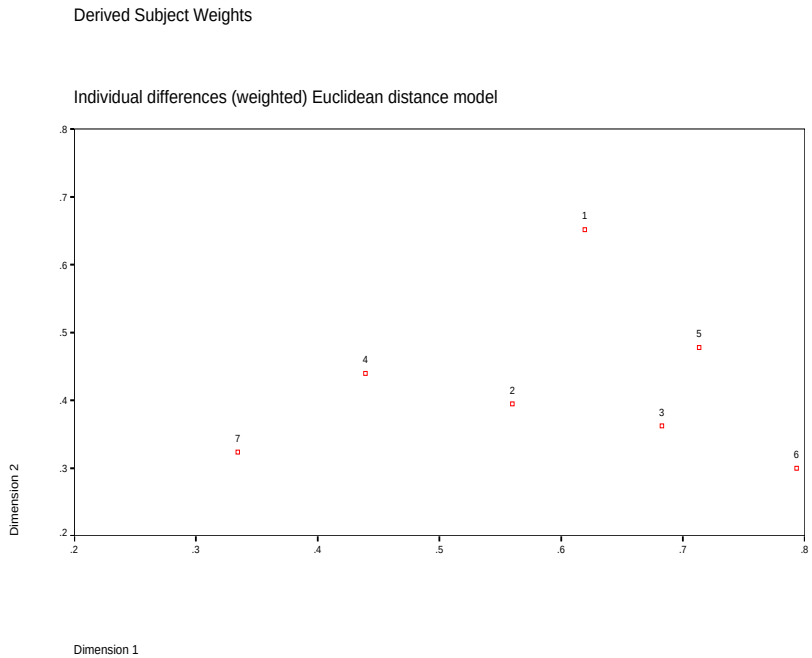
(d) Misc issues

Though this is not always something one should do, it is possible to compare the stim-

[5]Is this real or a result of overfitting? I don't know. The diagnostic that I gave in the MDS section of looking at the structure of the raw data matrix is difficult to apply here because there are six matrices, one from each subject. A sensible strategy would be to compute median data, run MDS, and double check whether the two dimensional solution is spurious. If two dimensions appear appropriate in the context of MDS, then that is some evidence in keeping two dimension in the INDSCAL solution.

Figure 10-7: Similarity judgments from 6 participants over 10 TV shows: stimulus configuration matrix and subject weights





ulus configuration from MDS with the stimulus configuration from INDSCAL by the multiple regression approach described above. Let the dimensions from one space be the predictor variables in the multiple regression, and each of the dimensions from the other solution be the dependent variables. It is also possible to compare two sets of configurations with canonical correlations, a technique we won't cover till the end of the term. Usually, MDS and INDSCAL provide similar stimulus configurations but INDSCAL provides the added bonus of subject weights.

Validating stimulus dimensions goes the same way as in regular MDS. One collects additional data and runs regressions on the data using the stimulus configuration as predictors. Similarly, it is possible to do additional analyses on the subject weights. For instance, save the subject weight for each subject on each dimension into a new data file. If you had additional data on the subjects (such as demographics, which experimental condition they were in the, etc) you could compare subjects on those additional variables. For instance, do men give more weight to dimension 1 than women? Do subjects in the experimental condition give more weight to dimension 1 than subjects in the control condition? In other words, you save the subject weights into a file and they can play the role of a variable that you can use in all kinds of analyses. Rather than "eyeballing" the subject weights, you can test rigorous hypotheses about them.

There have been some developments on the INDSCAL front. ANOVA and regression have been merged with INDSCAL so now it is possible to test hypotheses about subject weights all in one analyses. That is, subject weights, regression to predict those subject weights, and stimulus dimensions can all estimated simultaneously. For instance, do the weights for subjects in one group differ from the weight in subjects in another group? This permits tests of hypotheses involving subjects using different sets of dimensions. Details are available in Clarkson & Gonzalez (2001, *Psychometrika*).

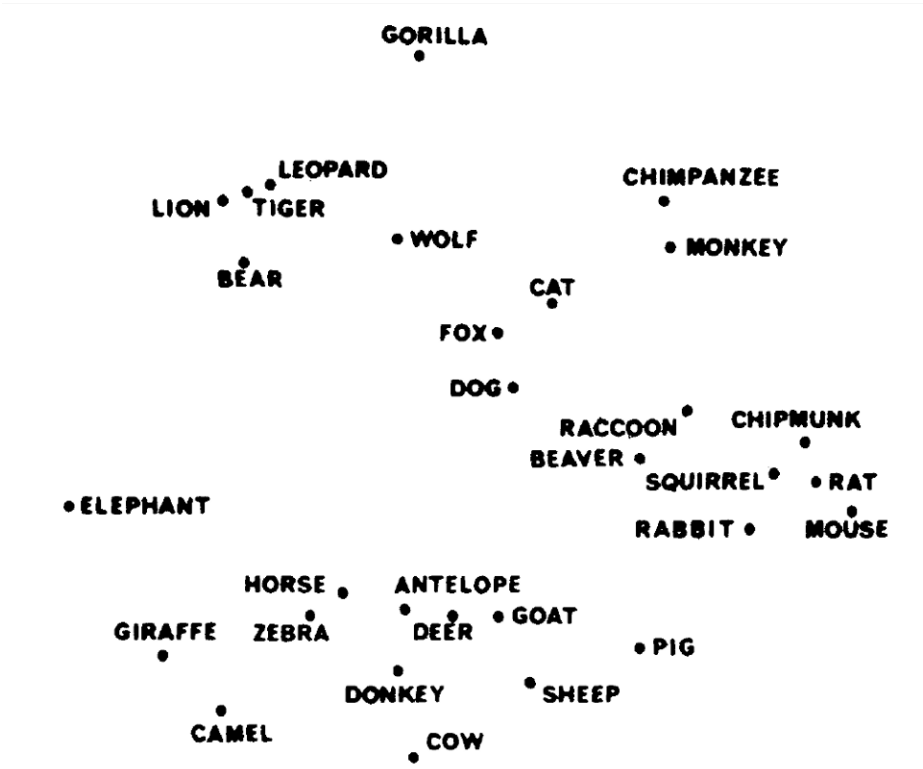
The INDSCAL approach has many applications that have not been fully utilized. One application is for analyzing complicated survey data. For instance, suppose a subject is asked to rate 10 different objects on 20 different scales. The objects could be products, cities to live in, universities, brothers/sisters in the subject's fraternity/sorority, etc. The scales can be typical attitude rating scales. For each of the N subjects it is possible to compute a correlation matrix of objects, so one has a 10x10 correlation matrix for each subject. It is possible to input these N correlation matrices to an INDSCAL analysis to find a common spatial representation and subject weight solution. It is also possible to compute a 20x20 matrix, subject by subject, over the scales (e.g., for subject 1, what is the correlation between scale 1 and scale 2 over the 10 objects), though this type of analysis will not be generally useful. As you think about these data structures, you realize that many more correlation matrices can be generated. For instance, for object 1, it is possible to correlate scale 1 and scale 2 over subjects, etc., leading to one 20x20 correlation matrix for each of the 10 objects. These matrices could also be subjected to an INDSCAL analysis; here the weight matrix would correspond to the 10 objects and the stimulus configuration would correspond to the 20 scales. If subjects respond

over several points in time, more possibilities for creating correlation matrices arise. In the years to come, I think INDSCAL analyses will take a more prominent role in the analysis of complicated psychological data.

For a complete discussion of MDS and related techniques see Borg and Groenen (1997) *Modern Multidimensional Scaling*.

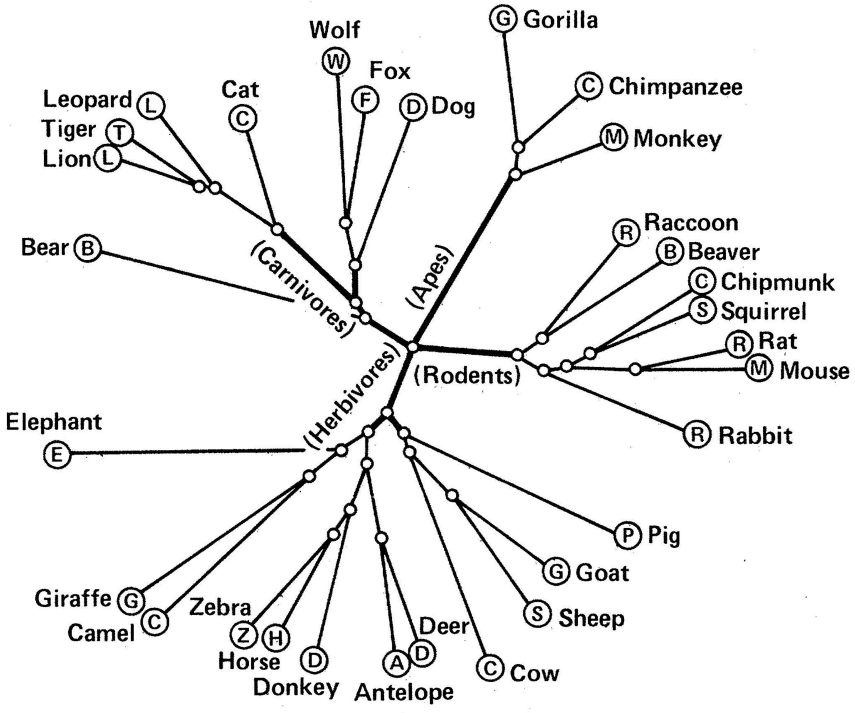
3. Additive Tree Structures (ADDTREE)

MDS and spatial solutions offer interesting representations of data. However, not everything satisfies the properties of dimensions and spatial arrangements. A classic example is from Henley who performed a study asking participants to rate the similarity of various animals such as dog, cat, lion, tiger, wolf, elephant, etc. Here is the MDS solution to the data in that study.

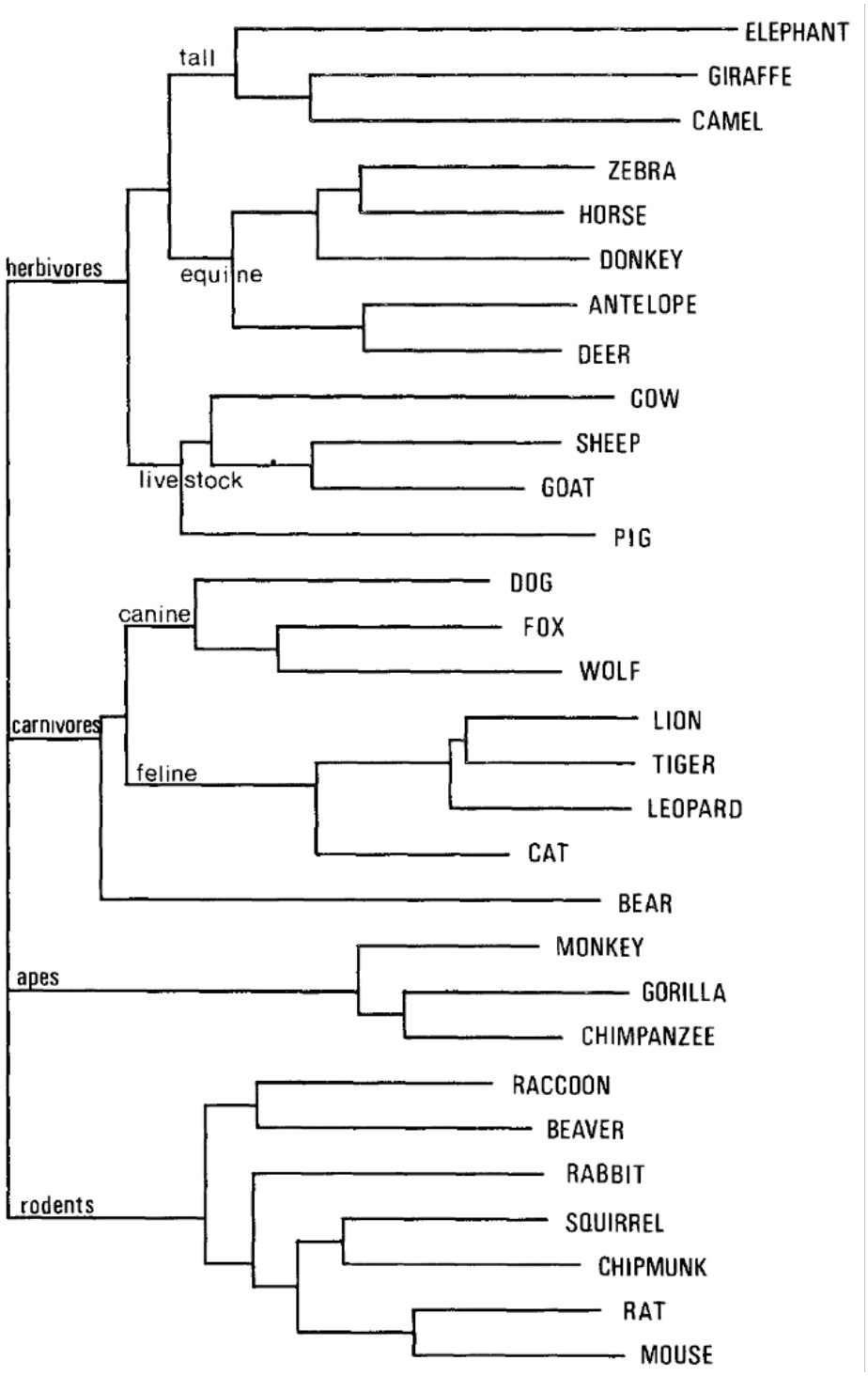


The MDS analysis turned out to be somewhat difficult to interpret with the solution not being completely satisfying. For example, here are two of the dimensions in the original paper. It is possible to interpret one dimension with a little rotation as animal size (smaller animals at one

end and larger animals at the other end) and the other dimension gets a little tricky but one can sort of see a "ferocity" dimension (animals that can harm you and those that can). Still, there is structure in the data that the dimension analysis misses. There is clustering, such as canines are close together, felines are close together, rodents are close together, farm animals are close together, etc.

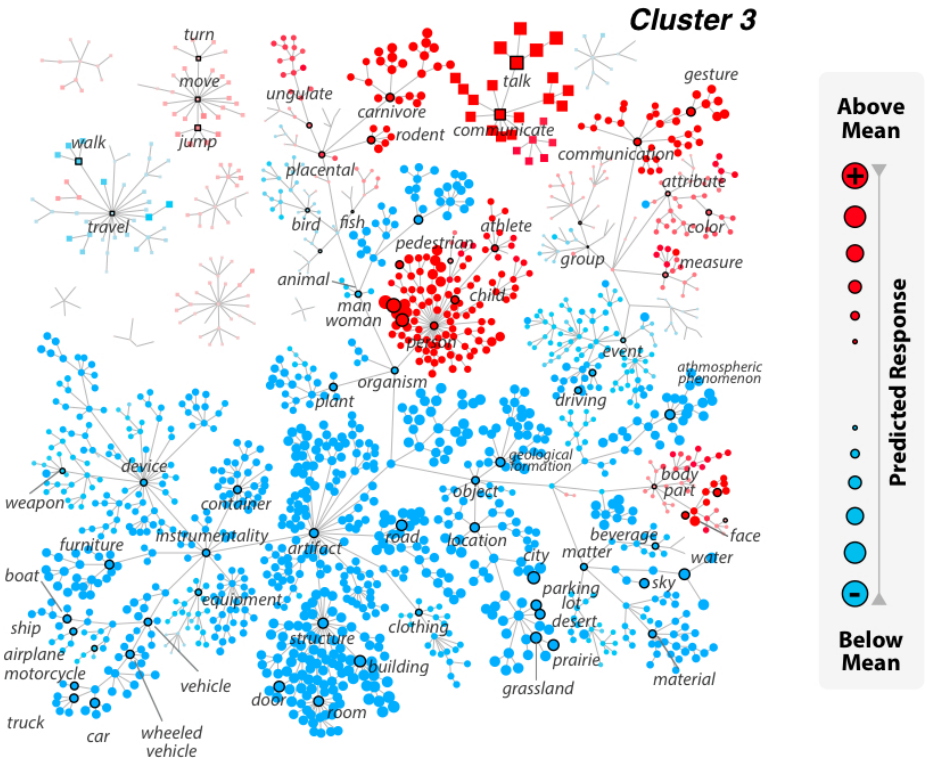


This led to the development of various clustering approaches to deal with data that do not seem to fit dimensional structures. Here is the analysis using the additive tree structure that we will cover in this subsection.



Some interesting recent uses of tree structures appear in brain imaging analyses. Here is one analysis from fMRI data of participants passively watching movies (in other studies, passively listening to podcasts led to similar results). For more examples of this type of application of

clustering to brain imaging data see gallantlab.org.



(a) The ADDTREE Model[6]

Additive trees are a special kind of tree structure. They are less restrictive than other types of tree structures that are typically used in social science research (e.g., the tree/clustering program available in SPSS). The logic of additive tree structures emerges from a simple model of similarity—similarity emerges from a comparison of common and distinctive features rather than a distance computation in a metric space.

In a tree model, the points are arranged in a tree structure, and one interprets clusters of points rather than dimensions. The distance between any two points, i and j, is defined as the sum of the lengths (or weights) associated with the links of the unique direct path between those two nodes. For instance, in Figure 10-8 the distance between points A and B is the sum of lengths α and β, and the distance between points A and D is the sum of lengths α, γ, and δ.

[6]Good references to ADDTREE for those wanting to do some outside reading include Sattath & Tversky, 1977, *Psychometrika*, 319-345 and a little book by Corter, *Tree Models of Similarity and Association* (Corter wrote the version of ADDTREE we will use in this class).

Figure 10-8: Four points in an additive tree.

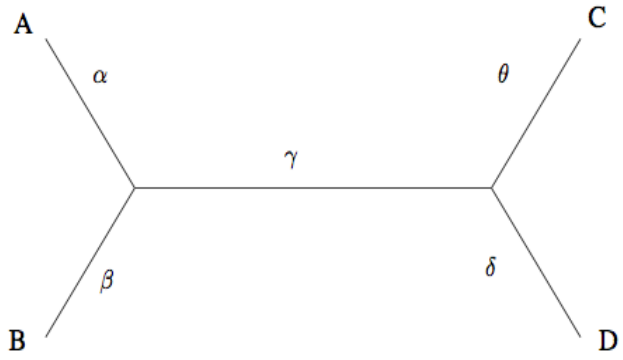
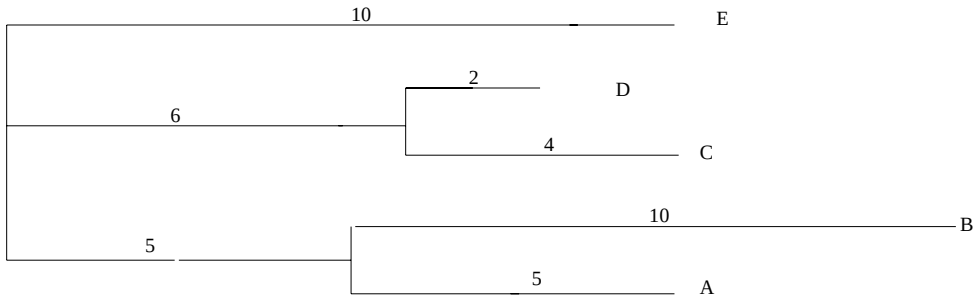


Figure 10-9: Example of an additive tree. Data matrix that generated this tree is given in the text.

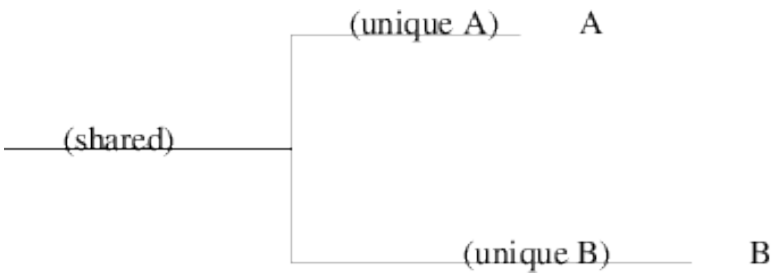


Here is a simple example with 5 stimuli (Corter, 1996, *Tree Models of Similarity and Association*). Assume this distance matrix (lowerhalf) between the five items:

	A	B	C	D	E
A					
B	15				
C	20	25			
D	18	23	6		
E	20	25	20	18	

This matrix perfectly satisfies an additive tree. Figure 10-9 shows this tree; note that the sum of the lengths are equivalent to the observed data. Only horizontal distances are interpreted on the graph. The vertical bars are placed in the graph only for convenience to help guide the eye.

