CHAPTER 40

MULTIVARIATE STATISTICS

There are some cases (few, but some) where it is forgivable not to have a response variable. Instead, you have lots of variables and you want to do condense the information they contain in one way or another. The general idea is to *find structure in the data*. The problem is that structure is rather easy to find, and all too often it is a feature of that particular data set alone. The real challenge is to find *general* structure that will apply to other data sets as well.

The data frame consists of a matrix in which the columns are variables and the rows are cases. The multivariate techniques implemented in S-Plus are

- principal components analysis
- factor analysis
- discriminant analysis
- cluster analysis
- neural networks

These techniques are *not* recommended unless you know exactly what you are doing, and exactly *why* you are doing it. Beginners are sometimes attracted to multivariate techniques because of the complexity of the output they produce, making the classic mistake of confusing the opaque for the profound.

The main division is between methods that *assume a given structure* and seek to divide the cases into groups, and methods that seek to *discover structure* from inspection of the data frame.

Principal Components Analysis (PCA)

The idea is to find a small number of *linear combinations of the variables* that captures most of the variation in the data frame as a whole. These linear combinations are called the principal components. The **princomp** function takes numeric vectors (*not* factors with levels designated by characters) and produces the numbers of standard deviations

attributable to the ranked components (one for each variable). Summary prints the proportion of the variance explained by each component.

```
attach(multivariate)
names(multivariate)
```

```
[1] "Temp"        "Industry"    "Population" "Wind"
[5] "Rain"        "Wet.days"
```

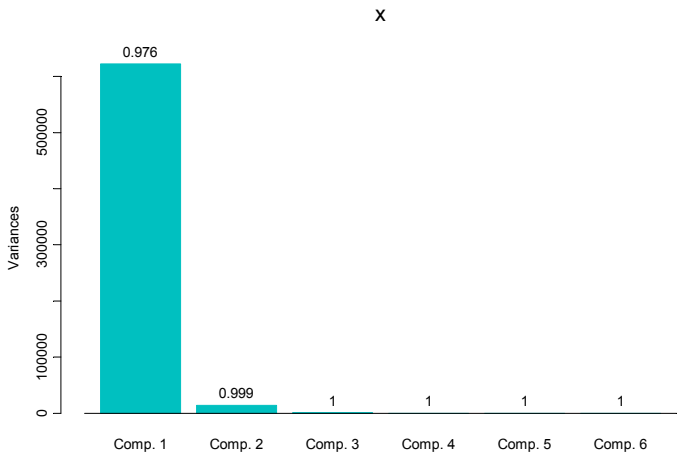
The PCA is carried out like this:

```
model<-princomp(multivariate)
summary(model)
```

```
Importance of components:
                          Comp. 1      Comp. 2      Comp. 3       Comp. 4       Comp. 5       Comp. 6
    Standard deviation 788.9652490 119.33045669 25.777597242 10.7941192696 3.51208857342 1.242269e+000
Proportion of Variance   0.9764163   0.02233685  0.001042327  0.0001827653 0.00001934863 2.420754e-006
 Cumulative Proportion   0.9764163   0.99875314  0.999795465  0.9999782306 0.99999757925 1.000000e+000
```

With PCA **plot** produces the following graph:

```
plot(model)
```



This shows that the first component explains more than 97% of the variance. It is good practice to base the calculations on the correlation matrix rather than the covariance matrix when (as here) the measurements are of different types.

```
model<-princomp(multivariate,cor=T)
```

Most of the interest concerns the loadings of the different variables within each component. These can be inspected using the **loadings=T** option of **summary**

```
summary(model,loadings=T)
```

