Multivariate Analysis - Cluster Analysis

Joseph J. Luczkovich

February 27, 2014

Introduction

There are times when we don't really know anything about our variables. That is we have collected many observations and cases, and measured lots of things, but we don't know which variables are predictors of another variable. There may not be a theory we can use to predict one variable from another, or perhaps there is not relationship of any kind. But we wish to explore the data to discover relationships that might exist. We can explore the data to discover **natural groupings** of multivariate data, to perhaps combine the variables in some way that makes sense. This is an example of Exploratory Data Analysis (EDA). Plotting variables is one way to explore the data. Using data reduction methods (using **inter-dependency methods** of multivariate analysis) can be a useful way to explore complex data with many variables and explanatory factors.

Clustering

In this chapter, we will learn to use **clustering strategies**, which are widely used in social and natural sciences. The goal in clustering is to discover natural groups of cases (fishes, people, stations), in a hierarchy of relations, based on multiple variables. For instance, in taxonomy, biologists are interested in what species (cases) are grouped into Families, Orders, Classes, Kingdoms based on genetic or morphological variables, i.e., how to put the like species together. In marketing, analysts want classify customers (cases) into similar groups based on buying habits, psychology, social and economic classes, gender, etc. This approach is based on the similarity of cases (rows) in terms of the variables (columns). One way to discover natural groupings of data is to use a clustering strategy. This is similar to a method used by most people, even by small children: grouping like object together, so if you have a collection of objects, you can group them by which ones are similar to another and put them in piles of similar colors, shapes, textures, etc. Clustering strategies are simply ways of doing that mathematically using some sort of similarity measure. But first let us examine the measures of similarity we will need to use.

Similarity, Dissimilarity, Distance

First we must make a measure of the similarity of objects. Or dissimilarity. This is already familiar to you: correlation is the similarity of x vs y. If both vary together, they are similar. So the Pearson correlation coefficient r is a similarity measure for two variables. But there are other types of similarity metrics. One is **Euclidean Distance**. Distance measures are used to compute similarity, but they are the inverse of similarity: when two cases are very different, they have a high euclidean distance. This is defined as the linear distance between points in a multivariate coordinate system. The simplest such system to visualize are 2D coordinates of x and y axes (two variables) that define two vectors on which the cases were measured (Figure 1). Each x and y measurement will define a point (x=(1,3), y= (3,1)), and there are different x and y measurements of each case (only one case as an example is shown in Figure 1). A plot of the x and y in coordinate space will show the points. Euclidean distance is the direct line distance between the points across the coordinate space, using the **Pythagorean Theorem** to compute this distance (hypotenuse of the right triangle).

$$(XP)^2 + (YP)^2 = 2^2 + 2^2 = \sqrt{8}$$

. You can visualize Euclidean distance in up to 3 dimensions. It is difficult to graph in higher dimensions, but it can always be computed, no matter how many dimensions (variables) you have. The distances are computed in higher dimensional vector space. Use your powerful mind (abstraction) here to understand how this might be done in n-dimensional vector space.

Distance is computed in many ways. Here are some methods using the **dist()** command in R:

- euclidean,usual square distance between the two vectors.
- maximum, maximum distance between two components.
- Manhattan, absolute distance between the two vectors, like using a taxicab on a city grid.
- Canberra, a weighted Manhattan measure of distances.
- binary the vectors are regarded as binary bits.
- Malinowski, generalization of both the Euclidean distance and the Manhattan distance.

Let's load some data to create a similarity matrix using Euclidean distance. We will use some fish stomach content analysis data from Core Sound, NC. In this file, rows are fish individuals collected inside and outside of the no trawling zones of Core Sound. The columns record the identity number of the individual fish (fish.ID, in this case, the species is pinfish), which are the cases, where they were collected (Open or Closed trawling zone, and each column after that reports the proportion of the mass in each stomach attributed to different prey

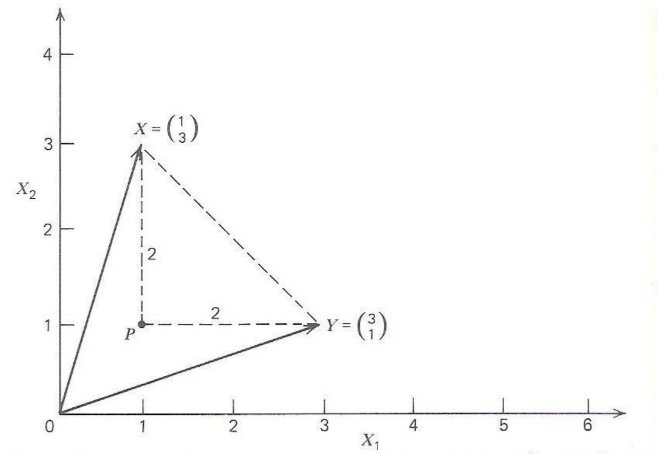


Figure I.3-3. Illustration of Euclidean distance. For $\triangle XYP$, we have $(XP)^2 + (YP)^2 = (XY$ or $2^2 + 2^2 = (XY)^2$, so $\|XY\| = \sqrt{8}$.

