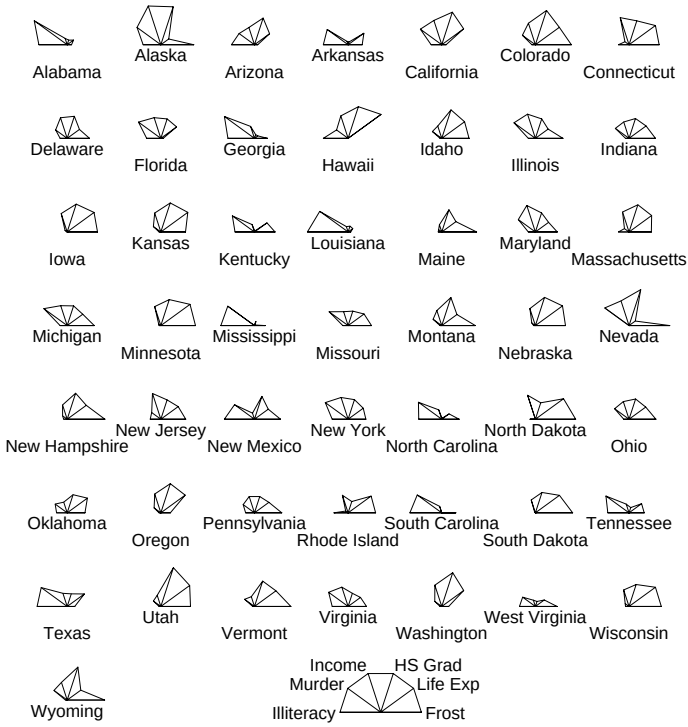

Figure 11.9: R version of `stars` plot of the `state.x77` dataset.

Such plots are often too 'busy' without a means of interaction to identify observations, sign-change and reorder variables, brush groups and so on (as is possible in XGobi and GGobi). As an example of a revealing parallel coordinate plot try

```
parcoord(log(ir)[, c(3, 4, 2, 1)], col = 1 + (0:149)%/%50)
```

on a device which can plot colour.

11.2 Cluster Analysis

Cluster analysis is concerned with discovering groupings among the cases of our n by p matrix. A comprehensive general reference is Gordon (1999); Kaufman and Rousseeuw (1990) give a good introduction and their methods are available in S-PLUS and in package `cluster` for R. Clustering methods can be clustered in many different ways; here is one.

- Agglomerative hierarchical methods (`hclust`, `agnes`, `mclust`).
  - Produces a set of clusterings, usually one with k clusters for each $k = n, \ldots, 2$, successively amalgamating groups.
  - Main differences are in calculating group–group dissimilarities from point–point dissimilarities.

  - Computationally easy.

- Optimal partitioning methods (`kmeans`, `pam`, `clara`, `fanny`).
  - Produces a clustering for fixed K.
  - Need an initial clustering.
  - Lots of different criteria to optimize, some based on probability models.
  - Can have distinct 'outlier' group(s).

- Divisive hierarchical methods (`diana`, `mona`).
  - Produces a set of clusterings, usually one for each $k = 2, \ldots, K \ll n$.
  - Computationally nigh-impossible to find optimal divisions (Gordon, 1999, p. 90).
  - Most available methods are *monothetic* (split on one variable at each stage).

Do not assume that 'clustering' methods are the best way to discover interesting groupings in the data; in our experience the visualization methods are often far more effective. There are many different clustering methods, often giving different answers, and so the danger of over-interpretation is high.

Many methods are based on a measure of the similarity or dissimilarity between cases, but some need the data matrix itself. A *dissimilarity coefficient* d is symmetric ($d(A, B) = d(B, A)$), non-negative and $d(A, A)$ is zero. A similarity coefficient has the scale reversed. Dissimilarities may be *metric*

$$d(A, C) \leqslant d(A, B) + d(B, C)$$

or *ultrametric*

$$d(A, B) \leqslant \max\big(d(A, C), d(B, C)\big)$$

but need not be either. We have already seen several dissimilarities calculated by `dist` and `daisy`.

Ultrametric dissimilarities have the appealing property that they can be represented by a *dendrogram* such as those shown in Figure 11.10, in which the dissimilarity between two cases can be read from the height at which they join a single group. Hierarchical clustering methods can be thought of as approximating a dissimilarity by an ultrametric dissimilarity. Jardine and Sibson (1971) argue that one method, single-link clustering, uniquely has all the desirable properties of a clustering method. This measures distances between clusters by the dissimilarity of the closest pair, and agglomerates by adding the shortest possible link (that is, joining the two closest clusters). Other authors disagree, and Kaufman and Rousseeuw (1990, §5.2) give a different set of desirable properties leading uniquely to their preferred method, which views the dissimilarity between clusters as the average of the dissimilarities between members of those clusters. Another popular method is complete-linkage, which views the dissimilarity between clusters as the maximum of the dissimilarities between members.