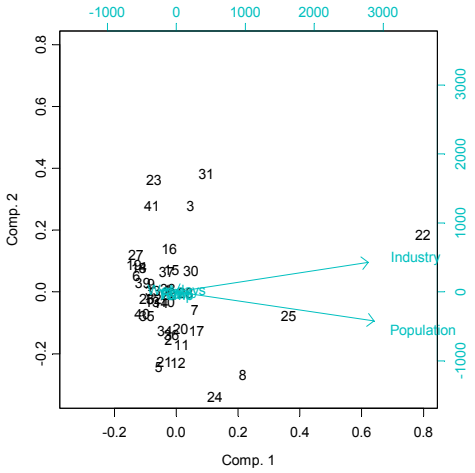
```
Importance of components:
                          Comp. 1   Comp. 2   Comp. 3   Comp. 4    Comp. 5     Comp. 6
    Standard deviation 1.4815042 1.2250751 1.1818840 0.8707642 0.33873542 0.185780560
Proportion of Variance 0.3658091 0.2501348 0.2328083 0.1263717 0.01912361 0.005752403
 Cumulative Proportion 0.3658091 0.6159440 0.8487523 0.9751240 0.99424760 1.000000000
```

Note that the change to the correlation matrix has made an enormous difference to the components. Now the first component only explains 36.6% of the variance (rather than 97%).

```
Loadings:
           Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5 Comp. 6
      Temp -0.329  -0.136  -0.669   0.308  -0.558   0.136
  Industry  0.611  -0.174  -0.271  -0.136   0.103   0.703
Population  0.578  -0.229  -0.346                  -0.695
      Wind  0.354   0.137   0.297   0.869  -0.113
      Rain          0.616  -0.514   0.170   0.568
  Wet.days  0.239   0.708          -0.313  -0.580
```

The loadings on the first component show that Industry is most important (0.611), followed by Population (0.578) with Temperature (-0.329) and Wind (0.354) in the next category of importance. Wet days is less important, and Rain does not figure in this component at all. However, both Rain (0.616) and Wet.days (0.708) figure prominently in the *second* component. We can view the first two components using **biplot** like this:

```
biplot(model)
```



This shows a tight cloud of data around the point (0,0) with an unusual point (#22) at the extreme of component 1 and a group of 4 points (#23, #31, #41 and #3) with high values of component 2. Perhaps the 2 points (#24 and #8) with large negative values on

component 2 deserve some special attention? The arrows for Industry and Population are long (showing their importance within component 1 as we saw from the loadings). The arrows for the other variables are in a tangled blur close to the origin. The directions of the arrows show that Industry was associated with positive values of component two, while population was associated with negative values, as judged by the relative slopes of the two arrows.

The overall interpretation is that a composite industry/population component is most important, followed by a composite precipitation component (formed of Rain and Wet.days). Remember that this model has no response variable. Compare this with the multiple regression model of the same data on p. xxx and the tree regression on p. xxx.

Factor analysis

Factor analysis attempts to explain correlations between variables in terms of underlying factors that are not themselves directly measurable (e.g. intelligence, social status, ecosystem stability). Compared with PCA, the variables themselves are of relatively little interest in factor analysis; it is an understanding of the underlying factors that is the main aim. The **factanal** function does the business (unfortunate name, don't you think?):

```
model<-factanal(multivariate,2)
```

The number 2 indicates the number of factors we want to estimate

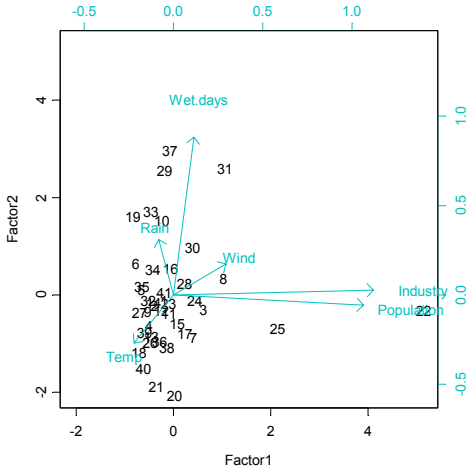
```
summary(model)
```

```
Importance of factors:
                  Factor1   Factor2
   SS loadings 1.9944022 1.2393427
Proportion Var 0.3324004 0.2065571
Cumulative Var 0.3324004 0.5389575

The degrees of freedom for the model is 4.

Uniquenesses:

      Temp   Industry Population      Wind      Rain    Wet.days
 0.8713365 0.01453403  0.1072747 0.8940372 0.8742299 0.004842805

Loadings:
           Factor1 Factor2
      Temp -0.195  -0.301
  Industry  0.992
Population  0.943
      Wind  0.260   0.195
      Rain          0.347
  Wet.days  0.101   0.992
```

