Displays for Statistics 5401

Lecture 41

December 13, 2005

Christopher Bingham, Instructor
612-625-1024

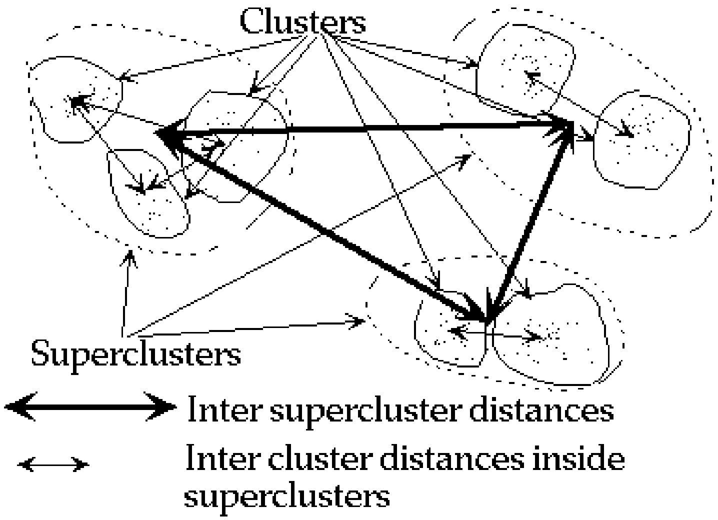
Class Web Page
http://www.stat.umn.edu/~kb/classes/5401

How many clusters are there?

Sometimes you have a prior idea as to the number of clusters there should be.

More usually, you base your decision on the history of the criterion = distances between the two clusters that were merged to reach a stage. This *always increases* or at least never decreases.

- While merging two objects or clusters that should be in the same cluster, the distance criterion should be relatively small.
- When you merge two distinct clusters, the criterion should take a jump, that is, be larger than the value at the previous merge.
- Sometimes real clusters are themselves grouped in "super clusters". When this occurs, the criterion should take a additional jumps when super clusters are merged.



This should give a little of the flavor of what's going on.

First the "dots" are gathered together to form the clusters, some of which are far apart and the others are close (<u>light double ended arrows</u>).

Then the clusters are gathered together to form super clusters, all of which are fairly well separated (<u>heavy double ended arrows</u>).

Here's another look at the "toy" example, 8 points from 3 separated populations.

```
Cmd> x
          x1          x2
2      15.606      27.451      Row labels are "true"
1      7.2295       29.53      population labels; not
1      9.9958      30.821      known in practice
3      17.241       31.21
2      16.212      25.889
1      10.644      28.937
3      20.954      31.244
2      14.528      24.695
```

First save the criterion values:

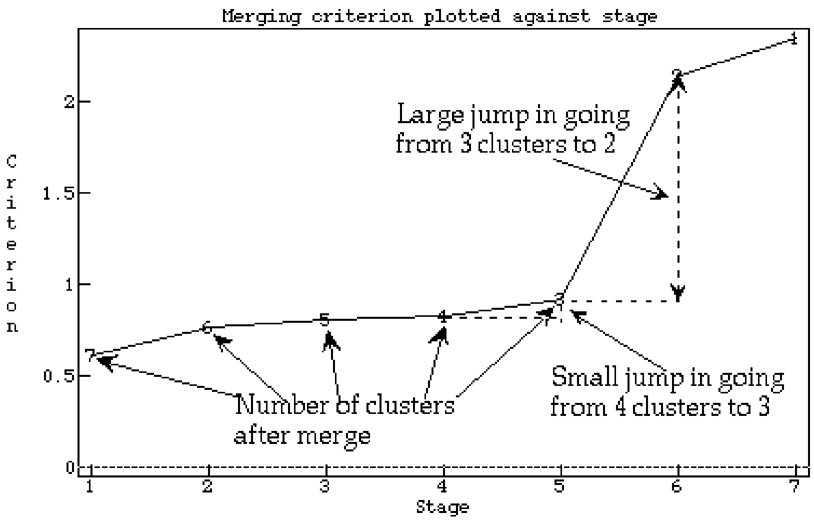
```
Cmd> criterion <- reverse(cluster(x,keep:"crit")); criterion
(1)     0.60943     0.76945     0.80378     0.83117     0.91951
(6)      2.1453      2.3442
```

When the largest number of clusters output is M (=8 here), **length(criterion)** = m = M-1.

I used **reverse()** so that **criterion[1]** is the distance between clusters M and M-1 as they are merged, **criterion[2]** is the distance at the next merge, ..., **criterion[m]** is the distance between clusters 1 and 2 when they merge.

You can now plot **criterion** vs stage and look for jumps.

```
Cmd> m <- length(criterion); n <- nrows(x)

Cmd> plot(Stage:n-m,Criterion:criterion,symbols:run(m,1),\
      lines:T,ymin:0,\
      title:"Merging criterion plotted against stage")
```

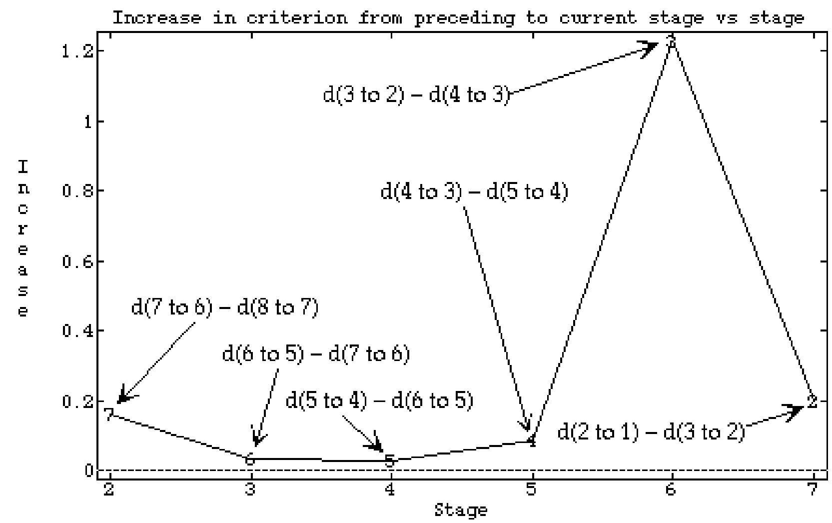


As compared to previous changes, the criterion takes a big jump going from 3 clusters to 2. This suggests g = 3 clusters.

- Stage 0 is before any merging, when there are N clusters.
- At stage j, there are N - j clusters.
- The criterion value for stage 1 is the smallest inter-object distance d_{ij}.