The function `hclust` implements these three choices, selected by its `method` argument which takes values `"compact"` (the default, for complete-linkage,

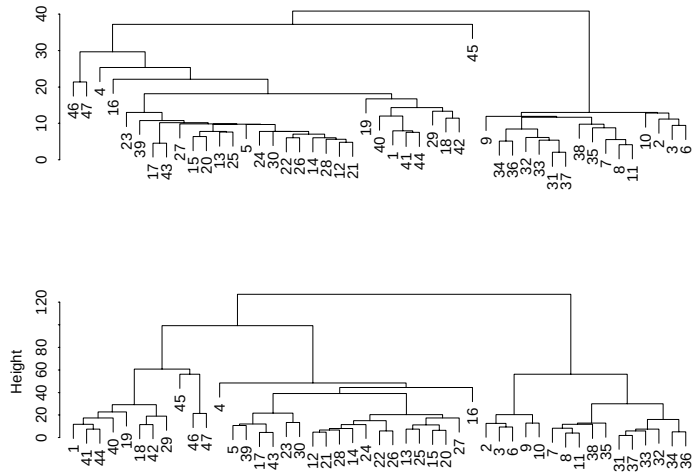

Figure 11.10: Dendrograms for the socio-economic data on Swiss provinces computed by single-link clustering (top) and divisive clustering (bottom).

R called `"complete"` in R), `"average"` and `"connected"` (for single-linkage, called `"single"` in R). Function `agnes` also has these (with the R names) and others.

The S dataset[12] `swiss.x` gives five measures of socio-economic data on Swiss provinces about 1888, given by Mosteller and Tukey (1977, pp. 549–551). The data are percentages, so Euclidean distance is a reasonable choice. We use single-link clustering:

```
# S: h <- hclust(dist(swiss.x), method = "connected")
# R: data(swiss); swiss.x <- as.matrix(swiss[, -1])
# R: h <- hclust(dist(swiss.x), method = "single")
plclust(h)
cutree(h, 3)
# S: plclust( clorder(h, cutree(h, 3) ))
```

The hierarchy of clusters in a dendrogram is obtained by cutting it at different heights. The first plot suggests three main clusters, and the remaining code reorders the dendrogram to display (see Figure 11.10) those clusters more clearly. Note that there appear to be two main groups, with the point 45 well separated from them.

Function `diana` performs *divisive* clustering, in which the clusters are repeatedly subdivided rather than joined, using the algorithm of Macnaughton-Smith *et al.* (1964). Divisive clustering is an attractive option when a grouping into a few large clusters is of interest. The lower panel of Figure 11.10 was produced by `pltree(diana(swiss.x))`.

[12] In R the numbers are slightly different, and the provinces has been given names.

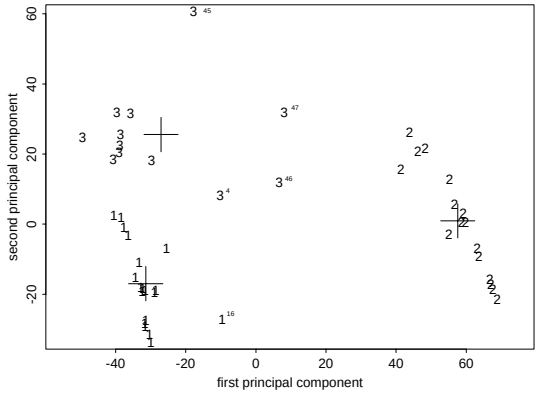

Figure 11.11: The Swiss provinces data plotted on its first two principal components. The labels are the groups assigned by K-means; the crosses denote the group means. Five points are labelled with smaller symbols.

