Data Science for Quantitative Psychology and Economics

John M. Noble,
Mathematical Statistics,
Institute of Applied Mathematics and Mechanics,
Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw,
ul. Banacha 2,
02-097 Warszawa, Poland

Contents

Chapter 1

Geometrical Representation and Distances

1.1 The Data Matrix

Consider p variables, and a *random sample* $\underline{x}_1, \ldots, \underline{x}_n$, where $\underline{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jp})'$. Each observation is a p vector, and there are n observations. A *random sample* means that $\underline{x}_1, \ldots, \underline{x}_n$ is an observation of $\underline{X}_1, \ldots, \underline{X}_n$, where the $(\underline{X}_j)_{j=1}^n$ are independent, identically distributed random p-vectors.

Notation A random p-vector, where each component corresponds to a different variable, is usually taken as a *column* vector, but when presented in a data matrix of n independent observations, the transpose is taken and each p-variate observation is taken as a *row*.

Sampling If the observations were selected from a total population of N p-vectors, then a *random sample* would mean that any subset of n vectors from N was chosen with probability $\frac{1}{\binom{n}{N}}$ and each ordering of the n vectors occurred with probability $\frac{1}{n!}$. In general, a *random sample* is a sample that has the properties of such a sample for $N >> n$.

The most widely used standard is to store the data in an $n \times p$ matrix, denoted $\mathbf{x}$, where

$$\mathbf{x} = \begin{pmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \ldots & x_{np} \end{pmatrix} = \begin{pmatrix} \underline{x}_1^t \\ \underline{x}_2^t \\ \vdots \\ \underline{x}_n^t \end{pmatrix}. \tag{1.1}$$

1.2 One Way Representations of Data Matrices: Andrews Curves

When considering a one way representation of a two dimensional data matrix, one can represent either the n units, or the p variables. Each variable, may be represented by an appropriate curve or solid pattern that highlights the similarities or disssimilarities between the constructions.

One example is the method of *Andrews Curves*. For each unit (or p-variate observation) i of the data matrix, set

$$f_i(t) = \frac{1}{\sqrt{2}} x_{i1} + \sum_{j=1}^{[p/2]} x_{i,2j} \sin(jt) + \sum_{j=1}^{[p/2]} x_{i,2j+1} \cos(jt) \quad t \in [-\pi, \pi].$$

Properties The Andrews curve satisfies the following properties:

1. Let $\overline{f} = \frac{1}{n} \sum_{i=1}^{n} f_i(t)$, then

$$\overline{f}(t) = \frac{1}{\sqrt{2}} \overline{x}_{.1} + \sum_{j=1}^{[p/2]} \overline{x}_{.2j} \sin(jt) + \sum_{j=1}^{[p/2]} \overline{x}_{.,2j+1} \cos(jt) \quad t \in [-\pi, \pi].$$

2. This function representation preserves the Euclidean distance between the variables. That is, if

$$d_{ij}^2 = \sum_{k=1}^{n} (x_{ik} - x_{jk})^2$$

then

$$\frac{1}{\pi} \int_{-\pi}^{\pi} (f_i(t) - f_j(t))^2 dt = d_{ij}^2.$$

3. Suppose $(X_1, \ldots, X_p)$ are independent variables, each with variance σ^2, then for each i,

$$\mathrm{Var}(f_i(t)) = \begin{cases} \frac{\sigma^2}{2} p & p \quad \text{odd} \\ \frac{\sigma^2}{2}(p-1) + \sigma^2 \cos^2(pt) & p \quad \text{even.} \end{cases}$$

The following features should be noted:

- An *outlier* appears as single Andrews' curves that looks different from the rest.
- A subgroup of data is characterised by a set of simular curves.
- The order of the variables plays an important role for interpretation.
- For more than 20 observations we may obtain a bad "signal-to-ink-ratio", i.e., too many curves are overlaid in one picture.

Implementation There is a package in R named **pracma**, which may be installed by:

```
> install.packages("pracma")
```

if it isn't already installed and activated by:

```
> library(pracma)
```

The following illustrates the Andrews curve for the Iris data set, which is a data set contained in R.

```
data(iris)
A = as.matrix(iris[,-5])
f=as.integer(iris[,5])
andrewsplot(A,f,style="pol",npts=200)
andrewsplot(A,f,style="cart",npts=200)
```

The plot for datum 1 is:

$$f_1(t) = \frac{5.1}{\sqrt{2}} + 3.5\sin(t) + 1.4\cos(t) + 0.2\sin(2t).$$

1.3 Subspace Projections

The data matrix $\mathbf{x}$ described by Equation (1.1) of p quantitative measurements on n units may be described either in the *object space* or the *variable space*, as defined below.

Definition 1.1 (Object Space)**.** *Let* $\overline{\underline{x}} = \frac{1}{n}\sum_{j=1}^{n}\underline{x}_j$ *denote the sample mean vector. The* object space *is the* p *dimensional space with origin at* $\overline{\underline{x}}$*.*

Multivariate analysis studies how the variables relate to each other; their covariance and correlation. Centralising around the sample average helps to keep this in view. When studying object space, n points in $\mathbb{R}^p$ are considered, labelled $(\underline{y})_{j=1}^{n}$, where $\underline{y}_j = \underline{x}_j - \overline{\underline{x}}$.

Definition 1.2 (Variable Space)**.** *Let* $\overline{x}_{.k} = \frac{1}{n}\sum_{j=1}^{n} x_{jk}$*, the sample average for variable* k*. Consider the vectors* $\underline{z}_k = \underline{x}_k - \overline{x}_{.k}\mathbf{1} \in \mathbb{R}^n$*,* $k = 1,\ldots,p$*, where* $\mathbf{1} = (1,\ldots,1)^t \in \mathbb{R}^n$*. These vectors are all perpendicular to* $\mathbf{1}$*. The* variable space *is defined as the space spanned by these vectors. The variable space is therefore a space of dimension less than or equal to* p*, embedded in the* $n-1$ *dimensional subspace of* $\mathbb{R}^n$ *perpendicular to the vector* $\mathbf{1}$*.*

In the *variable space*, the scalar product c_{kl} between $\underline{z}_k$ and $\underline{z}_l$ is given by

$$c_{kl} = \sum_{i=1}^{n} z_{ik}z_{il}.$$

The quantity $s_{kl} = \frac{1}{n-1}c_{kl}$ is defined as the sample covariance between variate k and variate l. This is an *unbiased estimator* of the population covariance. The *sample correlation* between these variables is defined as

$$\cos(\alpha_{kl}) = r_{kl} := \frac{c_{kl}}{\sqrt{c_{kk}c_{ll}}},$$

where α_{kl} is the *angle* between vector $\underline{z}_k$ and $\underline{z}_l$.

Note: It should be clear that the projection of the vector $\underline{z}_k$ onto the one dimensional subspace of $\mathbb{R}^n$ spanned by the vector $\underline{z}_l$ is simply the linear regression of $\underline{z}_k$ onto $\underline{z}_l$;

$$r_{kl}\sqrt{\frac{c_{kk}}{c_{ll}}}\underline{z}_l.$$

Definition 1.3. *Set* $S_{kl} = \frac{1}{n-1}c_{kl}$. *The matrix* S *is the* sample covariance *matrix of the data matrix* $\mathbf{X}$. *The matrix* R *with entries* r_{kl} *is the* sample correlation *matrix.*

Remark When the n observations are considered in object space, their respective distances from each other may be represented by the $n \times n$ matrix $(d_{ij})_{1\le i\le n, 1\le j\le n}$. Considered in *variable space*, the observations lead to the $p \times p$ correlation and covariance matrices and the $p \times p$ matrix of angles α_{kl}, all representing the similarity between the variables.

Lemma 1.4. *The matrices* S *and* R *are non-negative definite.*

Proof Consider any p vector $\underline{a}$. Then

$$\underline{a}^t S \underline{a} = \frac{1}{n-1}\sum_{i=1}^{n}\sum_{kl} a_k a_l z_{ik} z_{il} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\sum_k a_k z_{ik}\right)^2 \ge 0.$$

$$\underline{a}^t R \underline{a} = \sum_{i=1}^{n} a_k a_l \frac{z_{ik}}{\sqrt{c_{kk}}}\frac{z_{il}}{\sqrt{c_{ll}}} = \sum_{i=1}^{n}\left(\sum_{k=1}^{p}\frac{z_{ik}a_k}{\sqrt{c_{kk}}}\right)^2 \ge 0.$$

□

1.4 Distances and Proximity Matrices

When the p variables are numerical and observations of continuous random variables, the *distance* between unit i and j in object space may be given by the Euclidean distance;

$$d_{ij} = \sqrt{\sum_{k=1}^{p}(y_{ik} - y_{jk})^2}.$$

Data sets often also give information in the form of *categorical* variables and it is useful to be able to incorporate both numerical and categorical variables. Also, there is a common problem of *missing data*; for an observation i, the datum x_{ik} may be missing for some, but not all, values of k.

The following measure of distance between observations is known as *Gower's dissimilarity*, which deals with missing data and also with categorical data.

Let

$$\delta_{ijk} = \begin{cases} 1 & x_{ik}, x_{jk} \quad \text{can be compared} \\ 0 & \text{otherwise} \end{cases}$$

$$s_{ijk} = 0 \qquad \text{if} \qquad \delta_{ijk} = 0.$$

If either x_{ik} or x_{jk} are missing, then both $\delta_{ijk} = 0$ and $s_{ijk} = 0$. For $\delta_{ijk} = 1$, if variable k is a numerical (quantitative) variable, let

$$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{\max_{a,b}|x_{ak} - x_{bk}|}.$$

If variable k is a categorical variable, let

$$s_{ijk} = \begin{cases} 1 & x_{ik} = x_{jk} \\ 0 & \text{otherwise} \end{cases}$$

Gower then constructs a *distance* by:

$$d_{ij} = \sum_k \frac{(1 - s_{ijk})\delta_{ijk}}{\sum_k \delta_{ijk}}.$$

If greater weight is attached to some of the variables, this can be modified using weights;

$$d_{ij} = \sum_k \frac{w_k(1 - s_{ijk})\delta_{ijk}}{\sum_k w_k \delta_{ijk}}.$$

Constructing a 'Virtual' data set from distances There are situations where the data matrix $\mathbf{x}$ is not given, but instead the distance matrix (d_{ij}) is given. The following discussion describes how to construct a virtual data matrix $\mathbf{x}$, which preserves the correct distances.

Let $\mathbf{x}$ be an $n \times p$ data matrix with entries x_{ij} and let $H_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^t$ where $\mathbf{1}$ denotes an n-vector where each entry is 1. Then it is an easy computation to see that

$$(H_n\mathbf{x})_{ij} = x_{ij} - \overline{x}_{.j}.$$

Set

$$Q = (H_n\mathbf{x})(H_n\mathbf{x})^t,$$

then it is clear that

$$Q_{ij} = \sum_{k=1}^{p} (x_{ik} - \overline{x}_{.k})(x_{jk} - \overline{x}_{.k}).$$

Set $y_{ij} = x_{ij} - \overline{x}_{.j}$. If the distance d_{ij} is the Euclidean distance, then

$$\begin{aligned} d_{ij}^2 &= \sum_k (y_{ik} - y_{jk})^2 \\ &= \sum_k y_{ik}^2 + \sum_k y_{jk}^2 - 2\sum_k y_{ik}y_{jk} \\ &= Q_{ii} + Q_{jj} - 2Q_{ij}. \end{aligned}$$

Note that $Q_{ij} = Q_{ji}$ and that $\sum_{i=1}^n Q_{ij} = 0$ for each j. Summing both sides over both i and j gives:

$$2n \sum_i Q_{ii} = \sum_{i,j} d_{ij}^2.$$

Summing over j and using $\sum_j Q_{jj} = \frac{1}{2n}\sum_{i,j} d_{i,j}^2$ gives:

$$Q_{ii} = \frac{1}{n}\sum_k d_{ik}^2 - \frac{1}{2n^2}\sum_{i,j} d_{ij}^2$$

Therefore:

$$Q_{ij} = -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{n}\sum_{k=1}^n d_{kj}^2 - \frac{1}{n}\sum_{k=1}^n d_{ik}^2 + \frac{1}{n^2}\sum_{ij} d_{ij}^2\right). \tag{1.2}$$

If the data matrix is not given, but instead the distances $(d_{ij})_{(i,j)\in\{1,\ldots,n\}}$, then a matrix Q may be constructed using the formula given by Equation (1.2). The matrix constructed in this way is clearly symmetric and can be diagonalised as

$$Q = P\Lambda P^t,$$

where P is orthonormal and Λ is diagonal. If the matrix $(d_{ij})_{(i,j)\in\{1,\ldots,n\}^2}$ is a distance, in the sense that it is symmetric, the entries are non negative and $d_{ij} \le d_{im} + d_{mj}$ for all $(i,j,m) \in \{1,\ldots,n\}^3$, then $\Lambda = \mathrm{diag}(\lambda_1,\ldots,\lambda_n)$, $\lambda_1 \ge \ldots \ge \lambda_n \ge 0$. Let $\sqrt{\lambda_j}$ denote the positive square root of λ_j and let $\Lambda^{1/2} = \mathrm{diag}(\sqrt{\lambda_1}\ldots,\sqrt{\lambda_n})$.

Definition 1.5 (Data Matrix obtained by Metric Scaling)**.** *Let*

$$\mathbf{x} = P\Lambda^{1/2},$$

then $\mathbf{x}$ *is the data matrix corresponding to* (d_{ij}) *obtained by metric scaling.*

Recall that the situation considered here is where the original data is not given; rather, the analyst has been presented with a matrix of distances between the original data points. The 'data matrix' obtained in this manner will preserve the distances between the original data.

Remarks

1. Metric scaling only works if the matrix Q is non negative definite (i.e. positive semi definite). This holds if and only if the input matrix (d_{ij}) satisfies the triangle inequality;

$$d_{ij} \le d_{im} + d_{mj} \qquad \forall (i,j,k) \in \{1,\ldots,n\}^3.$$

2. By construction, the data matrix $\mathbf{x}$ obtained in this way is already centred; $\mathbf{x} = H\mathbf{x}$.

3. Since $\mathbf{x} = H\mathbf{x}$, it follows that $\mathrm{rank}(Q) \le n-1$, at least one eigenvalue is zero. If

$$\frac{\lambda_1 + \ldots + \lambda_m}{\lambda_1 + \ldots + \lambda_{n-1}}$$

is sufficiently large for some $m < n-1$, then an approximate data matrix can be constructed from the first m columns of P by taking $\mathbf{x}$ as the $n \times m$ matrix with entries $\mathbf{x}_{ij} = P_{ij}\sqrt{d_j}$ $i \in \{1,\ldots,n\}$, $j \in \{1,\ldots,m\}$.

1.5 Measuring and Testing Multivariate Distances

Often in multivariate analysis, the n observations are not an observed random sample from a single population, but rather come from m different populations. Often, the aim is *classification*; to decide, based on the p-variate observation, which population the observation belongs to.

Consider m populations (for example, 7 different types of dog), where p features (variables) are measured (for example, p different bones within the body may be considered and the length of each

measured for each animal). Suppose that $n = n_1 + \ldots + n_m$, where n_b denotes the number of different animals from population b, for each population $b = 1, \ldots, m$. Let x_{abc} denote the observation: observation a, population b, variable c. Suppose you are given an observation, but you are not told which population the observation comes from. As a first step for making a guess, it is useful to have a measure of distance between the various populations.

1.6 Penrose and Mahalanobis Distance

Penrose Distance Let $n = \sum_{b=1}^{m} n_b$ denote the total number of observations and let

$$s_c^2 = \frac{\sum_{b=1}^{m}(n_b - 1)s_{bc}^2}{n - m}.$$

The observed *Penrose distance* between two populations α and β is defined as

$$p_{\alpha,\beta} = \frac{1}{p}\sum_{k=1}^{p}\frac{(\bar{x}_{.,\alpha,k} - \bar{x}_{.,\beta,k})^2}{s_k^2}.$$

Formal tests, of whether or not an observed Penrose distance is significantly different from zero, may be carried out under distributional assumtions. If it is assumed that the the observations x_{abc} are from *independent* random variables X_{abc}, where

$$X_{abc} \sim N(\mu_{bc}, \sigma_c^2)$$

(that is, the variables are normal and for variate c, the population variance is the same for each population $b = 1, \ldots, m$), then the distribution of

$$P_{\alpha,\beta} = \frac{1}{p}\sum_{k=1}^{p}\frac{(\overline{X}_{.,\alpha,k} - \overline{X}_{.,\beta,k})^2}{S_k^2}$$

under the null hypothesis that $\underline{\mu}_{\alpha,.} = \underline{\mu}_{\beta,.}$ may be computed.

The Mahalanobis Distance The Penrose distance does not take into account correlations between the variables. The Mahalanobis distance is a modification of the Penrose distance that takes into account possible correlations. If the independence assumption holds, then the Penrose distance is better, because there are fewer parameters involved. Let $\underline{X}_a$ denote a random vector that models population a, with $\mathbb{E}[\underline{X}_a] = \underline{\mu}_a$ and $\mathbf{C}(\underline{X}_a) = C$ (the notation $\mathbf{C}$ is used to denote a covariance matrix), where C is the same for *each* population $a = 1, \ldots, m$. Let $\bar{x}_{aj}$ denote the jth component of the vector $\underline{\bar{x}}_a$, the sample average from population a. Let S denote the pooled estimate of the covariance matrix and let $V = S^{-1}$. The Mahalanobis distance between two populations α and β is *defined* as

$$D_{\alpha\beta} = \sum_{r=1}^{p}\sum_{s=1}^{p}(\bar{x}_{.,\alpha,r} - \bar{x}_{.,\beta,r})V_{rs}(\bar{x}_{.,\alpha,s} - \bar{x}_{.,s\beta}) = (\underline{\bar{x}}_\alpha - \underline{\bar{x}}_\beta)^t V(\underline{\bar{x}}_\alpha - \underline{\bar{x}}_\beta).$$

To test whether the sample Mahalanobis distance, computed from the sample means and sample covariance matrix is statistically significant, one uses Hotelling's T^2 distribution; under the null hypothesis (of no difference),

$$\frac{n_a + n_b - p - 1}{(n_a + n_b - 2)p} \frac{n_a n_b}{n_a + n_b} D_{ab} \sim F_{p, n_a + n_b - p - 1}.$$

Note that there are $p(p+1)/2$ terms to be estimated in the covariance matrix for the Mahalanobis distance, while there are only p variances to be estimated for the Penrose distance. Therefore, if there is reason to believe that an independence assumption gives an accurate model, the Penrose distance is a better measure of distance; rather many observations are required to obtain the whole matrix S^{-1} with accuracy.