Figure 10-10: Basic cluster in an additive tree.



The critical axiom for an additive tree is the *four-point property*, which is defined as:

For any four points (nodes), h,i,j,k,

$$d(h,i) + d(j,k) \quad \leq \quad \max((d(h,j) + d(i,k)), (d(h,k) + d(i,j))) \qquad (10\text{-}9)$$

This condition is satisfied for the five possible quadruples in the previous example. For instance, check that d(A,B)+d(C,D) $\leq$ max(d(A,C) + d(B,D), d(A,D) + d(B,C)). Under symmetry, the four point condition (vacuously) implies the triangle inequality.[7]

The ADDTREE program forces data to satisfy the triangle inequality by adding a constant to all distances. Similarity is handled by converting all data to negative numbers and then adding a positive constant to make the data positive (i.e., similarity is linearly transformed to distance).

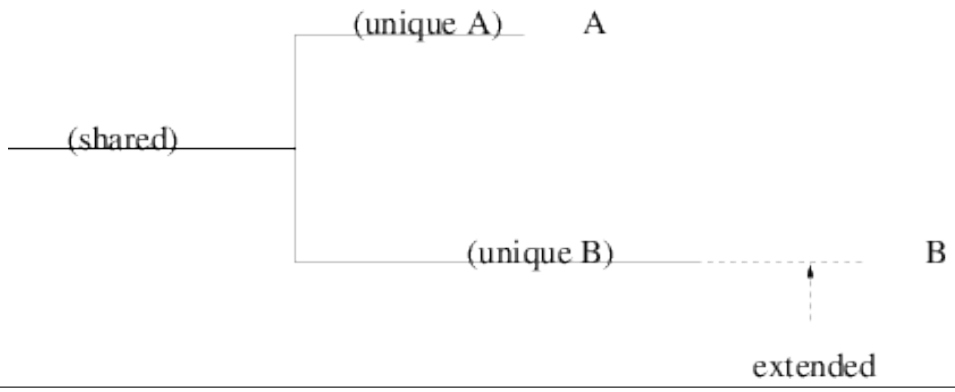
In an ADDTREE solution, the length of the cluster can be interpreted as a measure of shared features, and the lengths of the cluster to the individual items can be interpreted as a measure of features unique to that item (see Figure 10-10). Even though additive trees have better properties than the more commonly used hierarchical trees (see below), they still have limitations. For instance, additive trees are designed to handle clusters that are either nested or disjoint. Hierarchical tree models are even more restrictive—they handle cases where all clusters are nested.

[7] To see this, let k=j (that is k and j are the same point), and re-write the four point condition replacing all k's with j's and recalling that d(j,j) = 0. The left hand side of Equation 10-9 becomes simply d(h,i) and the right hand side becomes

$$\begin{aligned} \max((d(h,j) + d(i,k)), d(h,k) + d(i,j)) \quad &= \quad \max((d(h,j) + d(i,j)), d(h,j) + d(i,j)) \\ &= \quad d(h,j) + d(i,j) \\ &= \quad d(h,j) + d(j,i) \end{aligned}$$

and thus we have d(h,i) $\leq$ d(h,j) + d(j,i), which is the triangle inequality.

Figure 10-11: Basic cluster in an extended tree.



But what about the case when stimuli are not nested and clusters overlap? Corter, who developed the ADDTREE program, developed the EXTREE approach to handle this situation. In an extended tree, regions are added to the tree solution that relate to features shared with other items in other clusters. So, when computing distance between two items, different path lengths will be computed depending on whether the two items are in the same cluster or different clusters. This idea of an EXTENDED tree has not been completely developed yet and is not well-understood. If you are interested in tinkering with EXTENDED tree structures, you can download the software from Corter's web site (see the appendix on ADDTREE).

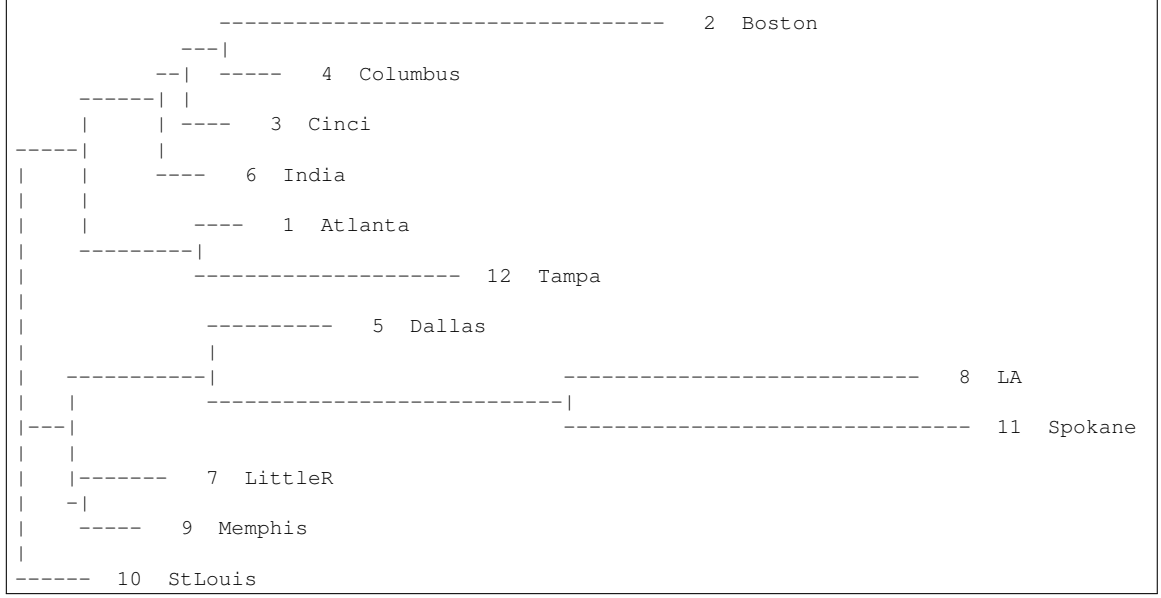
As natural as tree structures are, they don't always apply in all situations. For instance, a tree structure on the air distances between US cities makes little sense. See Figure 10-12.

There is quite a bit of research activity around additive clustering methods such as ADDTREE. Work by Michael Lee (2001, Journal of Mathematical Psychology, 45, 131-48) used a Bayesian approach to determine the number of clusters in additive clustering models. Hutchinson and Mungale (1997, *Psychometrika, 62*, 85-117) provide a non-metric algorithm for ADDTREE and EXTREE like models.

(b) Related Techniques: Hierarchical Clustering Methods

If the data matrix can be permuted (rows and columns) such that clusters form in the data matrix, then there is a special type of tree called hierarchical clustering. This kind of tree satisfies (in addition to the three metric axioms) the ultrametric property:

Figure 10-12: ADDTREE fit to distance between 12 US cities. Previously analyzed as an MDS solution. This tree is meaningless.



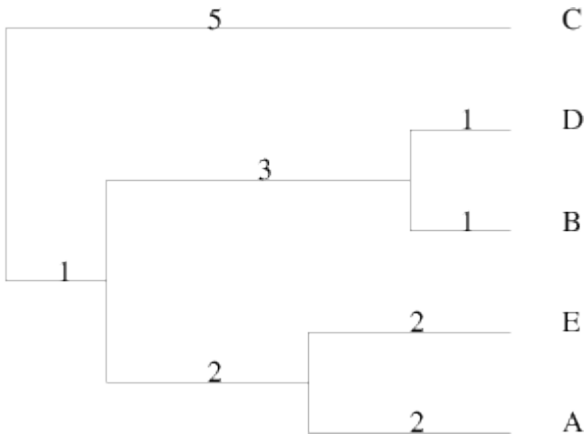
$$d(i,k) \leq \max(d(i,j), d(j,k)) \tag{10-10}$$

The ultrametric inequality implies that all "triangles" are acute isosceles triangles (i.e., triangles in which the two longer sides are always equal). The ultrametric condition is stronger than the additive tree condition. The ultrametric condition implies that, given two disjoint clusters, all intracluster distances are smaller than all intercluster distances, and all intracluster distances are equal, both of which are very strong restrictions on the data. In an additive tree structure, however, it is possible for intracluster distances to exceed intercluster distances (see Sattath & Tversky, 1977).

SPSS and R implement the hierarchical tree model using several different algorithms. Two popular algorithms are the single link method (connectedness) and the complete link method (compactness). The former uses a strategy by which the next object to be added to a cluster is the one with the smallest distance to the closest object in the cluster. It tends to produce long, chainlike clusters that may be difficult to interpret. The latter uses a rule where the next object to be added to a cluster is the one with the smallest distance to the farthest object in the cluster (tends to produce easier to understand trees).

Here is a simple example with 5 stimuli (Corter, 1996, *Tree Models of Similarity and Association*). Assume this distance matrix (lowerhalf) between the five items:

Figure 10-13: Example of an ultrametric tree. Data matrix that generated this tree is given in the text.



	A	B	C	D	E
A					
B	8				
C	10	10			
D	8	2	10		
E	4	8	10	8	

This matrix perfectly satisfies the ultrametric axiom. Figure 10-13 shows this tree; note that the sum of the lengths are equivalent to the observed data. This tree has the property that all pairs within a cluster have the same length.

As I stated above, a data matrix that perfectly satisfies the ultrametric property can be permuted so that the proximity matrix shows clusters. Here's one permutation that shows clusters with the required pattern:

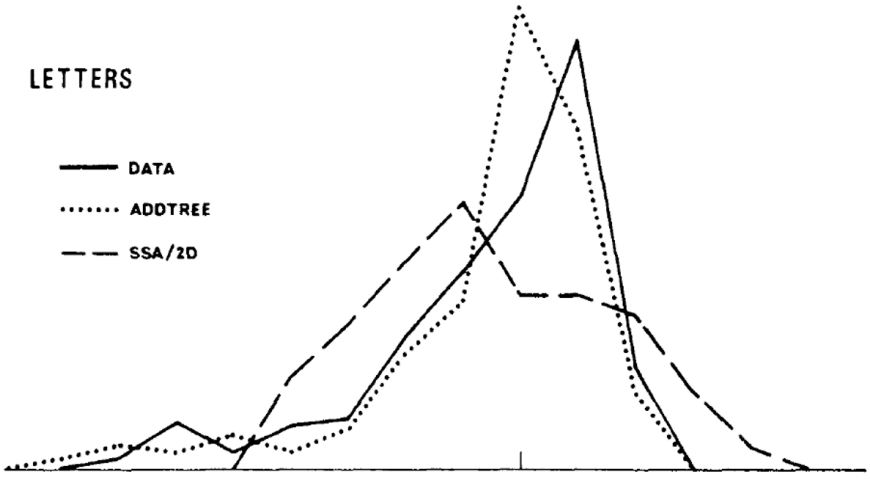
	D	B	E	A	C
B	2				
E	8	8			
A	8	8	4		
C	10	10	10	10	

Ultrametric trees have a single, unique root because all the points must be equidistant from the root. However, additive trees do not have a unique root. Much like picking

up a child's mobile in different places produces a structure that "looks" different, one needs to be careful of statements like "two major clusters were found" when dealing with additive trees. Are the clusters you see real or are they an artifact of the arbitrary "root" the program chose?

(c) Comparing tree and dimensional representations (e.g., ADDTREE vs. MDS)

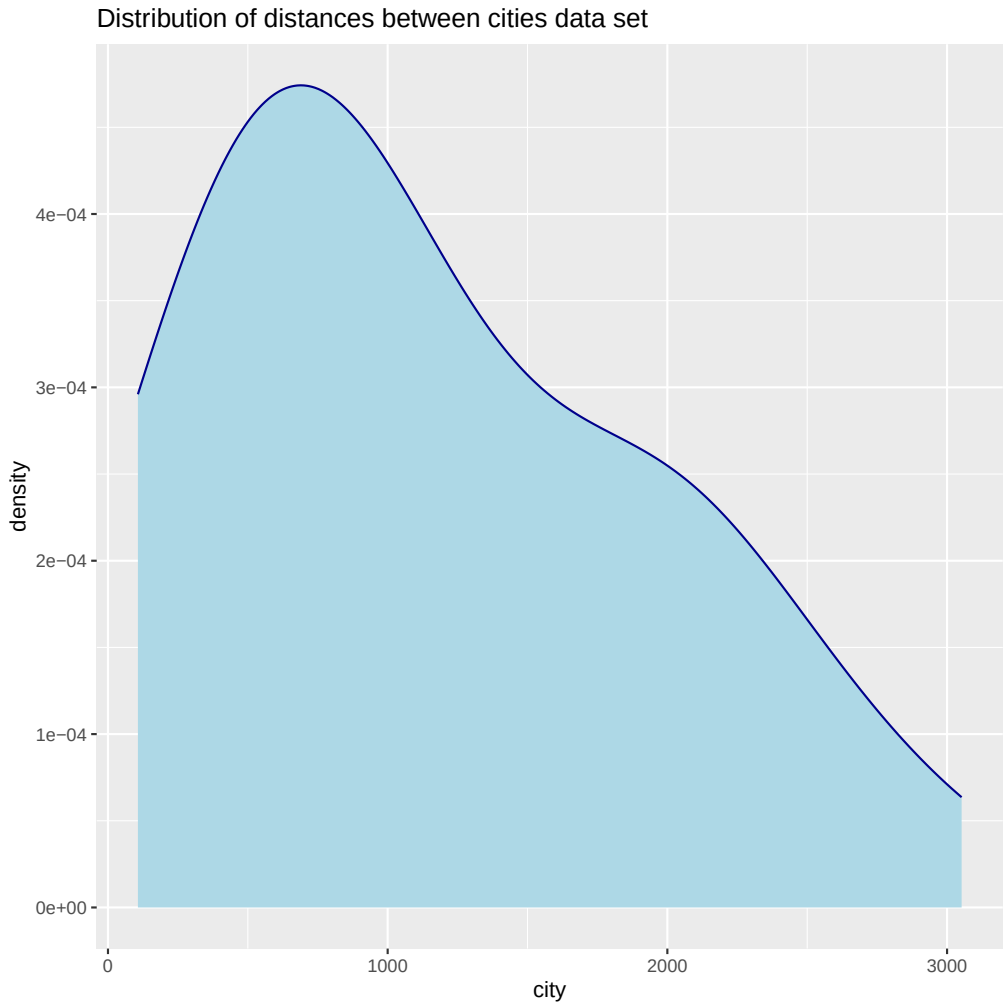
Sattath & Tversky (1977) showed that histograms of (1) actual distances, (2) distances from the MDS solution, and (3) distance from tree solution give a tell-tale pattern. Tree distances tend to be skewed to the left (the tail is in the left) because there are fewer distances that belong in the same cluster and more distances that belong in different clusters; in other words, few small distances and more large distances. However in MDS, the distances are skewed to the right (tail is on the right); in other words, more small distances and fewer large distances. By comparing data distributions of the observed proximities to each of the estimated tree and MDS distributions, it is possible to distinguish an MDS structure from a tree structure. A proof of this result is given in Sattath & Tversky (1977). A simple assessment of skewness can be made by simply comparing the mean and the median: a left skewed distribution has the mean greater than the median, whereas a right skewed distribution has the mean less than the median. Pruzansky, Tversky, and Carroll also suggest using the standard measure of skewness (third moment) as an index. A measure isn't necessary though because the key idea is whether the data histogram resembles the tree histogram or the MDS histogram, and usually the comparison is obvious as in the next figure.



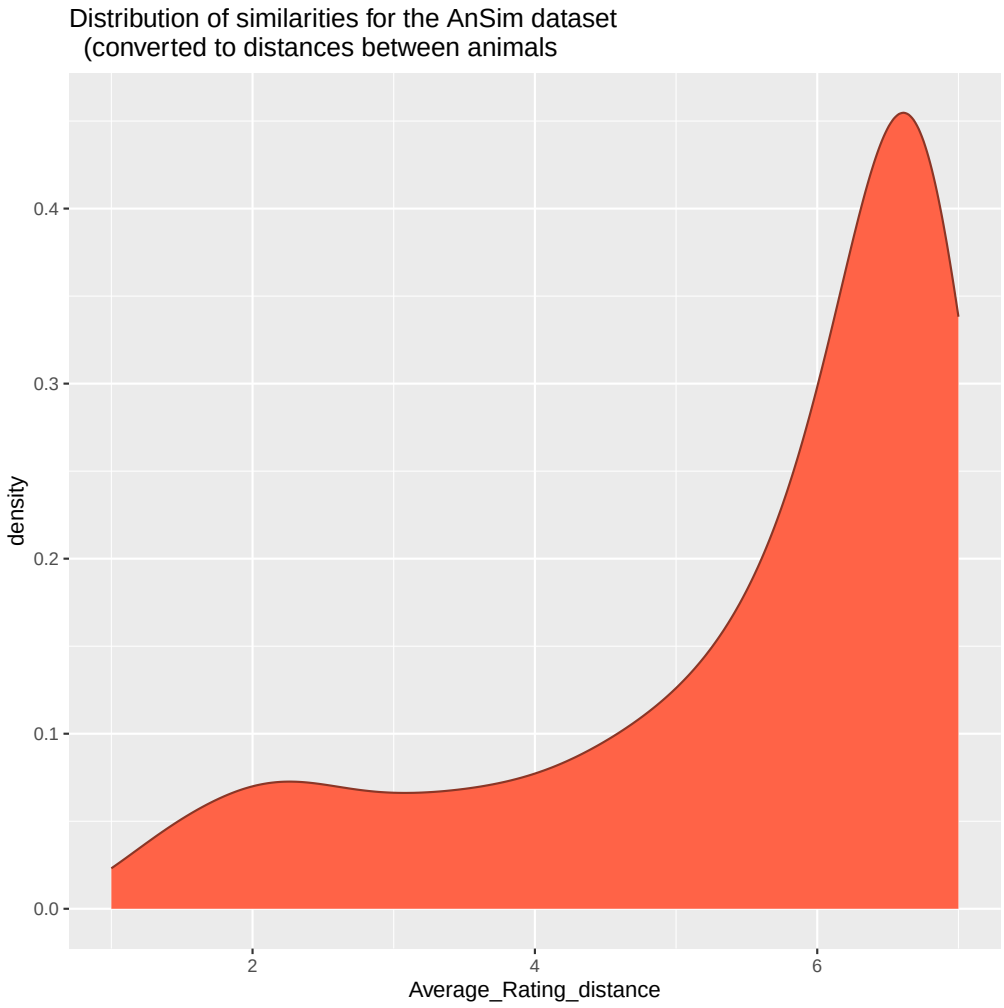
To verify this skewness result I'll show that for the distances between cities data we examined earlier, the data are skewed in one direction, whereas for the animal similarity

data the distances are skewed in the other direction. Similarity data need to be converted to the distance scale to apply this skewness test; recall that for distance data, small numbers reflect "close" but for similarity data, large numbers reflect "close". So, I convert the 1-7 similarity scale to a 1-7 distance scale by subtracting the rating from 8 (i.e., a similarity rating of 1 becomes 8 - 1 = 7 and a similarity rating of 7 becomes 8 - 7 = 1). I cannot find the Henley animal data used earlier in these lecture notes, but I found a larger data set of animal similarity ratings out of Boston University called the AnSim data having 3785 pairs of animal similarity ratings, so I'll use those data to illustrate the skewness pattern.

```
#R code to read data and produce density plots
#use adjust = 1.5 to smoooth out the distribution
library(ggplot2)
d <- as.matrix(read.table(
  "city.dat"))
d.data <- d[lower.tri(d)]
skew.data.demo <- data.frame(city=d.data)
ggplot(skew.data.demo, aes(x=city)) +
  geom_density(color="darkblue", fill="lightblue", adjust=1.5) +
  ggtitle("Distribution of distances between cities data set")
```




```
#convert similarities to distances
animal.sim.data <- read.csv("animal-similarity.csv")
animal.sim.data$Average_Rating_distance <- 8 -
  animal.sim.data$Average_Rating
ggplot(animal.sim.data, aes(x=Average_Rating_distance)) +
  geom_density(color="tomato4", fill="tomato1", adjust=1.5) +
  ggtitle("Distribution of similarities for the AnSim dataset
  (converted to distances between animals")
```



Another difference between tree and spatial structures is "centrality", or the concept of nearest neighbor. High centrality seems to be an indicator of a tree, or a spatial solution that requires a high dimensional space. If there is time, I'll discuss this concept in lecture.