Interpretation is the same as with PCA. The first factor is loaded almost equally on Industry and Population. The second factor is again precipitation, but Wet.days is loaded much more heavily than Rain. The two factors only account for 54% of the variance between them.

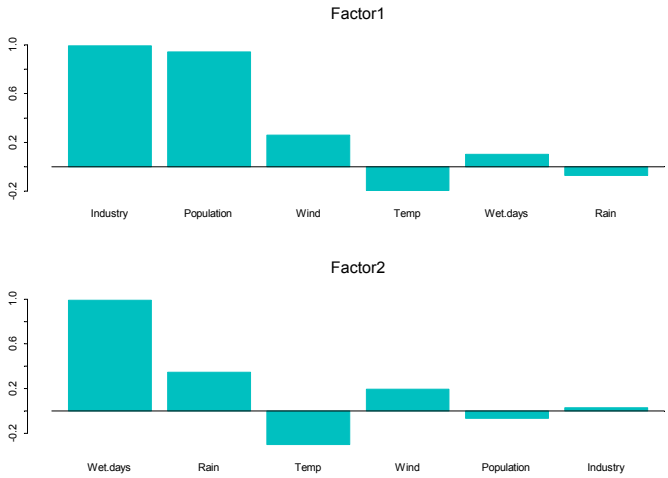
The biplot shows rather more detail than it did with PCA:

```
biplot(model)
```



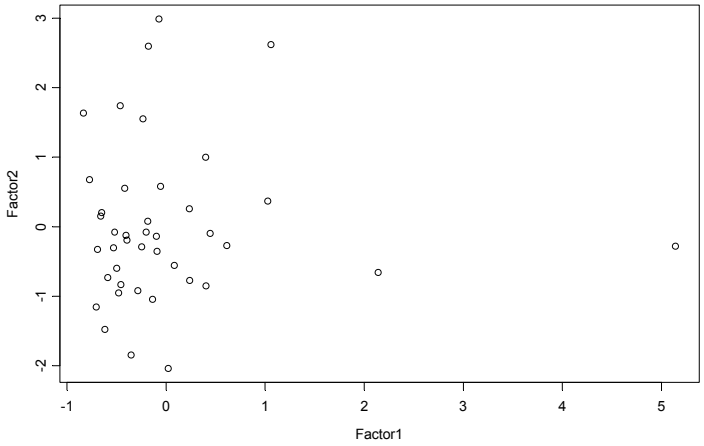
The role of Wet.days on factor 2 is prominent, and an effect of Temperature associated with negative values of factor 2 has appeared. As before, the effects of Industry and Population are both similar and important. The groups of outlying points are rather different. Point #22 is still the odd one out on factor 1, but two new points join #31 for positive values of factor 2 (#37 and #39) and there are two different outliers at the negative end of factor 2 (#20 and #21). To see the relative importance of the different variables on the two factors we plot the loadings like this:

```
plot(loadings(model))
```



You might want to save the factor scores for use in subsequent modelling (once a response variable has been unearthed):

```
fscores<-model$scores
plot(fscores)
```



As you can see, the two factor scores are independent of one another.

Discriminant analysis

In taxonomy, one often has a set of quantitative observations of several variables for a number of individuals. The questions are these:

- how many groups are there?
- which individuals belong to which groups ?