Partitioning methods

The K-means clustering algorithm (MacQueen, 1967; Hartigan, 1975; Hartigan and Wong, 1979) chooses a pre-specified number of cluster centres to minimize the within-class sum of squares from those centres. As such it is most appropriate to continuous variables, suitably scaled. The algorithm needs a starting point, so we choose the means of the clusters identified by group-average clustering. The clusters *are* altered (cluster 3 contained just point 45), and are shown in principal-component space in Figure 11.11. (Its standard deviations show that a two-dimensional representation is reasonable.)

```
h <- hclust(dist(swiss.x), method = "average")
initial <- tapply(swiss.x, list(rep(cutree(h, 3),
   ncol(swiss.x)), col(swiss.x)), mean)
dimnames(initial) <- list(NULL, dimnames(swiss.x)[[2]])
km <- kmeans(swiss.x, initial)
(swiss.pca <- princomp(swiss.x))
Standard deviations:
  Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
  42.903 21.202  7.588 3.6879 2.7211
   ....
swiss.px <- predict(swiss.pca)
dimnames(km$centers)[[2]] <- dimnames(swiss.x)[[2]]
swiss.centers <- predict(swiss.pca, km$centers)
eqscplot(swiss.px[, 1:2], type = "n",
         xlab = "first principal component",
         ylab = "second principal component")
text(swiss.px[, 1:2], labels = km$cluster)
points(swiss.centers[,1:2], pch = 3, cex = 3)
identify(swiss.px[, 1:2], cex = 0.5)
```

By definition, K-means clustering needs access to the data matrix and uses Euclidean distance. We can apply a similar method using only dissimilarities if we confine the cluster centres to the set of given examples. This is known as the k-medoids criterion (of Vinod, 1969) implemented in `pam` and `clara`. Using `pam` picks provinces 29, 8 and 28 as cluster centres.

```
> library(cluster)          # needed in R only
> swiss.pam <- pam(swiss.px, 3)
> summary(swiss.pam)
Medoids:
     Comp. 1   Comp. 2 Comp. 3 Comp. 4  Comp. 5
[1,] -29.716  18.22162  1.4265 -1.3206  0.95201
[2,]  58.609   0.56211  2.2320 -4.1778  4.22828
[3,] -28.844 -19.54901  3.1506  2.3870 -2.46842
Clustering vector:
 [1] 1 2 2 1 3 2 2 2 2 2 2 3 3 3 3 3 1 1 1 3 3 3 3 3 3 3 3 3
[29] 1 3 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
    ....
> eqscplot(swiss.px[, 1:2], type = "n",
           xlab = "first principal component",
           ylab = "second principal component")
> text(swiss.px[,1:2], labels = swiss.pam$clustering)
> points(swiss.pam$medoid[,1:2], pch = 3, cex = 5)
```

The function `fanny` implements a 'fuzzy' version of the k-medoids criterion. Rather than point i having a membership of just one cluster v, its membership is partitioned among clusters as positive weights u_{iv} summing to one. The criterion then is

$$\min_{(u_{iv})} \sum_v \frac{\sum_{i,j} u_{iv}^2 u_{jv}^2 d_{ij}}{2 \sum_i u_{iv}^2}.$$

For our running example we find

```
> fanny(swiss.px, 3)
 iterations objective
         16    354.01
Membership coefficients:
          [,1]     [,2]     [,3]
 [1,] 0.725016 0.075485 0.199499
 [2,] 0.189978 0.643928 0.166094
 [3,] 0.191282 0.643596 0.165123
    ....
Closest hard clustering:
 [1] 1 2 2 1 3 2 2 2 2 2 2 3 3 3 3 3 1 1 1 3 3 3 3 3 3 3 3 3
[29] 1 3 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
```

The 'hard' clustering is formed by assigning each point to the cluster for which its membership coefficient is highest.

Other partitioning methods are based on the idea that the data are independent samples from a series of group populations, but the group labels have been lost, so the data can be regarded as from a mixture distribution. The idea is then to find the