Sometimes it isn't easy to represent data either by a tree or by a dimensional space—like mother/father/son/daughter. Though additive trees tend to do better than hierarchical trees in some of those settings.

Eric Holman (1972, *Journal of Mathematical Psychology*) discusses a set of techniques/axioms for distinguishing dimensional representations from tree structures that satisfy the ultra-metric axiom (i.e., hierarchical trees).

Sattath & Tversky suggest that as a heuristic it is okay to compare fit metrics such as

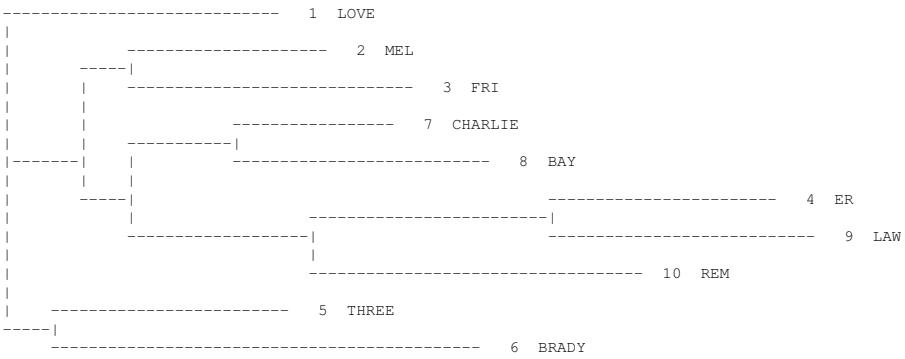
stress between an additive tree structure and an MDS solution. But one needs to be careful because each of these models differs in the number of parameters that are being estimated so using a measure like stress may be misleading. Also, some MDS and TREE programs use different definitions of stress so you need to make sure that across the programs you use, stress is defined equivalently. Corter points out that an ultrametric tree has roughly the same number of parameters as a one dimensional solution, and an additive tree has roughly the same number of parameters as a two-dimensional solution. So it is not fair to compare, say, a 3 or 4 dimensional spatial solution to an additive tree because they differ on the number of parameters. Much like when we compare regression equations, you want to be sure that you are accounting for the number of predictors properly, that is, we need to keep the number of parameters on equal footing when comparing tree and spatial models, or use metrics that consider the number of parameters.

I've probably erred on the side of making the distinction between dimensional space and tree structures too strong. The same developers of nonmetric MDS and INDSCAL also worked on tree extensions (respectively, MAPCLUS and INDCLUS). Much of the development of these models was done by a behavioral science division with AT&T. These extensions are interesting because they allow one to model tree structures within the context of individual differences. These programs are now outdated and little research on these topics has been done since AT&T decided to redefine its behavioral research division. How these type of tree structures differ from the tree structures imposed by ADDTREE and EXTREE still remain to be studied and there is plenty of opportunity to revisit that program of research with modern approaches, algorithms and software.

A paper by Kiers, Vicari, and Vichi (2005, *Psychometrika, vol 70*) proposes a technique that combines in the same simultaneous technique: MDS, clustering and regression using external variables. It computes the spatial configuration and simultaneously computes clusters of points (so, e.g., you can draw circles/ovals around points in the same cluster) and a regression analysis on the MDS dimensions. Matlab code is available from the authors. There is a benefit to computing everything simultaneously compared to doing an MDS, then running regression on the dimensions to aid in the interpretation of factors, then running a clustering algorithm, all as three separate analyses.

One standard heuristic to compare the fit of tree and dimensional solutions is to superimpose the tree clusters on the dimensional solution by drawing circles/ovals around points in the same cluster. If the two solutions coincide, then there won't be any funny shapes to those circles; one sees nice concentric circles/ovals. However, if two points are far apart in the dimensional solution, but the tree solution puts them in the same cluster, then one may need to draw a banana shape to go around other points belonging to different clusters. This suggests the two solutions are not compatible and one of the solutions is likely inappropriate. I'll demonstrate with the TV example given earlier using INDSCAL (Figure 10-7). The same data yielded this additive tree structure presented

in the original output from the ADDTREE program.



Those clusters can be superimposed on the MDS/INDSCAL configuration (see Figure 10-14). If nice concentric ovals appear, then the tree and dimensional solutions align. In this example, they align nicely. However, if two points in the same ADDTREE cluster are far apart in the dimensional solution in such a way that ovals have to bend around other points that are not in the same cluster (like a banana), then that suggests the two solutions do not align very well. But this is just a heuristic and doesn't always identify mismatch. For example, take the MDS dimensional space in Figure 10-1 and apply the ovals implied by the ADDTREE structure in Figure 10-12; you will see that the ovals are mostly well-behaved and they don't have to bend around other points, yet we know this is a dimensional space.

(d) Checking the distance and tree axioms

Axiom testing in this domain is not as easy as it may appear at first. Testing the triangle inequality becomes trivial if we allow constants to be added to the distance data, in that case with a large enough constant any matrix of distance data can be forced to satisfy the triangle inequality. We need a method that is not sensitive to addition of constants. One possibility is to test the condition in the special case when it yields equality (e.g., in the Euclidean case, when point k falls on the straight line path between points i and j, the triangle inequality becomes an equality). In the case of equality, the trick of additive constants won't work. However, testing equality in the presence of noisy data presents new problems. For instance, is the reason that subjects are providing the same distances because they are saying something to the effect of "close enough" (related to the concept of just noticeable differences, or JND). If so, with a more sensitive measuring procedure it may be possible that some of the previously observed "equalities" may turn into inequalities.

Tversky developed a simple ordinal method to test triangle inequality but it requires the ability of the experimenter to create stimuli that have certain characteristics (e.g.,

Figure 10-14: Overlap of INDSCAL dimensional space and ADDTREE clusters: TV shows example

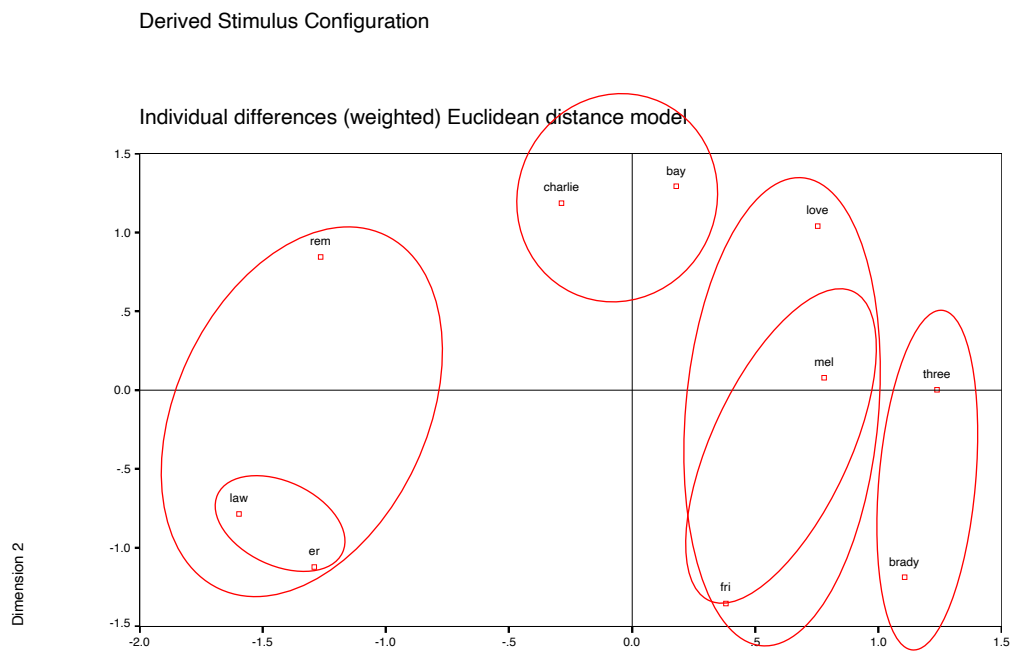
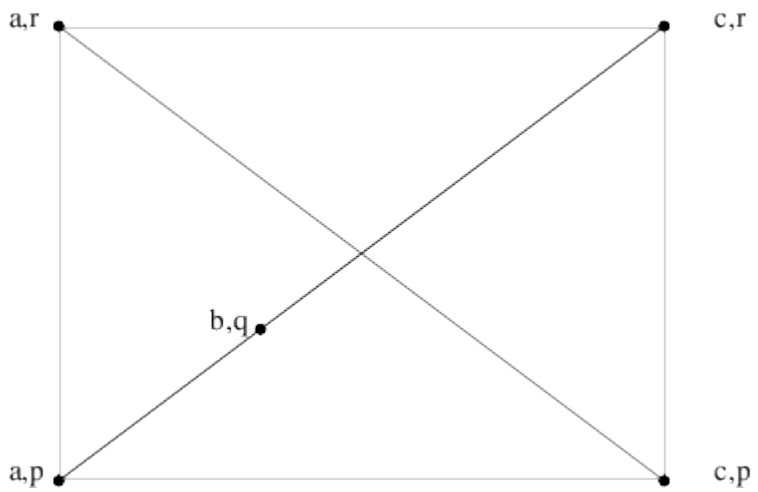


Figure 10-15: Ordinal test of the triangle inequality. The point b,q is defined somewhere along the diagonal.



if the stimuli are faces, then faces need to be morphed to create new faces with certain properties). The condition proposed by Tversky is called the "corner inequality".

Let's assume that our stimuli vary in two dimensions or factors (e.g., rectangles can vary in length and width). A stimuli is denoted as (a,p) meaning it has the value a on the first dimension and the value p on the second dimension. A metric implies that the sum of distances along one dimension should not exceed an analogous distance along both dimensions. For instance, take the four stimuli (a,r), (a,p), (c,r), and (c,p) as four corners of a rectangle. Somewhere along the minor diagonal create the stimulus (b,q), which has a value between a and c on the first dimension and a value between p and r on the second dimension. See Figure 10-15 for an illustration of how the stimuli need to be arranged (most likely, specially created for this test). Now, the metric axioms are violated if both of the following inequalities hold

$$d(ap, cp) \geq d(ap, bq) \quad (10\text{-}11)$$
$$d(cp, cr) \geq d(bq, cr) \quad (10\text{-}12)$$

For more details see Tversky and Gati (1982).

4. K-means Clustering

SPSS and R implement yet another kind of clustering algorithm called K-means that is com-

monly used by behavioral scientists and now part of the data science/machine learning ecosystem. The intuition is sort of the reverse of ANOVA. In ANOVA we have group membership and we compute a decomposition of sum of squares based on between and within sources. K-means clustering tries to find the "optimal" way to group subjects into K groups (where you as the analyst specify K in advance) to yield the greatest possible discrepancy between sum of squares between and sum of squares within. Sometimes though this algorithm gives silly results, like putting a single outlier subject into one group by itself (that has the effect of reducing SSW and increasing SSB). So use this technique with caution and double check output for any strange behavior.

Here's an example taken from Johnson and Wichern. Consider four subjects on two variables.

Subject	Variable 1	Variable 2
1	5	3
2	-1	1
3	1	-2
4	-3	-2

The SPSS syntax and output follow. I asked for K=2 groups. The subcommand save cluster will add a new column to the data set that contains the class codes for each case. These codes can then be used in subsequent analyses such as color coding a scatterplot by class membership.

```
QUICK CLUSTER
  var1 var2
  /CRITERIA= CLUSTER(2) MXITER(10) CONVERGE(.02)
  /METHOD=KMEANS(NOUPDATE)
  /SAVE CLUSTER
  /PRINT INITIAL ANOVA CLUSTER DISTAN.

Case listing of Cluster membership.
 Case ID   Cluster       Distance
       1         1           .000
       2         2          2.000
       3         2          2.236
       4         2          2.236


 Final Cluster Centers.
  Cluster       VAR1            VAR2
        1         5.0000         3.0000
        2        -1.0000        -1.0000


ANOVA TABLE
```

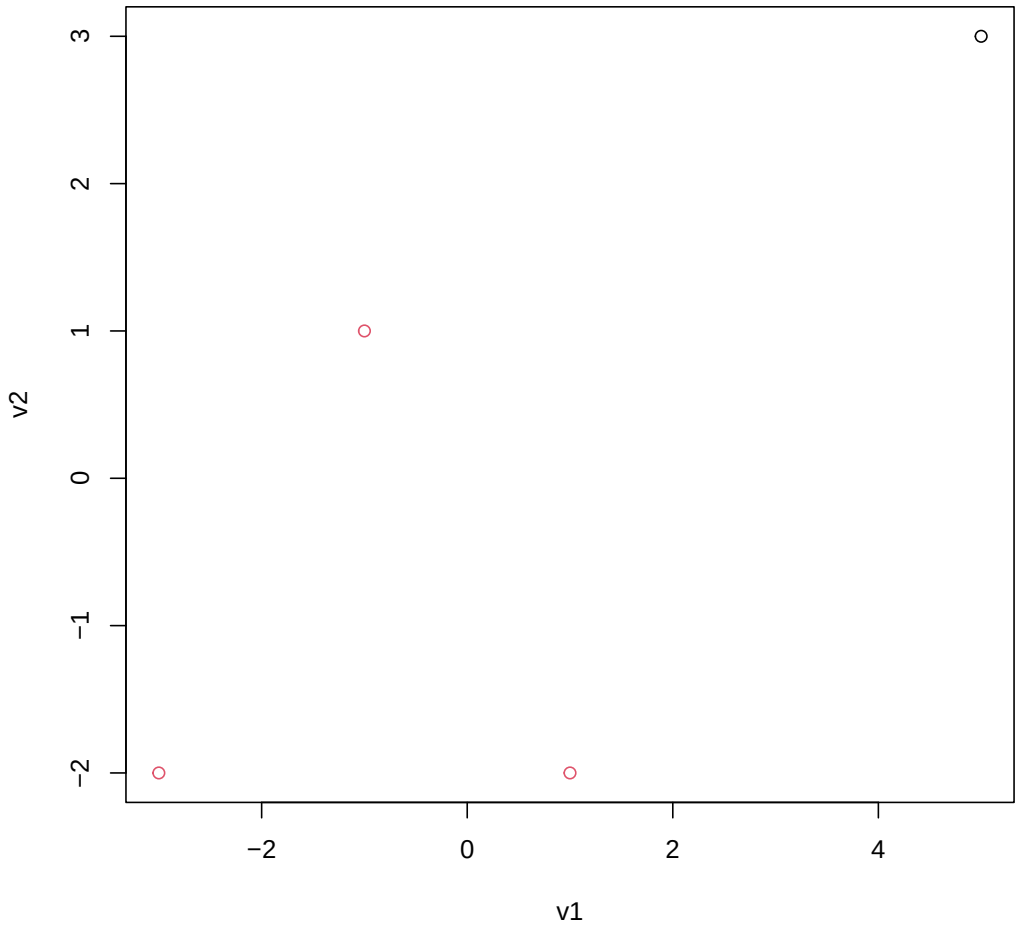
```
Variable    Cluster MS   DF   Error MS    DF         F     Prob
VAR1          27.0000    1    4.000       2.0     6.7500   .122
VAR2          12.0000    1    3.000       2.0     4.0000   .184


 Number of Cases in each Cluster.
  Cluster      unweighted cases    weighted cases
        1                   1.0               1.0
        2                   3.0               3.0
```

The algorithm fit two clusters—a cluster with only the first subject as a member and a second cluster with the remaining three subjects. The cluster centers are merely the group means on each variable. When using this algorithm, one plays around with different K's until an interpretable number of groups emerges. SPSS doesn't provide enough information for one to create measures that would permit a "scree plot" to help select an appropriate value of K. The resulting cluster definition is based on all variables together. The resulting "grouping code" that emerges can be compared with other variables in the data (e.g., a known grouping code, another variable that wasn't used in the K-means clustering, etc).

The analogous command in R can be done with the built in kmeans command. Here I reproduce the SPSS example.

```
data <- cbind(subject = 1:4, v1 = c(5, -1, 1, -3), v2 = c(3, 1, -2,
    -2))
out.kmeans <- kmeans(data, 2)
out.kmeans



## K-means clustering with 2 clusters of sizes 1, 3
##
## Cluster means:
##   subject v1 v2
## 1       1  5  3
## 2       3 -1 -1
##
## Clustering vector:
## [1] 1 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]  0 16
##  (between_SS / total_SS =  72.4 %)
##
## Available components:
##
```

```
## [1] "cluster"      "centers"      "totss"
## [4] "withinss"     "tot.withinss" "betweenss"
## [7] "size"         "iter"         "ifault"


# plot one cluster in black, one in red
plot(data[, 2:3], col = out.kmeans$cluster)
```



An example using kmeans clustering is to find properties of flowers. The iris data set involves 50 examples of three iris species (so 150 flowers in all) where the sepal width and length as well as pedal width and length (all in cm) are recorded. I'll ask kmeans to solve for 3 classes, so it identifies the clusters with the maximum separation in means across the four variables.

You can see the algorithm correctly identified all setosa flowers, but made a few mistakes for the other two. For the versicolor it correctly put 48 in one class but for the virginica it correctly put 36 into one class, the rest were errors of confusing versicolor and virginica.

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length
## 1          5.1         3.5          1.4
## 2          4.9         3.0          1.4
## 3          4.7         3.2          1.3
## 4          4.6         3.1          1.5
## 5          5.0         3.6          1.4
## 6          5.4         3.9          1.7
##   Petal.Width Species
## 1         0.2  setosa
## 2         0.2  setosa
## 3         0.2  setosa
## 4         0.2  setosa
## 5         0.2  setosa
## 6         0.4  setosa
```

```
out.kmeans <- kmeans(iris[, 1:4], 3)
out.kmeans
```

```
## K-means clustering with 3 clusters of sizes 62, 50, 38
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length
## 1     5.901613    2.748387     4.393548
## 2     5.006000    3.428000     1.462000
## 3     6.850000    3.073684     5.742105
##   Petal.Width
## 1    1.433871
## 2    0.246000
## 3    2.071053
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [23] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [45] 2 2 2 2 2 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [67] 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1
```

```
##  [89] 1 1 1 1 1 1 1 1 1 1 1 3 1 3 3 3 3 1 3 3 3
## [111] 3 3 3 1 1 3 3 3 3 1 3 1 3 1 3 3 1 1 3 3 3 3
## [133] 3 1 3 3 3 3 1 3 3 3 1 3 3 3 1 3 3 1
##
## Within cluster sum of squares by cluster:
## [1] 39.82097 15.15100 23.87947
##  (between_SS / total_SS =  88.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"
## [4] "withinss"     "tot.withinss" "betweenss"
## [7] "size"         "iter"         "ifault"
```

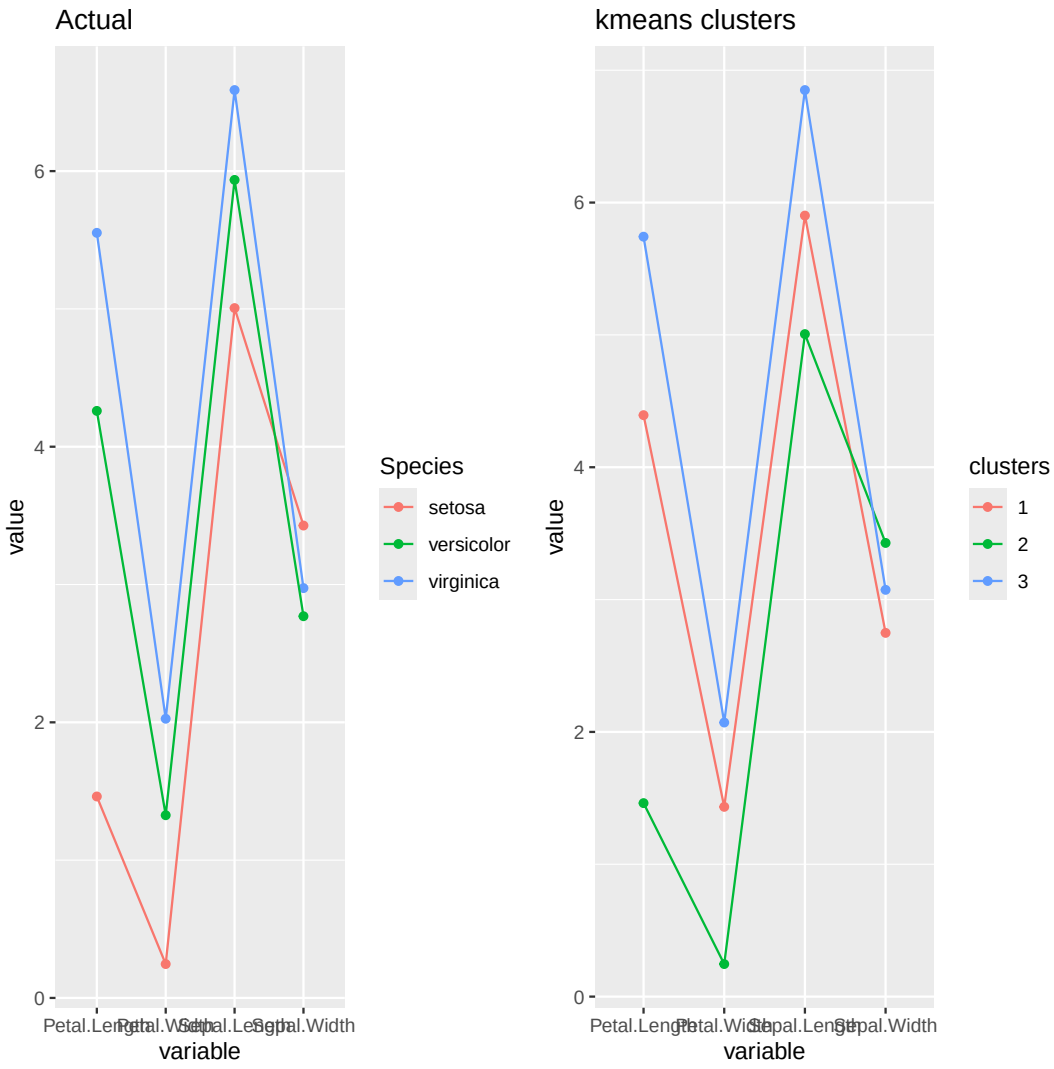
```
iris$clusters <- factor(out.kmeans$cluster)

table(iris$clusters, iris$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         48        14
##   2     50          0         0
##   3      0          2        36
```

We can plot the means of the actual three flowers and compare to the means of the classes that kmeans "found". The assignment of the labels class 1, class 2, etc., is arbitrary because kmeans doesn't know how to assign the label "1" to setosa. So, one would need to reorder so that the colors coincide across the two plots. Not a bad result overall, actually impressive because kmeans did not know the three species but found the four means across three classes of iris flowers. It is common to draw line segments in these plots to help the eye read horizontally for the patterns across the variables for the classes.