Example 1.1 (Egyptian Skull Data)**.**

The data set on Egyptian skulls, found in `skulls.dat` on the course home page gives the measurements X_1 = maximum breadth, X_2 = basibregmatic height, X_3 = basialveolar length and X_4 = nasal height. The data is for a total of 150 skulls, 30 from each of 5 groupings; −4000 Early Predynastic, −3300 Late Predynastic, −1850 12th and 13th Dynasties, −200 Ptolemaic Period, 150 Roman Period.

Firstly, the sample mean vector for $(X_1, X_2, X_3, X_4)^t$ is computed for each period, and the pooled covariance matrix. That is, firstly S_a, the sample covariance matrix for period a is computed for each of the 5 periods and then

$$S = \frac{\sum_{a=1}^{5} 29 S_a}{145}.$$

Here S is a 4×4 covariance matrix, with the sample variances along the diagonal.

The Penrose distances may now be computed directly; to compute the Mahalanobis distances, the inverse S^{-1} is required. These distances turn out to be:

Penrose

	I	II	III	IV	V
I	–				
II	0.023	–			
III	0.216	0.163	–		
IV	0.493	0.404	0.108	–	
V	0.736	0.583	0.244	0.066	–

Mahalanobis

	I	II	III	IV	V
I	–				
II	0.091	–			
III	0.903	0.729	–		
IV	1.881	1.594	0.443	–	
V	2.697	2.176	0.911	0.219	–

Due to the change of scale (the Penrose is divided by a $1/p$) it does not make sense to compare the *absolute* values of these distances, but the *ratios* should be comparable, giving the change between one group and another. The ratio of the $I \to II$ and $I \to V$ distance is 0.736/0.023 = 32.0 for the Penrose and 2.697/0.091 = 29.6 for the Mahalanobis measure; the results are similar. □

Implementation The computation may be implemented in R as follows: the data is found on the course home page under `skulls.dat` and is loaded as follows:

```
www2 = "https://www.mimuw.edu.pl/~noble/courses/MultivariateStatistics/data/skulls.dat"
skulls = read.table(www2,header=T)
```

To get the means for each variable by category, try

```
> x2 <- by(skulls[,-5],skulls[,5],colMeans)
> y<-simplify2array(x2)
```

and for the covariance matrices by category, try

```
> S <- by(skulls[,-5],skulls[,5],cov)
> Sarray<-simplify2array(S)
```

To obtain the pooled covariance,

```
> Spooled <-
29*(Sarray[,,1]+Sarray[,,2]+Sarray[,,3]+Sarray[,,4]+Sarray[,,5])/145
> Spooled
            MB          BH          BL        NH
MB 21.11080460  0.03678161  0.07908046  2.008966
BH  0.03678161 23.48459770  5.20000000  2.845057
BL  0.07908046  5.20000000 24.17908046  1.133333
NH  2.00896552  2.84505747  1.13333333 10.152644
```

To obtain the Mahalanobis distance,

```
> mahalanobis(y[,1],y[,2],Spooled,inverted=FALSE)
[1] 0.09103424
```

I have not found an existing R script that gives the whole array of Mahalanobis distances presented in a single matrix, but this is a relatively easy exercise.

1.7 The Mantel Randomisation Test

The Mantel test (1967) was introduced to detect space / time clustering of diseases. Suppose that n objects are being studied and suppose that there are observations on two sets of observations. Let M be the $n \times n$ matrix where M_{ij} is the distance between object i and object j based on the first set of variables and let E be a matrix of distances between the objects based on the second set of variables. Mantel's test assesses whether or not the elements in M and E show some significant correlation. Let

$$Z = \sum_{j=2}^{n} \sum_{k=1}^{j-1} M_{jk} E_{jk}.$$

This is compared with observations

$$Z_\sigma = \sum_{j=2}^{n} \sum_{k=1}^{j-1} M_{\sigma(j)\sigma(k)} E_{jk},$$

where σ is a randomly chosen permutation of $(1, \ldots, n)$. The values z_σ are computed for *each* of the $n!$ permutations σ and then it is seen if Z is a 'typical' observation of this distribution (i.e. does it land between the $\frac{\alpha}{2} \times 100$ and $1 - \frac{\alpha}{2} \times 100$ percentiles of this empirical distribution, where α is the significance level?)

For the Egyptian skulls data, the n objects are the n different skulls. To perform a Mantel randomisation test, the two sets of variables are: Set 1 (on which M is based) are the measurements of the skulls and Set 2 (on which E is based) is the single variable, the period from which the skull comes.

Example 1.2 (Ozone data)**.**

To perform a Mantel test, the `ade4` package may be used:

```
> install.packages("ade4")
> library("ade4")
```

Load the `ozone.csv` data set.

```
> ozone <- read.csv("~/ozone.csv")
>   View(ozone)
> head(ozone)
  Station   Av8top      Lat       Lon
1      60 7.225806 34.13583 -117.9236
2      69 5.899194 34.17611 -118.3153
3      72 4.052885 33.82361 -118.1875
4      74 7.181452 34.19944 -118.5347
5      75 6.076613 34.06694 -117.7514
6      84 3.157258 33.92917 -118.2097
```

This contains ozone measurements from thirty-two locations in the Los Angeles area aggregated over one month. The dataset includes the station number (Station), the latitude and longitude of the station (Lat and Lon), and the average of the highest eight hour daily averages (Av8top). We want to test whether the differences in ozone measurements are smaller for stations that are closer together than for stations that are far apart.

To run a Mantel test, generate two distance matrices, one containing spatial distances and one containing distances between measured outcomes at the given points. In the spatial distance matrix, entries for pairs of points that are close together are lower than for pairs of points that are far apart. In the measured outcome matrix, entries for pairs of locations with similar outcomes are lower than for pairs of points with dissimilar outcomes. This may be done using the `dist` function. The Mantel test function requires objects of this 'distance' class.

```
> station.dists <- dist(cbind(ozone$Lon, ozone$Lat))
> ozone.dists <- dist(ozone$Av8top)
> as.matrix(station.dists)[1:5, 1:5]
          1         2         3         4         5
1 0.0000000 0.3937326 0.4088031 0.6144127 0.1854888
2 0.3937326 0.0000000 0.3749446 0.2206810 0.5743590
3 0.4088031 0.3749446 0.0000000 0.5116772 0.4994034
4 0.6144127 0.2206810 0.5116772 0.0000000 0.7944601
5 0.1854888 0.5743590 0.4994034 0.7944601 0.0000000
> as.matrix(ozone.dists)[1:5, 1:5]
         1        2        3        4        5
1 0.000000 1.326612 3.172921 0.044354 1.149193
2 1.326612 0.000000 1.846309 1.282258 0.177419
3 3.172921 1.846309 0.000000 3.128567 2.023728
4 0.044354 1.282258 3.128567 0.000000 1.104839
5 1.149193 0.177419 2.023728 1.104839 0.000000
```

These are the two matrices which the function will test for a correlation. The test consists of calculating the correlation of the entries in the matrices, then permuting the matrices and calculating the same test statistic under each permutation and comparing the original test statistic to the distribution of test statistics from the permutations to generate a p-value. The number of permutations defines the precision with which the p-value can be calculated. The function to perform the Mantel test is `mantel.rtest` and the required arguments are the two distance matrices. The number of permutations can also be specified by the user; the default value is 99.

```
> mantel.rtest(station.dists, ozone.dists, nrepet = 9999)
Monte-Carlo test
Observation: 0.1636308
Call: mantel.rtest(m1 = station.dists, m2 = ozone.dists, nrepet = 9999)
Based on 9999 replicates
Simulated p-value: 0.0293
```

Based on these results, we can reject the null hypothesis that these two matrices, spatial distance and ozone distance, are unrelated with $\alpha = .05$. The observed correlation, $r = 0.1636308$, suggests that the matrix entries are positively associated. So smaller differences in ozone are generally seen among pairs of stations that are close to each other than far from each other. Note that since this test is based on random permutations, the same code will always arrive at the same observed correlation but rarely the same estimate of the p-value. □

Lecture 1: Exercises

1. Consider the data matrix *iris*, which is available as one of the *datasets* in R. Type

   ```
   > iris
   ```

   to see the names of the variables, their recorded values and the numbers of observations. Typing

   ```
   >?iris
   ```

   will give a brief description in the *help* window (bottom right).

   Plot the Andrews curves corresponding to the first ten individuals of each of the three groups into one diagram using x_1 as sepal length, x_2 as sepal width, x_3 as petal length, x_4 as petal width. Try to spot clusters and outliers.

   Now plot the corresponding Andrews curves with x_1 as petal length, x_2 as petal width, x_3 as sepal length, x_4 as sepal width.

   The pattern has changed significantly, indicating that with Andrews curves, some permutations of the variables can be more informative than others.

2. Consider the data for ozone measurements from thirty two locations in the Los Angeles area, found in the file `ozone.csv` in the course data directory. Perform a Mantel test to see whether the differences between ozone measurements are smaller for stations that are closer together.

Chapter 2

Principal Component and Factor Analysis

2.1 Introduction

Let $\mathbf{x}$ denote an $n \times p$ data matrix of n p-variate observations. Principal Component Analysis is a technique applied when some of the variables are highly correlated. The aim is to find m linear combinations of the variables, where $m < p$, which describe the sample covariance or correlation structure of the data set.

PCA may be carried out on either S, the sample covariance matrix, or R, the sample correlation matrix. The sample correlation matrix is preferable if the p variables in the data set have widely varying scales.

The aim is

- data reduction (reducing p variables to m, p linear combinations of the variables)
- interpretation (we examine which variables influence the principal components and, from this, try to determine hidden factors; the principal components are *factors*).

2.2 Principal Component Analysis

Since PCA is concerned with the covariance / correlation structure of the variables, the data matrix is first *centred*, so that the columns are all mean zero. Let

$$H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^t \tag{2.1}$$

where I_n denotes the $n \times n$ identity matrix and $\mathbf{1}_n$ denotes the n-vector with each entry 1. Let

$$\mathbf{z} = H\mathbf{x},$$

then the entries of the $n \times p$ matrix $\mathbf{z}$ are

$$z_{ij} = x_{ij} - \overline{x}_{.j}.$$

The sample covariance matrix S of $\mathbf{x}$ is given by:

$$S = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki} - \overline{x}_{.i})(x_{kj} - \overline{x}_{.j}) = \frac{1}{n-1}\mathbf{z}^t\mathbf{z}. \tag{2.2}$$

A principal component analysis simply finds the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ of the sample covariance matrix S and the corresponding eigenvectors $P_{.j} : j = 1, \ldots, p$. The principal components are the uncorrelated linear combinations; $\mathbf{y} = \mathbf{z}P$, where $\mathbf{y}_{.1}$ has the largest possible statistical variance among orthonormal transformations of $\mathbf{z}$ and $\mathbf{y}_{.q}$ has the largest statistical variance under the constraint that it is uncorrelated with $(\mathbf{y}_{.1}, \ldots, \mathbf{y}_{.,q-1})$.

Lemma 2.1. *Let S be the sample covariance matrix defined by Equation* (2.2) *and let λ_1 be the* largest *eigenvalue of S and let $\underline{\gamma}$ denote the corresponding normalised eigenvector; namely,*

$$S\underline{\gamma} = \lambda_1\underline{\gamma}, \qquad \sum_{j=1}^{p}\gamma_j^2 = 1.$$

Let $z_{ij} = x_{ij} - \bar{x}_{.j}$. Then, for any p- vector $\underline{a}$, with $\sum_{j=1}^{p} a_j^2 = 1$,

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{p} a_j z_{ij}\right)^2 \leq \sum_{i=1}^{n}\left(\sum_{j=1}^{p}\gamma_j z_{ij}\right)^2.$$

Proof

$$\begin{aligned}\sum_{i=1}^{n}\left(\sum_{j=1}^{p} a_j z_{ij}\right)^2 &= \sum_{i=1}^{n}\sum_{jk=1}^{p} a_j a_k z_{ij} z_{ik} \\ &= (n-1)\sum_{jk} a_j a_k S_{jk} = (n-1)\underline{a}^t S\underline{a} = (n-1)\underline{a}^t P^t D P\underline{a} = (n-1)\underline{b}^t D\underline{b},\end{aligned}$$

where $\underline{b} = P\underline{a}$. Note that $\underline{b}^t\underline{b} = \underline{a}^t P^t P\underline{a} = \underline{a}^t\underline{a} = 1$, so $\underline{b}$ is a unit vector. Since $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ where $\lambda_1 \geq \ldots \geq \lambda_p$, it follows that the expression is maximised if $\underline{b} = (1, 0, \ldots, 0)$, so

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{p} a_j z_{ij}\right)^2 \leq (n-1)\lambda_1.$$

Meanwhile, since $\underline{\gamma}$ is a unit eigenvector of S with eigenvalue λ_1, it follows that

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\gamma_j z_{ij}\right)^2 = (n-1)\underline{\gamma}^t S\underline{\gamma} = (n-1)\lambda_1\underline{\gamma}^t\underline{\gamma} = (n-1)\lambda_1$$

and the result follows. □

Notation λ_k will be used to denote the kth largest eigenvalue.

Definition 2.2 (Principal Component, Loading Vector)**.** *Let* $\mathbf{x}$ *denote the* $n \times p$ *data matrix,* n *multivariate observations on* p *variables. Let* P *denote the orthonormal matrix and* D *the diagonal matrix with elements arranged in decreasing order such that* $S = PDP^t$*. Let* H *be the* $n \times n$ *matrix defined by Equation* (2.1)*. The* columns *of the matrix*

$$\mathbf{y} = H\mathbf{x}P \tag{2.3}$$

are called the sample principal components*. The* i*th element of the* k*th column represents the* score *of the* k*th principal component for the* i*th observation. The* k*th column of the orthonormal matrix* P *is the* loading vector *for the* k*th principal component.*

The following theorem is stated without proof; it is left as an exercise.

Theorem 2.3. *Let* $\mathbf{x}$ *be an* $n \times p$ *data matrix;* n *p-variate observations. Let* P *denote the orthonormal matrix and* D *the diagonal matrix with elements from highest to lowest such that* $S = PDP^t$*, where* S *is the unbiased sample covariance matrix. Let*

$$\mathbf{y} = H\mathbf{x}P$$

where

$$H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^t.$$

Let

$$S^{(\mathbf{y})} = \frac{1}{n-1}\mathbf{y}^t\mathbf{y}.$$

Then

1. $$S^{(\mathbf{y})} = D$$

2. *Let* $\underline{z}_k = (z_{1,k}, \ldots, z_{n,k})^t$ *where* $z_{jk} = x_{jk} - \frac{1}{n}\sum_{i=1}^n x_{ik}$*. Then the linear combinations*

   $$P_{1m}\underline{z}_1 + \ldots + P_{pm}\underline{z}_p, \qquad m = 1, \ldots, q \leq p$$

   span the parallelepiped with the largest volume among all parallelepipeds spanned by standardised linear combinations of $\mathbf{x}$ *in variable space.*

3. *Let* $c_{kl}^{(Y)} = \sum_{i=1}^n y_{ik}y_{il}$*. The largest volume is given by*

   $$\sqrt{det\left((c_{kl}^{(Y)})_{(k,l)\in\{1,\ldots,q\}^2}\right)} = (n-1)^{q/2}\sqrt{d_{11}\ldots d_{mm}}.$$

4. $$tr(S^{(\mathbf{x})}) = tr(S^{(\mathbf{y})}) = \lambda_1 + \ldots + \lambda_p.$$

Recall For a symmetric $p \times p$ matrix, the sum of the trace is equal to the sum of the eigenvalues. It follows that $\text{tr}(S) = \sum_{j=1}^{p} \lambda_j = \text{tr}(D)$.

Interpretation

- Plotting $\mathbf{y}_{.,1}, \ldots, \mathbf{y}_{.,q}$ for $q < p$ in object space gives the 'best' q dimensional summary of $\mathbf{x}$ if one is looking for the parallelepiped of largest volume spanned by standardised linear combinations of $\mathbf{x}$ in variable space (recall that the point $\underline{\overline{x}}^t := (\overline{x}_{.,1}, \ldots, \overline{x}_{.,p})$ is the origin in variable space.

- It often turns out that all the loadings for the *first* principal component are positive. If this is the case, then it can be interpreted as a measurement of size. If this is the case, then it necessarily follows that all the other principal components have both positive and negative loadings and are therefore interpreted in terms of shape. Since the general idea is to reduce the data and to only use m principal components where $m < p$, they will not cover all possibilities for shape.

- The sum of the unbiased sample variances, that is $\text{tr}(S)$ is also called the total sample variation of $\mathbf{x}$. If only m principal components are used, then

$$\frac{\lambda_1 + \ldots + \lambda_m}{\lambda_1 + \ldots + \lambda_p}$$

  represents the proportion of the variance explained by the first m sample principal components. There are two usual criteria for deciding how many to use:

  1. The m sample prinicipal components explain 90% of the variation.
  2. (Kaiser's criterion) The variances of the sample principal components beyond the mth principal components account for less than the average $\frac{1}{p}\text{tr}(S)$.

  When $p \leq 20$, the second of these tends to include too few components.

- After deciding on the number of principal components m to include, the data is represented only using the first m principal components: using

$$\mathbf{y} = H\mathbf{x}P = \mathbf{z}P,$$

  it follows that

$$\mathbf{x} = \mathbf{1}_n \underline{\overline{x}}^t + \mathbf{y}P^t$$

  the components $m+1, \ldots, p$ are estimated by 0, giving $\widehat{\mathbf{x}}$, the estimate of $\mathbf{x}$ as:

$$\hat{\mathbf{x}} = \mathbf{1}\bar{\mathbf{x}}^t + (\mathbf{y}_{.,1} | \ldots | \mathbf{y}_{.,m}) \begin{pmatrix} P_{11} & \ldots & P_{p1} \\ \vdots & \ddots & \vdots \\ P_{1m} & \ldots & P_{pm} \end{pmatrix}.$$

  Recall that the vector $(P_{1k}, \ldots, P_{p,k})^t$ is the kth loading vector.