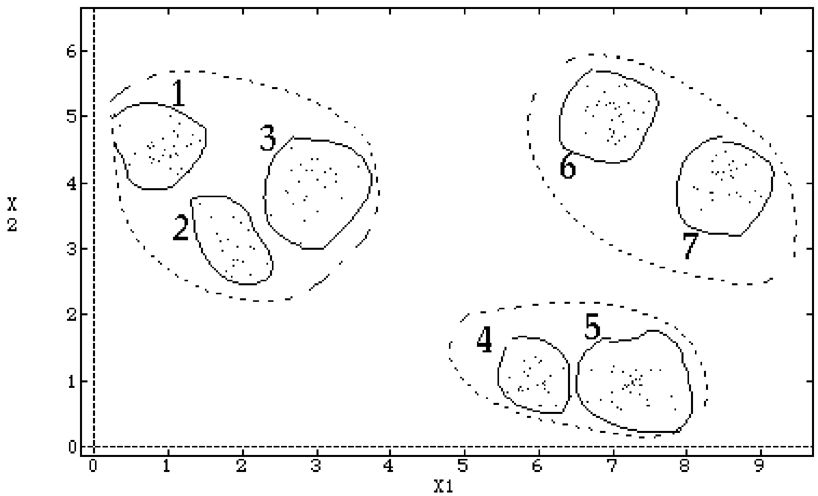
I find the changes or *differences* in the distances are more informative. While building clusters, changes should be small. When merging two real clusters they should should show a jump and sometimes a peak.

```
Cmd> plot(n-m+1,criterion[-1]-criterion[-m],symbols:run(m,2),\
    xlab:"Stage", ylab:"Increase", ymin:0,lines:T, title:\
 "Increase in criterion from preceding to current stage vs
stage")
```



There is large jump in differences when 3 clusters go to 2, suggesting there are 3 clusters. The peak is at the 2^{nd} from last point and 2 + 1 = 3 = guess at g.

Here's an analysis of an artificial bivariate example where there are 7 clusters in 3 super clusters.

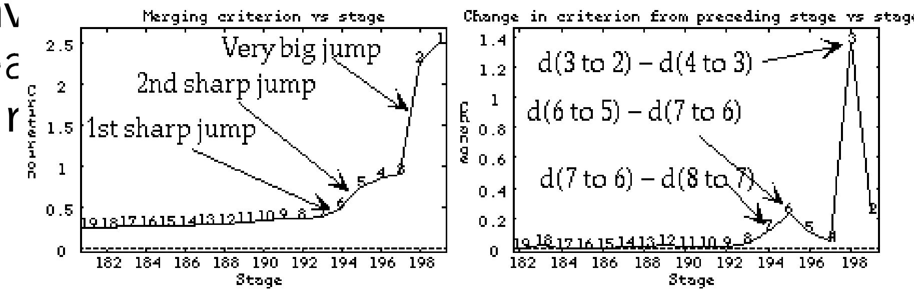


The numbers are labels for the actual populations sampled.

You ought to be able to do fairly good job of recovering the "true" clusters as they are all more or less circular. This is an almost ideal case.

```
Cmd> N <- nrows(y1)

Cmd> criterion <- cluster(y1,nclust:20,keep:"criterion")

Cmd> classes <- cluster(y1,nclust:20,keep:"classes")
```


`cluscritplot()` in file
mv
ea
cr



These are criterion plots for average linkage.

On the left, the jump from 7 to 6 is the distance when groups 4 and 5 merged.

The jump from 6 to 5 is the distance when groups 2 and 3 merged.

The plot on the right suggests 7 or 6 clusters that later merge into 3 super-clusters.

Here are confusion tables of the average linkage 7 cluster and 6 cluster solutions with true cluster membership.

```
Cmd> print(format:"6.0f",matrix(tabs(,True,classes[,6]),\
        labels:structure("True ","Clus ")))
MATRIX:
```

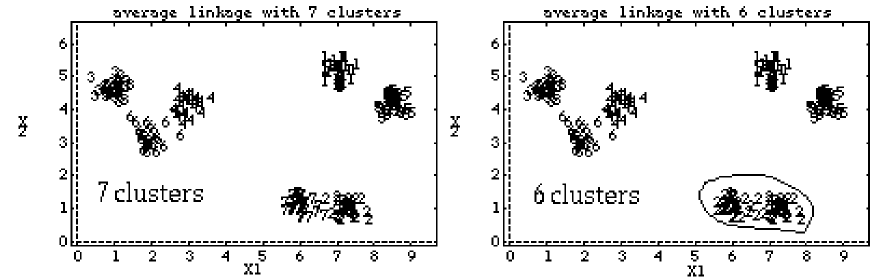
	Clus 1	Clus 2	Clus 3	Clus 4	Clus 5	Clus 6	Clus 7
True 1	0	0	32	0	0	0	0
True 2	0	0	0	0	0	21	0
True 3	0	0	0	24	0	1	0
True 4	0	1	0	0	0	0	27
True 5	0	32	0	0	0	0	1
True 6	33	0	0	0	0	0	0
True 7	0	0	0	0	28	0	0

There is almost perfect clustering, with only three cases in a cluster that don't correspond to their populations. Two of these are in the next clusters to be merged (clusters 7 and 2).

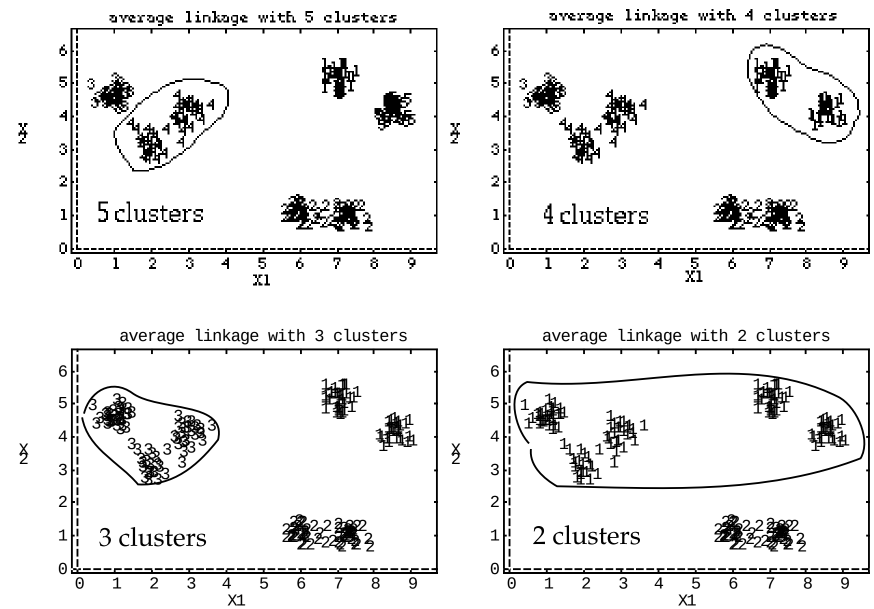
```
Cmd> print(format:"6.0f",matrix(tabs(,True,classes[,5]),\
        labels:structure("True ","Clus ")))
MATRIX:
```