```
library(tidyverse)
library(gridExtra)
actual <- iris %>%
  gather(variable, value, Sepal.Length:Petal.Width) %>%
  ggplot(aes(variable,value,color=Species,group=Species)) +
  stat_summary(fun="mean",geom="point") +
  stat_summary(fun="mean",geom="line") + ggtitle("Actual")

kmeans.clusters <- iris %>%
```

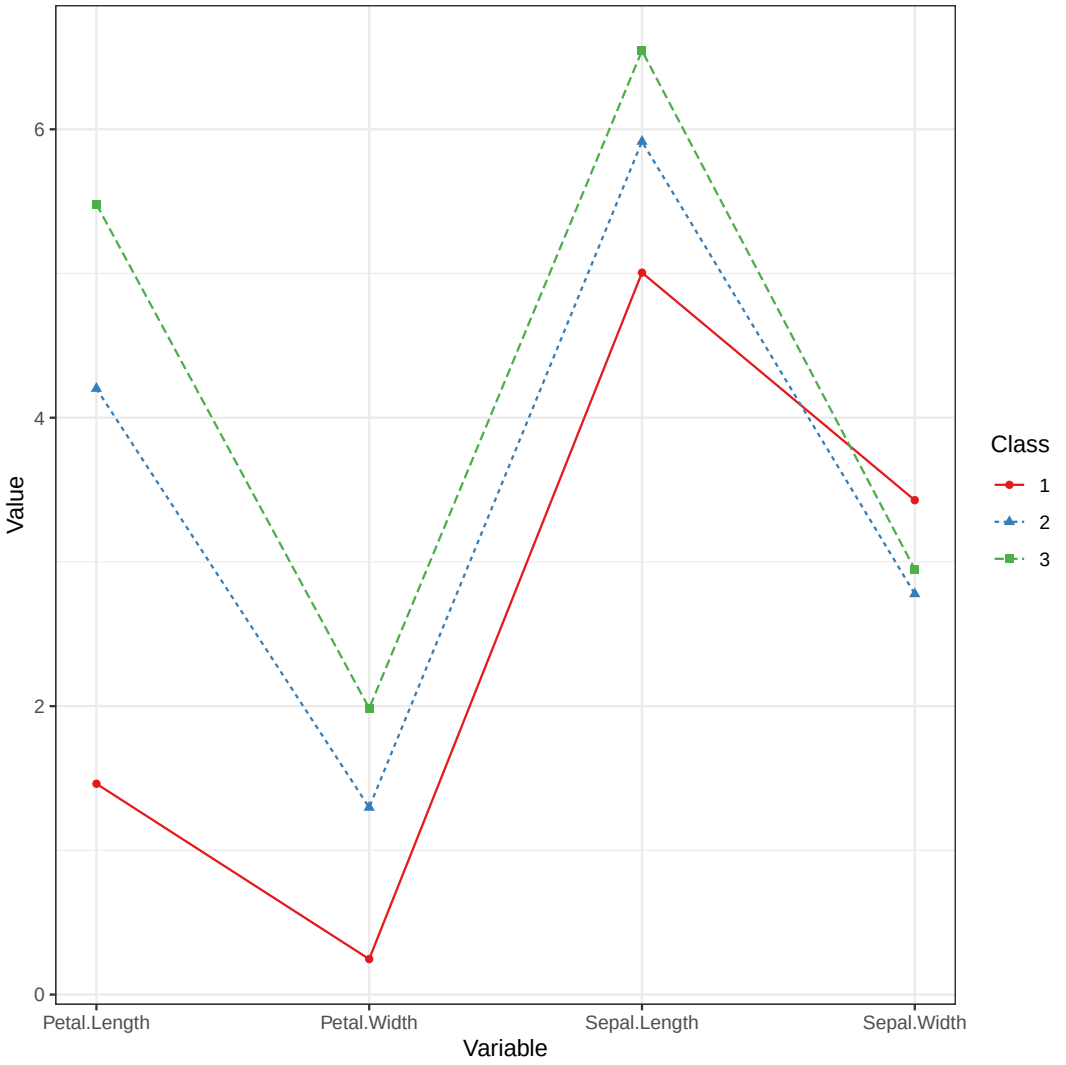
```
  gather(variable, value, Sepal.Length:Petal.Width) %>%
  ggplot(aes(variable,value,color=clusters,group=clusters)) +
  stat_summary(fun="mean",geom="point") +
  stat_summary(fun="mean",geom="line") + ggtitle("kmeans clusters")

grid.arrange(actual,kmeans.clusters,ncol=2)
```



An extension of k-means clustering is Latent Profile Analysis (LPA). LPA adds an error structure, including variances and covariances across the variables. The model can impose constraints such as forcing variances or covariances to be equal across clusters. Here I show a less restrictive version that allows variances and covariances across the four variables to differ across the three classes (called "model 6" in this nomenclature). These results are sim-

ilar to kmeans, but LPA is on stronger statistical footing given the ability to model variances and covariances, essentially adding a "plus epsilon" to what kmeans is doing. LPA also assigns a probability of class membership to each case (something kmeans cannot do because it does not have an underlying statistical model). We will see in LN11 that factor analysis analogously adds a "plus epsilon" to PCA.

```
library("tidyLPA")
output.lpa <- iris %>%
  select(Petal.Length, Petal.Width, Sepal.Length,
         Sepal.Width) %>%
  estimate_profiles(3, variances="varying",
                    covariances="varying")
output.lpa



## tidyLPA analysis using mclust:
##
##  Model Classes AIC    BIC    Entropy prob_min
##  6     3       448.37 580.84 0.97    0.98
##  prob_max n_min n_max BLRT_p
##  1.00     0.30  0.37  0.01



output.lpa %>%
  plot_profiles(rawdata=F,ci=NULL, sd=F,add_line=T)
```




```
#compare actual flowers to LPA clusters
table(output.lpa[[1]][[4]]$Class,iris$Species)
```

```
##
##      setosa versicolor virginica
##    1     50          0         0
##    2      0         45         0
##    3      0          5        50
```

```
#to see probabilities of class membership
#check out full output matrix output.lpa[[1]][[4]]
```

```
#see LPA and mclust in R for more info
```

Examining the table of LPA assignments and actual iris flower species, we see that LPA made fewer errors. Not bad given that neither kmeans nor LPA are classification approaches and neither had information about the actual iris flower species.

For both kmeans and LPA one should run multiple models asking for different number of classes and then plotting the analogous scree plot to identify the ideal number of classes. The tidyLPA and mclust approaches in R provide the ability to compare multiple models, including models that vary not only in the number of classes but also the types of constraints imposed across classes, such as equal or varying covariances.

5. Correspondence Analysis

Just a tickler because we won't have time to cover this technique in class. . . .

Correspondence analysis is an MDS-type technique that is used for contingency tables. An algorithm is available through an SPSS add-on or in the R package FactoMineR, which we will cover in LN11 for PCA. The data necessary to compute correspondence analysis is different than for MDS. Rather than collect similarity data directly, one uses a standard contingency table. For example, if for each US state you counted the number of democrats, republicans, independent, and "other", you would have a frequency table of 50 rows (states) and 4 columns (the four categories). The resulting spatial solution places both the rows and the columns in the same space, somewhat in the spirit of INDSCAL that places items and subject ideal points in the same space. This is known as a biplot. So, for example, if you request a two dimensional space you would have states and party affiliation (4 levels as described above) on the same plane (so 54 points in all). The closeness of the states tell you something about the similarity of the states over the four levels of party affiliation. One can interpret distances between row points and distances between column points, but not distances between a row point and a column point (though a projection method is possible as in unfolding, see next lecture notes). Dimensions can be interpreted in a similar manner to MDS (e.g., if you find that the southern states are at one end of a dimension and the northern states are at another end, that may lead to a natural interpretation for that dimension).

It turns out that correspondence analysis is a simple extension of the standard χ^2 test for contingency tables (see Borg and Groenen, 1997).

6. Classification and Regression Trees (CART)

CART

CART models are interesting models. They were developed in the 1980s and build on earlier

developments going back to the 1960s. CART models are now included as part of the standard data science/machine learning ecosystem. SPSS has this implemented CART in an add-in that costs quite a bit of money and until recently didn't use a very solid algorithm. In R, you can download the TREE package (also the rpart and the party packages provide alternative approaches to computing CART models).

The logic of CART models is to perform regression by clustering data rather than the usual linear fits. The benefit of this style of regression is that it helps you find clusters of predictors for the dependent variable in a more "pattern-centered" approach. There is still much theoretical work to do on these techniques, including building appropriate models for statistical inference. CART works on both categorical and continuous data, and the convention is to say classification for categorical dependent variables and regression for continuous dependent variables. Given our earlier discussion in Lecture Notes 9 about how some types of categorical data can be modeled using a regression approach such as logistic regression, this distinction is perhaps less critical than typically expressed, though there are other approaches to handling categorical data that CART can handle that are outside the regression approach.

I'll give an example of CART using the Titanic survival data. This file has the passenger class (such as first class ticket, second class ticket), whether or not the passenger survived, passenger sex, passenger age, number of siblings or spouses aboard and number of parents or children aboard. Of course, many factors determined whether an individual survived, including was the individual placed given a seat on a life boat, how healthy they were, etc.

Suppose you have this regression with the survival column as the dependent variable Y and the following five predictors:

$$Y = \beta_0 + \beta_1 class + \beta_2 sex + \beta_3 age + \beta_4 sibspouse + \beta_5 parchild \quad (10\text{-}13)$$

But you aren't sure whether the relations are linear, whether there are interactions, you are worried about skewness in some of the predictors, you know you should be careful about treating some variables as categorical rather than numerical (such as passenger first, second and third class tickets) and you think the relations between the predictors may be more complicated than what can be expressed in simple formulas. For example, maybe there is a complicated relation in that if age is below a particular value AND the passenger has 1 or 2 siblings on board, then there is a higher chance of survival. These kinds of complex "if-then" relations are related to interactions but are difficult to capture with the usual definition of interactions (multiplication of two or more predictors) because it is not sufficient given that there may also be a cutoff that needs to be estimated for the "if-then" relations. Further, there may be more than one cutoff that breaks the age variable. Actually, the typical regression approach of "multiply by slope and add the products" (and add the intercept—being mindful of the link function since this is a regression on a binary outcome variable) seems somewhat silly by comparison.

CART models provide a more flexible option to implement this type of more general predic-

tion problem. The general form of the regression in this framework is as follows:

$$Y = f(class, sex, age, sibspouse, parchild) \quad (10\text{-}14)$$

The issue is to characterize the unknown function f in some general manner while also handling the predictors in a general manner (treating categorical and numerical predictors appropriately, addressing complicated interactions or higher order terms, etc). A multiple regression is a special case of this more general approach where each predictor is weighted by a slope and those products are summed. More general versions would be, for example, a series of "if-then" statements, such as "if sex=female and age¡12, then predict survival."

CART finds a classification/clustering of the predictors to fit optimally the dependent variable. Figure 10-16 displays an example. The predicted outcome (died or survived) is at the bottom of the tree structure, along with the percentage of the sample in that cluster and the proportion of the sample that survived. So, the branch that represents females have a predicted outcome of "survived" with 73% of the females surviving; whereas, males older than 9.5 years have a predicted outcome of "died" (the output reports 17% survived, or equivalently 83% died).

The Titanic data set has these variables:

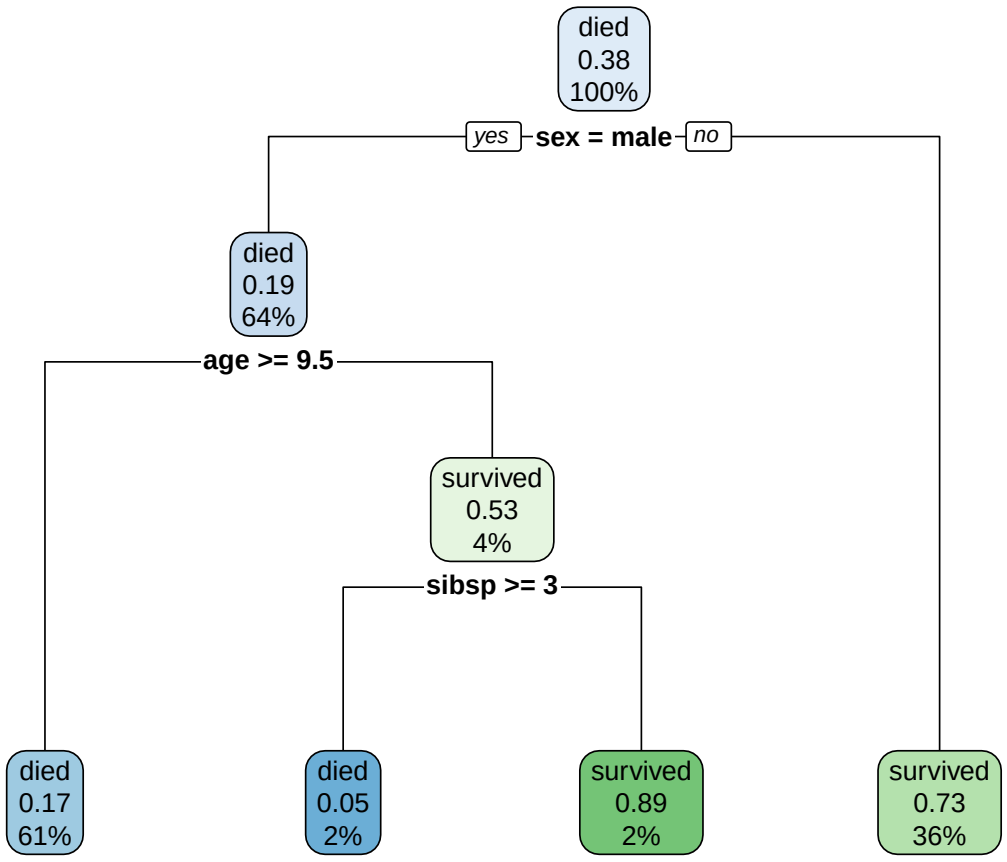
```
pclass passenger class, unordered factor: 1st 2nd 3rd
survived factor: died or survived
sex unordered factor: male female
age age in years, min 0.167 max 80.0
sibsp number of siblings or spouses aboard, integer: 0...8
parch number of parents or children aboard, integer: 0...6
```

You can see by looking at Figure 10-16 that there is a similarity between the tree structures we covered as part of MDS-like procedures and CART. The difference is that with the MDS-like clustering models data are typically in the form of a distance matrix, whereas for CART the data are the more typical columns of variables with one column being treated as a dependent variable and the other columns being treated as predictor variables. This CART example partitioned the data set into four class: females (36% of the sample), males older than 9.5 years (61% of the sample), males younger than 9.5 with 3 or more siblings (2% of the sample) and males younger than 9.5 with fewer than 3 siblings (2% of the sample).

Deciding on the CP value involves using several heuristics. The one I used in these notes is to select the CP corresponding to the lowest cross-validation error called xerror within one

Figure 10-16: Example of CART.

```
library(rpart)
library(rpart.plot)

data(Titanic)
data(ptitanic)
tree <- rpart(survived ~ ., data = ptitanic, cp = 0.02)
# cp=.02 because want small tree for demo
rpart.plot(tree)
```



standard error. To find this, print out the cp table and search for the row with the lowest xerror value.

```
# rerun full tree without setting cp parameter
tree <- rpart(survived ~ ., data = ptitanic)
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = survived ~ ., data = ptitanic)
##
## Variables actually used in tree construction:
## [1] age    parch  pclass sex    sibsp
##
## Root node error: 500/1309 = 0.38197
##
## n= 1309
##
##         CP nsplit rel error xerror     xstd
## 1 0.424000      0     1.000  1.000 0.035158
## 2 0.021000      1     0.576  0.576 0.029976
## 3 0.015000      3     0.534  0.548 0.029438
## 4 0.011333      5     0.504  0.538 0.029238
## 5 0.010000      9     0.458  0.536 0.029198
```

The lowest xerror is .548. Now, take that value and add its standard error (the xstd value in that same row), which yields .548+.0294=.577. This value serves as the threshold: go back to the xerror column and find the xerror closest to this threshold. For these data the value is .576, which corresponds to a CP of .021. I rounded off to .02 to yield a tree with 3 splits (4 nodes). You can read the rpart vignette for more details, especially how the cross-validation error is computed and for other heuristics for selecing the CP value. There are many other approaches to decide how much/little to prune tree

The predictions of this tree (the first one I ran with CP=.02) are binary.

```
table(predict(tree, type = "class"))
```

```
##
##     died survived
##      918      391
```

They can be displayed in the form of classification probabilities by the argument type="class". It is possible to use the tree to make predictions for new data such as what is the predicted outcome of a 25 year old female with a 2nd class passenger ticket traveling with her partner:

```
newdata <- data.frame(pclass="2nd", sex="female",
                      age=25, sibsp=1, parch=0)
predict(tree, newdata, type=c("class"))
```

```
##        1
## survived
## Levels: died survived
```

```
predict(tree, newdata, type=c("prob"))
```

```
##    died survived
## 1 0.068    0.932
```

This type of prediction is analogous to the predictions conducted in regression in the sense that one has predictor values, those values are plugged in the model, and a prediction on the dependent variable emerges.

One way to conceptualize CART is by examining the recursive partitioning algorithm. I'll use the kyphosis data in R to illustrate the concept underlying recursive partitioning. Kyphosis is a binary dependent variable (present/absent) and there are two predictors, Age in months and Start (first topmost vertebra operated on). The data set includes 81 children who had corrective spinal surgery.