Figure 1: The euclidean distance visualized in two dimensions, x, y. These are two measurements, or variables, taken across multiple cases. Here just a single case measurements x=(1,3), y=(3,1) is shown. P is a point at right angles to the x,y. The variables form a 2D vector space. From [1]

types consumed by the fish. Actually, we will make a dissimilarity (rather than similarity) matrix using the **dist()** command, which will operate only on the columns with continuous measurements of our prey items in pinfish diet data:

```
> pinfish_diet<-read.csv("~/CRM7008/Clustering/Pinfish_diet.csv")
> pinfish_diet
```

```
   Fish.number   Area   Plankton     Plant     Epifauna     Infauna   Fouling
1       CLOSED Closed  0.6258179 96.836161  1.549007230  0.55999317  0.000000
2      CLOSED2 Closed  0.0000000 99.430834  0.003932112  0.30093033  0.000000
3      CLOSED3 Closed  0.2004361 92.617130  0.851531255  0.52598368  0.000000
4      CLOSED4 Closed  0.0000000 80.444874  9.110825290  0.00000000  0.000000
5      CLOSED7 Closed  0.0000000 46.521205  0.407168489  0.00000000  0.000000
6      CLOSED8 Closed 15.2400962 81.971564  0.000000000  2.73428175  0.000000
7      CLOSED9 Closed  0.3101242 50.925040  0.725567128  7.21379967  0.000000
8     CLOSED10 Closed  0.0000000 99.801633  2.368230914  0.02920775  0.000000
9     CLOSED11 Closed  0.0000000 92.143465  1.046490235  0.96130017  0.000000
10    CLOSED12 Closed  0.0000000 27.158980  1.163215291  0.30078257  0.000000
11    CLOSED13 Closed  0.0000000 49.230652  1.656090993  1.91265888  0.000000
12       OPEN1   Open  0.0000000 90.395816  9.364389841  0.21234308  0.000000
13       OPEN5   Open 11.0626751 12.249153 65.022380380  7.37859119  2.666142
14       OPEN6   Open  0.0000000 50.722702 18.254838800 19.49602083 16.196510
15       OPEN7   Open  1.6407267 12.531904  5.440881254 80.14602110  0.000000
16       OPEN8   Open  0.4651242 30.282063  0.000000000  4.48772342  6.696965
```

```
17       OPEN9   Open  1.3036137 51.604957 10.715423670 32.87126052  0.000000
18      OPEN10   Open  0.0000000 13.216017  0.928382240 62.69442508  7.570645
19      OPEN11   Open  0.4194916 19.561325  0.447708271 25.71700725 42.673005
20      OPEN12   Open  0.0000000 98.025418  0.693246953  0.41623789  1.062957
21      OPEN13   Open  1.2997738  2.338562  3.956611394  1.66878320  2.624613
22      OPEN14   Open  0.0869813 22.348482 14.653372200  2.46342587 42.581546
23      OPEN15   Open  0.0000000 35.446637  0.524122614  3.93167556 54.101267
24      OPEN16   Open  0.1466321 48.561367  0.729186303 42.91372835  0.000000
           POM
1   0.00000000
2   0.11383878
3   0.00000000
4  10.07640173
5  51.68915922
6   0.00000000
7  40.05571447
8   0.16915935
9   3.57907386
10 70.79541494
11 39.82116557
12  0.02745106
13  1.62105789
14  3.42818334
15  0.00000000
16  0.47804048
17  2.67615400
18  3.21637062
19 11.38538246
20  0.00000000
21 84.00982780
22  4.51701742
23  5.99629839
24  6.47894783
```