	Clus 1	Clus 2	Clus 3	Clus 4	Clus 5	Clus 6
True 1	0	0	32	0	0	0
True 2	0	0	0	0	0	21
True 3	0	0	0	24	0	1
True 4	0	28	0	0	0	0
True 5	0	33	0	0	0	0
True 6	33	0	0	0	0	0
True 7	0	0	0	0	28	0

Now populations 4 and 5 are grouped together in cluster 2.

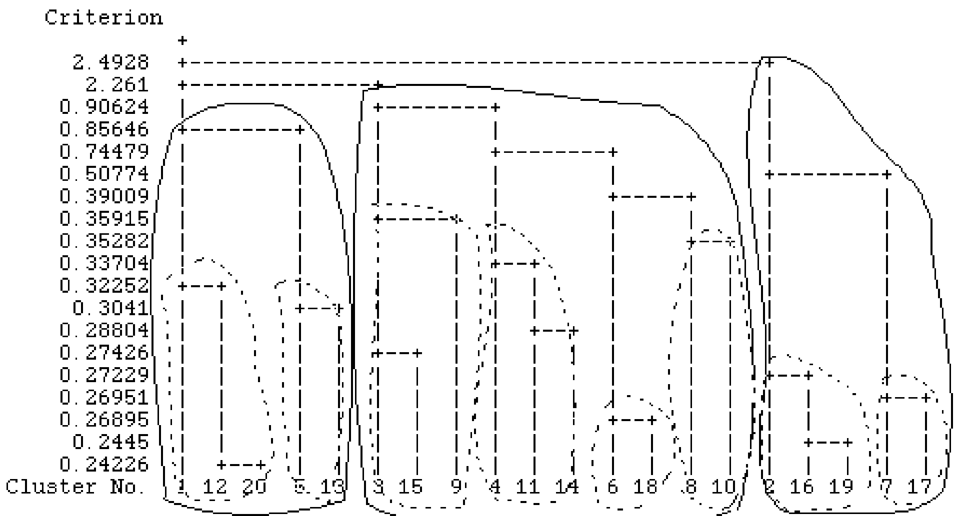


The distance between clusters 2 and 7 is not much bigger than the average of distances between clusters. That's why it doesn't show up much in the criterion plots.



The three cluster solution corresponds to the super clusters.

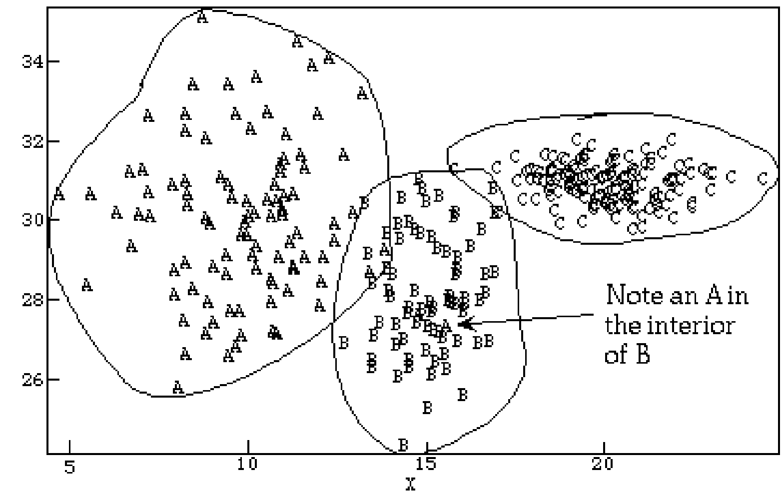
Here is the dendrogram down through 20 clusters:



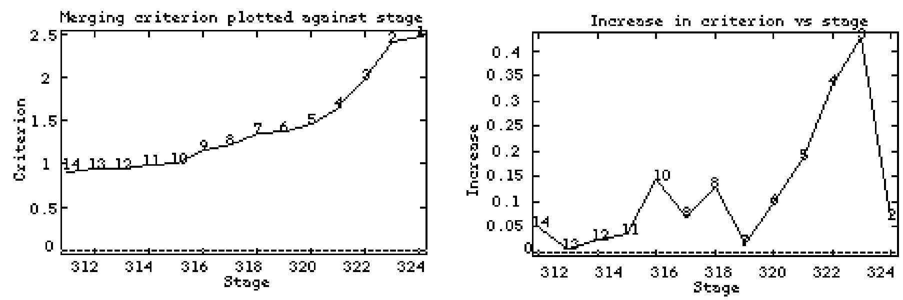
- Solid loops: Histories of the three super clusters
- Dashed loops: Histories of the seven clusters.

There's a lot of history missing because only the top part of the dendrogram with 20 or fewer clusters is shown.

Here is a N = 325 sample from the same populations as the N = 8 sample.



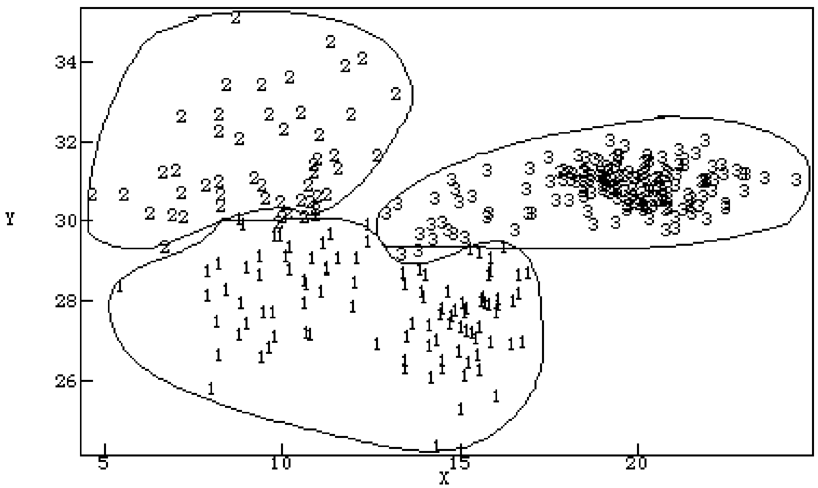
Plots for average linkage



The biggest change is again the second from last which suggests g is 3. The preceding change was also quite big so you might reasonably guess g=4.