- Sometimes it is better to use *standardised* variables. This occurs if variables on a *smaller* scale give significant information, which can be lost if the raw data is used. The solution is to start by standardising the data (namely, for each column, removing the sample mean and dividing through by the sample standard deviation) and applying a principal component analysis to the standardised data. This is equivalent to carrying out a PCA on the *correlation* matrix rather than the *covariance* matrix.

  The analysis is not scale invariant; the relative importance of the eigenvalues may change, the loadings will change and their interpretation may change. Whether the data or the standardised data is to be used depends on the situation.

- It may be possible to decide that some of the p variables are redundant, based on the PCA analysis on the *correlation* matrix. This is done by considering the *last* sample principal component and discarding the variable assigned to the loading with largest absolute value. Then continue with the loadings of the *second* last principal component, and so on. Stop discarding if certain criteria are satisfied: for example,

  1. the eigenvalue corresponding to the loadings is greater than 0.7 (this seems to work in practice)
  2. the sample principal components corresponding to the loadings you have not yet considered explain less than 80% of the variation.

  Using either criterion, at least four variables should always be retained.

The columns of the matrix P are often called the *coefficients*.

Prinipal component analysis is only useful as a tool if some of the eigenvalues of the statistical correlation matrix are very small. Absolutely nothing is achieved by a principal component analysis if all the eigenvalues of the correlation matrix are significant.

2.3 How to do a Principal Component Analysis

Throughout this discussion, variance refers to statistical variance, covariance to statistical covariance and correlation to statistical correlation. Firstly, suppose that the PCA is being carried out on the *covariance*. The procedure is as follows: suppose there are n independent observations from $(X_1, \ldots, X_p)$. Firstly, the data is centralised:

$$\mathbf{z} = H\mathbf{x}$$

and the statistical covariance is computed;

$$S = \frac{1}{n-1}\mathbf{z}^t\mathbf{z}.$$

The *first* principal component is

$$\mathbf{y}_{.1} = P_{11}\mathbf{z}_{.1} + \ldots + P_{p1}\mathbf{z}_{.p},$$

where $P_{11}, \ldots, P_{p1}$ are chosen to maximise $\mathrm{Var}(\mathbf{y}_{.1})$ subject to the constraint that $\sum_{k=1}^{p} P_{k1}^2 = 1$. That is, to maximise

$$P_{.1}^t S P_{.1}$$

where $P_{.1}$ is taken as a column vector, subject to the constraint. Once the first component has been established, the second component

$$\mathbf{y}_{.2} = P_{12}\mathbf{z}_{.1} + \ldots + P_{p2}\mathbf{z}_{.p}$$

is established by finding $(P_{12}, \ldots, P_{p2})$ that maximises $\mathrm{Var}(\mathbf{y}_{.2})$ subject to the constraints

$$\sum_{k=1}^{p} P_{k2}^2 = 1,$$

$$P_{.2}^t S P_{.2} = 0.$$

That is, $P_{.2}$ is chosen to ensure that the *statistical* correlation is zero. Inductively, once $P_{.j}$ have been established for $j = 1, \ldots, k-1$, $P_{.k}$ are established by maximising the estimate of $\mathrm{Var}(\mathbf{y}_k)$,

$$P_{.k}^t S P_{.k}$$

subject to the constraints that

$$\sum_{l=1}^{p} P_{lk}^2 = 1$$

and the statistical covariances $\mathrm{Cov}(\mathbf{y}_{.,j}, \mathbf{y}_{.,k})$ are zero for $j = 1, \ldots, k-1$. That is

$$P_{.j}^t S P_{.k} = 0, \qquad j = 1, \ldots, k-1.$$

Note that the statistical variances of the principal components are the eigenvalues of the sample covariance matrix and that the columns $P_{.k}$ are the eigenvectors.

Recall that, for a symmetric $m \times m$ matrix C, with eigenvalue $\lambda_1, \ldots, \lambda_m$,

$$\mathrm{tr}(C) = \sum_{j=1}^{m} \lambda_j.$$

Let λ_j denote the estimates of $\mathrm{Var}(Z_j)$. It follows that

$$\sum_{j=1}^{p} S_{jj} = \sum_{j=1}^{p} \lambda_j.$$

Since principal component analysis considers dependence and independence, it is usual to code the variables $\mathbf{x}_{.1}, \ldots, \mathbf{x}_{.p}$ so that they each have mean 0 and variance 1 at the beginning of the analysis. The procedure with this modification is therefore as follows:

1. Compute $\bar{x}_{.k}$ for $k = 1, \ldots, p$ and $S_{kl} = \frac{1}{n-1}\sum_{j=1}^{n}(x_{jk} - \bar{x}_{.k})(x_{jl} - \bar{x}_{.l})$, the sample means and sample covariance matrix.

2. Compute the coded variables, $y_{jk} = \frac{x_{jk} - \bar{x}_{.k}}{\sqrt{S_{kk}}}$.

3. Compute the correlation matrix
$$R_{kl} = \frac{S_{kl}}{\sqrt{S_{kk}S_{ll}}}.$$
This is the covariance matrix for the coded variables.

4. Find the eigenvalues $\lambda_1, \ldots, \lambda_p$ and the corresponding eigenvectors $P_{.1}, \ldots, P_{.p}$ in the way described above.

5. Discard any principal components that do not account for a significant variation in the data. This means that $\mathbf{y}_{.k}, \ldots, \mathbf{y}_{.p}$ are discarded for k such that $\sum_{j=k+1}^{p}\lambda_j \leq \alpha \leq \sum_{j=k}^{p}\lambda_j$ where α is the level of the variation that is to be ignored. Usually, this is roughly 20% or, when the data is standardised, components corresponding to eigenvalues less than 1 are ignored.

2.4 Confidence Intervals for PCA Eigenvalues and Eigenvectors

There exists some results in the literature. The proofs of these are long and technical. Much more seriously, they all rely on the assumption that the data comes from i.i.d. p-variate Gaussian variables and that n is large.

Theorem 2.4 (Lawley (1956))**.** *If λ_i is a distinct eigenvalue of the covariance (correlation) matrix, then*
$$\mathbb{E}[\widehat{\lambda}_i] = \lambda_i + \frac{\lambda_i}{n}\sum_{j\neq i}\frac{\lambda_j}{\lambda_i - \lambda_j} + O(n^{-2})$$
so that the estimate is asymptotically unbiased and:
$$\mathbf{V}(\widehat{\lambda}_i) = \frac{2\lambda_i^2}{n}\left(1 + \frac{1}{n}\sum_{j\neq i}\left(\frac{\lambda_i}{\lambda_i - \lambda_j}\right)^2\right) + O(n^{-3}).$$

Also, let $\widehat{h}_i$ denote the estimate of the ith eigenvector h_i with λ_i the ith eigenvalue, then

1.
$$\sqrt{n}(\underline{\widehat{\lambda}} - \underline{\lambda}) \longrightarrow_{(d)} N_p(\underline{0}, 2\Lambda^2)$$
where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$

2.
$$\sqrt{n}(\underline{\widehat{h}}_i - \underline{h}_i) \longrightarrow_{(d)} N_p(\underline{0}, E_i)$$
where
$$E_i = \lambda_i \sum_{k\neq i}\frac{\lambda_k}{(\lambda_i - \lambda_k)^2}\underline{h}_k\underline{h}_k^t.$$

3. For each $i = 1, \ldots, p$, $\widehat{\lambda}_i \perp \widehat{\underline{h}}_i$.

Since the 'normality' assumption for these results is not usually satisfied and size of the data set is often insufficient for a 'central limit theorem effect', these results are of limited value. To find confidence intervals for eigenvalues, bootstrap methods may be used.

If we have an $n \times p$ data matrix, a bootstrap method takes randomly chosen subsets of size m, were $m \leq n$ and performs the PCA on the subset of size m. By taking M such randomly chosen subsets, an empirical distribution for the estimate of the eigenvalue λ_i may be constructed.

2.5 Implementation in R

2.5.1 Sparrow Data

The implementation in R is shown by application to the Bumpus Sparrow data set.

Example 2.1 (Bumpus Sparrow Data)**.**

In 1898, H.C. Bumpus collected data from 49 female sparrows, which he picked up after a severe storm. Birds 1 to 21 survived; birds 22 to 49 died. The variables measured were X_1 total length, X_2 alar extent X_3 length of beak and head, X_4 length of humerus, X_5 length of keel of sternum, X_6 returns a 1 if the bird survived and a 0 otherwise. This data set is found in `sparrow.dat` on the course home page.

```
www<-"https://www.mimuw.edu.pl/~noble/courses/QPEDataScience/data/spar
row.dat"
sparrow <- read.table(www,header=T,quote="\"")
View(sparrow)
```

A principal component analysis can be carried out quite simply using the command `prcomp`.

```
pca <- prcomp(sparrow[,-6],scale=TRUE)
print(pca)
Standard deviations:
[1] 1.8834858 0.7399002 0.6203523 0.5756578 0.4345230

Rotation:
              PC1         PC2        PC3        PC4        PC5
LENGTH  0.4528873 -0.08190723  0.6184031  0.5613384 -0.3010998
ALAR    0.4481098  0.40195407  0.4199159 -0.5570644  0.3885620
HEADBK  0.4559146  0.26939394 -0.5192954  0.4895259  0.4585472
HUMERUS 0.4749343  0.16267408 -0.4015361 -0.2960348 -0.7064748
STERNUM 0.4008366 -0.85597207 -0.1017315 -0.2174999  0.2213285
```

The correlation matrix may be obtained in the following way:

```
> cormat <- cor(sparrow[,-6])
> cormat
           LENGTH      ALAR    HEADBK   HUMERUS   STERNUM
LENGTH  1.0000000 0.6761409 0.6618119 0.6452841 0.6051247
ALAR    0.6761409 1.0000000 0.6433941 0.7287194 0.4887925
HEADBK  0.6618119 0.6433941 1.0000000 0.7631899 0.5262701
HUMERUS 0.6452841 0.7287194 0.7631899 1.0000000 0.6066493
STERNUM 0.6051247 0.4887925 0.5262701 0.6066493 1.0000000
```

The eigenvalues and eigenvectors from the correlation matrix can be obtained in the following way:

```
> ev <- eigen(cor(sparrow[,-6]))
> ev
$values
[1] 3.5475186 0.5474524 0.3848369 0.3313819 0.1888102


$vectors
           [,1]        [,2]       [,3]       [,4]       [,5]
[1,] -0.4528873 -0.08190723  0.6184031  0.5613384 -0.3010998
[2,] -0.4481098  0.40195407  0.4199159 -0.5570644  0.3885620
[3,] -0.4559146  0.26939394 -0.5192954  0.4895259  0.4585472
[4,] -0.4749343  0.16267408 -0.4015361 -0.2960348 -0.7064748
[5,] -0.4008366 -0.85597207 -0.1017315 -0.2174999  0.2213285
```

Note that $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 5$, the sum of the trace.

The first component accounts for $\frac{3.5475}{5} \times 100 = 70.95\%$ of the total variance. The other principal components account for 10.95%, 7.70%, 6.63% and 3.77% respectively of the total variance.

Another way of looking at it is as follows: after standardisation, all the original variables have variance 1. Therefore, the first principal component has a variance 3.616 times as much as one of the original variables, while the second only accounts for half as much as any of the original variables. The first principal component is clearly by far the most important.

The first principal component, in terms of the standardised variables, is

$$Y_1 = 0.4529Z_1 + 0.4481Z_2 + 0.4559Z_3 + 0.4749Z_4 + 0.4008Z_5.$$

The coefficients are all nearly equal, so Y_1 is an index of the size of the sparrows. Therefore, about 72.3% of the variation in the data is due to differences in the size of the sparrows.

The second principal component is

$$Y_2 = -0.0819Z_1 + 0.4020Z_2 + 0.2694Z_3 + 0.1627Z_4 - 0.8560Z_5.$$

This contrasts Z_2, Z_3 and Z_4 on the one hand, with the length of the keel of the sternum Z_5 on the other. Here Y_2 represents a shape difference between the sparrows. □

2.5.2 Bootstrap for Confidence Intervals

Consider the Fisher Iris data. Suppose we want a 95% confidence interval for the loading of the first principal component with respect to Sepal length.

```
library(boot)

getPrcStat <- function (samdf,vname,pcnum){
  prcs <- prcomp(samdf[1:4]) # returns matrix
  return(prcs$rotation[ vname,pcnum ])   # pick out the thing we need
}

bootEst <- function(df,d){
   sampledDf <- df[ d, ]  # resample dataframe
   return(getPrcStat(sampledDf,"Sepal.Length",1))
}

bootOut <- boot(iris,bootEst,R=10000)
boot.ci(bootOut,type=c("basic"))
```

This gives the output:

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
  Based on 10000 bootstrap replicates

  CALL :
  boot.ci(boot.out = bootOut, type = c("basic"))

  Intervals :
  Level      Basic
  95%   ( 0.3364,  1.1086 )
  Calculations and Intervals on Original Scale
```

2.5.3 Using the Principal Components

In the 'Sparrow' data set, we see that by far the most of the variation is accounted for by the first two principal components, so these two components should be useful for most analysis of the data; the other three components should not add much.

The question we now consider is whether the 5 quantitative variables can be used to show differences between the two groups, those that survived the storm and those that did not.

A *scree plot* gives the proportion of the variance attributable to each component; this is found in Figure 2.1.

```
> screeplot(pca)
```

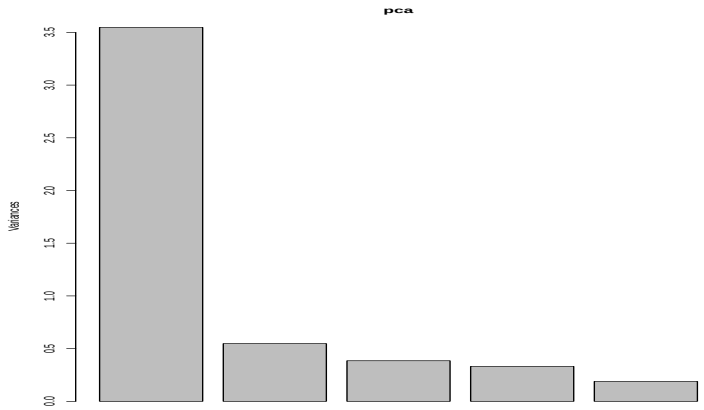

Figure 2.1: Scree plot for Sparrow principal components

Now let us make a scatter plot of the first two principal components. A nice package for plots and visualisation is **ggplot2**. Firstly, we add the first two principal components to the 'sparrow' data frame as follows:

```
spa <- sparrow
spa$pc1 <- pca$x[,1]
spa$pc2 <- pca$x[,2]
```

Using the plotting command from **ggplot2**:

```
qplot(pc2,pc1,colour=SURVIVE,data=spa)
```

gives the scatterplot in Figure 2.2. This illustrates that the population of birds that did not survive has more 'extreme' values than the population that did survive.

2.6 Weighted Projection Methods

Let $\mathbf{x}$ be the $n \times p$ data matrix, corresponding to n p-variate observations $\underline{x}_1, \ldots, \underline{x}_n$. Let $\underline{y}_1, \ldots, \underline{y}_n$ denote the corresponding n points obtained by projecting onto a q dimensional subspace of the object space. The following properties characterise the q dimensional subspace found by PCA.

1. The points $\underline{x}_1, \ldots, \underline{x}_n$ are projected perpendicularly onto $\underline{y}_1, \ldots, \underline{y}_n$.

2. The data points $\underline{y}_1, \ldots, \underline{y}_n$ have the greatest variance among standardised q dimensional subspace projections.

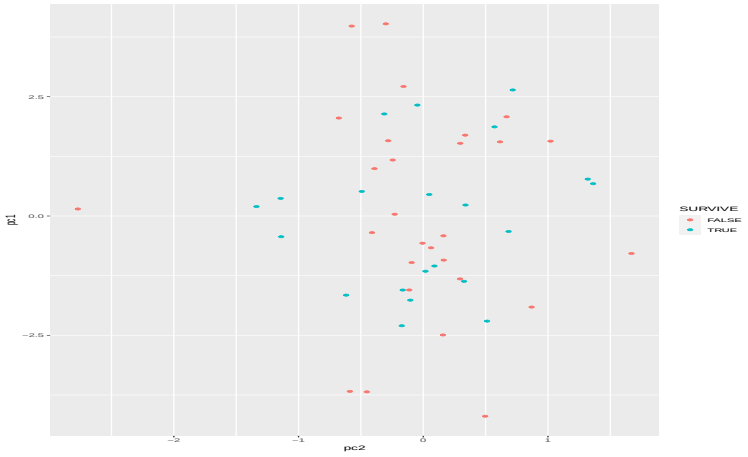

Figure 2.2: Scatter plot for first two Sparrow principal components

The points are those in the q dimensional space that minimise:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(d_{ij}-\hat{d}_{ij})^2$$

where

$$d_{ij}=\sqrt{\sum_{k=1}^{p}(x_{ik}-x_{jk})^2}, \qquad \hat{d}_{ij}=\sqrt{\sum_{k=1}^{p}(y_{ik}-y_{jk})^2}.$$

A disadvantage of the constraint that $\underline{y}_1,\ldots,\underline{y}_n$ have the largest possible variation is that observations close to the centre in the projected space may be far from the centre in the higher dimensional space. A better quantity to minimise is

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\omega_{ij}(d_{ij}-\hat{d}_{ij})^2$$

where ω_{ij} controls the accuracy of the comparisons. For example, take $\omega_{ij}=d_{ij}$ if accurate representation of large distances is required and $\omega_{ij}=\frac{1}{d_{ij}}$ if accurate representation of small distances is required.

2.7 Factor Analysis

As usual with descriptive statistics, 'Var' represents a *statistical* variance and 'Cov' represents *statistical* covariance; the terms refer to the statistics computed from the data and not to any features of the population distribution.

Factor analysis may be seen as an extension of Principal Component Analysis. Given p variables $X_1,\ldots,X_p$, it is hoped that they can be expressed, or mostly expressed, by a reduced number of *factors*, which are linear combinations of the variables. Based on the original variables, it is hoped that these factors may have an interpretation.

Suppose there are n observations, $(x_{j1}, \ldots, x_{j_p})_{j=1}^n$ of the p variables. One starts by applying a *principal component analysis* on the *correlation* (that is, on the standardised data). A suitable value of m is chosen and principal components after level m are neglected. Let $Y_1, \ldots, Y_p$ denote the principal components, with corresponding eigenvalues $\lambda_1 \geq \ldots \geq \lambda_p$. Suppose that the data has been standardised. Then $Y = P^t X$. In co-ordinates,

$$Y_k = P_{1k}X_1 + \ldots + P_{pk}X_p, \qquad k = 1, \ldots, p.$$

These linear combinations of the variables, as discussed earlier, are statistically uncorrelated. Now, choose m, the number of factors in the model. As discussed earlier, there are two usual methods; either let m equal the number of eigenvalues greater than or equal to 1 (Kaiser's method) or else let m denote the lowest number of eigenvalues that account for more than 80% of the variation.

Recall that P is orthonormal and hence $P^{-1} = P^t$. It follows that $X = PY$. In co-ordinates,

$$X_j = P_{j1}Y_1 + \ldots P_{jp}Y_p, \qquad j = 1, \ldots, p.$$

Set $F_j = \frac{Y_j}{\sqrt{\lambda_j}}$ for $j = 1, \ldots, m$, then the $(F_j)_{j=1}^m$ are uncorrelated and $\mathrm{Var}(F_j) = 1$ for each $j = 1, \ldots, m$.

Let

$$A_{jk} = \sqrt{\lambda_k}P_{jk} \qquad j = 1, \ldots, p, \qquad k = 1, \ldots m.$$

Let $\epsilon_a = \sum_{k=m+1}^p A_{ak}Y_k$. Then, for $j = 1, \ldots, p$,

$$X_j = A_{j1}F_1 + \ldots + A_{jm}F_m + \epsilon_j, \qquad j = 1, \ldots, p.$$

The $F_1, \ldots, F_m$ are uncorrelated factors with $\mathrm{Var}(F_j) = 1$ for all $j = 1, \ldots, m$.

Definition 2.5 (Specificity). *The quantity $Var(\epsilon_a)$ is known as the* specificity *of X_a, the part of the variance that is unrelated to the common factors.*

The elements $A_{a1}, \ldots, A_{am}$ are known as the provisional *factor loadings* for variable a.

Definition 2.6 (Factor Loadings). *Once the errors $\epsilon_1, \ldots, \epsilon_p$ have been determined, along with the factors $F_1, \ldots, F_m$ to be used, the* factor loadings *for the factor a are the coefficients $A_{a1}, \ldots, A_{am}$ such that*

$$X_a = \sum_{j=1}^m A_{aj}F_j + \epsilon_a.$$

Definition 2.7 (Communality). *The* communality *of a variable X_j in a factor analysis is defined as $\sum_{k=1}^m A_{jk}^2$, where m is the number of factors. It gives the correlation between X_j and the part of X_j explained by the factors.*

An orthonormal transformation of uncorrelated variables yields uncorrelated variables. Therefore, any orthonormal transformation D yielding factors F^* given by

$$\mathbf{F}^* = D\mathbf{F}$$

will produce a suitable decomposition of X into uncorrelated factors. The second stage of the analysis is to find a rotation matrix D that produces rotated factors that are most convenient.

The last stage is to calculate the factor scores $(F^*_{j1}, \ldots, F^*_{jm})$ for each observation $j = 1, \ldots, n$.

Note that the factors produced by a principal component analysis are orthogonal (i.e. uncorrelated). In the second stage, an orthonormal transformation will preserve this feature. If other transformations are used, the factors will not be independent.

The Varimax Rotation This is the transformation taken from the *orthogonal* transforms that maximises the variance of the squared loadings; that is, choose D to maximise

$$\mathcal{V} := \frac{1}{p}\sum_{l=1}^{k}\left(\sum_{j=1}^{p} A_{jl}^4 - \left(\frac{1}{p}\sum_{j=1}^{p} A_{jl}^2\right)^2\right).$$

The logic behind this is that if this is large, then each values of A_{jk} is close to either 0 or 1, so that the variable is explained as much as possible by a single factor.

Note that, by standardisation, $\mathrm{Var}(F_j) = 1$ for all j and $\mathrm{Cov}(F_j, F_k) = 0$ for $j \neq k$. If ϵ is small (as it should be if the variables are properly explained by m factors), then the correlation structure of $\mathbf{X}$ (where the variables have been standardised) is given by

$$\mathrm{Cov}(X_j, X_k) = \mathrm{Cov}\left(\sum_{a=1}^{m} A_{ja}F_a, \sum_{b=1}^{m} A_{kb}F_b\right) = \sum_{a=1}^{m} A_{ja}A_{ka}.$$

The Value of Factor Analysis

Factor analysis is often useful for gaining *qualitative* insight into the structure of multivariate data, but it should be regarded purely as a piece of *descriptive statistics*; it has no value whatsoever for formal inferential statistics. It is not appropriate if it is carried out on a single small sample that cannot be replicated and then assuming that the factors obtained must represent underlying variables. Simulations have shown that even if a postulated factor model is correct, the chance of recovering it using the available methods is not very high.

2.8 Example: Country Employment Profiles

This example considers the percentages of people employed in nine industry sectos in various European countries in the years from 1989 to 1995. 30 countries are considered and 9 different industry sectors (X_1 = AGR: agriculture forestry and fishing, X_2 = MIN: mining and quarrying, X_3 = MAN: manufacturing, X_4 = PS: power and water supplies, X_5 = CON: construction, X_6 = SER: services, X_7 = FIN: finance, X_8 = SPS: social and personal services and X_9 = TC: transport and communications. In addition, the countries were classified as to whether they belonged to the EU, or EFTA, or were

Eastern European, or 'other'. The data is from 1995 and uses the classifications that were appropriate then. For 'USSR' read 'former USSR', for 'Yugoslavia' read 'former Yugoslavia', etc ... The data set is found on the course home page under `employment.csv`.

```
www<-"https://www.mimuw.edu.pl/~noble/courses/QPEDataScience/data/empl
oyment.csv"
employment <- read.csv(www)
View(employment)
```

	country	group	AGR	MIN	MAN	PS	CON	SER	FIN	SPS	TC
1	Belgium	EU	2.6	0.2	20.8	0.8	6.3	16.9	8.7	36.9	6.9
2	Denmark	EU	5.6	0.1	20.4	0.7	6.4	14.5	9.1	36.3	7.0
3	France	EU	5.1	0.3	20.2	0.9	7.1	16.7	10.2	33.1	6.4
4	Germany	EU	3.2	0.7	24.8	1.0	9.4	17.2	9.6	28.4	5.6
5	Greece	EU	22.2	0.5	19.2	1.0	6.8	18.2	5.3	19.8	6.9
6	Ireland	EU	13.8	0.6	19.8	1.2	7.1	17.8	8.4	25.5	5.8
7	Italy	EU	8.4	1.1	21.9	0.0	9.1	21.6	4.6	28.0	5.3
8	Luxemb.	EU	3.3	0.1	19.6	0.7	9.9	21.2	8.7	29.6	6.8
9	Netherl.	EU	4.2	0.1	19.2	0.7	0.6	18.5	11.5	38.3	6.8
10	Portugal	EU	11.5	0.5	23.6	0.7	8.2	19.8	6.3	24.6	4.8
11	Spain	EU	9.9	0.5	21.1	0.6	9.5	20.1	5.9	26.7	5.8
12	U.K.	EU	2.2	0.7	21.3	1.2	7.0	20.2	12.4	28.4	6.5
13	Austria	EFTA	7.4	0.3	26.9	1.2	8.5	19.1	6.7	23.3	6.4
14	Finland	EFTA	8.5	0.2	19.3	1.2	6.8	14.6	8.6	33.2	7.5
15	Iceland	EFTA	10.5	0.0	18.7	0.9	10.0	14.5	8.0	30.7	6.7
16	Norway	EFTA	5.8	1.1	14.6	1.1	6.5	17.6	7.6	37.5	8.1
17	Sweden	EFTA	3.2	0.3	19.0	0.8	6.4	14.2	9.4	39.5	7.2
18	Switzerl.	EFTA	5.6	0.0	24.7	0.0	9.2	20.5	10.7	23.1	6.2
19	Albania	Eastern	55.5	19.4	0.0	0.0	3.4	3.3	15.3	0.0	3.0
20	Bulgaria	Eastern	19.0	0.0	35.0	0.0	6.7	9.4	1.5	20.9	7.5
21	Czech/Sl.	Eastern	12.8	37.3	0.0	0.0	8.4	10.2	1.6	22.9	6.9
22	Hungary	Eastern	15.3	28.9	0.0	0.0	6.4	13.3	0.0	27.3	8.8
23	Poland	Eastern	23.6	3.9	24.1	0.9	6.3	10.3	1.3	24.5	5.2
24	Romania	Eastern	22.0	2.6	37.9	2.0	5.8	6.9	0.6	15.3	6.8
25	USSR	Eastern	18.5	0.0	28.8	0.0	10.2	7.9	0.6	25.6	8.4
26	Yugoslav.	Eastern	5.0	2.2	38.7	2.2	8.1	13.8	3.1	19.1	7.8
27	Cyprus	Other	13.5	0.3	19.0	0.5	9.1	23.7	6.7	21.2	6.0
28	Gibraltar	Other	0.0	0.0	6.8	2.0	16.9	24.5	10.8	34.0	5.0
29	Malta	Other	2.6	0.6	27.9	1.5	4.6	10.2	3.9	41.6	7.2
30	Turken	Other	44.8	0.9	15.3	0.2	5.2	12.4	2.4	14.5	4.4

Firstly, the sample correlation matrix for the standardised variables for the nine industries may be obtained:

```
> cormat <- cor(employment[,3:11])
> cormat
           AGR         MIN         MAN          PS         CON         SER
AGR  1.0000000  0.31606875 -0.25438889 -0.38235660 -0.34861031 -0.60471243
MIN  0.3160688  1.00000000 -0.67193466 -0.38737805 -0.12902071 -0.40654843
MAN -0.2543889 -0.67193466  1.00000000  0.38789059 -0.03445846 -0.03294004
PS  -0.3823566 -0.38737805  0.38789059  1.00000000  0.16479638  0.15498141
CON -0.3486103 -0.12902071 -0.03445846  0.16479638  1.00000000  0.47308319
SER -0.6047124 -0.40654843 -0.03294004  0.15498141  0.47308319  1.00000000
FIN -0.1757533 -0.24805846 -0.27374053  0.09430991 -0.01802316  0.37928368
SPS -0.8114755 -0.31641839  0.05028408  0.23774016  0.07200705  0.38798122
TC  -0.4890732  0.04363923  0.24283766  0.10527250 -0.05581358 -0.08413452
            FIN         SPS          TC
AGR -0.17575329 -0.81147553 -0.48907320
MIN -0.24805846 -0.31641839  0.04363923
MAN -0.27374053  0.05028408  0.24283766
PS   0.09430991  0.23774016  0.10527250
CON -0.01802316  0.07200705 -0.05581358
SER  0.37928368  0.38798122 -0.08413452
FIN  1.00000000  0.16601516 -0.38953393
SPS  0.16601516  1.00000000  0.47766783
TC  -0.38953393  0.47766783  1.00000000
```

The eigenvalues and eigenvectors may be obtained by:

```
> ev <- eigen(cormat)
> ev
$values
[1] 3.113358e+00 1.808358e+00 1.496866e+00 1.063671e+00 7.102833e-01
3.113942e-01
[7] 2.927508e-01 2.033110e-01 7.945502e-06

$vectors
            [,1]          [,2]        [,3]        [,4]        [,5]        [,6]
 [1,]  0.5115354  0.0232344722 -0.27813086  0.01485514 -0.02439567 -0.04269187
 [2,]  0.3747526 -0.0007024389  0.51486608  0.11531920  0.34654797  0.19857793
 [3,] -0.2461080 -0.4314434990 -0.50242351  0.05793390 -0.23361655 -0.03084313
 [4,] -0.3159322 -0.1085180547 -0.29430765  0.02293593  0.85439646  0.20623295
```

```
 [5,] -0.2212554  0.2441275595  0.06938066  0.78219676  0.06182311 -0.50302464
 [6,] -0.3812441  0.4090143576  0.06449693  0.16822359 -0.26685196  0.67268477
 [7,] -0.1309033  0.5524403796 -0.09429308 -0.49138387  0.13088760 -0.40566601
 [8,] -0.4283699 -0.0552823165  0.36065300 -0.31507798 -0.04512635 -0.15820227
 [9,] -0.2063304 -0.5161419256  0.41300662 -0.04165603 -0.02322350 -0.14187821
             [,7]        [,8]       [,9]
 [1,]  0.16697139  0.53949206 0.58212143
 [2,] -0.21599546 -0.44673451 0.41883163
 [3,] -0.23891313 -0.43014036 0.44707205
 [4,]  0.06108782  0.15521495 0.03031326
 [5,]  0.01933405  0.03159870 0.12874070
 [6,] -0.17317838  0.20281109 0.24507813
 [7,] -0.45674441 -0.02535405 0.19070896
 [8,]  0.62182700 -0.04639853 0.41013704
 [9,] -0.48888289  0.50530953 0.06084925
```

The eigenvalues, with the percentages of the total of nine in parentheses, are
3.112(34.6%), 1.809(20.1%), 1.496(16.6%),
1.063(11.8%), 0.710(7.9%), 0.311(3.5%), 0.293(3.3%), 0.204(2.3%), 0.000(0.0%).
The last value is necessarily zero, because the data is *percentages* of the workforce, which necessarily adds up to 100. Therefore, although there are 9 variables, there are only 8 *free* variables. This information may be obtained as follows:

```
> pca <- prcomp(employment[,3:11],scale = TRUE)
> summary(pca)
Importance of components:
                          PC1    PC2    PC3    PC4     PC5    PC6     PC7
PC8      PC9
Standard deviation     1.7645 1.3448 1.2235 1.0313 0.84278 0.5580 0.54106
0.45090 0.002819
Proportion of Variance 0.3459 0.2009 0.1663 0.1182 0.07892 0.0346 0.03253
0.02259 0.000000
Cumulative Proportion  0.3459 0.5469 0.7132 0.8314 0.91028 0.9449 0.97741
1.00000 1.000000
```

It is a matter of judgement whether or not to use 4 or 5 components. The first 4 components account for 83% of the variation; the first 5 account for over 90% of the variation. From the eigenvectors,

$$Z_1 = 0.51(AGR) + 0.37(MIN)$$
$$-0.25(MAN) - 0.31(PS) - 0.22(CON) - 0.38(SER) - 0.13(FIN) - 0.42(SPS) - 0.21(TC),$$

$$Z_2 = -0.02(AGR) + 0.00(MIN)$$
$$+0.43(MAN) + 0.11(PS) - 0.24(CON) - 0.41(SER) - 0.55(FIN) + 0.05(SPS) + 0.52(TC).$$

and the others similarly.

The first component contrasts (AGR) and (MIN) on the one hand with (MAN), (PS), (CON), (SER), (FIN), (SPS) and (TC) on the other hand.
The second component gives little or no weight to (AGR), (MIN), (SPS) and constrasts (MAN), (PS), (TC) with (CON), (SER), (FIN).

Interpretations for the other components may be derived similarly.

While PC1 and PC2 are uncorrelated when taken with respect to the *whole* data set, ignoring the classifications, it is interesting to plot PC1 against PC2, using different symbols for the four categories (Western EU, EFTA, Eastern European, Other). When `prcomp` is used and the results stored in `pca`, the principal component values are stored under `pca$x`. They may be added to the data frame in the following way:

```
> emp <- employment
> emp$pc1 <- pca$x[,1]
> emp$pc2 <- pca$x[,2]

> library("ggplot2")
> qplot(pc2,pc1,colour=group,data=emp)
```

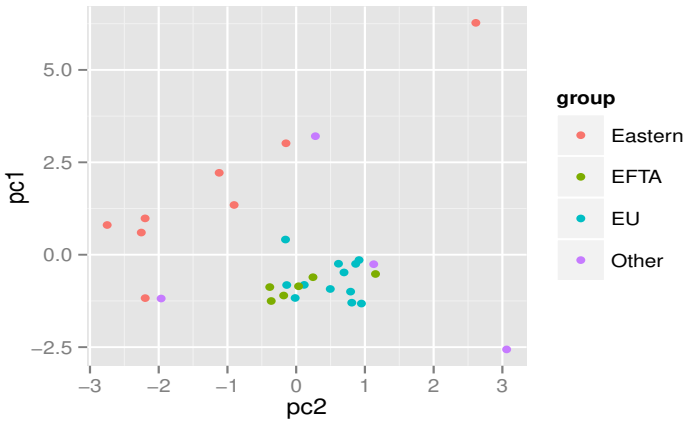

Figure 2.3: First two principal components for employment data, coloured by group

The analysis may be continued to give a *factor analysis*. There are four eigenvalues greater than 1 for the standardised variables in the principal component analysis, so the 'rule of thumb' suggests that *four* factors are appropriate using (initially) $F_j = \frac{Z_j}{\sqrt{\lambda_j}}$, giving

$$X_j = A_{j1}F_1 + A_{j2}F_2 + A_{j3}F_3 + A_{j4}F_4 + \epsilon_j, \ldots j = 1, \ldots, 9.$$

It is useful if each variable can be expressed in terms of as few factors as possible. The next step is therefore to try a rotation, which keeps the factors uncorrelated. That is, $\mathbf{F}^* = \Theta\mathbf{F}$, where Θ is a rotation matrix, which tries to ensure that for each variable the *loading* is weighted as much as possible towards *one predominant* factor. The varimax seems to work quite well.