Now, using the columns only with the proportions of prey in the diet (columns 3 - 13), let's create a distance matrix using the **dist()** command, and using the default metric of dissimilarity, euclidean distance:

```
> dist.pinf<-dist(pinfish_diet[,2:6])
> dist.pinf
```

```
            1           2           3           4           5           6
2   3.4601927
3   4.8047895   7.6840590
4  20.2039793  23.5449766  16.4581510
5  56.2761476  59.1574382  51.5430475  39.1562409
6  23.4962850  26.0530194  20.7702051  20.1579621  43.2526093
```

7	51.8758360	54.7861382	47.2094422	35.2467771	9.4624404	38.8487974
8	3.5599391	2.6928688	8.2313746	22.9169029	59.6096744	26.5305917
9	5.3416541	8.2635315	0.7842533	15.9222576	51.0235661	20.6145188
10	77.9063318	80.8127838	73.1860424	60.2353870	21.6667324	63.6784391
11	53.2507900	56.1848019	48.5411788	35.9436883	3.9621923	40.4296542
12	11.3507025	14.5455133	9.8451474	11.1316346	50.0656868	22.2845385
13	119.0548838	122.4771416	115.8763137	99.7094826	83.1153231	106.8184500
14	58.7845718	61.9881966	54.9763665	41.0352793	29.9225404	47.7355599
15	129.6983053	132.0933143	126.3731100	117.5357215	97.5108169	117.4148085
16	74.5595422	77.4540035	69.8404532	57.2240197	18.8491220	60.1371232
17	62.9916900	65.8082098	59.4425000	48.9455658	38.9602977	51.7133223
18	116.4785009	118.9894820	112.7469397	103.1822570	79.3733122	103.4146173
19	90.8675620	93.7115811	86.3998652	74.5268120	41.6589907	76.1831212
20	1.7884950	1.7548664	6.0546317	21.7974886	57.5862294	24.8957998
21	105.6957686	108.6628581	101.0097514	87.5473882	49.6132384	90.5023319
22	84.5876181	87.7566168	80.0932290	65.3068394	31.4910878	70.7046982
23	68.7521267	71.6539459	64.0333056	51.4056280	13.1395304	54.7555684
24	71.8087420	74.1965357	68.3524781	60.5022672	48.0348251	60.8170778

	7	8	9	10	11	12
2						
3						
4						
5						
6						
7						
8	55.2645255					
9	46.6134567	8.7509522				
10	27.6790337	81.2286961	72.6587331			
11	6.3181382	56.5848771	47.9946119	24.7487317		
12	45.8487730	13.1076872	9.5395668	71.2930967	46.8625606	
13	84.7463048	121.2815702	115.3233498	74.7727730	83.1803545	108.2730965
14	23.9339048	61.6449774	54.2602840	38.9844482	27.0859571	50.3097441
15	92.3116089	132.5086910	125.6494296	90.8996966	96.7230175	124.8512494
16	23.2947007	77.9316514	69.2872662	6.0055736	21.4662223	68.1895365
17	30.8129810	65.8871672	58.7026818	46.7892338	36.1912154	56.7324899
18	75.0017293	119.5097879	112.0297656	71.4792626	78.9937228	111.4220477
19	40.7145624	94.2220413	85.7434151	29.6731520	42.5522051	84.7621392
20	53.2065512	2.7636713	6.6161965	79.2329338	54.5904700	12.9151616
21	54.8043350	109.0065983	100.4712605	28.0048444	52.5108249	98.6606221
22	35.9377807	87.7200681	79.5200512	16.1945136	33.3895809	76.3502451
23	17.6949871	72.1127199	63.4786017	10.1413191	15.6267739	62.3638119
24	40.0015406	74.7274626	67.6343818	53.3166859	45.8587000	67.5288611

	13	14	15	16	17	18
2						
3						
4						

```
5
6
7
8
9
10
11
12
13
14  70.1483410
15 105.6758425  81.4236535
16  76.4342936  34.9378769  87.1079171
17  80.9570899  17.2560714  68.8253306  41.4700849
18  95.4672698  66.8305660  20.2509520  67.8265092  55.4596413
19  76.4275065  40.7277847  61.6274442  26.5946245  38.4737791  41.9523447
20 120.7614497  60.3116341 130.8205416  75.8817837  64.3266190 117.6396106
21  70.3132643  59.8431713  88.4936466  31.7241447  65.6358689  69.4019774
22  58.9879027  37.2186744  88.1634485  18.7717806  47.4020750  69.8172444
23  77.7207544  31.4242204  89.1668237   5.8601471  38.7967929  70.2444359
24  92.4263062  32.7915841  58.1893269  47.5831372  16.2472360  45.2856339
            19          20          21          22          23
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20  92.1749633
21  33.3171867 107.0623781
22  30.6269641  86.0674320  25.4194786
23  30.1480851  70.0757691  37.3285531  21.6032602
24  37.6975353  72.9104916  69.3671321  56.0939928  45.9845101
```