How well does the 3 cluster solution match the actual? Not well, except for population C.

Here are the clusters found when there are 3. Clusters 3 and 2 pretty much consist of populations C and A, and but cluster 1 overlaps populations A and B substantially.



If you compare this with the actual clusters, you see cluster 2 contains about half of group A, with cluster 1 containing the rest of A and most of B. Cluster 3 is primarily group C.

A "confusion matrix" shows the amount of overlap.

```
Cmd> classes <- cluster(y,keep:"class")

Cmd> tabs(,groups,classes[,2]) # confusion matrix,g = 3
(1,1)      47        51         2 True group A
(2,1)      56         0        19 True group B
(3,1)       0         0       150 True group C
   Cluster 1          2         3
```

- About half of cluster **1** is group A and about half is group B.
- Cluster **2** consists entirely of group A
- Cluster **3** is mainly group C which is entirely in it.

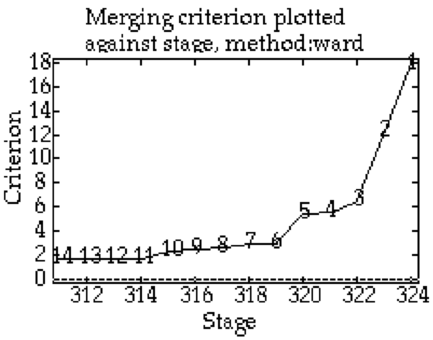
```
Cmd> tabs(,groups,classes[,3]) # confusion matrix,g = 4
(1,1)      47        51         2         0
(2,1)      53         0        19         3
(3,1)       0         0       150         0
```

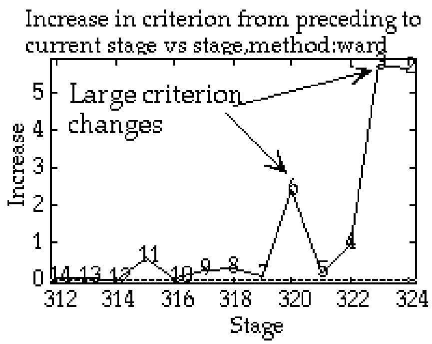
- When there were 4 clusters, the 4th consisted of 3 cases from group B which were later put in cluster 1

```
Cmd> tabs(,groups,classes[,4]) # confusion matrix,g = 5
(1,1)      47        34         2         0        17
(2,1)      53         0        19         3         0
(3,1)       0         0       150         0         0
```

- When there were 5 clusters, the 5th consisted of 17 cases from group A which were later put in cluster 2

Plots for Ward Method





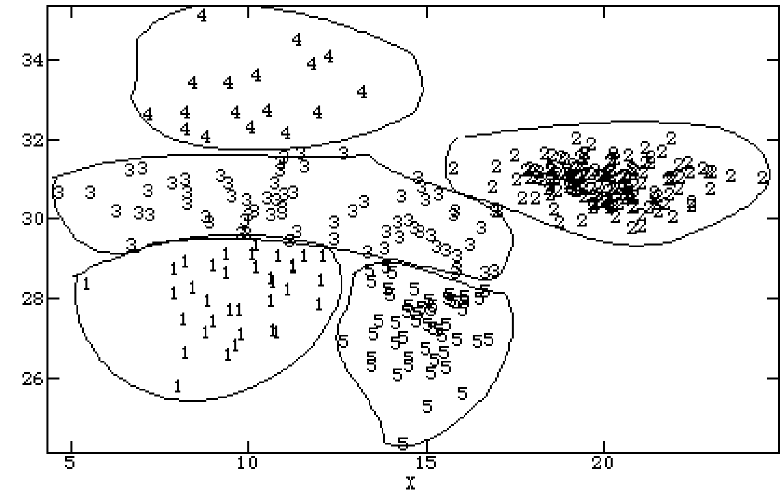
Differences show a big jump going from 3 to 2 clusters, indicating 3 clusters.

```
Cmd> classes <- cluster(y,keep:"class",method:"ward")

Cmd> tabs(,groups,classes[,2]) # 3 cluster confusion matrix
(1,1)      39          0         61
(2,1)      48          1         26
(3,1)       0        149          1

Cmd> tabs(,groups,classes[,3]) # 4 cluster confusion matrix
(1,1)      39          0         44         17
(2,1)      48          1         26          0
(3,1)       0        149          1          0

Cmd> tabs(,groups,classes[,4]) # 5 cluster confusion matrix
(1,1)      36          0         44         17          3
(2,1)       0          1         26          0         48
(3,1)       0        149          1          0          0
```

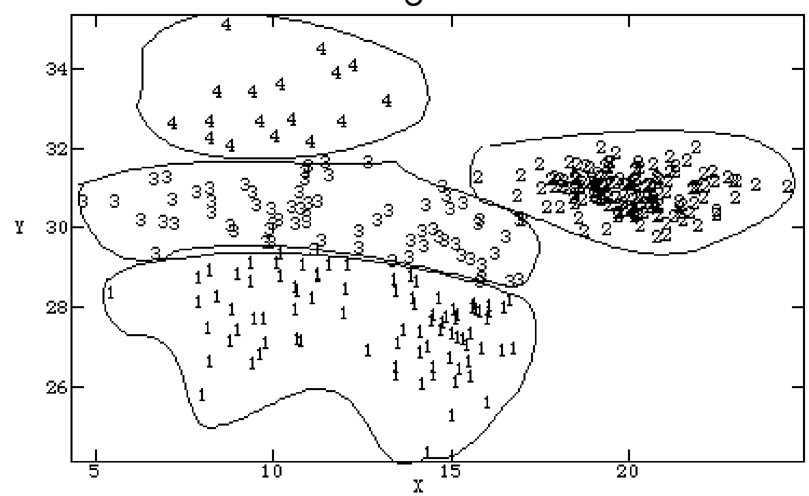
Together with the earlier peak, which by itself might suggest 6 clusters, there is a hint of a hierarchy of clusters, with 6 clusters grouped in 3 larger clusters.