```
> install.packages("GPArotation")
>library("GPArotation")
>install.packages("psych")
>library("psych")
> fit <- principal(emp[,3:11], nfactors=4, rotate="varimax")
> fit
Principal Components Analysis
Call: principal(r = emp[, 3:11], nfactors = 4, rotate = "varimax")
Standardized loadings (pattern matrix) based upon correlation matrix
      RC1   RC3   RC2   RC4   h2    u2
AGR -0.85 -0.27 -0.10 -0.36 0.93 0.068
MIN -0.11 -0.86 -0.30 -0.10 0.85 0.152
MAN  0.03  0.89 -0.32 -0.09 0.91 0.093
PS   0.19  0.64  0.04  0.14 0.46 0.538
CON  0.02  0.04 -0.08  0.95 0.92 0.082
SER  0.35  0.15  0.48  0.65 0.79 0.209
FIN  0.08  0.00  0.93 -0.01 0.88 0.125
SPS  0.91  0.12  0.18  0.04 0.88 0.123
TC   0.73  0.03 -0.56 -0.14 0.87 0.129

                       RC1  RC3  RC2  RC4
SS loadings           2.26 2.05 1.66 1.51
Proportion Var        0.25 0.23 0.18 0.17
Cumulative Var        0.25 0.48 0.66 0.83
Proportion Explained  0.30 0.27 0.22 0.20
Cumulative Proportion 0.30 0.58 0.80 1.00
```

For the employment data, this yields the model (where the communality is indicated on the right)

$$X_1 = -\mathbf{0.85}F_1^* - 0.10F_2^* - 0.27F_3^* - 0.36F_4^* + \epsilon_1 \qquad 0.93$$

$$X_2 = -0.11F_1^* - 0.30F_2^* - \mathbf{0.86}F_3^* - 0.10F_4^* + \epsilon_2 \qquad 0.85$$

$$X_3 = 0.03F_1^* - 0.32F_2^* + \mathbf{0.89}F_3^* - 0.09F_4^* + \epsilon_3 \qquad 0.91$$

$$X_4 = 0.19F_1^* + 0.04F_2^*\mathbf{0.64}F_3^* + 0.14F_4^* + \epsilon_4 \qquad 0.46$$

$$X_5 = 0.02F_1^* - 0.08F_2^* + 0.04F_3^* + \mathbf{0.95}F_4^* + \epsilon_5 \qquad 0.92$$

$$X_6 = 0.35F_1^* + 0.48F_2^* + 0.15F_3^* + \mathbf{0.65}F_4^* + \epsilon_6 \qquad 0.79$$

$$X_7 = 0.08F_1^* + \mathbf{0.93}F_2^* + 0.00F_3^* - 0.01F_4^* + \epsilon_7 \qquad 0.87$$

$$X_8 = \mathbf{0.91}F_1^* + 0.18F_2^* + 0.12F_3^* + 0.04F_4^* + \epsilon_8 \qquad 0.88$$

$$X_9 = \mathbf{0.73}F_1^* - \mathbf{0.57}F_2^* + 0.03F_3^* - 0.14F_4^* + \epsilon_9 \qquad 0.87.$$

The 'varimax' rotation has conveniently expressed each variable in terms of a *predominant* factor plus other less important factors for that variable. The only variable that seems to have *two* predominant factors is X_9.

Following the varimax rotation, the results are interpreted by considering the four factors in terms of the variables. From that, it may be possible to give useful labels to each factor.

Here, it is clear that F_1^* has high positive loadings for X_1 (agriculture, forestry and fishing) and high negative loadings for X_8 (social and personal services) and X_9 (transport and communications). Therefore, F_1^* measures the extent to which people are employed in agriculture rather than services and communications. It could be labelled 'rural industries rather than social service and communication'.

Factor F_2^* turns out to have a high negative loading for X_7 (finance). The loading for X_9 (transport and communication) seems to be higher than the others. A possible labelling could be 'lack of finance industries'. □

Chapter 3

Cluster Analysis

3.1 Introduction

Cluster Analysis is the most well-known example of *unsupervised learning*. Unsupervised learning means that we do not have a training set of examples where the classification is known; we learn the number of the classes and assign objects to classes purely using the data. It is a tool for arranging large quantities of multivariate data into natural groups.

Consider a sample of n independent observations on p variables, $(x_{j1}, \ldots, x_{jp})_{j=1}^{n}$. The data may come from various groups, but the number or size of each group may not be known in advance. The idea of cluster analysis is to derive, from the data, the characteristics of the various groups from which the outcomes in the sample came.

Absolutely nothing is assumed about the data, except that the n observations come from K groupings (where K is not known in advance), where for group j, $\mathbb{E}[\underline{X}] = \underline{\mu}_j$. The number K and the vectors $\underline{\mu}_j$, $j = 1, \ldots, K$ have to be estimated from the data through cluster analysis. In other words, cluster analysis only attempts to locate the collection of different average values and assigns the observations to the groups determined by these averages.

Cluster analysis techniques are purely numerical, measuring distances and assigning observations to groups based on the distance measures; the techniques do not involve statistics. On the one hand, the techniques do not seem to be very powerful; in a few cases, where the data seems to have some very pronounced centres, cluster analysis techniques can locate them. Having said that, they can work well for large data sets.The important advantage of clustering is that very few modelling assumptions are required.

There are numerous ways of clustering a data set of n independent observations on p correlated variables. We consider three:

1. **Clustering Observations** This is the usual use of the term 'clustering'; we divide the observations into K groups.

2. **Clustering Variables** In situations where there are large numbers of variables, many of which are highly correlated with each other (for example gene expression levels), We may wish to

partition the p *variables* into K distinct groups. For each cluster of variables, a *representative* variable is taken and used.

3. **Two-way clustering** In some situations, both variables and observations may be clustered *together*. For example, we may want to cluster both genes and tissue samples at the same time to see which subset of genes is most closely related to which subset of tissue samples.

In this lecture, we concentrate on the first of these tasks, clustering *observations*.

Warning In many situations, cluster analysis is a total waste of time; the algorithms will only detect the simplest patterns. Consider, for example, a bivariate data set, coming from two random variables; the first uniformly distributed in the circle of radius 1 and the second uniformly distributed in the annulus $\{(x,y)|1.5 \leq x^2+y^2 \leq 2.5\}$. A cluster analysis is unlikely to detect that there are two different groups.

3.2 Distance and Dissimilarity Measures

Let x_{ik} denote observation i on variable k, where there are n observations on p variables. Usually, *quantitative* variables are *standardised* before the analysis;

$$s_k = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(x_{jk}-\bar{x}_{.k})^2}$$

and

$$y_{ik} = \frac{x_{ik}-\bar{x}_{.k}}{s_k}.$$

For quantitative variables, a common distance measure in cluster analysis is the *Euclidean distance*:

$$d_{ij} = \sqrt{\sum_{k_1}^{p}(y_{ik}-y_{jk})^2}.$$

The statistical distance between two p-variate observations $\underline{x}$ and $\underline{y}$ is usually of the form

$$d(\underline{x},\underline{y}) = \sqrt{(\underline{x}-\underline{y})^t S^{-1}(\underline{x}-\underline{y})}$$

where S is the sample covariance matrix. This is the simply the *Mahalanobis* distance. The Penrose distance may be used instead. There are many choices of distance.

In situations where the data is categorical, such a distance measure is not meaningful. It is then useful to try and compare data by the presence and absence of certain characteristics in the p variables. The distance is then the number of differences. For example, suppose $p = 5$ and the five variables for items i and k are coded as

	1	2	3	4	5
i	1	0	0	1	1
k	1	1	0	1	0

Then the distance $d(i,k) = 2$.

In some cases, a $1-1$ match is stronger evidence of similarity than a $0-0$ match. For example, if two people can understand ancient Greek, this is stronger evidence of a similarity than if both of them do not read ancient Greek. Then the following table is useful:

$i\backslash k$	1	0	
1	a	b	$a+b$
0	c	d	$c+d$
	$a+c$	$b+d$	$p = a+b+c+d$

This leads to the definition of *similarity*, which is a number that lies between 0 and 1, equal to 1 if the two objects are identical and equal to 0 if the two objects have nothing in common. The following are common similarity measures that are used depending on the situation.

1. $s_{ik} = \frac{a+d}{p}$ if equal weight is given to $1-1$ and $0-0$ matches.
2. $s_{ik} = \frac{2(a+d)}{2(a+d)+b+d}$ if double weight is given to $1-1$ matches than $0-0$ matches
3. $s_{ik} = \frac{a}{p}$ if the $0-0$ matches do not contribute to similarity.
4. $s_{ik} = \frac{a}{a+b+c}$ if the $0-0$ matches are considered irrelevant.
5. $s_{ik} = \frac{2a}{2a+b+c}$ Double weight for $1-1$ matches, $0-0$ matches irrelevant
6. $s_{ik} = \frac{a}{a+2(b+c)}$ $0-0$ matches irrelvant, double weight for unmatched pairs.
7. $s_{ik} = \frac{a}{a+b}$ Ratio of matches to mis-matches, the $0-0$ matches excluded.

The *dissimilarity* is defined as $d_{ik} = 1 - s_{ik}$.

3.3 Clustering Techniques

We first consider *hierarchical* metohds for clustering. These methods start by computing the distances, or dissimilarities (depending on the type of data) from each object to every other object in the sample and storing this as a dissimilarity matrix. Groups are then formed by a process of *agglomeration* or *division*. Agglomeration techniques start with all individuals in groups of size 1. Close groups are merged until the process is complete. With division, all objects start in a single group. This group is split into two and then the subgroups are split into pairs until all the elements in each group are sufficiently similar and the groups as a whole are sufficiently far apart from each other.

There is also a *partitioning* approach to cluster analysis. For algorithms that fall into this class, objects are permitted to move in and out of groups at various stages in the analysis. First, some more or less arbitrary group centres are chosen. Objects are allocated to the nearest group centre. New centres are then calculated, representing the averages of the objects in the groups. An object is then moved to a new group if it is closer to that group's centre than the centre of the present group. Groups that are close together are merged; groups that are spread out are split, following some defined rules. The process continues iteratively until stability is achieved with a predetermined number of groups. For each run, the number of groups is fixed in advance. This is repeated over a range of values for the group number and the output is the most successful one, where some criteria for goodness of fit are given.

3.4 Hierarchic Methods

First, a matrix of distances between each object is calculated. For example, if there are 5 objects, the matrix might be

	object				
object	1	2	3	4	5
1	–				
2	2	–			
3	6	5	–		
4	10	9	4	–	
5	9	8	5	3	–

Firstly, each object is considered to be in a group on its own. Then run through all possible distances, from lowest to highest. As the distance is increased, a criterion for deciding whether groups should merge is decided upon.

Single Linkage With *single linkage*, also known as *nearest neighbour linkage*, the groups are merged as the distance increases if one of the objects in a group has distance less than or equal to that distance from an object in another group. In the example above, no merging takes place for a distance less than 2. At 2, (1) and (2) merge to form a new group, $(1,2)$. The groups, formed according to *nearest neighbour linkage*, are given below, as the distance increases.

distance	groups
$0,1$	$(1),(2),(3),(4),(5)$
2	$(1,2),(3),(4),(5)$
3	$(1,2),(3),(4,5)$
4	$(1,2),(3,4,5)$
5	$(1,2,3,4,5)$

This may be expressed as a *dendrogram*.

Complete Linkage With *complete linkage*, also known as *furthest neighbour linkage*, two groups merge only if the most distant members of the two groups are close enough. Starting with each member in a group on its own, the groups formed under furtherst neighbour linkage, as the distance is increased and using the groups already formed, are given below.

distance	groups
$0,1$	$(1),(2),(3),(4),(5)$
2	$(1,2),(3),(4),(5)$
3	$(1,2),(3),(4,5)$
5	$(1,2),(3,4,5)$
10	$(1,2,3,4,5)$

As with all hierarchical techniques, the dendrogram is a natural way to visualise it graphically. With *group average linkage*, two groups merge if the *average* distance between them is small enough. Merging groups based on *average* distance, letting the distance range through the continuum, yields

distance	groups
0	$(1),(2),(3),(4),(5)$
2	$(1,2),(3),(4),(5)$
3	$(1,2),(3),(4,5)$
4.5	$(1,2),(3,4,5)$
7.8	$(1,2,3,4,5)$

Average Linkage Again, the input into the algorithm may be distances or dissimilarities. The algorithm begins by searching for the closest objects. Suppose these are U and V. These are then merged to form a cluster (UV).

Now, suppose that (UV) and W are clusters. The distance between the two clusters is defined as

$$d_{(UV)W} = \frac{1}{N_{(UV)}N_W}\sum_{ik} d_{ik}.$$

The Ward Clustering Algorithm

The Ward Clustering Algorithm does not measure the distance between groups in terms of nearest neighbours, furthest neighbours or group centres. Rather, it tries to join groups that do not increase a given measure of heterogeneity too much. That is, the distance is related to the *variance* within the group; two groups are merged if the within-group variance of the merged group is lower than a specified level.

The Ward Clustering algoritm uses the following distance function: let the distance between groups containing *single* multivariate observations $\underline{x}$ and $\underline{y}$ be

$$d(\underline{x},\underline{y}) = \sqrt{\sum_{j=1}^{p}(x_j - y_j)^2}.$$

Then, for larger groups, the distance between them is computed inductively. If two *distinct* groups P and Q are merged to form a group $P \cup Q$, the distance between $P \cup Q$ and another group R, in terms of the previously calculated $d(P,R)$ and $d(Q,R)$ is given by

$$d(P \cup Q, R) = \frac{n_R + n_P}{n_R + n_P + n_Q} d(P,R) + \frac{n_R + n_Q}{n_R + n_P + n_Q} d(Q,R) - \frac{n_R}{n_R + n_P + n_Q} d(P,Q),$$

where n_R, n_P, n_Q are the numbers in R, P, Q respectively Two groups are merged when the Ward distance between them reaches the specified level.

Motivation The *inertia* of a group R is defined as

$$I_R = \frac{1}{n_R} \sum_{j=1}^{n_R} d^2(\underline{x}^{(j)}, \bar{\underline{x}}_R);$$

in other words, the inertia is the average squared distance from each group member to the group centre. When two distinct groups P and Q are merged, the new group $P \cup Q$ has a larger inertia than the sum of inertias of P and Q. That is, if P and Q are disjoint, then a straightforward computation yields that:

$$I_{P \cup Q} \geq I_P + I_Q.$$

More precisely, if d is the Ward measure, then a straightforward computation gives the following expression for $I_{P \cup Q} - I_P - I_Q$:

$$\Delta(P,Q) := I_{P \cup Q} - (I_P + I_Q) = \frac{n_Q n_P}{n_Q + n_P} d^2(P,Q).$$

With the Ward algorithm, therefore, the next pair of groups to be joined as the distance parameter increases is the pair of groups that currently gives the smallest value for $\Delta(P,Q)$.

3.5 Divisive Analysis (`diana`)

Divisive hierarchic methods are less common. Objects start in a single group. Then, the object furthest from the mean is split off. Then objects from the main group are moved to the new group if they are closer to the new group as it stands, than from the main group. The pair of groups from the original group has been established when all objects are closest to the centre of the group they are in.The most-used divisive hierarchical clustering procedure was proposed by MacNaughton-Smith, Williams, Dale, and Mockett (1964).

Suppose the algorithm has reached a stage where there are several clusters. Let the current collection of clusters be $\mathcal{C}$. The cluster $C \in \mathcal{C}$ that has the largest average dissimilarity between an item and the remaining items in the cluster is chosen.

A *splinter* group (say cluster A) is computed, where $A \subset C$. The splinter group is initiated by extracting the item that has the largest average dissimilarity from all the other items in A.

Let $B = C \backslash A$. For each of the items in B, we compute (1) the average dissimilarity between that item and all the other items in B and (2) the average dissimilarity between that item and all the items in A. We compute the difference (1)-(2) for each item in B. If all the differences are negative, we stop. If any of the differences are positive, we move the item with the largest (1)-(2) into A and then repeat until we have a division of C into two clusters, A and B. This is continued recursively; after each division, the largest average distance between each item and the other items in the cluster is recorded.

3.6 Non-hierarchical Clustering Methods

Non-hierarchical methods start from either (1) an initial partition of items into groups or (2) an initial set of seed points, which will form the nuclei of clusters. One way to start is to randomly select seed points from among the items or to randomly partition them into groups. This eliminates bias.

3.6.1 K-means method

This algorithm proceeds as follows:

1. Partition the items into K initial clusters.
2. Proceed through the list of items assigning an item to the cluster whose mean is nearest. Distance is usually computed using Euclidean measure with standardised variables. Recalculate the mean for the cluster receiving the new item and for the cluster losing the item.
3. Repeat step 2 until no more reassignments take place.

Example 3.1.

Consider 4 measurements, A, B, C, D of the two variables (X_1, X_2). The data is

	x_1	x_2
A	5	3
B	−1	1
C	1	−2
D	−3	−2

Suppose they are to be divided into two clusters. To start with, *randomly* assign them to two clusters. For example, suppose this were AB and CD. Then the co-ordinates for the cluster centres (means) are

$$(AB) : (\bar{x}_1, \bar{x}_2) = (\frac{5 + (-1)}{2}, \frac{3 + 1}{2}) = (2, 2)$$

$$(CD) : (\bar{x}_1, \bar{x}_2) = (\frac{1 + (-3)}{2}, \frac{-2 + (-2)}{2}) = (-1, -2).$$

Using Euclidean distance, the distance from each item to each group centre is computed and then the group centre updated to:

$$\bar{x}_{i,new} = \begin{cases} \frac{n\bar{x}_i + x_{ji}}{n+1} & \text{item j added} \\ \frac{n\bar{x}_i - x_{ji}}{n-1} & \text{item j removed} \\ \bar{x}_i & \text{no change} \end{cases}$$

In the first round, if A were re-assigned to group (ACD), the group centres would then be

$$\bar{\underline{x}}_{ACD} = (1, -\frac{1}{3}), \quad \bar{\underline{x}}_B = (-1, 1).$$

Since A is closer (using Euclidean distance) to the centre of AB than to ACD, it is not moved. By proceeding, the final partition is (A), (BCD).

3.6.2 K-medoids

A *medoid* is simply a representative object. The *K-medoids algorithm* searches for K 'representative objects' rather than the centroids. A dissimilarity-based distance is used instead of squared-Euclidean distance. Because it minimises a sum of dissimilarities instead of a sum of squared Euclidean distances, it is more robust to data anomalies such as outliers and missing values.

K-medoids starts with the proximity matrix D and an initial configuration into K clusters. The *representative object* is the object that minimises the total dissimilarity to all other items in the cluster. From this point onwards, the role of centroids in K-means is replaced by represenative objects in K-medoids and the sum of squares of Euclidean distances replaced by the sum of squares of dissimilarities. With these changes, the algorithm proceeds in the same way.

3.6.3 Partitioning Around Medoids (`pam`)

This clustering method is a modification of the *K-medoids algorithm*.The `pam` algorithm modifies K-medoids by introducing a swapping strategy by which the medoid of a cluster is replaced by the item in the cluster that minimises the value of an objective function that may be different from the sum of squares of the dissimilarities.

K-medoids and `pam` run well on small data sets, but are not efficient enough to use for clustering large data sets.

3.6.4 Silhouette Plot

With `kmeans` and `pam`, the result of the partition can be represented graphically in a so-called *silhouette plot*. Suppose the data is split into K clusters. Let $c(i)$ denote the cluster of item i. Let c be some cluster different from $c(i)$ and $d(i, c)$ the average dissimilarity between i and items of c. We compute $d(i, c)$ for all clusters other than $c(i)$. let

$$b_i = \min_{c \neq c(i)} d(i, c)$$

and a_i the average dissimilarity between i and all items of the same cluster. The ith silhouette value is

$$s_{iK} = \frac{b_i - a_i}{\max a_i, b_i}.$$

Large positive values indicate that item i is well clustered, large negative that it is badly clustered. A *silhouette plot* is a plot of these after they have been ranked in decreasing order for each cluster, where the length of the bar is s_{iK}.

The average silhouette width may also be used to find the best value of K, by maximising $\overline{s}_K$. The *silhouette coefficient* is $s_C = \max_K \overline{s}_K$.

3.7 Self Organising Maps (SOM)

The *self-organising map* algorithm is due to Kohonen (1982). The basic idea is *data reduction*. A set of n observations in (say) $\mathbb{R}^p$ is reduced to a number $K = K_1 \times K_2$ nodes, where $K << n$. These nodes are represented on a 2-d grid. To each node m is assigned a *representative value* $x_m \in \mathbb{R}^p$, or whatever the state space for the observations is. Observations are assigned to a node m if they lie within the neighbourhood of the representative value x_m. These nodes may be considered as nodes of a neural network and there is a link between two nodes a and b if their representative values lie within each others neighbourhood.

Two versions of the algorithm are available; the *on-line* version and the *batch* version. The end product of SOM, after a large number of iterative steps, is a graphical image known as an SOM plot. This is displayed in output space and consists of a grid (or network) of a large number of inter-connected *nodes* or artificial neurons. The SOM algorithm has much in common with K-means.

3.7.1 On-Line Version

For a 2-d SOM, firstly set up the map size $K_1 \times K_2$ (select K_1 and K_2 usually larger than intended for the final output). Let $\mathcal{K}$ denote the space of labels; $k = (i, j) \in \mathcal{K}$ for $i \in \{1, \ldots, K_1\}$ and $j \in \{1, \ldots, K_2\}$. For each $k \in \mathcal{K}$, choose at random a representative value $m_k \in \mathbb{R}^p$ (or whatever the input space is) for the standardised data matrix.

Firstly, the data set is standardised so that the columns have mean 0 and variance 1. At each stage, an input vector X is randomly selected and assigned to k^* where

$$k^* = \text{argmin}_k \|X - m_k\|.$$

k^* is declared to be the winning node. We then look at nodes that are 'neighbours' of the winning node. Two nodes k and k' are neighbours if the Euclidean distance between m_k and $m_{k'}$ is less than a given threshold c. We update the value of m_k for the winning node and for all its neighbours using:

$$m_k \leftarrow m_k + \alpha(X - m_k) \qquad k \in N(k^*).$$

Here $\alpha \in (0, 1)$ is the *learning factor*.

A useful rule of thumb is to run the algorithm for $500 \times K_1 \times K_2$ steps.

A *distance weighted* strategy is more popular:

$$m_k \leftarrow m_k + \alpha h_k(X - m_k)$$

where

$$h_k = \exp\left\{\frac{-\|m_k - m_{k^*}\|^2}{2\sigma^2}\right\}\mathbf{1}_{\{k\in N_c(k^*)\}}$$

where $N_c(k)$ denotes the neighbourhood of k with parameter c. Values of c, α and σ are provided by the user, but may change during the process.

Batch Version m_k is updated by listing all items X_i already examined whose $m_{k^*} \in N_c(k)$. These are averaged over and the average is used for the update. This turns out to be much faster.

3.8 Implementation in R

We'll see whether or not a cluster analysis on the European Employment Data can recover the four categories.

- step 1: standardise the nine variables.
- step 2: compute the Euclidean distances between *all* pairs of countries, using the standardised variables.
- step 3: construct a dendrogram using, for example, the agglomerative, nearest neigbour, hierarchic process.