This matrix is a result of the euclidean distance metric applied to each case,

```
> scatterplot3d(Plant,Epifauna,Infauna)
```

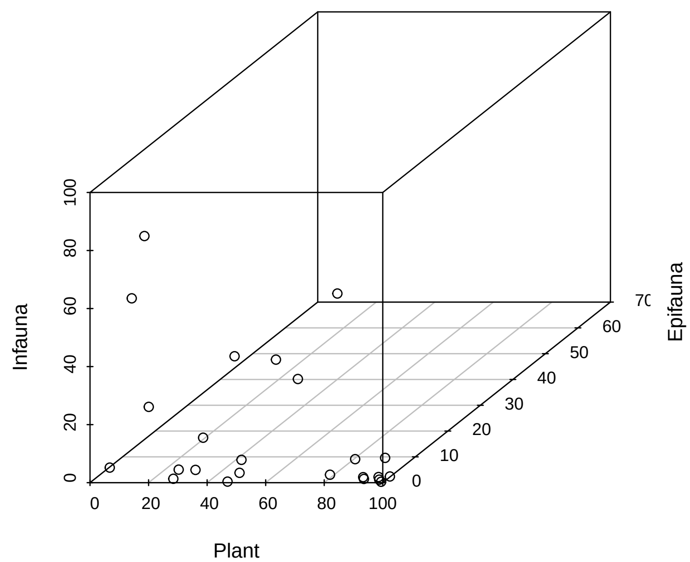


Figure 2: The 3D scatter plot of the proportion of plants, epifauna (small benthic critters), infauna (larger benthic worms) in the stomach contents of pinfish from Core Sound, NC.

across the columns of prey, so the similarity (low distance = high similarity) of the diet of fish 1 is given relative to that of fish 2,3,4,5 etc. To illustrate the similarity of the fish for just three of the prey types, I will create a 3D scatter plot (but download R package scatterplot3d first) with points representing the individual fish stomach contents along an axis devoted to three prey types:

```
> library("scatterplot3d", lib.loc="C:/Users/Joseph/Documents/R/win-library/3.0")
> attach(pinfish_diet)
```

The points in Figure 2 represent the diets of each pinfish. Some fish plot close together in the 3D plot, they have similar diets (based on only these three prey types). The pinfish plotting close to one another will have low euclidean distance, thus high similarity. Imagine a straight line connecting each pair of fish, that is the euclidean distance. The matrix we computed above considered all they prey in the diet, so it is a better estimate of similarity, but it is impossible to visualize with a graph.