Plots of the clusters for g = 5, 4 and 3:

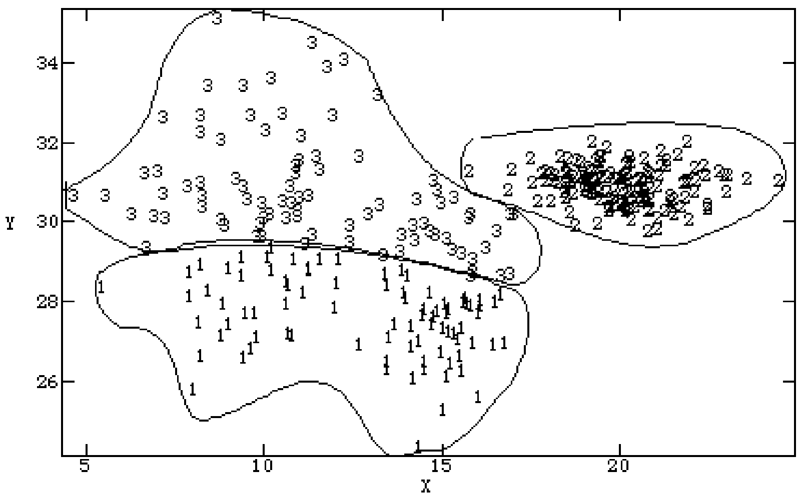
Ward method g = 5 clusters



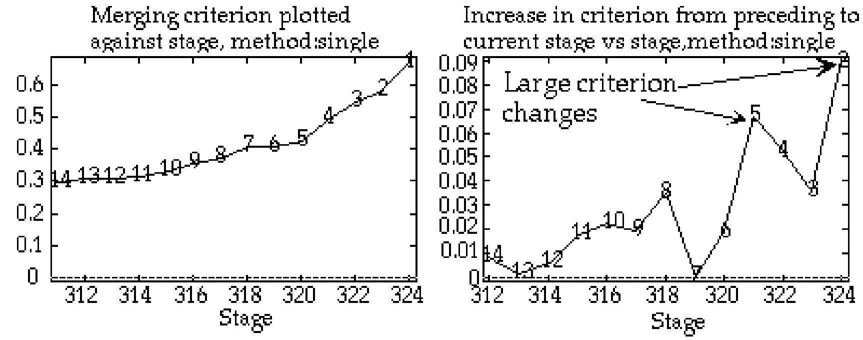
Ward method g = 4 clusters



Ward method g = 3 clusters



Plots for Single Linkage



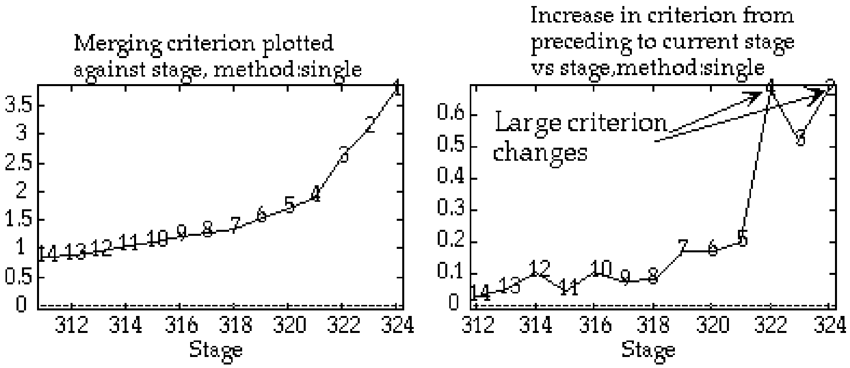
No clear indication of number of clusters, although it suggests 5 clusters split into 2 larger clusters. But it isn't so.

The confusion matrices tell the story.

```
Cmd> tabs(,groups,classes[,2])
(1,1)     99          1          0 Cluster 2 is singleton
(2,1)     74          0          1 Cluster 3 is singleton
(3,1)    150          0          0

Cmd> tabs(,groups,classes[,3])
(1,1)     98          1          0          1
(2,1)     74          0          1          0
(3,1)    150          0          0          0

Cmd> tabs(,groups,classes[,4])
(1,1)     97          1          0          1          1
(2,1)     74          0          1          0          0
(3,1)    150          0          0          0          0
```

Lots of singletons. With 15 classes, 8 were "singletons". Single linkage works terribly with this data even though there is a clear clustering.