```
> www =
"https://www.mimuw.edu.pl/~noble/courses/QPEDataScience/data/
employment.csv"
> employment <- read.csv(www,header=T)
> y<-scale(employment[,3:11])
> emp<-employment
> emp[,3:11]<-y
```

The 'Ward' method may be implemented as follows:

```
> d <- dist(y, method = "euclidean")
> fit <- hclust(d, method="ward.D2")
> plot(fit,labels=employment$group)
> rect.hclust(fit, k=4, border="red")
```

The plot is shown in figure 3.1.
At a distance of about 1, there are four clusters. The performance of the clustering algorithm, using the Ward distance is clear from the plot. Other distance measures give less satisfactory results.

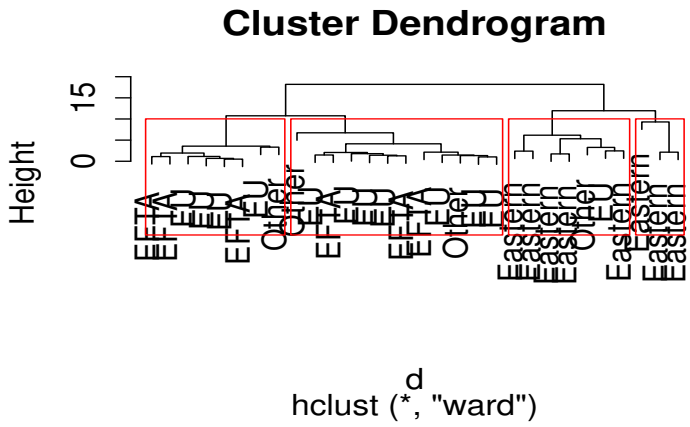

Figure 3.1: Dendrogram for employment data

Cluster Analysis: Example

The package **cluster** is extremely useful and contains most of the methods (for example `diana` and `pam`) discussed in the lecture. Install this package and check it out.

Consider the data set `satimage.txt` in the course data directory. Since 1972, Landsat satellites orbiting the Earth have been using a combination of scanning geometry, satellite orbit and Earth rotation to collect high-resolution multispectral digital information for detecting and monitoring different types of land surface cover characteristics. The 36 variables in the data set are arranged in groups of 4 spectral bands: 1,2,3 and 4, covering each pixel of a 3×3 neighbourhood (top-left TL, top-centre TC, top-right TR, centre-left CL, centre-right CR, centre-centre CC, bottom-left BL, bottom-centre BC, bottom-right BR. The centre pixel CC of each of the 4435 neighbourhoods is classified into one of six classes: 1-red soil, 2-cotton crop, 3-grey soil, 4-damp grey soil, 5-soil with vegetation stubble and 7-very damp grey soil. There is no class 6.

Warning The data file is rather large and therefore the clustering methods are rather slow and take a long time. You should select a random subset of the data (say 500 observations from the data set) and work with these. Make sure your sample contains sufficient numbers of observations from each class.

1. Do not use the class variable. Standardise the other variables and perform cluster analysis using SL (single linkage), AL-average linkage, CL-complete linkage, K-means and `pam` (partition around medioids. How do the various clustering methods perform?

2. Construct a silhouette plot for partitioning around medioids (`pam`) with $K = 6$ clusters. A *silhouette plot* is a ba plot of all the

$$s_{iK} = \frac{b_i - a_i}{\max_{j \in C} |b_j - a_j|}$$

   after they have been ranked in decreasing order, where the length of the ith bar is s_{iK}. Let $d(i,j)$ denote the dissimilarity between items i and j, then a_i is the average dissimilarity that item i has to other items in the same cluster and $b_i = \min_{c \neq c(i)} d(i,c)$ where $c(i)$ denotes the cluster containing item i and $d(i,c)$ is the average dissimilarity between item i and items in cluster c.

3. Construct a confusion table for `pam` clustering with $K = 6$ clusters.

4. Run the clustering algorithms for the `satimage.txt` data, but only using the centre pixels (i.e. the variables CC1, CC2, CC3, CC4) of each 3×3 neighbourhood. Compare your results with those obtained from the full data set.

5. There are several R packages tha deal with self organising maps. I draw attention to **kohonen**. We'll put it through its paces.

Make a 6×6 hexagonal batch-SOM plot of the Landsat satellite image data. The circles correspond to nodes and the projected points are plotted randomly within the appropriate circle to which they were deemed closest. Use the six classes of vegetation as plotting symbols.

Solution We start by loading the data:

```
> www =
"https://www.mimuw.edu.pl/~noble/courses/QPEDataScience/data/satimage.
txt"
> satellite = read.table(www,header=T)
```

The data frame is `satellite`. This data set (sadly) is too large for my small laptop, so I take a random subset of 500 rows.

```
> satdata = satellite[sample(nrow(satellite),500),]
```

and this is what we will use. Firstly, we *scale* the data; this operation centres the data (so that each column has mean 0 and scales the data (so that each column has standard deviation 1).

```
> scaleddata = sapply(satdata[,1:36],scale)
```

This operation does not return a data frame, so we make it a data frame.

```
> scaleddata = as.data.frame(scaleddata)
```

Now let us add the class variable:

```
> scaleddata$class = satdata$class
```

1. **Clustering: Illustration with Ward and PAM** Firstly, let us see how the *Ward* clustering algorithm performs.

```
> d = dist(scaleddata[,1:36],method="euclidean")
> satward=hclust(d,method="ward.D")
> plot(satward,labels=scaleddata$class)
```

There are 6 different classes in the data. We can see that we get 5 classes at a height of approximately 30. In fact, two of the classes are difficult to distinguish from each other.

Now let us try `pam`, partition around means. Let us see how it performs with 6 clusters, which is the true number.
The *partition around means* for this data is given, quite simply, by:

```
> satpam <-pam(scaleddata[,1:36],6)
```

2. **Silhouette Plot** Now try a silhouette plot:

```
> silpam = silhouette(satpam)
> plot(silpam,border=NA)
```

The `border = NA` is an RStudio issue; without this, the graph can go wrong. This indicates that the 6 cluster solution is reasonably good; not many of the entries are negative.

3. **Confusion Table** Now we make a confusion table, comparing with the value of the class variable.

```
> tabsat = table(satpam$clustering,scaleddata$class)
> tabsat
```

```
   1  2  3  4  5  7
1 37  9  0  5 17  0
2  2  0 26 28  3 36
3  0  1  0  3 35 89
4  0 59  0  0  0  0
5 71  0  0  0  2  0
6  2  0 67  8  0  0
```

We can see how well the clustering compares with the 'true' classes. The labels are clearly different, but some of the classes have been detected reasonably well. The dendrogram from the Ward clustering indicates that things are not clear cut. The confusion table results are in line with the silhouette plot.

4. **Self Organising Map** Now let us try a SOM. We need the package **kohonen**