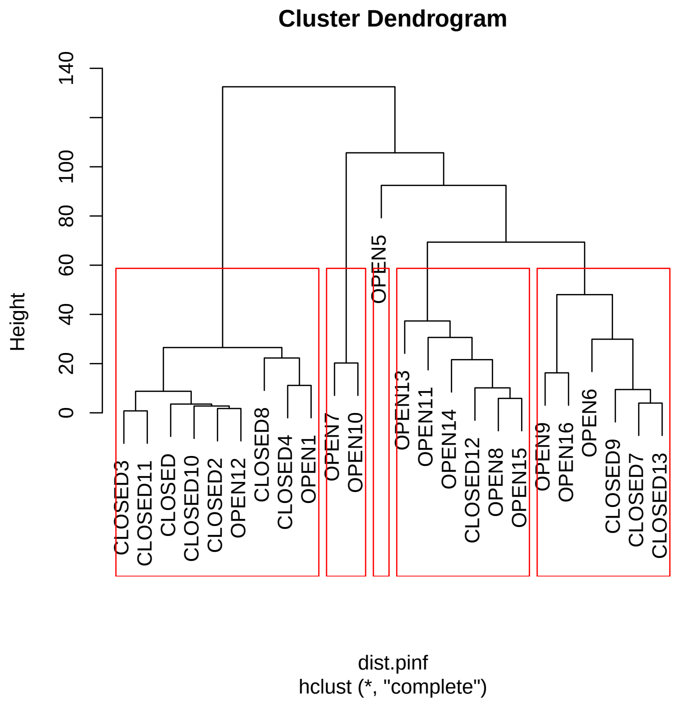
1 Clustering Strategies

A whole other area to consider in clustering multivariate data, beside which distance metric to use, is to choose an appropriate clustering algorithm, i.e., what is the best clustering strategy? We could start like a child with all her toys laid out separately and start by putting ones together in piles if they have a high similarity (low distance), creating some number of piles. These piles then could be grouped as well in a hierarchy of similarity. This is called an **agglomerative** clustering strategy. The child could also have one huge pile of toys and start dividing the huge pile into some number of different smaller piles based on similarity. This is called a **divisive** clustering strategy. Both are hierarchical. There is a type of strategy called k-means clustering. If you know how many large groups or piles you ultimately want, say three groups, you could specify k-means clustering with k = 3. Spam email is often sorted this way (known sender, unknown sender OK, unknown sender SPAM).
The clustering process is iterative. The agglomerative strategy starts with the separate cases and groups them together in larger and larger piles based on the linkage criterion that you specify. Do you group with the closest one in vector space (most similar by distance metric chosen? If the groups get bigger and bigger, which point or group of points is used to judge the similarity to other points and groups (piles)? The average, median or centroid similarity of cases within each group? The nearest group member in vector space for the two groups? Thus, there are many linkage methods to choose from. In R, a number of different clustering methods are provided. Each clustering algorithm uses a different method to link the cases or groups of cases.

- The *complete linkage* method finds similar clusters based on the farthest away points within each group's points. Default method in R
- The *single linkage* method finds similar clusters based on the nearest points to the ones in each cluster; it adopts a "friends of friends" clustering strategy. Creates chains of groups.
- *Ward's minimum variance* method aims at finding compact, spherical clusters.
- *Median linkage* use the median distance measure with each cluster to join groups.
- *Centroid linkage* computes coordinates for a central point within each cluster in vector space and uses this centroid to join groups.

The other methods can be regarded as aiming for clusters with characteristics somewhere between the single and complete link methods. Note however, that the methods median and centroid do not lead to monotonic distance measure, and the resulting **dendrograms** can have so-called inversions (which are hard to interpret). These choices are complicated and will give different cluster dendrograms (chaining in single linkage). You can try experimenting with different

```
> plot(pinfish.clust, labels = pinfish_diet[,1])#plots dendrogram and labels ends
> rect.hclust(pinfish.clust,h=50)#this makes a cluster cut-point at height=50
```



linkage methods; you should feel good if they seem to produce the same clusters over time. If they produce different clusters each time you use a new linkage method, there may be problems with extreme values in the underlying data. The command**hclust()** used on a distance matrix object is what will perform a clustering algorithm, using the complete linkage (default). Let's try this on our distance matrix from the pinfish diet data:

```
> pinfish.clust<-hclust(dist.pinf)
```

Then, we can plot the resulting cluster output object as a dendrogram, by using the **plot ()** command. Let's make a cluster dendrogram of the pinfish diet data Height in these dendrograms is equivalent to the euclidean distance. Fish with the most similar diets have low heights at which the linking method joined them. There are many ways to cut a dendrogram, you could also use k= 2 instead of h=50 in the code to split the clusters into two major clusters, for example. It is often best to leave it to the reader or viewer to do this task, the groups may be obvious. Here it is not so obvious that the pinfish ate different diets in the trawled and non-trawled (closed) areas. There may be one large cluster of fish from the open areas, but two from the closed are also grouped there. We could try using different distance metrics, or different linkage methods. There are other statistical methods to generate probability values for each branch of the cluster diagram.