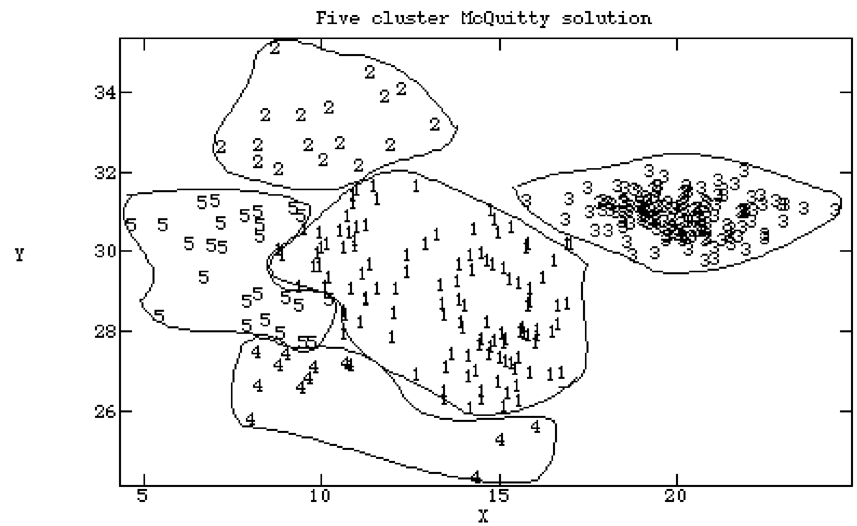
Plots for McQuitty method

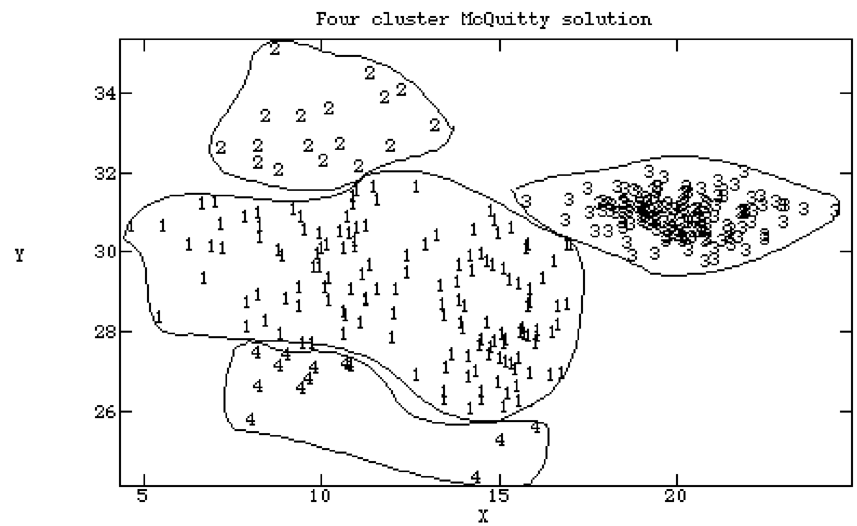


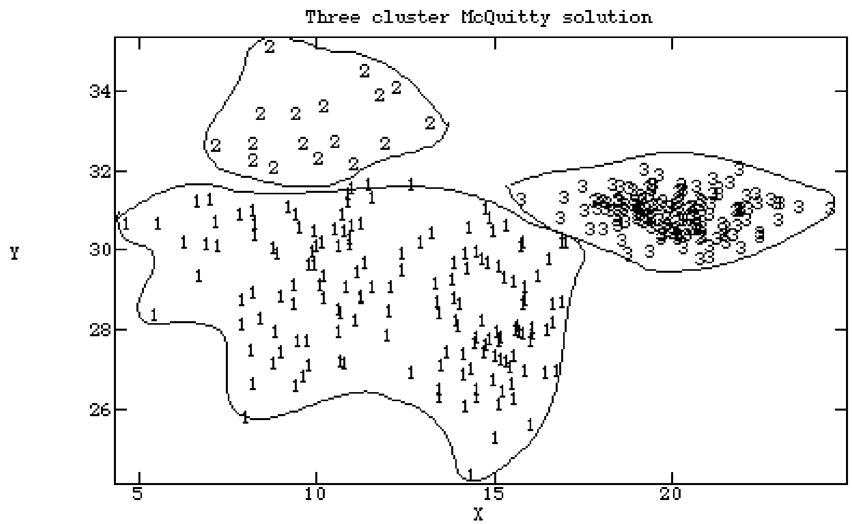
This shows big increase in change going from 4 clusters to 3, indicating 4 clusters.

```
Cmd> classes <- cluster(y,keep:"class",method:"mcquitty")

Cmd> tabs(,groups,classes[,2]) # g = 3
(1,1)      83         17          0
(2,1)      74          0          1
(3,1)       1          0        149

Cmd> tabs(,groups,classes[,3]) # g = 4
(1,1)      73         17          0         10
(2,1)      71          0          1          3
(3,1)       1          0        149          0

Cmd> tabs(,groups,classes[,24) # g = 5
(1,1)      47         17          0         10         26
(2,1)      71          0          1          3          0
(3,1)       1          0        149          0          0
```

McQuitty doesn't do too bad a job, but doesn't come close to accurately identifying original clusters.







Utility company data from text.

```
Cmd> data <- read("","t12_05")
T12_05    22    8 format labels
) Data from Table 12.5 p. 687 in
) Applied Mulivariate Statistical Analysis, 5th Edition
) by Richard A. Johnson and Dean W. Wichern, Prentice Hall, 2002
) These data were edited from file T12-5.DAT on disk from book
) Public Utility data (1975)
) Col. 1: X1 = fixed-charge coverage ratio (income/debt)
) Col. 2: X2 = rate of return on capital
) Col. 3: X3 = cost per KW capacity in place
) Col. 4: X4 = annual load factor
) Col. 5: X5 = peak KWH demand growth from 1974 to 1975
) Col. 6: X6 = sales (KWH use per year)
) Col. 7: X7 = percent nuclear
) Col. 8: X8 = total fuel costs (cents per KWH)
) Col. 9: Utility name (skipped by format)
Read from file "TP1:Stat5401:Data:JWData5.txt"

Cmd> stuff <- cluster(data,nclust:22,method:"average",\
keep:vector("criterion","classes"),reorder:T,class:T)
```

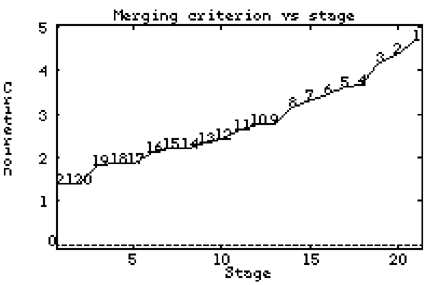
Case Number of Clusters

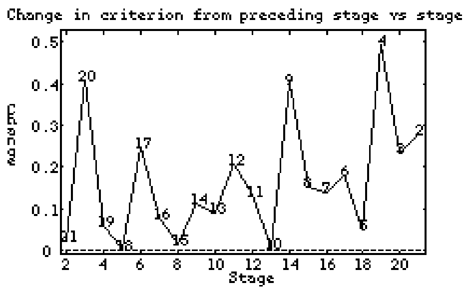
No.	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18	18	18	18	18
14	1	1	1	1	1	1	1	1	1	1	1	1	14	14	14	14	14	14	14	14	14
19	1	1	1	1	1	1	1	1	1	1	1	1	14	14	14	14	14	19	19	19	19
6	1	1	1	1	1	1	1	9	9	9	9	9	9	9	9	9	9	9	9	9	9
3	1	1	1	1	1	1	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	1	1	1	1	1	1	8	8	8	11	11	11	11	11	11	11	11	11	11	11	11
4	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
20	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	20	20	20
10	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	17	17	17	17	17	17
13	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	17	17	17	17	21	21
22	1	1	1	1	6	6	6	6	6	6	12	12	12	12	12	12	12	12	12	12	12
2	1	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
12	1	1	4	4	4	4	4	4	4	4	4	13	13	13	13	13	13	13	13	13	13
21	1	1	4	4	4	4	4	4	4	4	4	13	13	13	13	13	13	13	13	13	22
15	1	1	4	4	4	4	4	4	4	4	4	13	13	15	15	15	15	15	15	15	15
7	1	1	4	4	4	4	4	4	10	10	10	10	10	10	10	10	10	10	10	10	10
17	1	1	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
8	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
16	2	2	2	2	2	2	2	2	2	2	2	2	2	2	16	16	16	16	16	16	16
11	2	2	2	2	2	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7

Lines separate clusters & super clusters

`cluscritplot()` works fine with the value returned by
`cluster(x,keep:vector("class","crit")`

```
Cmd> cluscritplot(stuff,values:T)

Cmd> cluscritplot(stuff,changes:T)
```





This might suggest 9 clusters (separated by light lines) grouped in 4 super clusters (separated by heavy lines). Cluster 3 is a singleton all the way.

```
Cmd> compnames(stuff)
(1) "classes"
(2) "criterion"

Cmd> avelnkclass <- stuff$classes # save class table
```