In the general case, the groups have independent covariance matrices and this leads to a heteroscedastic model with a quadratic discriminant function of the form:

$$d(x) = \beta_{i0} + \beta_{i1}x + x^T\beta_{i2}x$$

Relationships of the explanatory variables to the groupings are expressed by their mean values and their variance-covariance matrices. Model simplification involves the use of similarities in variance-covariance matrices of different groups to reduce the number of estimated parameters.

Cluster analysis

Cluster analysis is a set of techniques that looks for groups (clusters) in the data. Objects belonging to the same group resemble each other. Objects belonging to different groups are dissimilar. Sounds simple, doesn't it ? The problem is that there is usually a huge amount of redundancy in the explanatory variables. It is not obvious which measurements (or combinations of measurements) will turn out to be the ones that are best for allocating individuals to groups. There are three ways of doing it

- partitioning into a number of clusters specified by the user, with functions like **kmeans**
- hierarchincal, starting with each individual as a separate entity and ending up with a single aggregation, using functions like **hclust**
- divisive, starting with a single aggregate of all the individuals and splitting up clusters until all the individuals are in different groups (like an inverse of hierarchical)

Partitioning

The function **kmeans** operates on a data frame in which the columns are variables and the rows are the individuals. Group membership is determined by calculating the centroid for each group. This is the multidimensional equivalent of the mean. Each individual is assigned to the group with the nearest centroid.

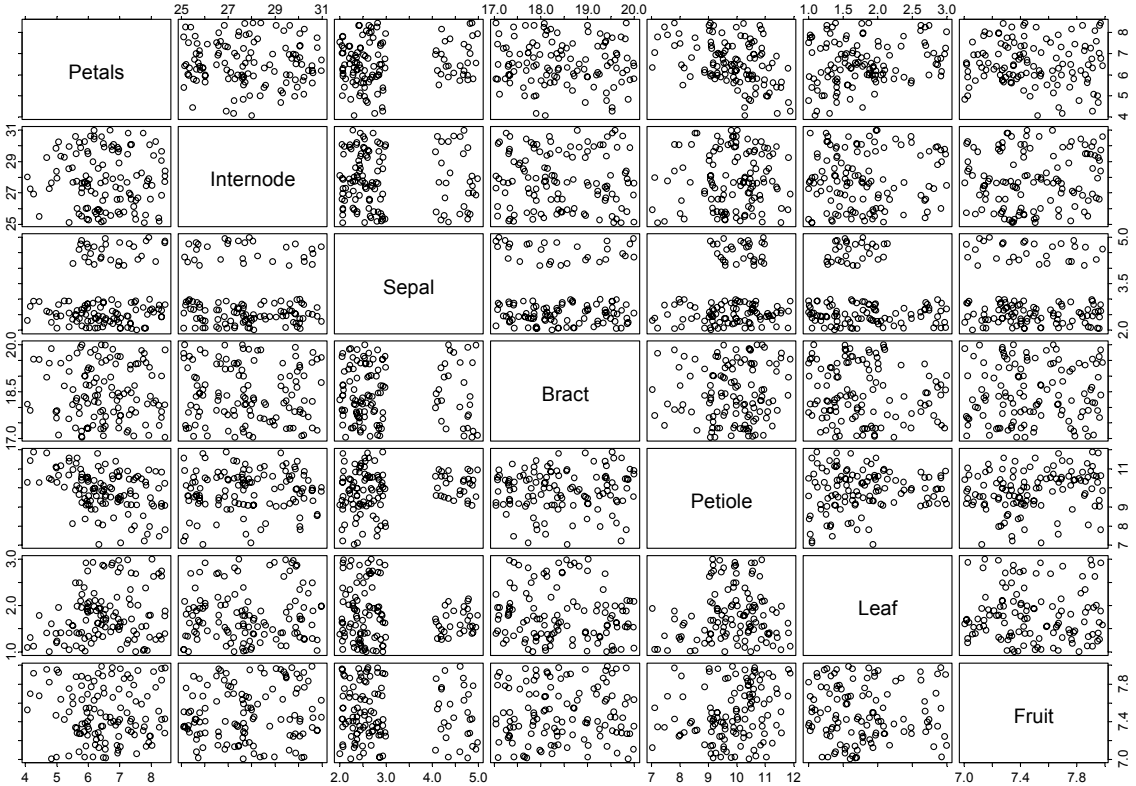
In this example we have measurements of 7 variables on 120 individual plants. The question is which of the variables (fruit size, bract length, internode length, petal width, sepal length, petiole length or leaf width) are the most useful taxonomic characters?

```
attach(taxa)
names(taxa)
```

```
[1] "Petals"    "Internode" "Sepal"     "Bract"
[5] "Petiole"   "Leaf"      "Fruit"
```