```
> library(kohonen)
> satdat=as.matrix(scaleddata[,1:36])
> somnet = som(satdat,somgrid(7,7,"hexagonal"))
> sommap = map(somnet,satdat)
> predict <- predict(somnet,satdat)
> plot(somnet, type="mapping", col = as.integer(scaleddata$class),
+      pchs = as.integer(scaleddata$class), bgcol = predict,
+      main = "another mapping plot", shape = "straight", border = NA)
```

Chapter 4

Conditional Independence and Graphical Models

The first lecture dealt with the data matrix, geometry, distance between different objects. The second lecture dealt with *principal component analysis*, the aim of which is to *reduce the dimension of the problem*. Related to this was exploratory *Factor Analysis*, whereby the important principal components could suggest *hidden* (or latent) *factors* which influenced the variables.

We now turn to the topic of *graphical models*, where again we try to reduce the computational complexity of the problem. This time, we do this by finding and exploiting the *independence structure* between variables. We'll also touch on *causality*; the attempt to ascertain whether one variable has a *causal* influence on other variables.

The idea of a *graphical model* is that the variables are represented as nodes on a *graph* and the edges in the graph represent a *direct* link between two variables. If two variables X and Y are graphically *separated* by a set of variables S, then *all* the influence that X and Y have on each other is mediated through the variables in S; X and Y are conditionally independent given S.

The problem of ascertaining statistical independence thereby becomes a problem of graphical separation and a powerful toolbox of graphical separation algorithms becomes available.

It is important to stress that, in a graphical model, graphical separation implies conditional independence, but the converse does not (in general) hold; there are often conditional independence relations that a graphical model cannot detect.

Introduction A *graphical model* for a probability distribution over several variables is, quite simply, a graph, where the random variables correspond to the node set of the graph and each graphical separation statement implies the corresponding conditional independence statement for the random variables. The opposite (that conditional independence implies graphical separation) in general does not hold. In a system with a large numbers of variables, the task of determining graphical separation statements is, in general, computationally far less demanding than the task of determining conditional independence.

A *Bayesian network* is the representation of a probability distribution on a directed acyclic graph

(DAG). In this setting, the most useful notion of separation is *D-separation*, defined later. If a probability distribution factorises along a DAG, then D-separation statements in the DAG imply the corresponding conditional independence statements (although the reverse implication is, in general, false).

In many problems, for example gene expression data where there are thousands of variables, it may not be either possible or desirable to obtain a complete description of the dependence structure. The aim for such problems is to learn a DAG which encodes the most important features of the dependence structure. In classification problems, a complete description of the dependence structure is usually unnecessary; algorithms only locate the key features of the dependence structure to ensure accurate classification.

4.1 Conditional Independence and Factorisation

Definition 4.1 (Independence)**.** *Two random vectors X and Y are independent if their joint probability distribution factorises as*

$$\mathbb{P}_{X,Y} = \mathbb{P}_X \mathbb{P}_Y.$$

X and Y are conditionally independent given a random vector X if

$$\mathbb{P}_{X,Y,Z} = \mathbb{P}_{X|Z} \mathbb{P}_{Y|Z} \mathbb{P}_Z.$$

This is written $X \perp Y|Z$.

Example 4.1 (Binary Variables)**.**

Suppose $X_1, \ldots, X_d$ are binary variables (i.e. each takes values in $\{0,1\}$). Then the state space is $\{0,1\}^d$, which has 2^d possible configurations. To specify the probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$, we therefore need to specify $2^d - 1$ values (since all the values sum to 1).

Suppose that $X_1, \ldots, X_d$ are mutually independent. Then we only need to specify d values; we need $\mathbb{P}_{X_1}(1), \ldots, \mathbb{P}_{X_d}(1)$, since $\mathbb{P}_{X_i}(0) = 1 - \mathbb{P}_{X_i}(1)$ for $i = 1, \ldots, d$.

For large d, there is therefore much computational advantage to be gained from exploiting the independence structure between the variables. □

4.2 Definition of a Bayesian Network

Consider a probability distribution over d variables $\mathbb{P}_{X_1,\ldots,X_d}$.

Recall that for any collection of events $A_1, \ldots, A_n$,

$$\mathbb{P}(A_1 \cap \ldots \cap A_n) = \mathbb{P}(A_1) \frac{\mathbb{P}(A_1 \cap A_2)}{\mathbb{P}(A_1)} \ldots \frac{\mathbb{P}(A_1 \cap \ldots \cap A_n)}{\mathbb{P}(A_1 \cap \ldots \cap A_{n-1})}$$

so that (using the definition $\mathbb{P}(A|B) = \frac{\mathbb{P}(A\cap B)}{\mathbb{P}(B)}$):

$$\mathbb{P}(A_1 \cap \ldots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2|A_1)\ldots\mathbb{P}(A_n|A_1 \cap \ldots \cap A_{n-1}).$$

Clearly, any probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ over $\mathcal{X}$ may be factorised as

$$\mathbb{P}_{X_1,\ldots,X_d} = \mathbb{P}_{X_{\sigma(1)}} \prod_{j=2}^{d} \mathbb{P}_{X_{\sigma(j)}|X_{\sigma(1)},\ldots,X_{\sigma(j-1)}}$$

for any permutation σ of $1,\ldots,d$. Let $\mathrm{Pa}^{(\sigma)}(j) \subset \{\sigma(1),\ldots,\sigma(j-1)\}$ satisfy

- $$\mathbb{P}_{X_1,\ldots,X_d} = \mathbb{P}_{X_{\sigma(1)}} \prod_{j=2}^{d} \mathbb{P}_{X_{\sigma(j)}|\mathrm{Pa}^{(\sigma)}(j)}$$

- $$\mathbb{P}_{X_1,\ldots,X_d} = \mathbb{P}_{X_{\sigma(1)}} \prod_{j=2}^{d} \mathbb{P}_{X_{\sigma(j)}|\Theta(j)}$$

  if any $\Theta(j)$ is a *strict* subset of $\mathrm{Pa}_j^{(\sigma)}$.

Unless otherwise stated, it will be assumed that the variables are labelled in such a way that $\sigma = I$, the identity.

For $\mathrm{Pa}_j = \{l_{j,1},\ldots,l_{j,m_j}\}$, the state space of $X_{\mathrm{Pa}(j)}$ is $\mathcal{X}_{l_{j,1}} \times \ldots \times \mathcal{X}_{l_{j,m_j}}$. For discrete variables, there are $q_j = \prod_{a=1}^{m_j} k_{l_{j,a}}$ configurations. These may be labelled $(\pi_j^{(l)})_{l=1}^{q_j}$ and the parameters required for the probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ are

$$\theta_{jil} = \mathbb{P}_{X_j|X_{\mathrm{Pa}(j)}}(i|\pi_j^{(l)}) \quad j = 1,\ldots,d \quad i = 0,\ldots,k_j-1, \quad l = 1,\ldots,q_j.$$

Estimating Parameters Suppose we have an $n \times d$ data matrix $\mathbf{x}$; to estimate the parameter θ_{jil}, we use

$$\widehat{\theta}_{jil} = \frac{\text{number of appearances of } (i,\pi_j^l) \text{ configuration in } \mathbf{x}}{\text{number of appearances of } \pi_j^l \text{ configuartion in } \mathbf{x}}.$$

Factorising a Probability Distribution along a Directed Acyclic Graph The factorisation of a Bayesian network may be represented by a *Directed Acyclic Graph*. For example, if the probability distribution over X,Y,Z,W satisfies

$$\mathbb{P}_{X,Y,Z,W} = \mathbb{P}_X\mathbb{P}_{Y|X}\mathbb{P}_{Z|X}\mathbb{P}_{W|Y,Z},$$

the factorisation may be represented by the graph in Figure 4.1.

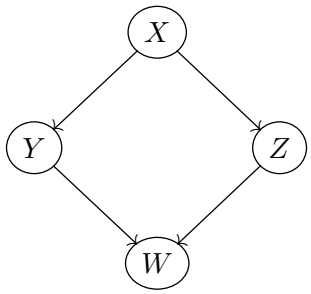

Figure 4.1: DAG representing the factorisation of a probability distribution

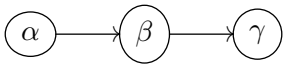

Figure 4.2: A Chain Connection

4.3 Connections in a Directed Acyclic Graph and Conditional Independence

Definition 4.2 (Instantiated)**.** *When the state of variable is known, the variable is said to be* instantiated.

Within a directed acyclic graph, there are three basic ways in which two nodes α, γ such that $\alpha \to \gamma \notin D$ and $\gamma \to \alpha \notin D$ can be connected via a third node. They are the *chain*, *fork* and *collider* connections respectively.

Chain Connections A chain connection between nodes α and γ is a connection via a node β such that the graph contains directed edges $\alpha \to \beta$ and $\beta \to \gamma$, but no edge between α and γ.

Consider a probability distribution over $(X_\alpha, X_\beta, X_\gamma)$ factorised according to the graph in Figure 4.2, as $\mathbb{P}_{X_\alpha}\mathbb{P}_{X_\beta|X_\alpha}\mathbb{P}_{X_\gamma|X_\beta}$.
Clearly, $X_\alpha \not\perp X_\gamma$ in general;

$$\mathbb{P}_{X_\alpha,X_\gamma}(x_1,x_3) = \mathbb{P}_{X_\alpha}(x_1)\sum_{x_2\in\mathcal{X}_2}\mathbb{P}_{X_\beta|X_\alpha}(x_2|x_1)\mathbb{P}_{X_\gamma|X_\beta}(x_3|x_2)$$

and, without further assumptions, this cannot be expressed in product form.

Conditioned on the instantiation $X_\beta = x_2$,

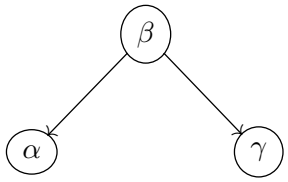

Figure 4.3: A Fork Connection

$$\begin{aligned}\mathbb{P}_{X_\alpha,X_\gamma|X_\beta}(.,.|x_2) &= \frac{\mathbb{P}_{X_\alpha,X_\beta,X_\gamma}(.,x_2,.)}{\mathbb{P}_{X_\beta}(x_2)} = \frac{\mathbb{P}_{X_\alpha}(.)\mathbb{P}_{X_\beta|X_\alpha}(x_2|.)\mathbb{P}_{X_\gamma|X_\beta}(.|x_2)}{\mathbb{P}_{X_\beta}(x_2)} \\ &= \left(\frac{\mathbb{P}_{X_\alpha}(.)\mathbb{P}_{X_\beta|X_\alpha}(x_2|.)}{\mathbb{P}_{X_\beta}(x_2)}\right)\left(\mathbb{P}_{X_\gamma|X_\beta}(.|x_2)\right) = \left(\mathbb{P}_{X_\alpha|X_\beta}(.|x_2)\right)\left(\mathbb{P}_{X_\gamma|X_\beta}(.|x_2)\right)\end{aligned}$$

where Bayes rule has been used and so $X_\alpha \perp X_\gamma|X_\beta$.

Fork Connections A *fork* connection between two nodes X_α and X_γ is a situation where there is no edge between X_α and X_γ, but there is a node X_β such that the graph contains directed edges $X_\beta \mapsto X_\alpha$ and $X_\beta \mapsto X_\gamma$. It is illustrated in Figure 4.3.

A distribution over the variables $(X_\alpha, X_\beta, X_\gamma)$ that factorises according to the DAG in Figure 4.3 has factorisation

$$\mathbb{P}_{X_\alpha,X_\beta,X_\gamma} = \mathbb{P}_{X_\beta}\mathbb{P}_{X_\alpha|X_\beta}\mathbb{P}_{X_\gamma|X_\beta}.$$

It is clear that $X_\alpha \not\perp X_\gamma$ in general;

$$\mathbb{P}_{X_\alpha,X_\gamma}(x_1,x_3) = \sum_{x_2\in\mathcal{X}_2} \mathbb{P}_{X_\beta}(x_2)\mathbb{P}_{X_\alpha|X_\beta}(x_1|x_2)\mathbb{P}_{X_\gamma|X_\beta}(x_3|x_2)$$

and, without further assumptions, this cannot be expressed in product form. Conditioned on X_β, though:

$$\mathbb{P}_{X_\alpha,X_\gamma|X_\beta} = \frac{\mathbb{P}_{X_\alpha,X_\gamma,X_\beta}}{\mathbb{P}_{X_\beta}} = \frac{\mathbb{P}_{X_\beta}\mathbb{P}_{X_\alpha|X_\beta}\mathbb{P}_{X_\gamma|X_\beta}}{\mathbb{P}_{X_\beta}} = \mathbb{P}_{X_\alpha|X_\beta}\mathbb{P}_{X_\gamma|X_\beta}.$$

It follows that $X_\alpha \perp X_\gamma|X_\beta$.

Collider Connections A *collider connection* between two nodes α and γ is a connection such that the graph does not contain an edge between α and γ, but there is a node β such that the graph contains directed edges $\alpha \mapsto \beta$ and $\gamma \mapsto \beta$. A collider connection is illustrated in Figure 4.4.

The factorisation of the distribution $\mathbb{P}_{X_\alpha,X_\beta,X_\gamma}$ corresponding to the DAG for the collider is

$$\mathbb{P}_{X_\alpha,X_\beta,X_\gamma} = \mathbb{P}_{X_\alpha}\mathbb{P}_{X_\gamma}\mathbb{P}_{X_\beta|X_\alpha,X_\gamma}.$$

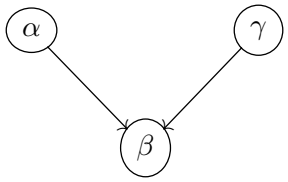

Figure 4.4: A Collider Connection

In general, $X_\alpha \not\perp X_\gamma | X_\beta$. But for each $(x, z) \in \mathcal{X}_\alpha \times \mathcal{X}_\gamma$,

$$\begin{aligned}
\mathbb{P}_{X_\alpha, X_\gamma}(x, z) &= \sum_{y \in \mathcal{X}_\beta} \mathbb{P}_{X_\alpha}(x)\mathbb{P}_{X_\gamma}(z)\mathbb{P}_{X_\beta | X_\alpha, X_\gamma}(y|x, z) \\
&= \mathbb{P}_{X_\alpha}(x)\mathbb{P}_{X_\gamma}(z) \sum_{y \in \mathcal{X}_\beta} \mathbb{P}_{X_\alpha | X_\beta, X_\gamma}(y|x, z) \\
&= \mathbb{P}_{X_\alpha}(x)\mathbb{P}_{X_\gamma}(z).
\end{aligned}$$

so that $X_\alpha \perp X_\gamma$.

A Causal Interpretation So far, the discussion has considered sets of random variables where, based on the ordering of the variables, the parent set of a variable is a subset of those of a lower order. The representation of a probability distribution by factorising along a Directed Acyclic Graph may be particularly useful if there are cause to effect relations between the variables, the ancestors being the cause and the descendants the effect. For a causal model, the connections have the following interpretations:

Fork Connection: Common cause For the fork connection, illustrated by Figure 4.2, X_β may be a *cause* that influences both X_α and X_γ which are *effects*. The variables are only related through X_β. The situation is illustrated by the following example, taken from a cartoon by Albert Engström; 'during a convivial discussion at the bar one evening, about the unhygienic nature of galoshes, one of the participants pipes up, "you have a very good point there. Every time I wake up wearing my galoshes, I have a sore head."

Let X_α denote the state of the feet and X_γ the state of the head. These two variables are related; $X_\alpha \not\perp X_\beta$. But there is a common cause; X_2, which denotes the activities of the previous evening. Once it is known that he has spent a convivial evening drinking, the state of the feet gives no further information about the state of the head; $X_\alpha \perp X_\gamma | X_\beta$.

Chain Connection This may similarly be understood as cause to effect. X_α influences X_β, which in turn influences X_γ, but there is no direct causal relationship between the values taken by X_α and those taken by X_γ. If X_β is unknown, then $X_\alpha \not\perp X_\gamma$, but once the state of X_β is established, X_α and X_γ give no further information about each other; $X_\alpha \perp X_\gamma | X_\beta$.

Collider Connection For the collider connection, X_α and X_β are unrelated; $X_\alpha \perp X_\gamma$. But they both influence X_β. For example, consider a burglar alarm (X_β) that is activated if a burglary takes place, but can also be activated if there is a minor earth tremor.

One day, somebody calls you while you are at work to say that your burglar alarm is activated. You get into the car to go home. But on the way home, you hear on the radio that there has been an earth tremor in the area. As a result, you return to work.

Once X_β is instantiated, the information that there has been an earth tremor influences the likelihood that a burglary has taken place; $X_\alpha \not\perp X_\gamma | X_\beta$.

This is known as *explaining away*.

4.4 Separation within a DAG

Attention is now turned to trails within a DAG, and characterisation of those along which information can pass.

Definition 4.3 (S-Active Trail). *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. Let $S \subset V$ and let $\alpha, \beta \in V \backslash S$. A trail τ between the two variables α and β is said to be* S-active *if*

1. *Every collider node in τ is in S, or has a descendant in S (that is, for each collider node $\alpha \in \tau$, there is a directed path $\alpha \to \beta_1 \to \ldots \to \beta_m \to \gamma$ for some $\gamma \in S$).*

2. *Every other node is outside S.*

Definition 4.4 (Blocked Trail). *A trail between α and β that is not S-active is said to be* blocked by S.

The following definition is basic; it will be seen that if a probability distribution factorises along a DAG $\mathcal{G}$ and two nodes α and β are D-separated by S, then $X_\alpha \perp X_\beta | X_S$.

Definition 4.5 (D-separation). *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph, where $V = \{1, \ldots, d\}$. Let $S \subset V$. Two distinct nodes α and β not in S are* D-separated *by S if all trails between α and β are blocked by S.*

Let A and B denote two sets *of nodes. If every trail from any node in A to any node in B is blocked by S, then the sets A and B are said to be D-separated by S. This is written*

$$A \perp\!\!\!\perp B \parallel_{\mathcal{G}} S. \tag{4.1}$$

The terminology D-separation is short for *directed* separation. The insertion of the letter 'D' points out that this is not the standard use of the term 'separation' found in graph theory.

Definition 4.6 (D-connected). *If two nodes α and β are not D-separated, they are said to be D-connected.*

Notation The notation $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ denotes that α and β are D-connected by S in the DAG $\mathcal{G}$. Here α and β may refer to individual nodes or sets of nodes.

Example 4.2.

Consider the chain connection $\alpha \mapsto \beta \mapsto \gamma$ in the DAG in Figure 4.2 and the fork connection of Figure 4.3. For the chain connection of Figure 4.2, the D-separation statements are: $\alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} \beta$ while $\alpha \not\perp\!\!\!\perp \gamma \| _{\mathcal{G}} \phi$ (ϕ denotes the empty set). For the DAG in Figure 4.3, $\alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} \beta$ while $\alpha \not\perp\!\!\!\perp \gamma \| _{\mathcal{G}} \phi$. These correspond to the conditional independence statements derived for probability distributions that factorise along these graphs. For Figure 4.4, $\alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} \phi$ while $\alpha \not\perp\!\!\!\perp \gamma \| _{\mathcal{G}} \beta$. Again, these statements correspond to the conditional independence statements that may be derived from the fact that a distribution factorises along the DAG of Figure 4.4. □

Let MB(α) denote the set of nodes which are either parents of α or children of α or a node which shares a common child with α. Then α is D-separated from the rest of the network by MB(α). This set of nodes is known as the *Markov blanket* of the node α.

Definition 4.7 (Markov Blanket)**.** *The* Markov blanket *of a node α in a DAG $\mathcal{G} = (V, D)$, denote MB(α), is the set consisting of the parents of α, the children of α and the nodes sharing a common child with α.*

4.4.1 Bayes Ball

The *Bayes ball* provides a convenient method for deciding whether or not two nodes are D-separated by a set S in a DAG $\mathcal{G} = (V, D)$. Variables are D-connected by a set S if the Bayes ball can be passed between them employing the following rule. The nodes which are *not* in S are depicted as unshaded; nodes in S as shaded.

Definition 4.8 (Instantiated Nodes)**.** *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. When considering statements $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ and $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$, the nodes in S are referred to as* instantiated.

Consider the three types of connection in a DAG; chain, collider and fork.

- For the *chain* connection illustrated in Figure 4.2, the Bayes ball algorithm indicates that *if* node β is instantiated, *then* the ball does not move from α to γ through β. The communication in the trail is *blocked*. If the node is *not* instantiated, then communication is possible.

- For the *fork* connection illustrated in Figure 4.3, the algorithm states that *if* node β is instantiated, then again communication between α and γ is *blocked*. If the node is *not* instantiated, then communication is possible.

- For the *collider* connection illustrated in Figure 4.4, the Bayes ball algorithm states that the ball *does* move from α to γ if node α or any of its descendants is instantiated. If β or a descendant is instantiated, this opens communication between the parents. If neither β nor any of its descendants are instantiated, then there is no communication.

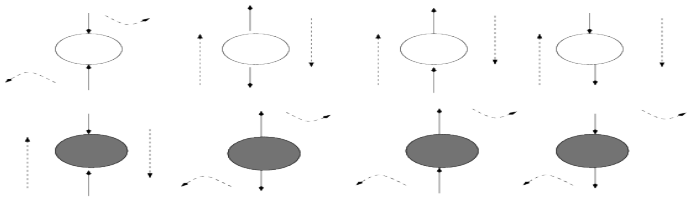

Figure 4.5: Bayes Ball

For a collider node β, instantiating any of the descendants of β *also* opens communication. If node β is *not* instantiated, *and* none of its descendants are instantiated, then there is no communication.

A DAG $\mathcal{G} = (V, D)$ satisfies the following important property:

Theorem 4.9. *A DAG $\mathcal{G} = (V, D)$ contains an edge between two nodes $\alpha, \beta \in V$ if and only if $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ for any $S \subseteq V \backslash \{\alpha, \beta\}$.*

Proof The proof of this is straightforward and left as an exercise. □

4.5 D-Separation and Conditional Independence

The following key result shows that if a probability distribution factorises along a given DAG $\mathcal{G}$, then every D-separation statement for the DAG implies the corresponding conditional independence statement for the distribution.

Theorem 4.10 (D-Separation Implies Conditional Independence)**.** *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph and let $\mathbb{P}$ be a probability distribution that factorises along $\mathcal{G}$. Then for any three disjoint subsets $A, B, S \subset V$, it holds that $X_A \perp X_B | X_S$ (X_A and X_B are independent given X_S) if $A \perp\!\!\!\perp B \| _{\mathcal{G}} A$ (A and B are D-separated by S).*

Proof of Theorem 4.10 Omitted □

Of course, the converse is not true in general; D-separation is a convenient way of locating some of the independence structure of a distribution. It does not, in general, locate the entire independence structure.

4.6 Queries

Once a probability distribution has been factorised according to a Bayesian Network, the next task is to use it to answer *queries*.

Definition 4.11 (Query)**.** *A* query *in probabilistic inference is simply a conditional probability distribution, over the variables of interest (the* query *variables) conditioned on information received.*

4.7 Bayesian Networks in R

4.8 Introduction

It has become clear that R is now the most effective and dominant language of statistical computing. There are excellent packages available in R for Bayesian Networks, for inference using a given Bayesian Network and for learning the structure of a Bayesian Network. This chapter introduces some of the software in R available for Bayesian Networks and discusses graphs in R and inference using networks that have already been defined. Parameter learning and structure learning are considered later.

The packages considered are **gRain** by Søren Højsgaard and **bnlearn**.

Having installed R and a suitable editor (for example Rstudio), the relevant packages have to be installed.

gRain and related packages Information for **gRain** is available on the author's web page:

```
http://people.math.aau.dk/~sorenh/software/gR/
```

The package, along with all the supporting packages, has to be installed. As pointed out on the web page, the package uses the packages **graph**, **RBGL** and **Rgraphviz**. These packages are *not* on CRAN, but on 'bioconductor'. To install these packages, execute

```
install.packages("BiocManager")
```

```
setRepositories()
```

and then make sure that all are activated (2 3 4 5 6 7 8)

Now install using:

```
install.packages("gRbase", dependencies=TRUE);
install.packages("gRain", dependencies=TRUE);
install.packages("gRim", dependencies=TRUE)
```

The package **bnlearn** also has some useful inference functions, although its main consideration is learning. Install it in the usual way:

```
> install.packages("bnlearn")
```

4.9 Graphs in R

This section considers the various graphs that appear in graphical modelling and how to render them in R. In addition to the packages mentioned so far, the package **ggm**, has some useful functions for graphical Markov models.