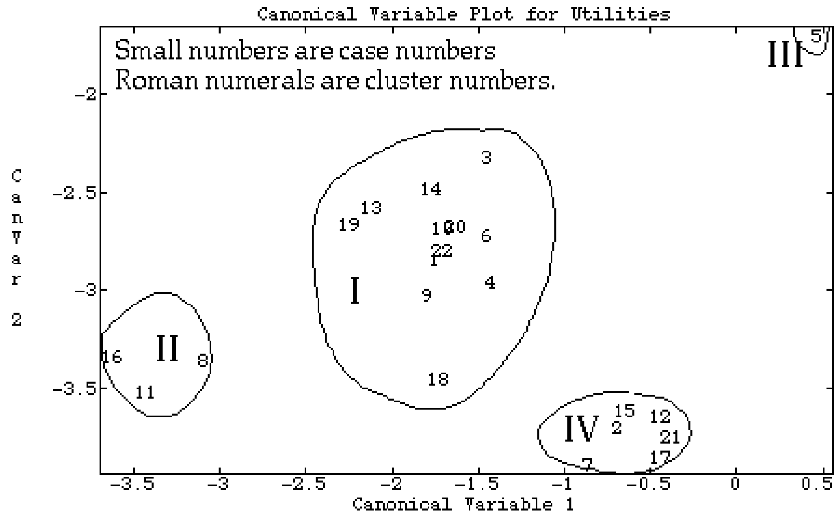

Do MANOVA using 4 clusters as groups. Column 1 of **avelnkclass** (saved class table) goes with 2 clusters, so **avelnkclass[,3]** goes with 4 clusters.

```
Cmd> manova("data = {factor(avelnkclass[,3])}", fstat:T)
Model used is data = {factor(avelnkclass[,3])}
WARNING: summaries are sequential
                    DF          SS          MS          F    P-value
CONSTANT             1
 FCCRatio                   27.306      27.306 1157.88993          0
 Return                     2535.9      2535.9  780.73073          0
 CapCost               6.2227e+05  6.2227e+05  539.63597          0
 LoadFact                    71421       71421 6959.85602          0
 PeakKWH                    231.08      231.08   25.23259 8.8296e-05
 Sales                 1.7481e+09  1.7481e+09  514.42426  1.088e-14
 Nuclear                      3168        3168   11.79440   0.002958
 TotalCost                  26.752      26.752  371.90192 1.8041e-13
{factor(avelnkclass[,3])} 3
 FCCRatio                  0.29044    0.096813    4.10524    0.02202
 Return                     47.284      15.761    4.85243   0.012041
 CapCost                     14875      4958.3    4.29983   0.018749
 LoadFact                   233.23      77.742    7.57580  0.0017585
 PeakKWH                    39.352      13.117    1.43234    0.26623
 Sales                 2.0348e+08  6.7827e+07   19.95971 5.9022e-06
 Nuclear                    1086.5      362.17    1.34834     0.2902
 TotalCost                  5.1993      1.7331   24.09334 1.5892e-06
ERROR1              18
 FCCRatio                  0.42449    0.023583
 Return                     58.467      3.2481
 CapCost                     20756      1153.1
 LoadFact                   184.71      10.262
 PeakKWH                    164.84      9.1579
 Sales                 6.1168e+07  3.3982e+06
 Nuclear                    4834.8       268.6
 TotalCost                  1.2948    0.071933
```

`fstats:T` results in this type of output. It should not be a surprise that F is "significant" for most variables.

Now find MANOVA <u>canonical variables</u> for plotting to try to display clusters.

```
Cmd> eigs <- releigen(SS[2,,],SS[3,,]);eigs$values# 3 non zero
(1)      20.197      6.0436      3.2648  1.0496e-15  1.6711e-16
(6) -1.4996e-16 -4.0731e-16 -1.7041e-15

Cmd> z <- data %*% eigs$vectors[,run(3)] # compute 3 can. vars.

Cmd> chplot(z[,1],z[,2],\
 title:"Canonical Variable Plot for Utilities",\
 xlab:"Canonical Variable 1",ylab:"CanVar 2",xaxis:F,yaxis:F)
```

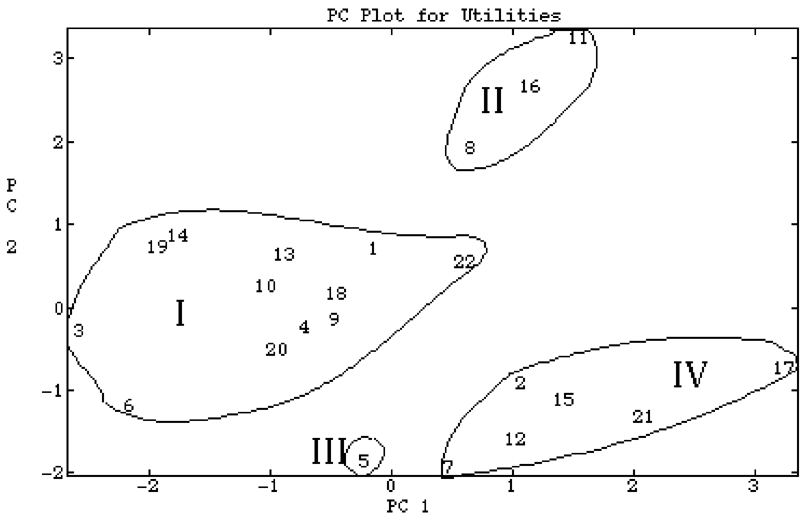



```
Cmd> split(run(nrows(data)), avelnkclass[,3]) # Useful "trick"
component: comp1  Cases in cluster 1
 (1)          1          3          4          6          9
 (6)         10         13         14         18         19
(11)         20         22
component: comp2  Cases in cluster 2
(1)          8         11         16
component: comp3  Case in cluster 3
(1)          5
component: comp4  Cases in cluster 4
(1)          2          7         12         15         17
(6)         21
```


Plotting the first two canonical variables does show the clusters as well separated.

Would that be the case in a *Principal Component plot* of the data?