A simple and sensible way to start is by looking at the data frame as a whole, using **pairs** to plot every variable against every other.

```
pairs(taxa)
```



There appears to be excellent data separation on sepal length, and reasonable separation on petiole length and leaf width, but nothing obvious for the other variables.

These data actually come from four Taxa (labelled I – IV), so in this contrived case we *know* that there are 4 groups. In reality, of course, we would not know this, and finding out the number of groups would be one of the central aims of the study. We begin, therefore, by seeing how well **kmeans** allocates individuals to 4 groups:

```
kmeans(taxa,4)
```

```
Centers:
       Petals Internode    Sepal    Bract   Petiole
[1,] 6.746014  29.99127 3.079527 18.30375  9.717186
[2,] 6.807612  26.20328 3.634420 18.39943 10.218505
[3,] 5.410618  28.17308 2.584850 18.64541 10.846860
[4,] 6.907184  27.02631 2.408701 18.49753  8.594522
         Leaf    Fruit
[1,] 2.036219 7.514235
[2,] 1.853898 7.463587
[3,] 1.492111 7.544768
[4,] 1.692353 7.425414

Clustering vector:
  [1] 3 3 3 2 2 2 3 3 3 2 3 2 3 3 3 3 3 3 3 3 1 3 2 3 3
```

```
 [26] 1 3 3 1 3 4 4 1 1 4 4 4 2 4 4 1 1 4 4 2 1 4 4 4 4
 [51] 4 2 4 4 1 4 1 4 1 4 1 2 4 2 1 1 2 4 2 1 1 2 1 4 4
 [76] 4 2 1 1 1 4 3 2 4 1 1 1 2 3 1 2 1 2 1 2 2 1 3 2 2
[101] 1 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 1 2 1 1

Within cluster sum of squares:
[1] 125.89857 162.47927  63.66186  90.72276
```

Because we *know* what the answer ought to be (the data are arranged so that the first 30 numbers are Taxon I, the next 30 are Taxon II, and so on) we can see that **kmeans** has made lots of mistakes. The output of the clustering vector should really look like this:

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [26] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [51] 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [76] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
[101] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
```

Let's try 3 groups:

kmeans(taxa,3)

Centers:

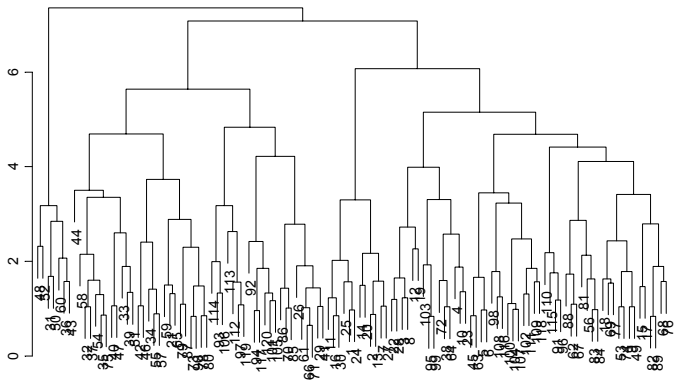
```
       Petals Internode    Sepal    Bract   Petiole
[1,] 6.187045  26.74649 3.398264 18.59997 10.489409
[2,] 6.456317  29.82515 2.965299 18.33530  9.931997
[3,] 7.091774  26.74340 2.520430 18.37322  8.891600
         Leaf    Fruit
[1,] 1.704769 7.477971
[2,] 1.900160 7.539722
[3,] 1.789907 7.426350