```
library(rpart)
data(kyphosis)
table(kyphosis$Kyphosis)
```

```
##
##  absent present
##      64      17
```

The next few slides illustrate what a tree algorithm is doing. We want to find whether we can use the variables Start and Age to classify the presence or absence of Kyphosis. We

want to know if we can chop up this Start by Age variable space into smaller regions that are predominantly red or predominantly black points, indicating absence (black, N=64) or presence (red, 17) of Kyphosis. The partitioning operates by creating successive rectangles where in each rectangle there is a predominance of one color (or all one color in the case of ideal prediction). The errors can be coded in terms of metrics such as correctly identifying disease present, false positive, false negative or correctly identifying disease absent.

Figure 10-17: Examine scatterplot

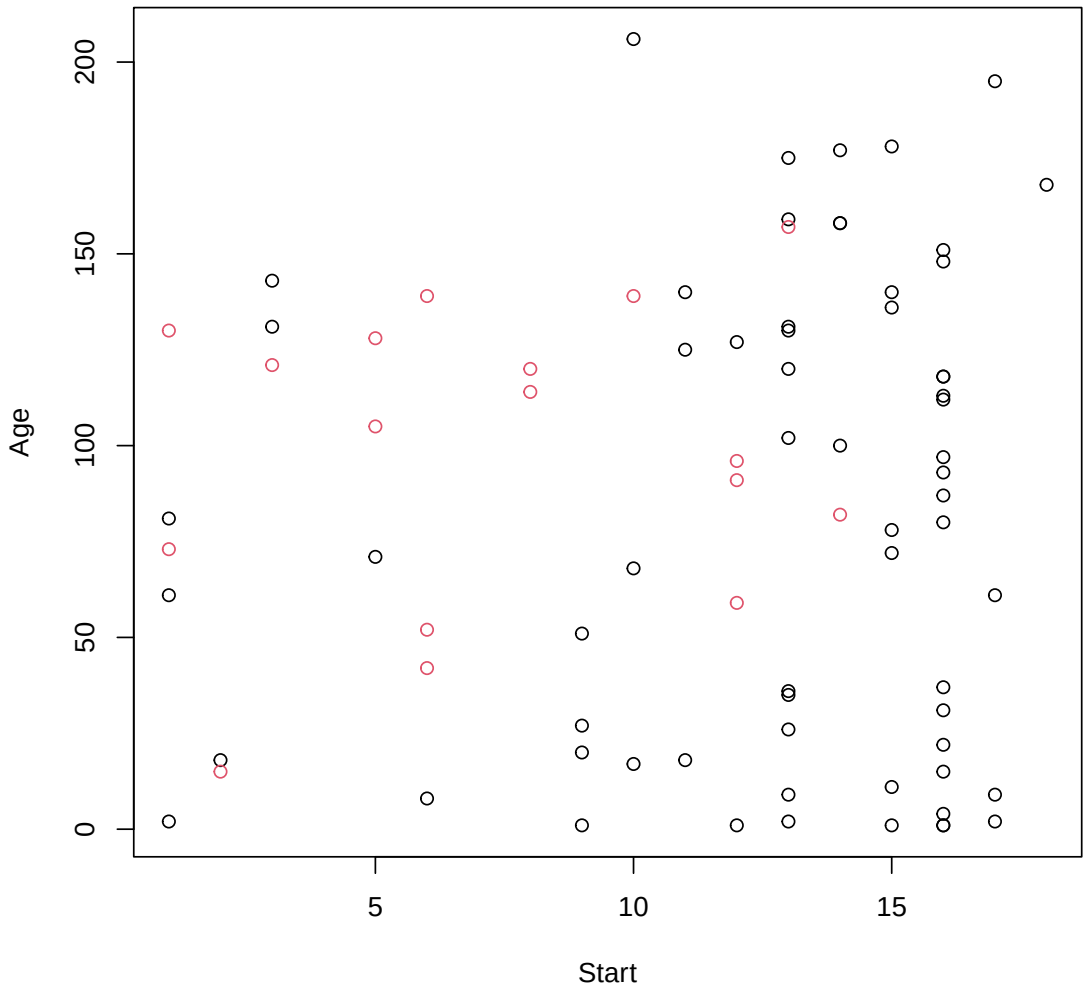


Figure 10-18: First cut to optimally separate presence/absence.

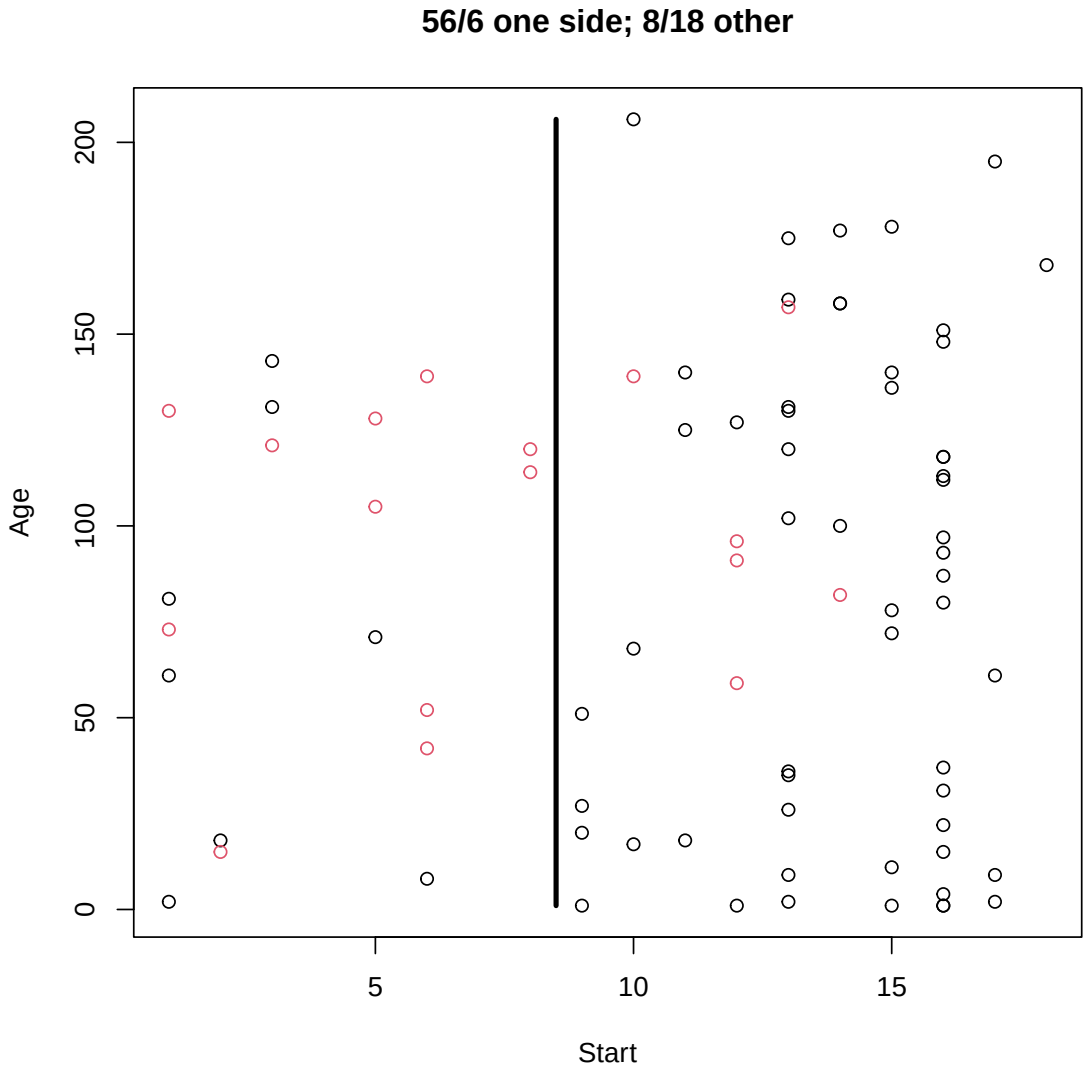


Figure 10-19: Second cut.

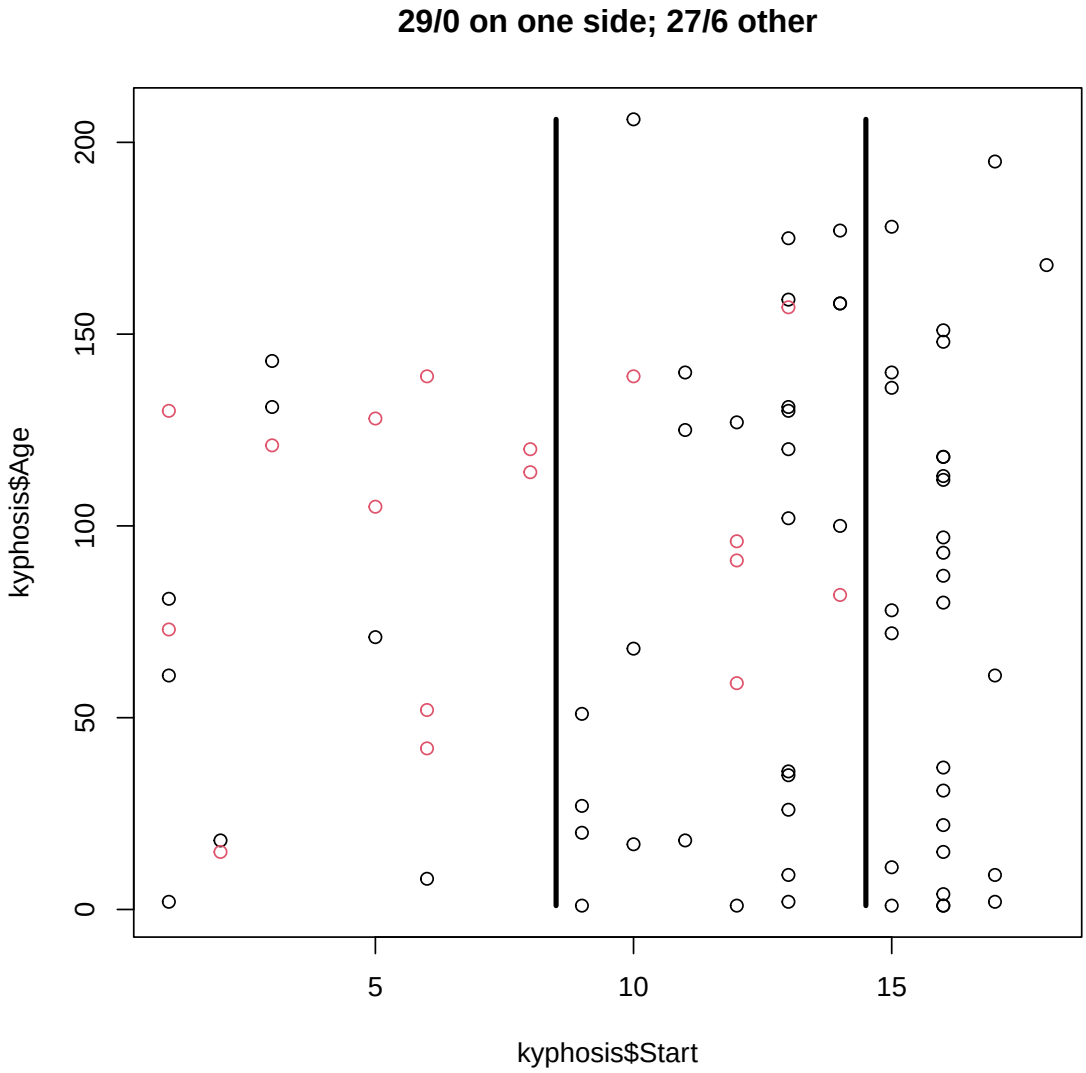


Figure 10-20: Third cut.

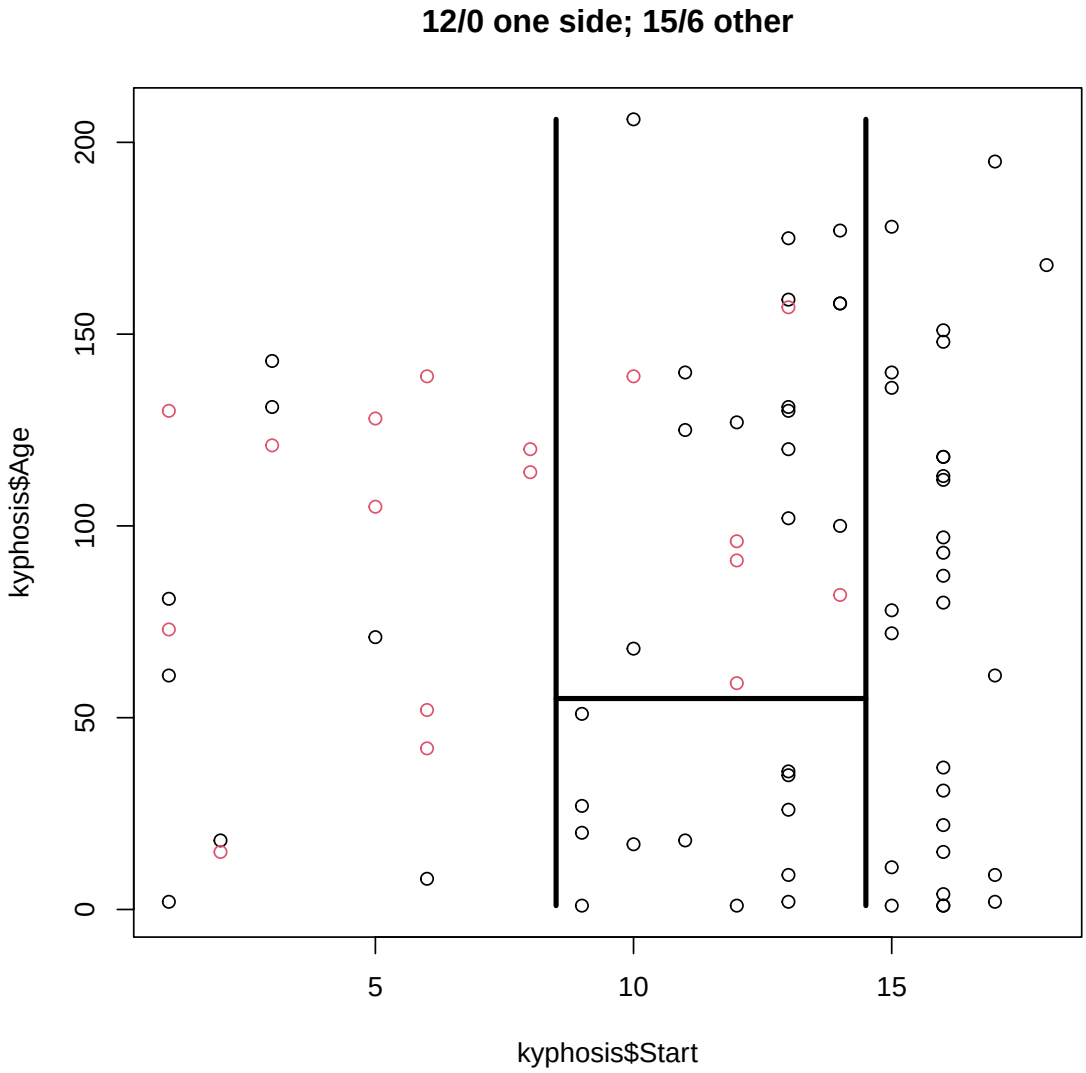
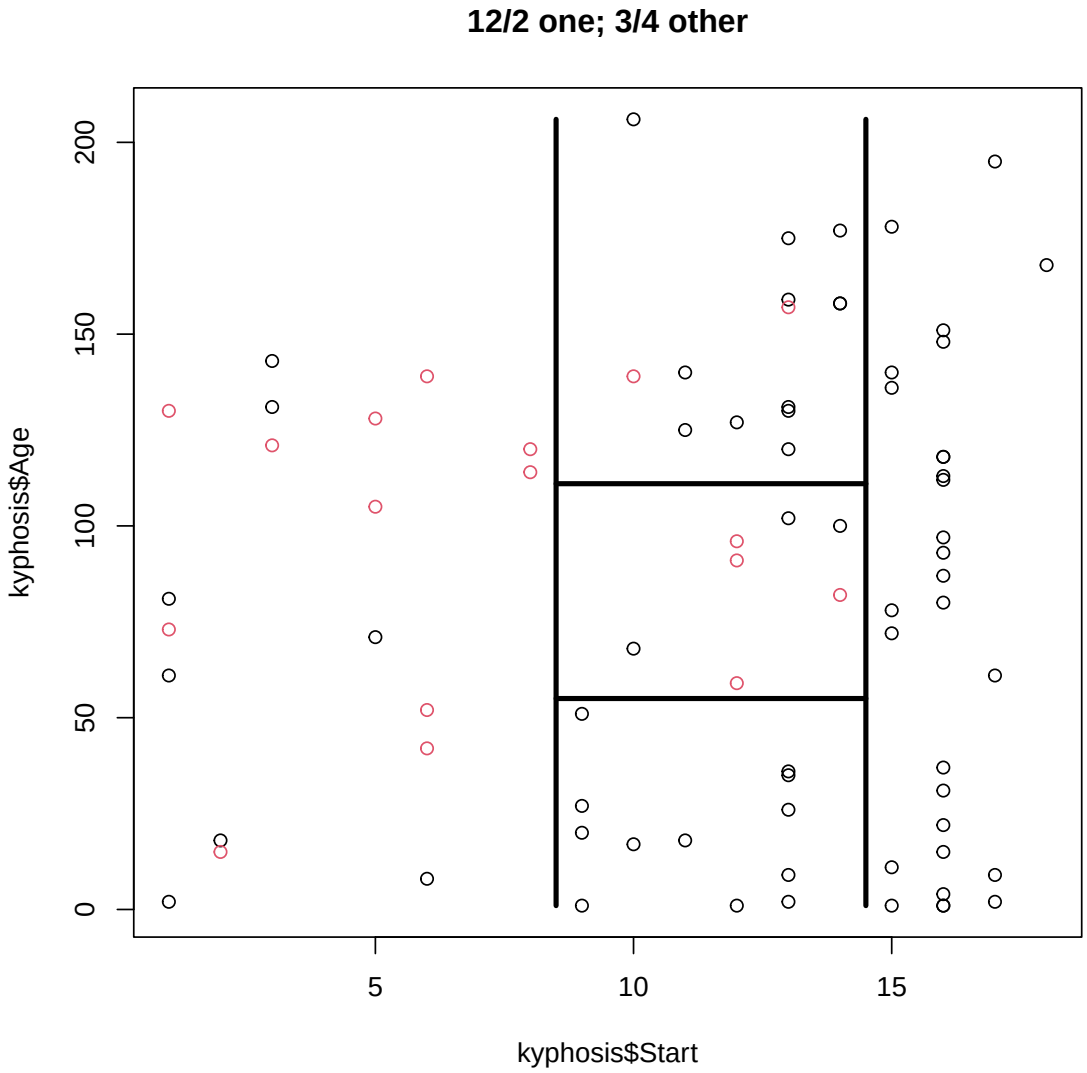
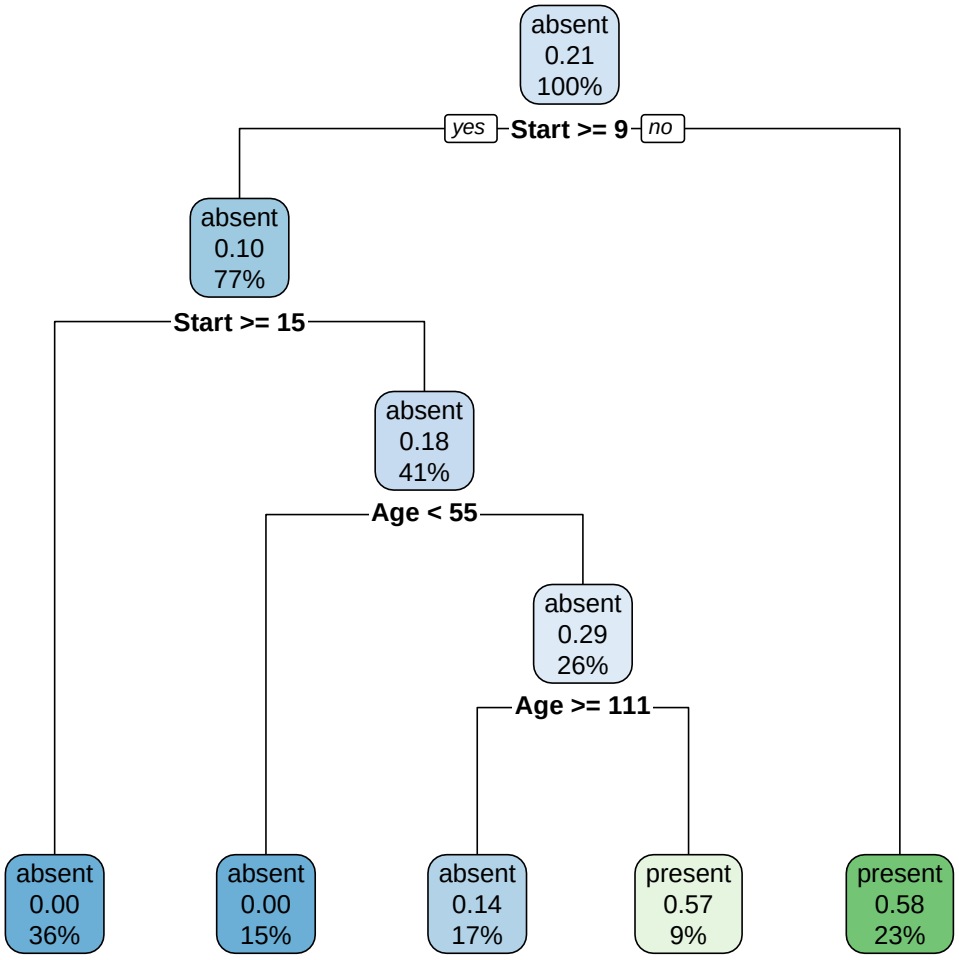


Figure 10-21: Fourth cut.



This series of plots shows that the algorithm makes cuts in a space of variables to optimize occurrence of a class within each rectangle. That is, define rectangles such as within a rectangle, there is maximal separation between classes (such as in the binary case rectangles that exhibit predominantly one class or the other). This series of cuts can be depicted as a tree:

```
kyphosis.tree <- rpart(Kyphosis ~ Age + Start, data = kyphosis)
rpart.plot(kyphosis.tree)
```



The five rectangles in my "series of cuts" illustration are depicted as five nodes in a tree.

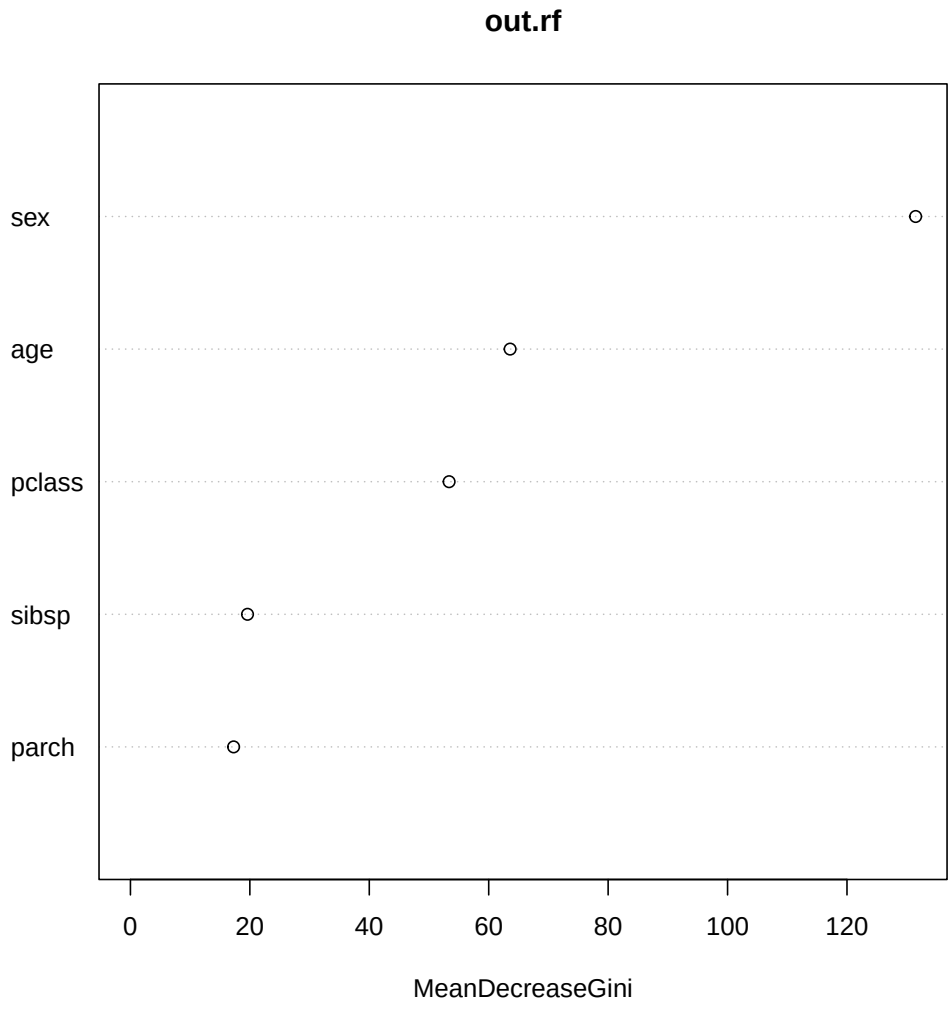
Random Forest

There are variants of CART, such as Random Forest (RF) models developed by Leo Breiman, the same person who developed CART in the 1980s and 1990s. RF creates many subsets of data using subsets of variables and subsets of subjects (in a manner analogous to the bootstrap), and computes a CART on each such subset. From all of those CART fits, one can do many things. A common computation is to identify "important" variables using the aptly named "importance criterion." Here, the algorithm searches for key predictors that appear across many trees and identifies them as important predictors; the predictors are then ranked by the importance criterion. Another common use of RF models is what is called an ensemble model, where one essentially performs a vote count by examining across all the trees that were estimated, the most common predicted values. In this way, one can enter a set of predictors values (much like one would enter a set of predictor scores in a regression equation

to compute $\hat{y}$), and the approach will apply those predictor values through all the trees and perform a vote count to identify the most likely predicted value.

I illustrate these two uses of random forest with the titanic data set. I'll run 2000 trees.

```
library(randomForest)
#this version of RF doesn't like to see missing data
out.rf <- randomForest(survived ~ ., data=na.omit(ptitanic),
                       ntree=2000)

#variable importance plot
varImpPlot(out.rf)
```



The variable importance plot suggests the predictor that had the most predictive power across the trees was sex and the second key predictor was age. The default metric is based on the Gini coefficient and there are several other possible metrics one can use. I can't go into the details of the many measures that are currently available, but it is easy to find references on variable importance metrics. You may want to take a look at Shapley values and the SHAP package in R, along with other more advanced methods for assessing variable importance such as in the treeshap r package that use newer approaches to generating trees such as XGBoost, which stands for extreme gradient boosting. The Shapley value comes close to the part correlation interpretation of a predictor contributing over and above the other predictors, but the Shapley value treats the problem a little differently using the intuition of how much a predictor contributes over and above all subsets of the predictors in the model. There is a new emphasis in machine learning for interpretable models and these kinds of metrics provide one way to assess the importance of predictors in a model that is otherwise difficult to understand (see Molnar's book *Interpretable Machine Learning,* 2022).

The second use of random forest is to make ensemble predictions where one counts the number of trees that make a particular prediction.

```
#prediction using ensemble of trees (a vote count over trees)
newdata <- data.frame(pclass=
             factor("2nd",levels=levels(ptitanic$pclass)),
             sex=factor("female",levels=levels(ptitanic$sex)),
             age=25, sibsp=1, parch=0)
#binary prediction
predict(out.rf, newdata, type="response")
```

```
##        1
## survived
## Levels: died survived
```

```
#probability of that predicted outcome (proportion of trees)
predict(out.rf, newdata, type="prob")
```

```
##     died survived
## 1 0.0555   0.9445
## attr(,"class")
## [1] "matrix" "array"  "votes"
```

```
#and the breakdown of the predictions of the 2000 trees
predict(out.rf, newdata, type="vote",norm.votes=FALSE)
```

```
##   died survived
## 1  111     1889
## attr(,"class")
## [1] "matrix" "array"  "votes"
```

```
#check proportions follow from vote count
died <- predict(out.rf, newdata, type="vote",norm.votes=FALSE)[1]
survived <- predict(out.rf, newdata, type="vote",norm.votes=FALSE)[2]
died
```

```
## [1] 111
```

```
survived
```

```
## [1] 1889
```

```
c(died, survived)/2000
```

```
## [1] 0.0555 0.9445
```

Ensemble models work like "vote counting" mechanisms. They have been successful in weather forecasting problems, such as in hurricane predictions where different models are averaged together to make the prediction. The idea is that if you have a large number of trees, you can use the trees to make predictions and then count the number of trees that make a particular prediction. The prediction that is made by the most trees is the one that is used as the final prediction. This is a powerful approach to prediction. It is also used in the context of neural networks, where the idea is to use a large number of neural networks to make predictions and then count the number of networks that make a particular prediction. The prediction that is made by the most networks is the one that is used as the final prediction. This is called an ensemble model. It is a powerful approach to prediction and is used in many machine learning applications. But these type of ensemble models do not typically lead to interpretable process models. In science we also want to understand a phenomenon and not just predict it. But it is important to know that there are other approaches to prediction that vary on their ability to provide understanding and interpretation. Regression may not make as great out-of-sample predictions as modern machine learning, but they are high on interpretability, can be easily diagnosed and subject to empirical testing.

7. Mixture Models

The clustering and CART model show that it is possible to partition a sample into subsamples based on a statistical model. This idea can be generalized to other types of statistical models such as ANOVA and regression. For example, in the context of a regression model it is possible to partition a sample into subsamples such that each subsample has different β values (i.e., different intercept and slopes). This is a data-driven approach in contrast to, say, the ANOVA approach where groups are already existing or are created by the experimenter. These mixture models are beyond the scope of this course but I mention them here so that you at least know they exist. An example of a mixture model already discussed is latent profile analysis. Other examples of mixture models include latent class analysis and growth mixture models. We will not be able to cover these methods in this course.

Historical note and caveat: Mixture models were originally created to estimate skewed and multimodal histograms. The idea of mixtures involves combinations of components (e.g., a three mode distribution can be modeled as a combination of three single mode distributions). There is currently active research to see if the notion of combination can be extended to how researchers are currently using these mixture models, which are more about subsets of subjects (as I described the technique in this subsection) rather than a combination of components.

Appendix 1: ALSCAL SYNTAX: MDS & INDSCAL

For ALSCAL in SPSS you can use the following (assuming similarity data):

```
ALSCAL variables = varlist
 /shape = symmetric
 /level = ordinal(similar)
 /model = euclid
 /criteria=dimens(2,2)
 /plot
 /print = data
```

For distances just omit the "(similar)" argument. You run this on the median data (i.e., for each cell in the distance or similarity matrix you compute the median for each subject). You should set the number of dimensions to whatever you want.

For INDSCAL just replace the word "euclid" with "indscal". Note that for INDSCAL you need to enter multiple matrices—one for each subject. In the output of INDSCAL you can ignore the standardized subject weights—all that matters is the relative weight a particular subject gives to each dimension. The minimum number of dimensions for INDSCAL is 2, unlike nonmetric MDS which can work with just one dimension.

Be careful with axes in your plots. Some version of SPSS plot the horizontal and vertical axes with different scales, making it difficult to compute distances directly from the plot to compare to the observed distance matrix. I've had to change the axes so that horizontal and vertical axes are on the same scale.

Remember your SPSS syntax. Commands must end in a period. Any line starting with / must be indented. Some versions of SPSS (such as the UNIX version) are more forgiving of syntax issues than other versions (such as the Windows version).

Here are some additional features. I haven't tested that the following works on all the newer versions of SPSS across different platforms.

- All my examples show complete data matrices being entered. ALSCAL only reads the lower triangle of the matrix when the option "symmetric" is given. So, anything can appear in the upper triangle (such as a bunch of zeros). Also, it is possible to enter only the lower triangle of the distance matrix, but that requires that you use the data list fixed option. I've gotten this to work with limited success.
- It is a good idea to add the subcommand /PRINT = DATA to the list of other subcommands given above. This forces ALSCAL to print the data that it is using so you can verify that the program read in the matrices the way you intended it to.

- I prefer that you use /LEVELS = ORDINAL because it invokes nonmetric MDS, which makes your solution invariant to monotonic transformations of your distance data. Other options instead of ORDINAL include INTERVAL, RATIO, and NOMINAL. These other three options do not have the "similar" argument like ORDINAL does. Use INTERVAL if you interpret your proximity data as unique up to a linear transformation, RATIO if you interpret your proximity data as unique up to a scalar transformation, and NOMINAL if your proximity matrix is a category like a binary 0/1 variable. The NOMINAL option is the least understood case so use with caution.

  When the similarity matrix happens to be a correlation matrix and option /LEVELS = INTERVAL is used, then MDS is identical to a principal components analysis. For a detailed discussion of MDS and the general idea of factor analysis see MacCallum (1974, *Psychological Bulletin, 31,* 505-516). We will cover this in Lecture Notes 11.

- The SHAPE=ASYMMETRIC with METHOD=EUCLID option permits asymmetric proximity matrices to be entered. I believe, though haven't verified, that the program simply averages the upper and lower triangles to arrive at a symmetric matrix. This makes sense if the reason you observe asymmetry is due to random error. However, SHAPE=ASYMMETRIC with METHOD=ASCAL or with METHOD=AINDS leads to a different kind of way to deal with asymmetry. See Young (1975, *Journal of Mathematical Psychology, 12,* 416-436) if you are interested in more details about these more general methods. On most versions of SPSS the word MODEL can be substituted for the word METHOD. There are some complicated numerical techniques that decompose an asymmetric distance matrix (see Borg and Groenen, 1997). Rodgers and Thompson (1992, *Applied Psychological Measurement, 16*, 105- review several methods of dealing with asymmetric data; Hutchinson (1989, *Psychometrika, 54*, 25-51) presents a method for dealing with nonsymmetric data using ideas from network theory. A complete discussion of the asymmetric case is beyond the scope of these lecture notes.

- ALSCAL has some ways of dealing with missing data. See the documentation for more detail, in particular the documentation surrounding /CRITERIA=NOULB.

- /CRITERIA=DIMENS(LOW, HIGH) You specify the number of dimensions through the DIMENS subcommand. If you want to fit multiple runs where you try a 2 dimensional solution, a 3 dimensional solution, and a 4 dimensional solution you specify DIMENS(2,4) and SPSS will run ALSCAL three times to produce a 2D, a 3D and a 4D solution. If you just want one solution, such as a 2 dimensional solution, you can either write DIMENS(2,2) or DIMENS(2).

- If you have convergence problems, you can try tinkering with /CRITERIA=CONVERGE(C), /CRITERIA=ITER(N), /CRITERIA=STRESSMIN(S) where c is a very small number (the default is .001), n is a large number (the default is 30), and s is a very small number (the default is .005).

- You may find that it is useful to add the subcommand /OUTFILE = FILENAME to the list of subcommands. This will save relevant output such as the stimulus configuration matrix (and subject weight matrix, if METHOD=INDSCAL) into a file called "filename". It will

require some SPSS tricks to merge the contents of “filename” to existing files you have, or to import the file to some other program such as Excel for further data manipulation or plotting.

ADDTREE PROGRAM & SYNTAX

ADDTREE is performed on a single median distance or similarity matrix (just like ALSCAL MDS above). Currently there is no analog to INDSCAL in the ADDTREE world so you can't get subjects weights (however, there is a now outdated program called INDCLUS which is derived from a very different viewpoint).

For those who want to their own copy of the free DOS/Windows/NT program you can download the ADDTREE program from this link

`http://www.columbia.edu/~jec34/`

There is also a link to this site on the class homepage.

For those of you who download the DOS executable program (Windows only) from the web.... To run the program, save the syntax file into a text only file, put this syntax file in the same directory as the ADDTREE DOS executable, and double click on the ADDTREE program. You will be prompted for an input file (give the name of the syntax file) and an output file (give the name of the file you want the program to write to; this should be a new name so you don't write over existing information). Open the output file with a text editor. I've noticed that the filename of the syntax file cannot be long (so keep it to 6 or so characters).

I have a copy of the ADDTREE source code in pascal and have compiled it successfully on both my mac and linux machines, though that was more than 10 years ago. If you are interested in trying to compile this code, feel free to contact me.

Here is a sample command file. This is what you would put into the file I called "infile". There should not be any blank lines:

```
heights, residuals, modeldistance
GIVE SOME TITLE HERE
8 sim lowerhalf 5 1
  9.0
  2.0  8.0
  8.0  8.0  4.0
 11.0  6.0  4.0  9.0
  7.0  3.0  3.0  3.0  4.0
  9.0  4.0 10.0  7.0 10.0  2.0
  3.0  2.0  8.0  2.0  3.0  8.0  2.0
A
B
C
D
E
```

```
F
G
H
```

The first line just leave as is. The second line is the title of your run. The third line has five commands: the number of stimuli, sim or dis (are data similarities or distances?), lowerhalf indicates that only half the matrix is given (this assumes symmetry), and the last two commands are the format of the data for output (number of spaces each number occupies and the number after the decimal). The manual reports that the last two numbers in line three refer only to output. Then comes the data matrix in the format given on the third line (each cell in the lowerhalf matrix uses 5 spaces with 1 decimal place. Be sure that the decimals line up. Note that only seven lines of data appear (there are 8 stimuli). Finally, the next lines are labels you want to assign to each of the eight stimulus (for printing purposes). You can use strings (like "apple" — with no quotes) rather than single letters.

I won't get into all the program options in this class. I'll mention one though. If you have a complete matrix, observe asymmetry and attribute it to noise, then change the command "lowerhalf" to "fullmatrix" and enter the entire matrix. That will read in a full matrix and automagically "symmetrize" (i.e., typically averaging the two estimates A v. B and B v. A into one). If you have a full matrix that is symmetric you can also use the fullmatrix option. For some references on dealing with asymmetric data see the ALSCAL appendix in these lecture notes on on page 10-77.

The keyword "modeldistance" in the first line will print out the model implied distances. If you have similarity data, the program will transform to distances by multiplying data by -1 and adding an appropriate constant to yield positive numbers. The program will print out the positive constant that was added.

There seems to be a limitation that the data cannot begin with a decimal value. For example, you need to type 0.5 rather than .5. Again, if you have blank lines in the syntax file, the program won't work.

The output of the program is pretty easy to read. The tree structure is clearly printed and the stress/R^2 information appears after the tree. The syntax I suggest also prints out the model implied distance matrix, the residual matrix (difference of model matrix and data matrix), and information about the lengths of the individual nodes. The latter information is useful if you want to label trees with specific information like I did in Figure 10-9, where the lengths for each segment are labelled by length. For this info you take the portion of the output that appears right before the tree. The segments that correspond to the stimuli are printed first (and numbered 1 to number of stimuli), then segments that connect two stimuli are listed (and numbered), then segments that connect segments, etc. Keep in mind that ADDTREE adds a constant to meet the triangle inequality, which you have to factor in if you want to see how the fit to the original distance matrix was calculated.

Stress formula 1 is the standard stress formula that has the sum of the square of the model distances in the denominator. Stress formula 2 has the variance of the model distances in the denominator. R^2 monotonic is simply equal to 1 - (stress formula 2)^2. R^2 p.v.a.f is not easy to explain (see the pascal code of the program for the actual definition).

For those students who use Matlab, there are some additive tree routines (as well as MDS programs) that are available such as those written by Michael Lee

`http://www.psychology.adelaide.edu.au/personalpages/staff/michaellee/homepage/` and those found on L. Hubert's website

`http://cda.psych.uiuc.edu/srpm_mfiles/` (with a companion book *The Structural Representation of Proximity Matrices with MATLAB* by Hubert et al.

R SYNTAX

MDS

I haven't done a thorough investigation of how the different algorithms in R compare to Kruskal's original algorithm and the ALSCAL approach developed by Young, which is the one implemented in SPSS.

Nonmetric MDS in R appears in several functions, such as the isoMDS function (only works on distance matrices not similarity matrices) in the MASS library (a very good library, by the way) and the metaMDS function in the vegan library. See `http://strata.uga.edu/software/pdf/mdsTutorial.pdf` for a tutorial. Lately, I've been using the package smacof because it handles all three key dimensional analyses we cover in this course: MDS and INDSCAL (these lecture notes) and unfolding (LN11) plus many bells and whistles not offered by the other packages. The package smacof use a different algorithm than the ALSCAL approach so there may be some minor differences in the solutions. See Borg and Groenen (2005) for a detailed comparison.

Here is the R syntax needed to reproduce the cities example at the beginning of these lecture notes. Assumes distances are already read into a 12x12 matrix with 0s in the diagonal. I'll use the MASS package.