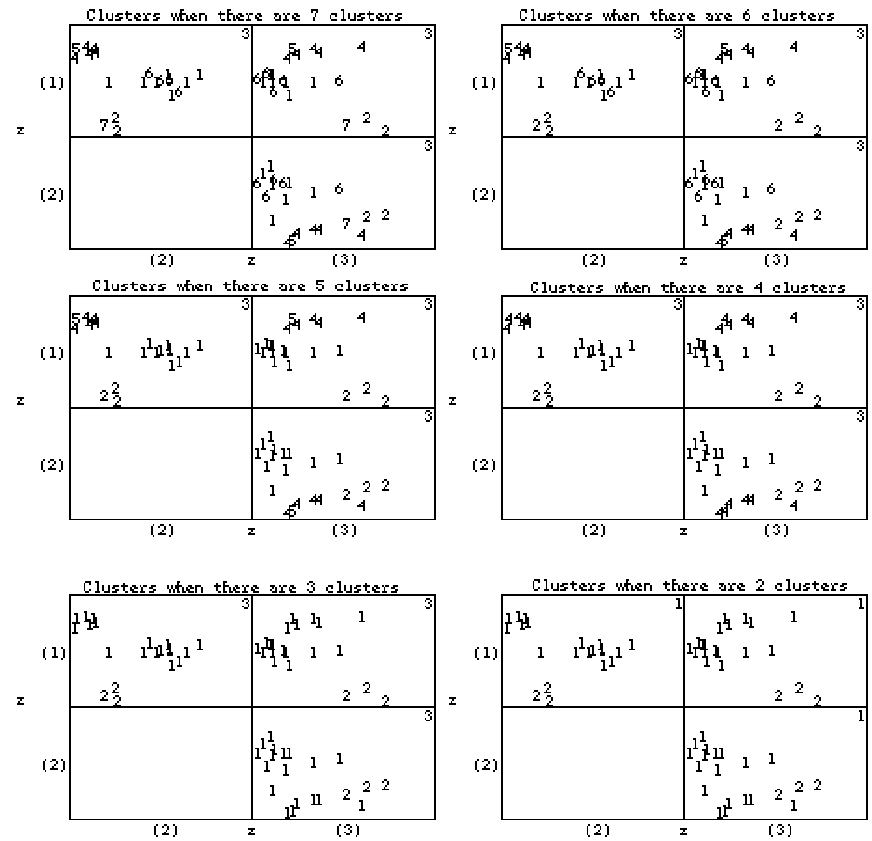
```
Cmd> eigspc <- eigen(cor(data))#eigen vals & vecs of cor matrix

Cmd> eigspc$values # Eigenvalues of correlation matrix
(1)      2.1729      1.9003      1.3235     0.99674     0.64902
(6)     0.57166      0.2165     0.16939

Cmd> zpc <- standardize(data) %*% eigspc$vectors[,run(3)]

Cmd> chplot(zpc[,1],zpc[,2],title:"PC Plot for Utilities",\
    xlab:"PC 1",ylab:"PC 2",xaxis:F,yaxis:F)
```



The clusters we found still consist of somewhat neighboring points, but the clustering is less apparent.

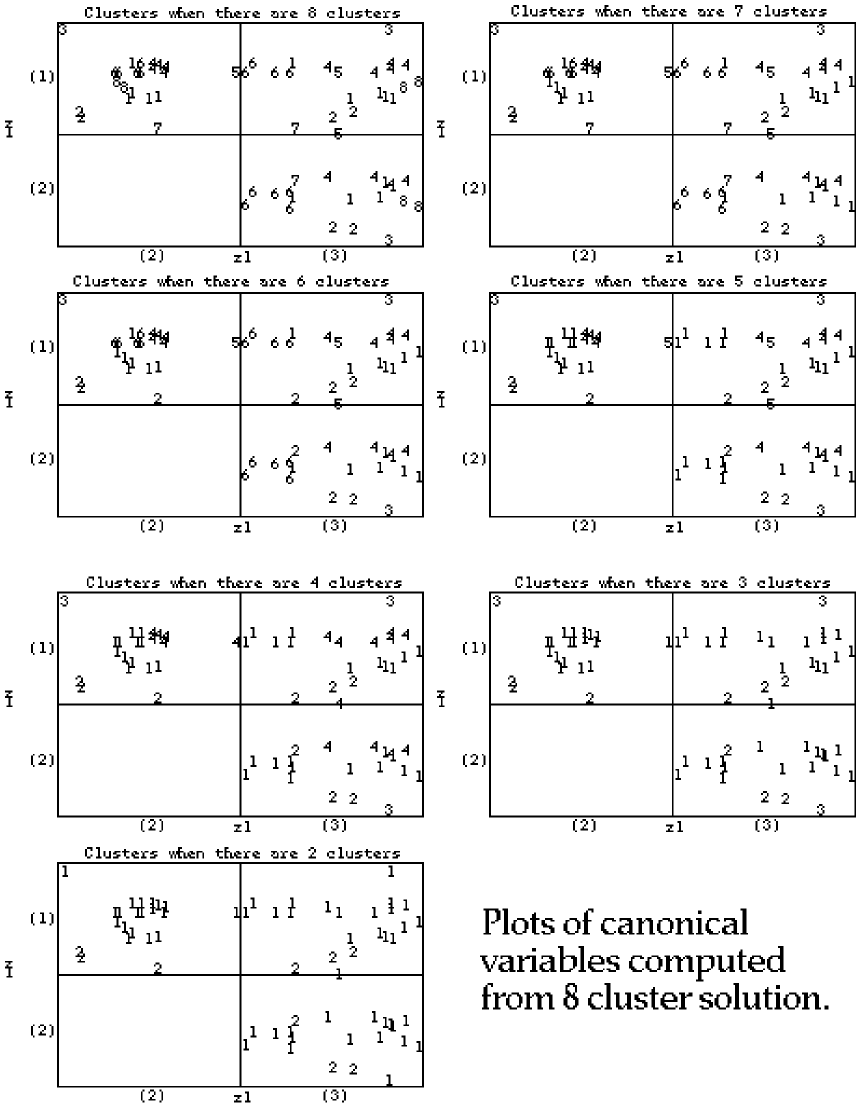
Canonical variable plots for 7, 6, 5, 4, 3, and 2 average linkage clusters using z.

```
Cmd> for(j,1,6){plotmatrix(z,upper:T,xaxis:F,yaxis:F,\
        symbols:avelnkclass[,7-j],\
  title:paste("Clusters when there are",8-j,"clusters"),wind:j)}
```



There isn't much visual evidence of 9 clusters grouped in 4, at least not in these dimensions computed from g=4.

```
Cmd> eigs <- releigen(SS[2,,],SS[3,,]);eigs$values#
(1)      68.346      17.222      7.8351      5.1326      2.4588
(6)      1.7082     0.56552     0.11127
```



Plots of canonical variables computed from 8 cluster solution.