Clustering vector:
  [1] 2 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 2 1 1 1 2 1 1 2 2
 [26] 2 1 1 2 2 3 3 2 2 3 3 3 3 3 3 2 2 3 3 3 2 3 3 3 3
 [51] 3 3 3 3 2 3 2 2 2 3 2 3 3 1 2 2 3 3 1 2 2 1 2 3 3
 [76] 3 1 2 2 2 3 1 3 3 2 2 2 3 1 2 1 2 1 2 1 1 2 1 1 1
[101] 2 1 1 1 2 1 1 1 1 1 1 2 2 3 1 1 2 1 2 2

Within cluster sum of squares:
[1] 211.8718 175.1567 128.7676
```

Not very impressive at all. Let's try a different method.

Hierarchical clustering

The first step is to turn the matrix of measurements on individuals into a dissimilarity matrix. In the data frame, the columns are variables and the rows are the individuals.

```
h<-hclust(dist(as.matrix(taxa)))
plclust(h)
```



We can edit this down in various ways: to 4 groups, for instance

```
cutree(h,k=4)
```

```
  [1] 3 3 2 2 2 2 3 2 2 2 3 2 3 3 2 3 2 2 2 3 1 2 2 3 3
 [26] 1 3 2 1 3 4 1 1 1 1 4 1 2 1 1 1 1 4 1 2 1 1 4 2 4
 [51] 1 4 2 1 1 2 1 1 1 4 1 2 2 2 1 1 2 2 2 1 1 2 1 2 1
 [76] 2 2 1 1 1 2 2 2 2 1 1 1 2 2 1 2 1 1 1 2 2 1 2 2 2
[101] 1 2 2 2 1 1 2 2 2 2 2 1 1 1 2 2 1 2 1 1
```

or to 3 groups

```
cutree(h,k=3)
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
 [26] 2 1 1 2 1 3 2 2 2 2 3 2 1 2 2 2 2 3 2 1 2 2 3 1 3
 [51] 2 3 1 2 2 1 2 2 2 3 2 1 1 1 2 2 1 1 1 2 2 1 2 1 2
 [76] 1 1 2 2 2 1 1 1 1 2 2 2 1 1 2 1 2 2 2 1 1 2 1 1 1
[101] 2 1 1 1 2 2 1 1 1 1 1 2 2 2 1 1 2 1 2 2
```

The computer was doing its classification blind. But we know the 4 islands from which the plants were collected. Can we write a key that ascribes taxa to islands better than the blind schemes adopted by the computer's multivariate techniques? The answer is definitely yes. When we used a **classification tree model** on the same data, it discovered a faultless key on its own (see p. xxx).

Neural Networks

These are computationally intensive methods for finding pattern in data sets that are so large, and contain so many explanatory variables, that standard methods like multiple

regression are impractical (they would simply take too long to plough through). The key feature of neural network models is that they contain a *hidden layer*: each node in the hidden layer receives information from each of many inputs, sums the inputs, adds a constant (the bias) then transforms the result using a fixed function. Neural networks can operate like multiple regressions when the outputs are continuous variables, or like classifications when the outputs are categorical. They are described in detail by Ripley, B.D. (1996) *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge. There are libraries for analysing neural networks in Venables and Ripley's S-Plus library called MASS.

Further reading

Bishop, Y. M. M., S. J. Fienberg, et al. (1980). Discrete Multivriate Analysis: Theory and Practice. New York, John Wiley.

Harman, H. H. (1976). Modern Factor Analysis. Chicago, University of Chicago Press.

Mardia, K. V., J. T. Kent, et al. (1979). Multivariate Statistics. London, Academic Press.