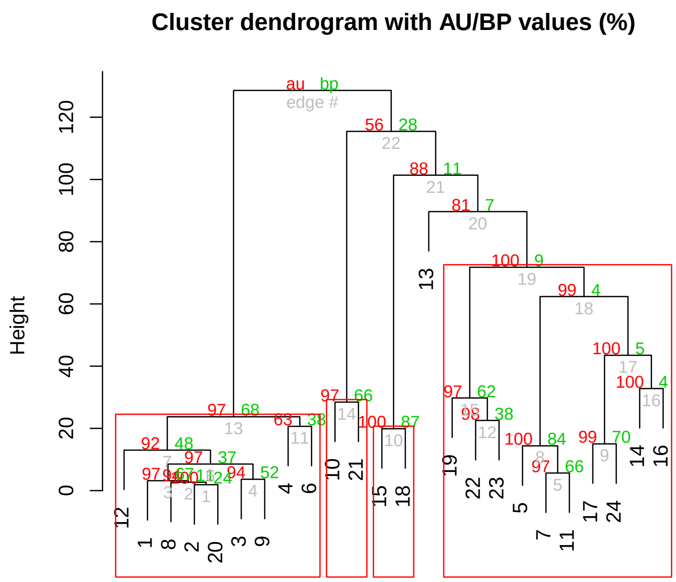
Probability clustering

There is an algorithm in R that allows you to generate a probability for the cluster groups based on bootstrap methods. The name for "bootstrap" comes from the phrase "to pull oneself up by one's bootstraps" - this is impossible feat. That is what the developers of bootstrap methods have done, created some statistical method that is impossible, or nearly so. It belongs to the general approach of re-sampling statistics - the data themselves from a small sample are used to generate a distribution to test against. The bootstrap approach in **pvclust()** generates many possible clusters using the **hclust()** algorithm (but dropping one point in each re-sampling run and then computing the clusters again) and compares the re-sampled distribution of these clusters to the observed clusters to see if the clusters you have discovered are "real". The package is called pvclust (http://www.is.titech.ac.jp/ shimo/prog/pvclust/). It computes p-values giving the uncertainty of the clusters, with 100% very certain. The hypothesis is that the clusters are "real" if the p-value is > 95%, that is, there is a 5% error of being wrong and the clusters are false. Two kinds of p-values are produced: approximately unbiased multi-scale bootstrapped probability (AU) and normal bootstrapped probability (BP). The AU is better because it accounts for bias in the bootstrapping approach. There is another function that allows you to discover clusters with very large and small probabilities. When you run it, it may take a long while to compute the final cluster dendrogram. You should see progress message "Bootstrap (r = 0.5)... Done" as it re-samples your data and updates the user. Here is an example of **pvclust()** with pinfish diet data from fish collected inside and outside Core Sound. pvclust likes the columns to be what is clustered, so we will first transpose the diet matrix:

```
> library("pvclust", lib.loc="C:/Users/Joseph/Documents/R/win-library/3.0")
> pinfish.tr<-t(pinfish_diet[,3:8])#just get the columns with prey, transpose
> pinfish.pvclust<-pvclust(pinfish.tr,method.hclust="complete",
+          method.dist="euclidean")#run pclust on transposed matrix
```

```
Bootstrap (r = 0.5)... Done.
Bootstrap (r = 0.5)... Done.
Bootstrap (r = 0.67)... Done.
Bootstrap (r = 0.67)... Done.
Bootstrap (r = 0.83)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.17)... Done.
Bootstrap (r = 1.17)... Done.
Bootstrap (r = 1.33)... Done.
```

```
> plot(pinfish.pvclust)
> pvrect(pinfish.pvclust,alpha=0.95)#red boxes around clusters exceeding p=95%
>
```



Distance: euclidean
Cluster method: complete

Next, we plot the dendrogram: So, we can see that there are some reliable clusters with P=100%. The red boxes show which clusters have a high probability of being "real": they are likely to be true and present in future studies 95 times out of 100 (if the bootstrap probabilities are correct). There seems to be a pattern that the fish from the open trawling areas group together in the large cluster on the right side (fish IDs 14-24), these fish ate a lot of infauna, but little plant material. But there are some exceptions: fish IDs 12 and 20 are not in this open-trawling cluster, they grouped with the closed-area fishes; when you examine their diets in the original data, they ate a lot of plants, so are grouped by the clustering strategy with the closed-area fish, which also ate plants predominantly. Fish IDs 5,7,11 were collected in closed trawling areas and are grouped with the open trawling area fish. Furthermore, there are low AU p-values (in red color text) associated with edge numbers for some clusters(the gray letters mark the dendrogram divergences 20,21,22). At edge number 19 and below, the AU p-values are all high, over 90%. Fishes in the middle two clusters (IDs 10,21, and IDs 15,18) are not reliably grouped with either main trawling-area groups classified here. They are intermediate in the groupings. These results are not surprising, as the fish could easily migrate between the areas, there was no fence separating trawling and non-trawling areas. The areas were adjacent to one another, so we might expect some overlap. Nonetheless, the clustering suggests there might be some trawling effects on pinfish diets. Thus,in conclusion, we have found some natural groupings associated with the trawling in Core Sound that is apparent in the pinfish diet data, and we can do some additional hypothesis testing and experimentation on the ideas that have emerged from the cluster dendrograms.

References

[1] William R Dillon and Matthew Goldstein. *Multivariate analysis: Methods and applications*, volume 45. John Wiley & Sons New York, 1984.