```
library(MASS)
d <- as.matrix(read.table("city.dat"))
city <- c("Atlanta", "Boston", "Cinci", "Columbus", "Dallas", "India",
    "LittleR", "LA", "Memphis", "StLouis", "Spokane", "Tampa")
dimnames(d) <- list(city, city)
d


##          Atlanta Boston Cinci Columbus Dallas
## Atlanta        0   1068   461      549    805
## Boston      1068      0   867      769   1819
## Cinci        461    867     0      107    943
## Columbus     549    769   107        0   1050
## Dallas       805   1819   943     1050      0
## India        508    941   108      172    882
## LittleR      505   1494   618      725    325
## LA          2197   3052  2186     2245   1403
## Memphis      366   1355   502      586    464
## StLouis      558   1178   338      409    645
## Spokane     2467   2747  2067     2131   1891
## Tampa        467   1379   928      985   1077
##          India LittleR   LA Memphis StLouis
## Atlanta    508     505 2197     366     558
```

```
## Boston     941    1494 3052    1355    1178
## Cinci      108     618 2186     502     338
## Columbus   172     725 2245     586     409
## Dallas     882     325 1403     464     645
## India        0     562 2080     436     234
## LittleR    562       0 1701     137     353
## LA        2080    1701    0    1831    1848
## Memphis    436     137 1831       0     294
## StLouis    234     353 1848     294       0
## Spokane   1959    1988 1227    2042    1820
## Tampa      975     912 2480     779    1016
##          Spokane Tampa
## Atlanta     2467   467
## Boston      2747  1379
## Cinci       2067   928
## Columbus    2131   985
## Dallas      1891  1077
## India       1959   975
## LittleR     1988   912
## LA          1227  2480
## Memphis     2042   779
## StLouis     1820  1016
## Spokane        0  2821
## Tampa       2821     0
```

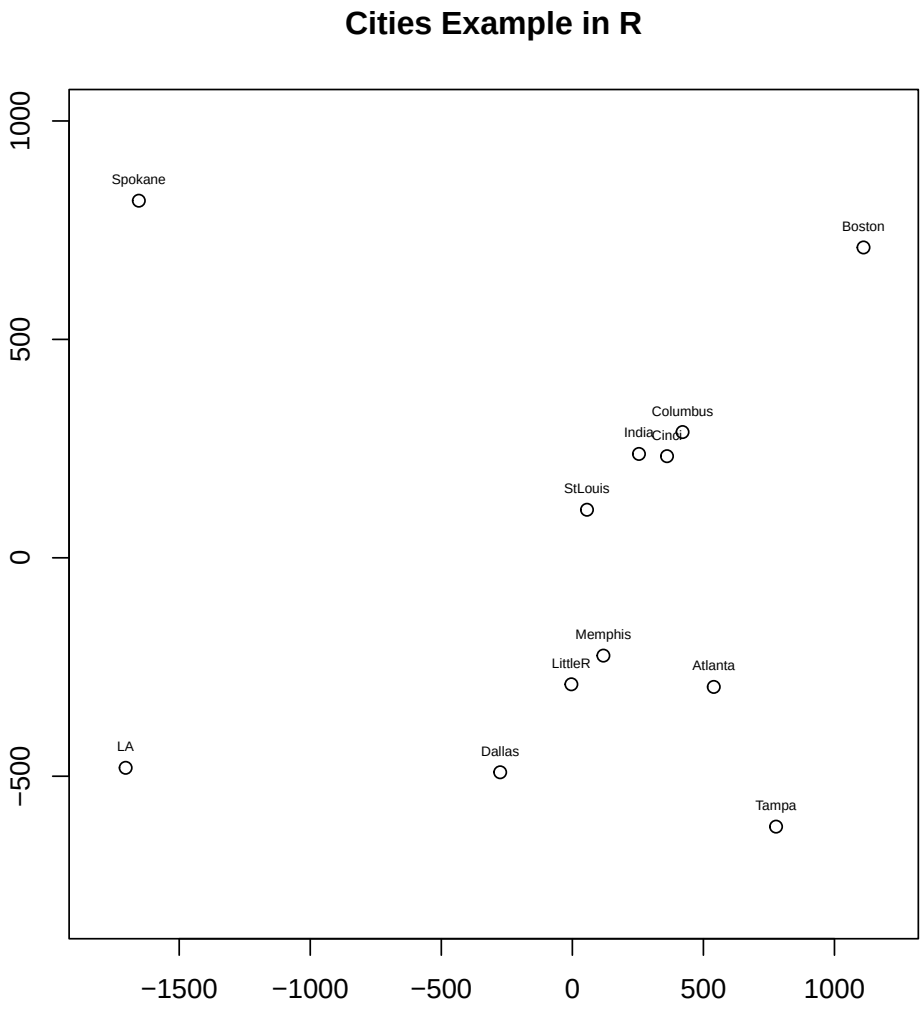
```
# run nonmetric MDS with 2 dimensions
output <- isoMDS(d, k = 2)
```

```
## initial  value 1.313509
## final  value 1.313172
## converged
```

```
output
```

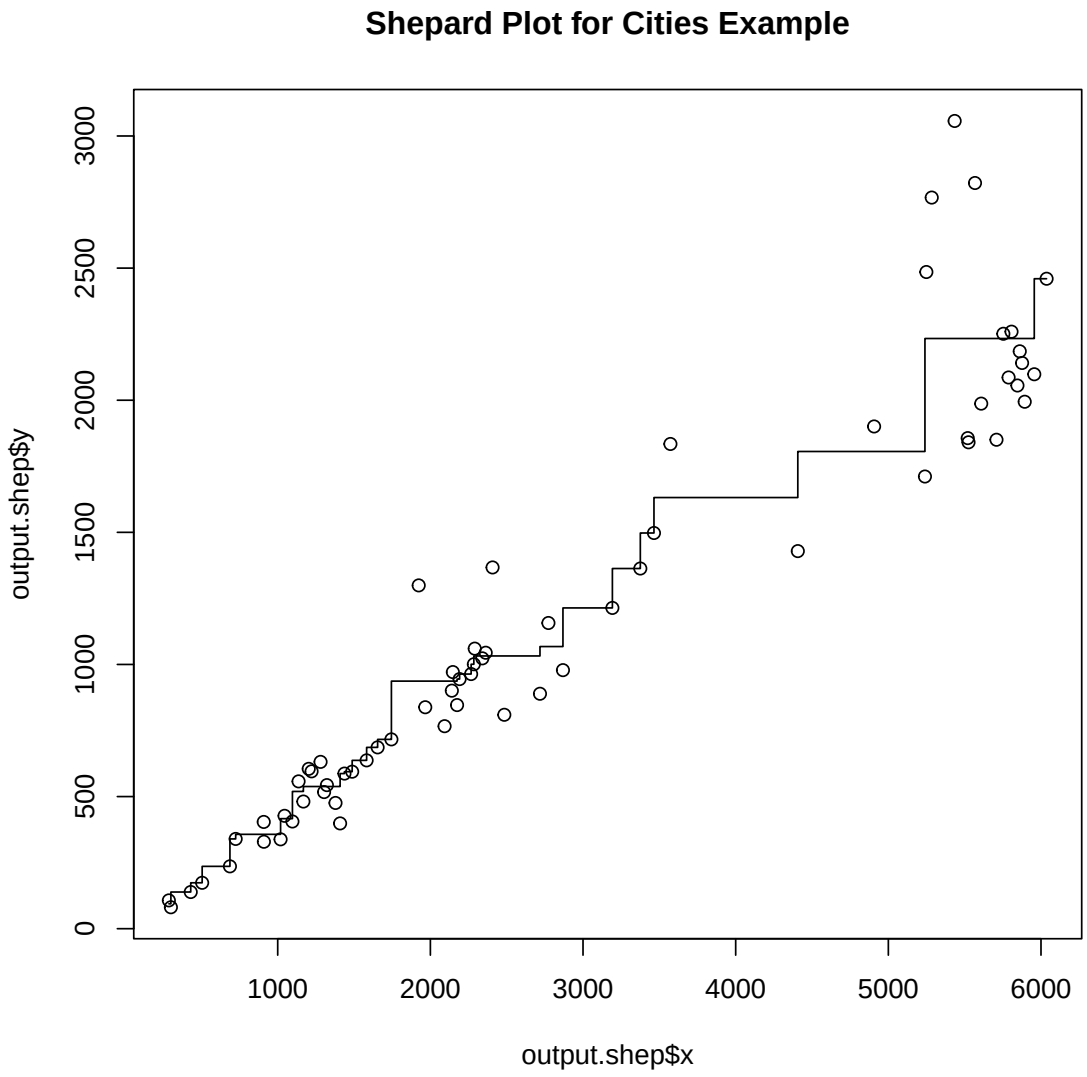
```
## $points
##                  [,1]      [,2]
## Atlanta   -539.548583  295.5791
## Boston   -1110.892907 -710.4100
## Cinci     -361.090474 -232.6195
## Columbus  -420.411900 -287.8384
## Dallas     275.415604  490.9497
## India     -254.116418 -237.7730
```

```
## LittleR      3.784687  289.5554
## LA        1704.312830  480.7391
## Memphis   -118.427332  223.8960
## StLouis    -55.803676 -110.0113
## Spokane   1654.001149 -817.4729
## Tampa     -777.222982  615.4058
##
## $stress
## [1] 1.313172


# set plot to be square
par(pty = "s")

# plot configuration MDS is unique up to rotation and scaling,
# reflection so may need to flip the plot
flip <- -1
plot(flip * output$points, xlim = c(-1800, 1200), ylim = c(-800, 1000),
    xlab = "", ylab = "", main = "Cities Example in R")
text(flip * output$points, labels = city, pos = 3, cex = 0.5)
```



One needs to be careful when creating plots from coordinates that emerge from an MDS analysis. In this cities example, I intentionally made the ranges of the x and y axes different. It makes for a plot that uses the plotting region reasonable well. However, a drawback is that distances in this plot (such as the distance from LA to New York) are difficult to interpret because the horizontal and vertical axes are on different scales. For that reason, it is better to maintain equal ranges in the plot even if it means having some dead space.

```
# plot monotonic regression
output.shep <- Shepard(dist(d), output$points)
plot(output.shep, main = "Shepard Plot for Cities Example")
lines(output.shep$x, output.shep$yf, type = "S", xlab = "", ylab = "")
```



INDSCAL in R

The package smacof can run INDSCAL and unfolding analyses (covered in the next lecture notes). There is also an INDSCAL command in the SensoMineR package.

```
inddata <- read.table(
  "indscal.dat")
names(inddata) <- c("love","mel","fri", "er","three","brady",
                    "charlie","bay","law","rem")
library(smacof)
sub <- factor(rep(1:7, each=10))
```

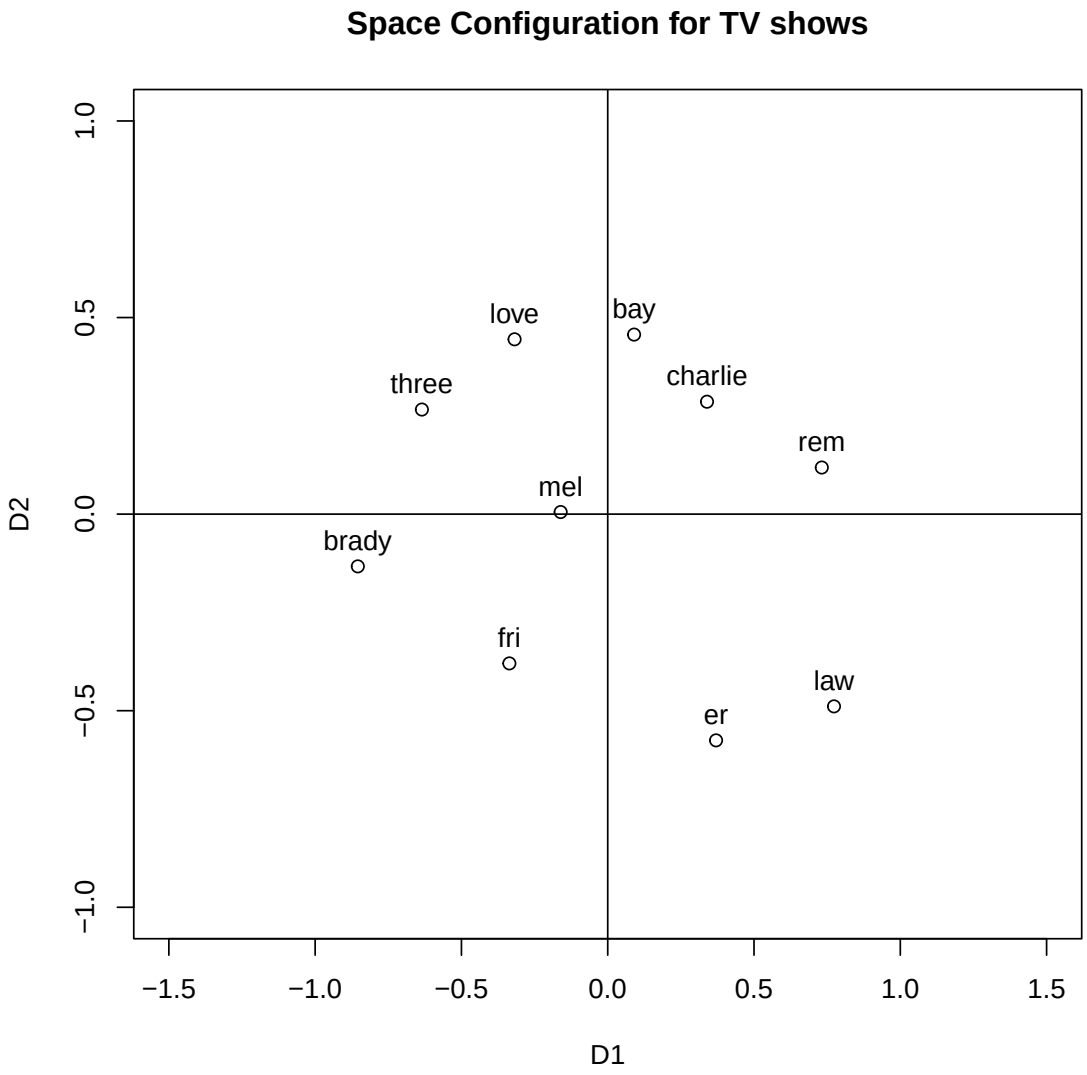
```r
#put data into a list and convert similarity to distances
inddatalist <- list()
for (i in 1:7)
  inddatalist[[i]] <-  (21-inddata[sub==i,])
#smacof now has a sim2diss function that one use with something like
#inddatadisslist <- lapply(inddatasimlist, sim2diss, method=21)
#assuming inddatasimlist is already a list of each subject's
#similarity matrices
#there are other options to sim2diss

output <- indscal(inddatalist,ndim=2, type="ordinal",itmax=2000)
#default is iteration is 1000 but these data hit it so
#increase iter number
output
```

```
##
## Call: smacofIndDiff(delta = delta, ndim = ndim, type = type, constraint = "ir
##     weightmat = weightmat, init = init, ties = ties, verbose = verbose,
##     modulus = modulus, itmax = itmax, eps = eps, spline.degree = spline.degre
##     spline.intKnots = spline.intKnots)
##
## Model: Three-way SMACOF
## Number of objects: 10
## Stress-1 value: 0.177
## Number of iterations: 1201
```

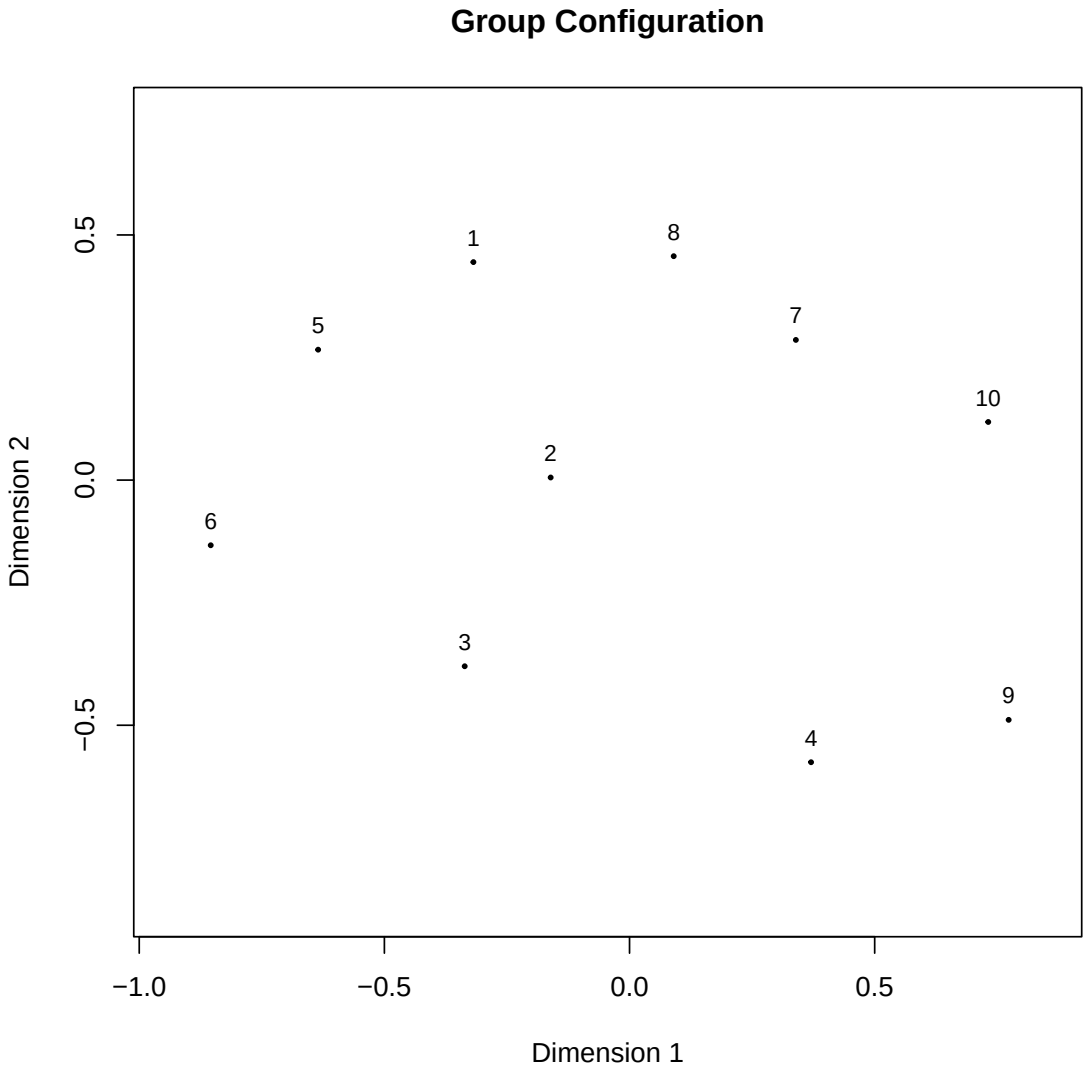
```r
summary(output)
```