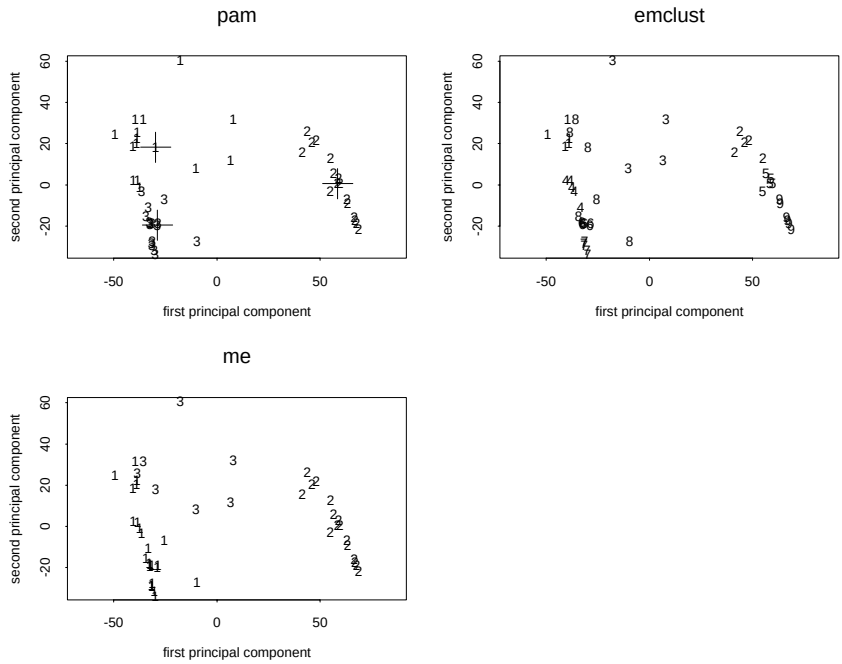

Figure 11.12: Clusterings of the Swiss provinces data by `pam` with three clusters (with the medoids marked by crosses), `me` with three clusters and `emclust` with up to nine clusters (it chose nine).

mixture distribution, usually as a mixture of multivariate normals, and to assign points to the component for which their posterior probability of membership is highest.

S+ S-PLUS has functions `mclust`, `mclass` and `mreloc` based on 'maximum-likelihood' clustering in which the mixture parameters and the classification are optimized simultaneously. Later work in the `mclust` library section[13] uses sounder methods in which the mixtures are fitted first. Nevertheless, fitting normal mixtures is a difficult problem, and the results obtained are often heavily dependent on the initial configuration supplied.

K-means clustering can be seen as 'maximum-likelihood' clustering where the clusters are assumed all to be spherically symmetric multivariate normals with the same spread. The `modelid` argument to the `mclust` functions allows a wider choice of normal mixture components, including `"EI"` (equal spherical) `"VI"` (spherical, differing by component), `"EEE"` (same elliptical), `"VEV"` (same shape elliptical, variable size and orientation) and `"VVV"` (arbitrary components).

Library section `mclust` provides hierarchical clustering via functions `mhtree` and `mhclass`. Then for a given number k of clusters the fitted mixture can be optimized by calling `me` (which here does not change the classification).

[13] Available at `http://www.stat.washington.edu/fraley/mclust/` and for R from CRAN.

```
library(mclust)
h <- mhtree(swiss.x, modelid = "VVV")
(mh <- as.vector(mhclass(h, 3)))
 [1] 1 2 2 3 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[29] 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 3 3 3
z <- me(swiss.x, modelid = "VVV", z = (ctoz(mh)+1/3)/2)
eqscplot(swiss.px[, 1:2], type = "n",
         xlab = "first principal component",
         ylab = "second principal component")
text(swiss.px[, 1:2], labels = max.col(z))
```

Function `mstep` can be used to extract the fitted components, and `mixproj` to plot them, but unfortunately only on a pair of the original variables.

Function `emclust` automates the whole cluster process, including choosing the number of clusters and between different `modelid`'s. One additional possibility controlled by argument `noise` is to include a background 'noise' term, that is a component that is a uniform Poisson process. It chooses lots of clusters (see Figure 11.12).

```
> vals <- emclust(swiss.x) # all possible models, 0:9 clusters.
> sm <- summary(vals, swiss.x)
> eqscplot(swiss.px[, 1:2], type = "n",
           xlab = "first principal component",
           ylab = "second principal component")
> text(swiss.px[, 1:2], labels = sm$classification)
```

11.3 Factor Analysis

Principal component analysis looks for linear combinations of the data matrix X that are uncorrelated and of high variance. Independent component analysis seeks linear combinations that are independent. *Factor analysis* seeks linear combinations of the variables, called *factors*, that represent underlying fundamental quantities of which the observed variables are expressions. The examples tend to be controversial ones such as 'intelligence' and 'social deprivation', the idea being that a small number of factors might explain a large number of measurements in an observational study. Such factors are to be inferred from the data.

We can think of both the factors of factor analysis and the signals of independent component analysis as *latent variables*, unobserved variables on each experimental unit that determine the patterns in the observations. The difference is that it is not the factors that are assumed to be independent, but rather the observations conditional on the factors.

The factor analysis model for a single common factor f is

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\lambda} f + \boldsymbol{u} \tag{11.1}$$

where $\boldsymbol{\lambda}$ is a vector known as the *loadings* and $\boldsymbol{u}$ is a vector of *unique* (or *specific*) factors for that observational unit. To help make the model identifiable, we