```
##
## Group Stimulus Space (Joint Configurations):
##         D1      D2
## 1  -0.3184  0.4446
## 2  -0.1608  0.0053
## 3  -0.3362 -0.3797
## 4   0.3703 -0.5754
## 5  -0.6351  0.2660
## 6  -0.8542 -0.1329
## 7   0.3393  0.2859
## 8   0.0901  0.4567
## 9   0.7733 -0.4890
## 10  0.7317  0.1185
##
```

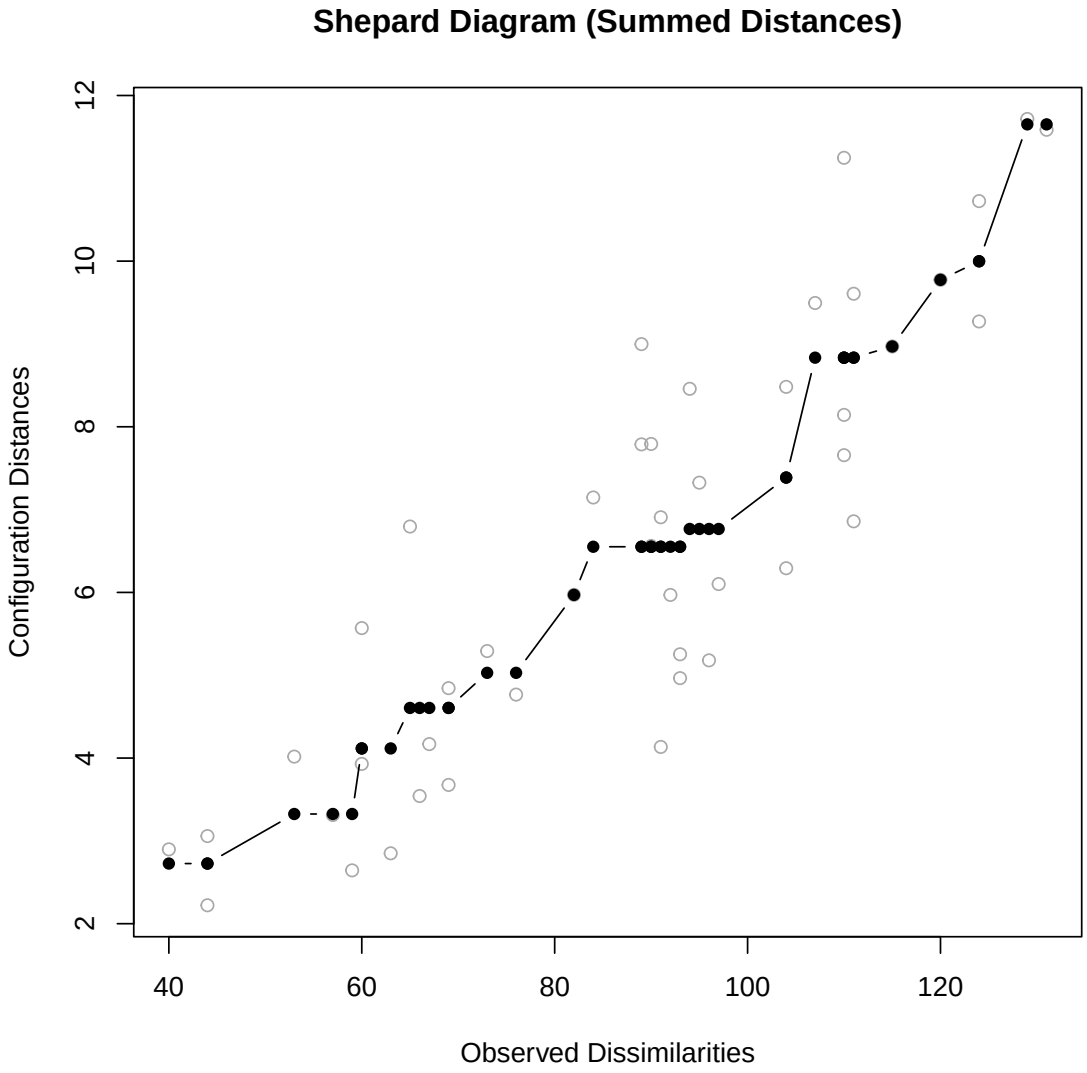
```
##
## Stress per point:
##     1     2     3     4     5     6     7     8
## 10.86  7.53  9.54 11.17 10.35 12.51  9.02 11.45
##     9    10
##  8.23  9.34


k <- output$gspace
dimnames(k) <- list(names(inddata),c("D1","D2"))

# similar to spss
plot(k,xlim=c(-1.5,1.5),ylim=c(-1,1),
     main="Space Configuration for TV shows")
  abline(v=0,h=0)
  text(k,names(inddata),pos=3)
```



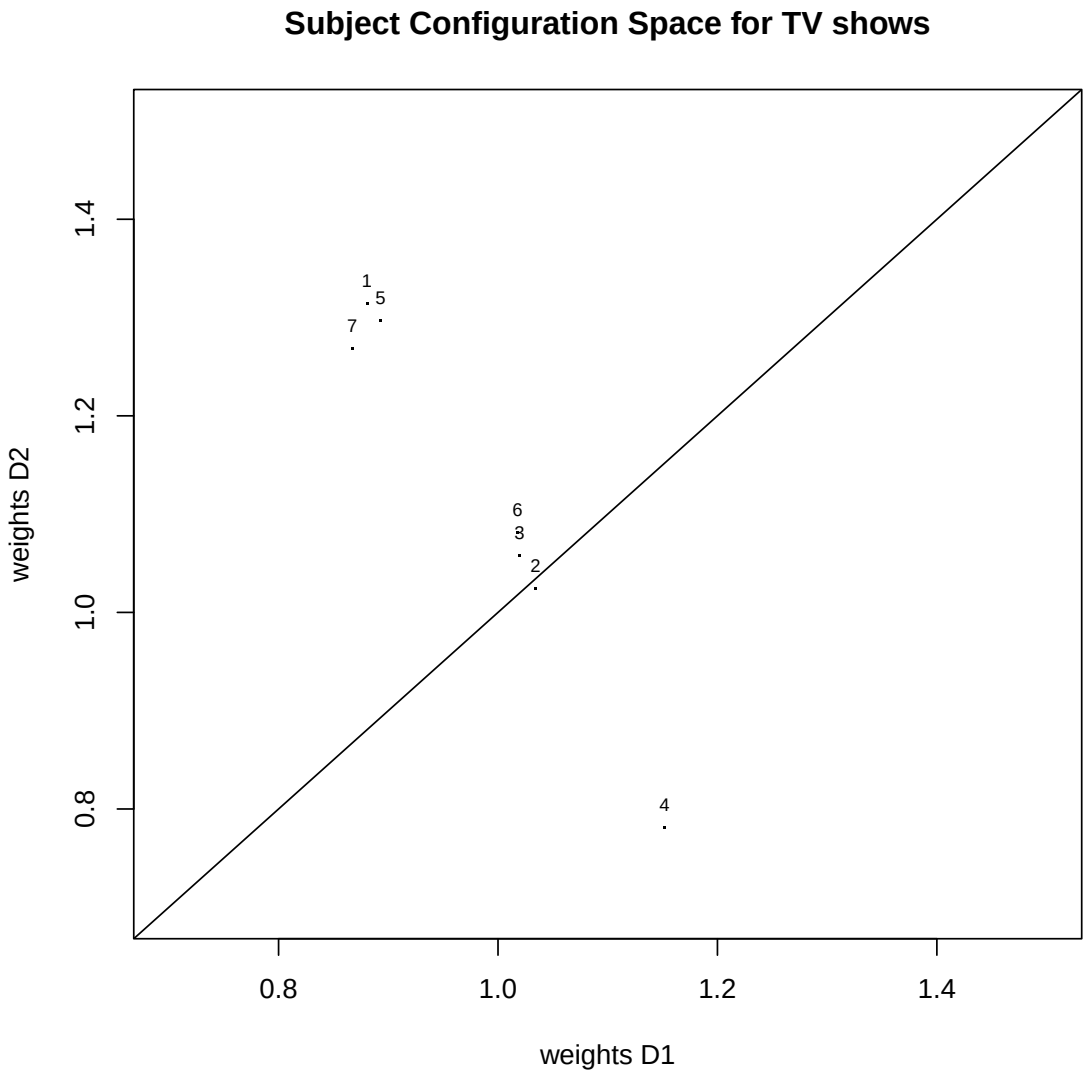

```
#or more conveniently
plot(output, plot.type = "confplot")
```




```
plot(output, plot.type = "Shepard")
```




```
weight <- matrix(0,7,2)
for (i in 1:7)
    weight[i,] <- diag(output$cweights[[i]])

lim <- c(0.7,1.5)
plot(weight,xlab="weights D1", ylab="weights D2",
     main="Subject Configuration Space for TV shows",
          xlim=lim, ylim=lim, pch=".")
text(weight,as.character(1:7),pos=3, cex=.7)
abline(0,1)
```




```
#these are raw weights and differ from the ones in SPSS because
#SPSS gives normalized weights, which are easier to interpret;
#careful when interpreting raw weights, see MacCallum (1977,
#Psychometrika)
```

ADDTREE and Clustering

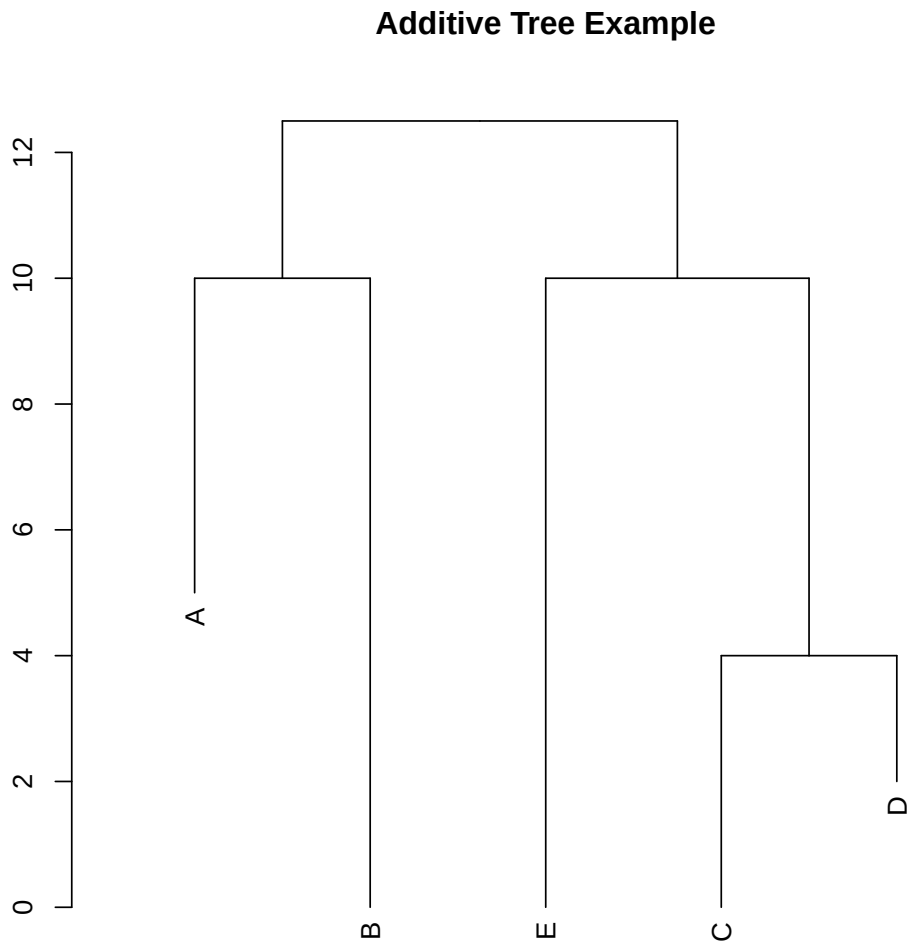
The package clue has several clustering algorithms including an addtree function. NOTE: If you do a search and find the term "additive regression trees" in R, that is something different than what I mean here as the clustering ADDTREE approach.

There are dozens of clustering packages available in R. For a summary see

https://cran.r-project.org/web/views/Cluster.html

Here is the simple (perfect fit) addtree example using the clue package. This is the same tree as in SPSS even though it is printed slightly different. The package clue also has functions to run ultrametric trees.

```
library(clue)
# create the distance matrix
distmat <- cbind(c(0, 15, 20, 18, 20), c(15, 0, 25, 23, 25), c(20,
    25, 0, 6, 20), c(18, 23, 6, 0, 18), c(20, 25, 20, 18, 0))
rownames(distmat) <- c("A", "B", "C", "D", "E")
colnames(distmat) <- c("A", "B", "C", "D", "E")
distmat


##    A  B  C  D  E
## A  0 15 20 18 20
## B 15  0 25 23 25
## C 20 25  0  6 20
## D 18 23  6  0 18
## E 20 25 20 18  0


out.addtree <- ls_fit_addtree(distmat)
plot(out.addtree, main = "Additive Tree Example")
```




```
# predicted dissimilarities
out.addtree


## Dissimilarities using Additive tree distances:
##    A  B  C  D
## B 15
## C 20 25
## D 18 23  6
## E 20 25 20 18
```

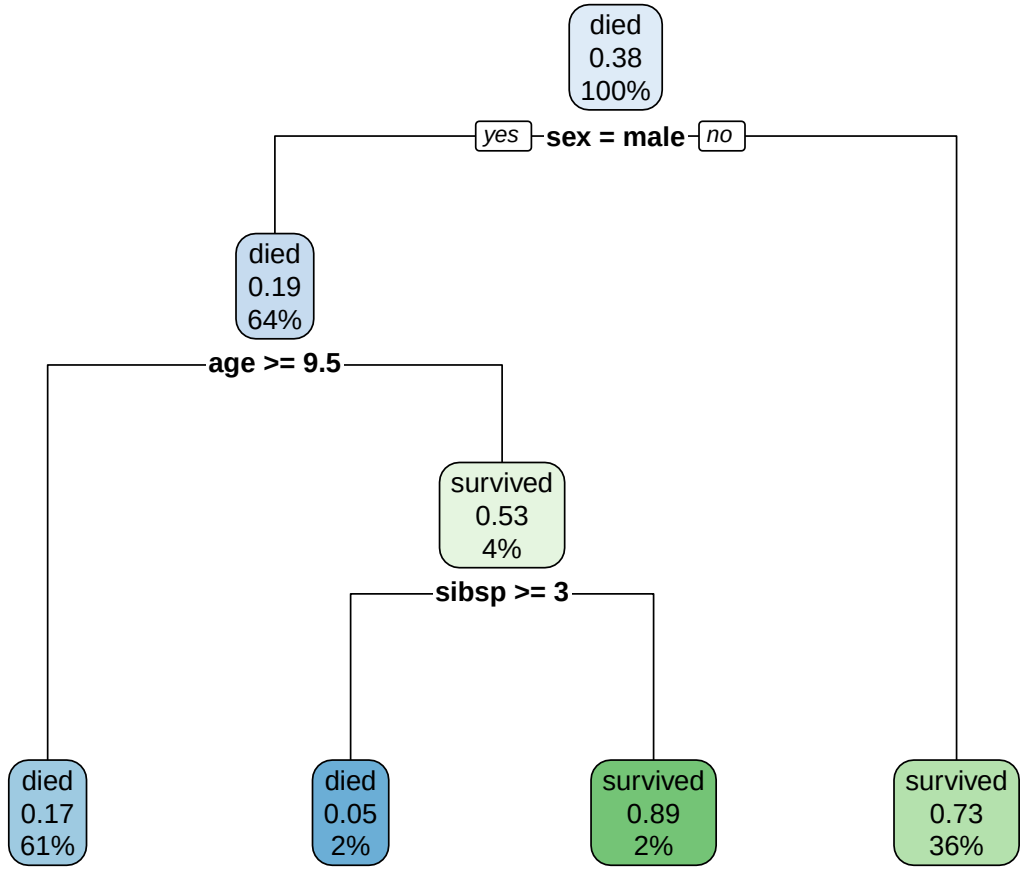
CART

There are several classification and regression tree programs in R. I've used the packages rpart and party to do various clustering algorithms. The programs are sophisticated in that they do the right thing depending on the dependent variable (binary dv is treated differently than continuous dv) and predictors (if a predictor is a factor, then it is treated as a categorical predictor rather than a continuous predictor). The party package also has versions of classification/regression tree that offer tests of significance including Bonferroni corrections for the multiple testing that goes on at each branch. Here is an example with the Titanic data set first with a regular recursive tree (rpart) and then again with the version of a classification tree with hypothesis testing and Bonferroni. You may also want to take a look at a newer alternative approach in the partykit package that has an associated ggparty package to produce nicer graphics using ggplot.

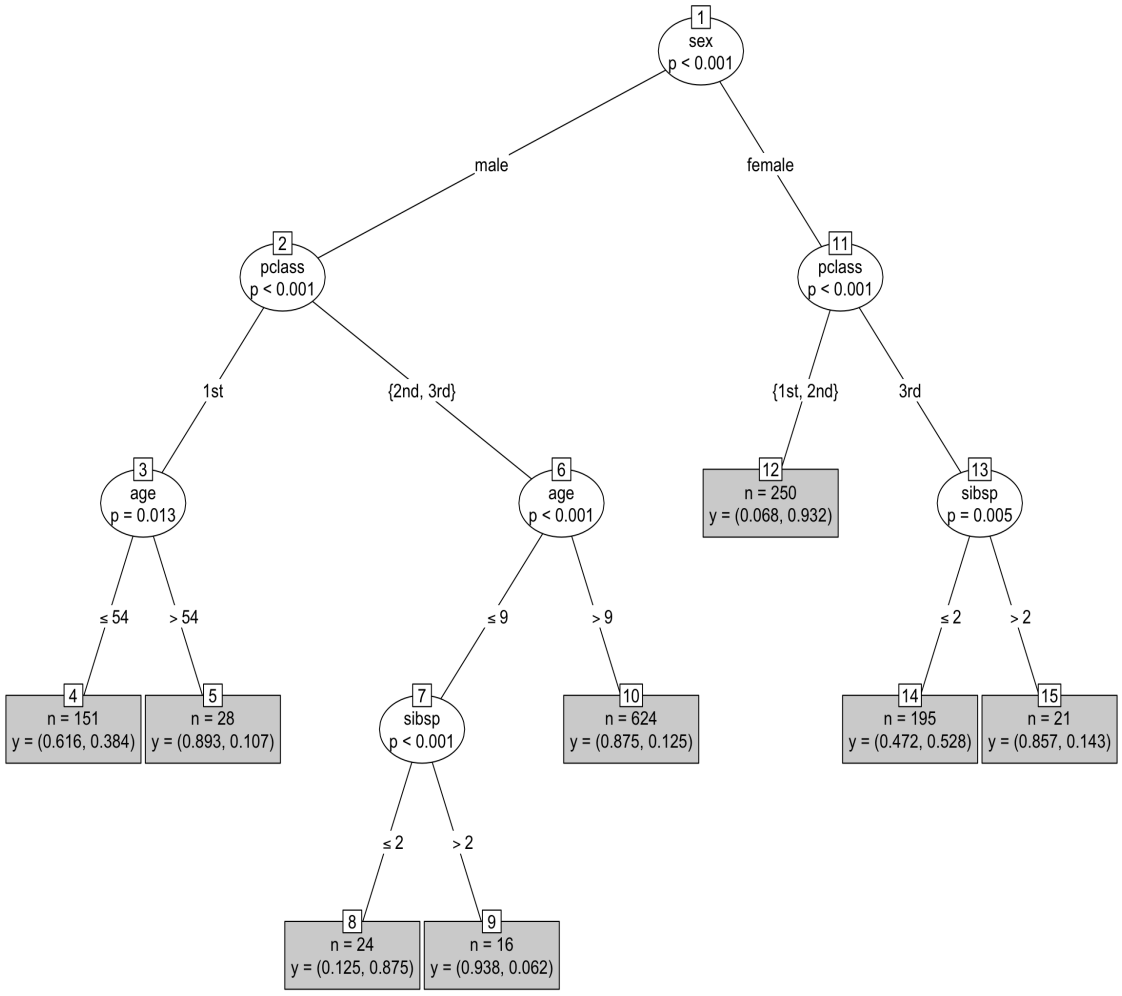
The Titanic data set has these variables:

```
pclass passenger class, unordered factor: 1st 2nd 3rd
survived factor: died or survived
sex unordered factor: male female
age age in years, min 0.167 max 80.0
sibsp number of siblings or spouses aboard, integer: 0...8
parch number of parents or children aboard, integer: 0...6
```

```
library(partykit)
library(rpart)
library(rpart.plot)

tree <- rpart(survived ~ ., data=ptitanic, cp=.02)
#cp=.02 because I want a small tree for demo
#tree can be pruned
test <- prune(tree,cp=.01)
rpart.plot(test)
```




```
#hypothesis testing version
ctr <- ctree(survived ~ ., data=ptitanic,
             control=ctree_control(testtype="Bonferroni"))
plot(ctr, type="simple")
```



More traditional clustering algorithms, such as the class of hierarchical trees that assume the ultrametric property, appear in the comprehensive clustering package CLUSTER.