```
>install.packages("ggm")
```

Another useful graphics package is **igraph**

```
>install.packages("igraph")
```

We also need the package **RBGL**, which is not available on CRAN, but is only available on BioConductor. So set repositories with

```
setRepositories()
```

make sure that the appropriate repositories are checked and then

```
install.packages("RBGL")
```

These packages should be activated:

```
> library("bnlearn")
> library("gRain")
> library("ggm")
> library("igraph")
> library("RBGL")
> library("gRbase")
```

4.10 Example: 'Asia' by Lauritzen

We consider the 'Asia' example of Lauritzen et. al. You have returned from holiday in Asia and you are feeling unwell. There may be nothing seriously wrong with you, but you could be suffering from tuberculosis, lung cancer or bronchitis. The causal diagram is shown in Figure 4.6. Let A denote 'visit to Asia' with values 'yes' or 'no'. Similarly, all the other variables are binary and are llabelled S for smoker, T for tuberculosis, L for lung cancer, B for bronchitis, E for either, X for X-ray ('yes' for indication of a problem, 'no' for clear), D for dyspnoea (shortness of breath)

$$\mathbb{P}(A = \text{yes}) = 0.01$$

$$\mathbb{P}(T = \text{yes}|A = \text{yes}) = 0.05 \qquad \mathbb{P}(T = \text{yes}|A = \text{no}) = 0.01$$

$$\mathbb{P}(S = \text{yes}) = 0.5$$

$$\mathbb{P}(L = \text{yes}|S = \text{yes}) = 0.1 \qquad \mathbb{P}(L = \text{yes}|S = \text{no}) = 0.01$$

$$\mathbb{P}(B = \text{yes}|S = \text{yes}) = 0.6 \qquad \mathbb{P}(B = \text{yes}|S = \text{no}) = 0.3$$

$$\mathbb{P}(E = \text{yes}|L = \text{yes}, B = \text{yes}) = 1, \quad \mathbb{P}(E = \text{yes}|L = \text{yes}, B = \text{no}) = 1$$

$$\mathbb{P}(E = \text{yes}|L = \text{no}, B = \text{yes}) = 1, \qquad \mathbb{P}(E = \text{yes}|L = \text{no}, B = \text{no}) = 0$$

$$\mathbb{P}(X = \text{yes}|E = \text{yes}) = 0.98 \qquad \mathbb{P}(X = \text{yes}|E = \text{no}) = 0.05$$

$$\mathbb{P}(D = \text{yes}|B = \text{yes}, E = \text{yes}) = 0.9 \quad \mathbb{P}(D = \text{yes}|B = \text{yes}, E = \text{no}) = 0.7$$

$$\mathbb{P}(D = \text{yes}|B = \text{no}, E = \text{yes}) = 0.8 \qquad \mathbb{P}(D = \text{yes}|B = \text{no}, E = \text{no}) = 0.1$$

We can programme the network into R as follows. We need the packages **gRain** and **gRbase**. The conditional probability potentials may be specified as follows:

```
> library("gRain")
Loading required package: gRbase
> yn <- c("yes","no")
> a<-cptable(~asia, values=c(1,99),levels=yn)
> t.a<-cptable(~tub+asia,values=c(5,95,1,99),levels=yn)
> s<-cptable(~smoke, values=c(5,5),levels=yn)
> l.s<-cptable(~lung+smoke,values=c(1,9,1,99),levels=yn)
> b.s<-cptable(~bronc+smoke,values=c(6,4,3,7),levels=yn)
> e.lt<-cptable(~either+lung+tub,values=c(1,0,1,0,1,0,0,1),levels=yn)
> x.e<-cptable(~xray+either,values=c(98,2,5,95),levels=yn)
> d.be<-cptable(~dysp+bronc+either, values=c(9,1,7,3,8,2,1,9), levels
= yn)
```

The + operator could be considered slightly misleading. There are other ways to enter the conditional probability potentials:

```
> t.a<-cptable(~tub|asia,values=c(5,95,1,99),levels=yn)
> t.a<-cptable(c("tub","asia"),values=c(5,95,1,99),levels=yn)
```

There are also special functions `ortable()` and `andtable`. For example, `e.lt()` could be entered by:

```
> e.lt <-ortable(~either+lung+tub, levels=yn)
```

4.10.1 Building the Network

A network is created with the function `grain()`, which returns an object of class `grain`:

```
> plist<-compileCPT(list(a, t.a, s, l.s, b.s, e.lt, x.e, d.be))
> grn1<-grain(plist)
> summary(grn1)
Independence network: Compiled: FALSE Propagated: FALSE
 Nodes : chr [1:8] "asia" "tub" "smoke" "lung" "bronc" "either" ...
> plot(grn1)
```

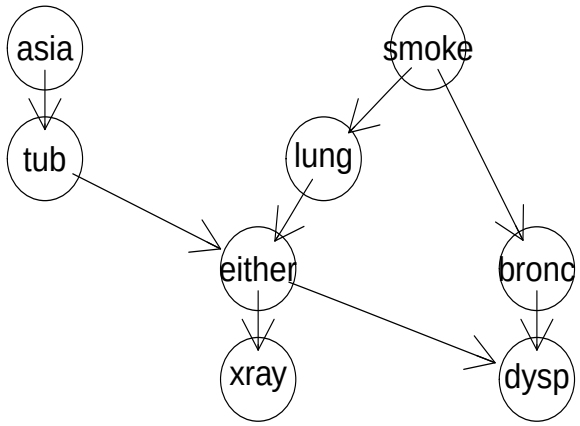

Figure 4.6: Asia Network

The plot is shown in Figure 4.6.

4.10.2 Compilation

The network has to be compiled and propagated before queries can be made.

```
> grn1c<-compile(grn1)
> summary(grn1c)
Independence network: Compiled: TRUE Propagated: FALSE
 Nodes : chr [1:8] "asia" "tub" "smoke" "lung" "bronc" "either" ...
 Number of cliques:                 6
 Maximal clique size:               3
 Maximal state space in cliques:    8
```

4.10.3 Absorbing Evidence and Answering Queries

Evidence may be entered as follows: for example, suppose we have evidence that someone has visited asia and has dyspnoea. This is entered as follows:

```
> grn1c.ev<-
+ setFinding(grn1c,nodes=c("asia","dysp"),states=c("yes","yes"))
```

This creates a new `grain` object. The `grain` objects with (`grn1c.ev`) and without (`gran1c`) can be queried to give marginal probabilities:

```
> querygrain(grn1c.ev,nodes=c("lung","bronc"),type="marginal")
$lung
lung
       yes         no
0.09952515 0.90047485

$bronc
bronc
      yes        no
0.8114021 0.1885979

> querygrain(grn1c,nodes=c("lung","bronc"),type="marginal")
$lung
lung
  yes    no
0.055 0.945

$bronc
bronc
 yes   no
0.45 0.55
```

The evidence in a `grain` object can be retrieved with the `getFinding()` function, while the probability of observing the evidence is obtained using the `pFinding()` function:

```
> getFinding(grn1c.ev)
Finding:
asia: yes
dysp: yes
Pr(Finding)= 0.004501375
> pFinding(grn1c.ev)
[1] 0.004501375
```

Joint and conditional distributions may be computed as follows:

```
> querygrain(grn1c.ev,nodes=c("lung","bronc"),type="joint")
     bronc
lung         yes         no
  yes 0.06298076 0.03654439
```

```
  no  0.74842132 0.15205354
> querygrain(grn1c.ev,nodes=c("lung","bronc"),type="conditional")
     bronc
lung         yes        no
  yes 0.07761966 0.1937688
  no  0.92238034 0.8062312
```

These are both conditioned on the evidence; the former the joint distribution of `lung` and `bronc` conditioned on the evidence, while the latter is the conditional distribution of `lung` given `bronc` and the evidence.

Chapter 5

Intervention Calculus

5.1 Causal Models and Bayesian Networks

In many applications, a Bayesian network is *constructed* as a *causal* model, where for each variable, its parent variables are considered to be direct causes that influence the value taken by the variable.

For example, an earth tremor or a burglary can *cause* the burglar alarm to go off and the arrows in the associated collider DAG represent *cause* to *effect* relations. It is self evident, but nevertheless has to be stated, that only *associations* can be inferred from an $n \times d$ data matrix $\mathbf{x}$ of instantiations; directions of cause to effect cannot be inferred from data alone. When conditional independence statements are learned from data, this can be interpreted as a Markov model and it may be possible to construct an efficient factorisation of the distribution using these conditional independence statements. Clearly, this factorisation cannot be understood as a *causal* model, unless there are other modelling assumptions. For example, consider a model containing observable variables A, B, C, where there are hidden variables H_1, H_2 that are unknown to the experimenter. If the causal diagram representing the causal relations between these variables is given by the DAG on the left in Figure 5.1, then the learned DAG, along which the distribution of A, B, C can be factorised, is the DAG on the right of Figure 5.1.

This is the correct DAG, in that it preserves the d-connection properties between A, B, C, but the collider connection cannot be interpreted as A and B having a *causal* effect on C; they are *effects* of the latent common causes H_1 and H_2.

If a Bayesian network is to be interpreted as a causal model, then the possible directions of cause

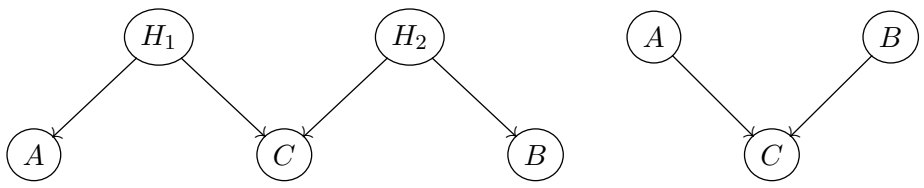

Figure 5.1: Hidden causes and the learned DAG

to effect must be part of the modelling assumptions before the data is analysed, determined by other considerations. The data analysis only determines which directed edges remain and which are removed. From data, one can determine whether or not there is an association between earth tremors and alarms triggered; it is not possible to determine from the data what causes what.

This is self evident, but surprisingly it turns out that it is necessary to state this. An article by Freedman and Humphreys from 1999 pointed out the obvious fact that causality could not be inferred from data alone and was a necessary response to obvious errors in the literature, where the term 'causal discovery' has been used in surprising ways, even after it had been established, with simple concrete and obvious examples, that the concept was ridiculous and long after publication of the Friedman Humphreys article illustrating that it was ridiculous. The article by Freedman and Humphreys is a good article; it is surprising that the literature had degenerated to such an extent that it was necessary for the authors to write it.

To define a *causal* network, an additional ingredient is needed; this is the concept of *intervention*, introduced by Judea Pearl in a seminal article from 1995.

5.2 Conditioning by Observation and by Intervention

Let X and Y be two random variables and suppose that $X = x$ is *observed*. Then the conditional probability of $Y = y$ is defined as

$$\mathbb{P}_{Y|X}(y|x) = \frac{\mathbb{P}_{X,Y}(x,y)}{\mathbb{P}_X(x)}.$$

This formula describes the way that the probability distribution of the random variable Y changes after $X = x$ is observed. If, instead, the value $X = x$ is *forced* by the observer, irrespective of other considerations, the conditional probability statement is invalid.

If random variables are linked through a *causal* model, expressed by a directed acyclic graph, where parent variables have a causal effect on their children, some attempt can be made to compute the probability distribution over the remaining variables when the states of some variables are forced.

In a controlled experiment, a variable is *forced* to take a particular value, chosen at random, irrespective of the other variables in the network. In terms of the directed acyclic graph, the variable is instantiated with this value, the directed edges between the variable and its parents are removed (because the parents no longer have influence on the state of the variable) and all other conditional probabilities remain unaltered.

5.3 The Intervention Calculus for a Bayesian Network

Definition 5.1 (The Intervention Formula)**.** *The conditional probability of $X_{V\setminus A} = x_{V\setminus A}$, given that the variables X_A were forced to take the values x_A independently of all else, is written*

$$\mathbb{P}_{V\setminus A\|A}(x_{V\setminus A}|X_A \leftarrow x_A) \quad or \quad \mathbb{P}_{V\setminus A\|A}(x_V\|x_A)$$

and defined as

$$\mathbb{P}_{V\backslash A\|A}(x_{V\backslash A}|X_A \leftarrow x_A) = \mathbb{P}_{V\backslash A\|A}(x_{V\backslash A}\|x_A) = \prod_{v\in V\backslash A} \mathbb{P}_{v|Pa(v)}(x_v|x_{Pa(v)}). \tag{5.1}$$

Note that (5.1) is equivalent to:

$$\mathbb{P}_{V\backslash A\|A}(x_{V\backslash A}\|x_A) = \frac{\mathbb{P}_V(x_V)}{\prod_{v\in A}\mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(v)})}. \tag{5.2}$$

The last expression of Equation (5.1) is in terms of the required factorisation; *instantiation of the variables indexed by the set* A and *elimination of those edges in* D *which lead from the parents of the nodes in* A *to the nodes in* $V\backslash A$. The terminology 'local surgery' is used to describe such an elimination. A local surgery is performed and the conditional probabilities on the remaining edges are multiplied. This yields a factorisation along a *mutilated graph* where the *direct causes* of the manipulated variable are put out of effect.

The intervention formula (5.1) is obtained by wiping out those factors from the factorisation which correspond to the interventions. An explicit translation of intervention in terms of 'wiping out' equations was first proposed by Strotz and Wold (1960).
The quantity $\mathbb{P}_{V\backslash A\|A}(.\|x_A)$ from Definition 5.1 defines a family of probability measures over $\mathcal{X}_{V\backslash A}$, which depends on the values x_A, which may be considered as *parameters*. These are the values forced on the variables indexed by A. This family includes original probability measure; if $A = \phi$, then $\mathbb{P}_{V\backslash A\|A}(.|x_A) = \mathbb{P}_X(.)$. This family is known as the *intervention measure*. In addition, the expression on the right hand side of (5.1) is called the *intervention formula*.

Intervention An 'intervention' is an action taken to force a variable into a certain state, without reference to its own current state, or the states of any of the other variables. It may be thought of as choosing the values x_A^* for the variables X_A by using a random generator independent of the variables X.

Remark In the same style of notation, *conditioning by observation* is

$$\mathbb{P}_{X_{V\backslash A}|X_A}(x_{V\backslash A}|\mathrm{see}(x_A)) = \mathbb{P}_{X_{V\backslash A}|X_A}(x_{V\backslash A}|x_A) \tag{5.3}$$

where, by the standard definition of conditional probability,

$$\mathbb{P}_{V\backslash A|A}(x_{V\backslash A}|x_A) = \frac{\mathbb{P}_V(x)}{\mathbb{P}_X(x_A)}. \tag{5.4}$$

Example 5.1.

Consider the DAG given in Figure 5.2, for 'X having causal effect on Y'.
The factorisation of $\mathbb{P}_{X,Y}$ along the DAG in Figure 5.2 is

$$\mathbb{P}_{X,Y}(x,y) = \mathbb{P}_{Y|X}(y|x)\mathbb{P}_X(x)$$

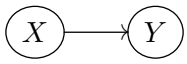

Figure 5.2: A DAG for X having causal effect on Y

and the intervention formula gives

$$\mathbb{P}_{Y\|X}(y\|x) = \mathbb{P}_{Y|X}(y|x).$$

Since X is a parent of Y, the intervention to force $X = x$ produces exactly the same conditional probability distribution over Y as *observing* $X = x$. But if instead Y is forced, the intervention formula yields

$$\mathbb{P}_{X\|Y}(x\|y) = \mathbb{P}_X(x).$$

Clearly, $\mathbb{P}_{X\|Y}(x\|y) \neq \mathbb{P}_{X|Y}(x|y)$ as functions unless X and Y are independent. □

Example 5.2 (The DAG for a wet pavement)**.**

The 'wet pavement' example is a classic illustration, introduced by Judea Pearl. The DAG represents a causal model for a wet pavement and is given in Figure 5.3. The *season* A has four states; spring, summer autumn, winter. Rain B has two states; yes / no. Sprinkler C has two states; on / off. Wet pavement D has two states; yes / no. Slippery pavement E has two states; yes / no.

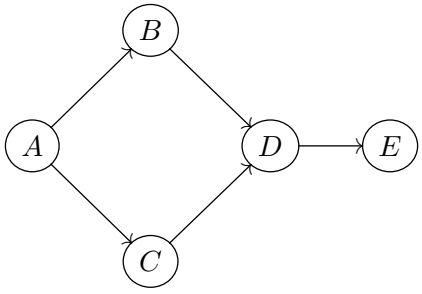

Figure 5.3: DAG for wet pavement, no intervention

The joint probability distribution is factorised as

$$\mathbb{P}_{A,B,C,D,E} = \mathbb{P}_A\mathbb{P}_{B|A}\mathbb{P}_{C|A}\mathbb{P}_{D|B,C}\mathbb{P}_{E|D}.$$

Suppose, without reference to the values of any of the other variables and without reference to the current state of the sprinkler, 'sprinkler on' is now *enforced*. This could be, for example, regular maintenance work, which is carried out at regular intervals, irrespective of the season or other considerations. Then

$$\mathbb{P}_{A,B,D,E\|C}(.\|C \leftarrow 1) = \mathbb{P}_A \mathbb{P}_{B|A} \mathbb{P}_{D|B,C}(.|.,1)\mathbb{P}_{E|D}.$$

After *observing* that the sprinkler is on, it may be inferred that the season is dry and that it probably did not rain and so on. If 'sprinkler on' is enforced, without reference to the state of the system when the action is taken, then no such inference should be drawn in evaluating the effects of the intervention. The resulting DAG is given in Figure 5.4. It is the same as before, except that $C = 1$ is fixed and the edge between C and A disappears. The deletion of the factor $\mathbb{P}_{C|A}$ represents the understanding that whatever relationships existed between sprinklers and seasons prior to the action, found from

$$\mathbb{P}_{A,B,D,E|C}(.,.,.,.|1)$$

are no longer in effect when the state of the variable is *forced*, as in a controlled experiment, without reference to the state of the system.

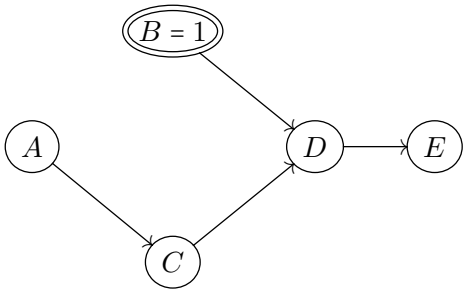

Figure 5.4: Sprinkler 'on' is forced

After *observing* that the sprinkler is on, it may be inferred that the season is dry, that it probably did not rain and so on. No such inferences may be drawn in evaluating the effects of the intervention 'ensure that the sprinkler is on'. □

5.4 Causal Models

Having defined the family of intervention measures, the concept of *causal model* may now be defined.

Definition 5.2 (Causal Model)**.** *Let $X = (X_1, \ldots, X_d)$ be a random vector and let $V = \{1, \ldots, d\}$ denote the indexing set. A* causal model *consists of the following:*

1. *A* Bayesian Network *for $\mathbb{P}_X$, that is, an ordering σ of the indices V, a factorisation of the probability distribution*

$$\mathbb{P}_V = \prod_{j=1}^{d} \mathbb{P}_{\sigma(j)|Pa^{(\sigma)}(j)} \tag{5.5}$$

where $Pa^{(\sigma)}(j) \subseteq \{\sigma(1), \ldots, \sigma(j-1)\}$ and is the smallest such subset such that (5.5) holds.

2. *The node set V consists of two types of nodes; V_I and V_N, where $V_I \cap V_N = \phi$ and $V_I \cup V_N = V$. The nodes V_I are the* interventional *nodes and V_I are the* non-interventional *nodes, where no intervention is possible. The intervention formula* (5.1) *holds for each subset $A \subseteq V_I$ of interventional nodes and each $x_A \in \mathcal{X}_A$.*

The arrows $\alpha \mapsto \beta$ of the DAG for either α or β (or both) in V_I are causal *arrows, indicating* direct cause *to* effect. *The remaining arrows are* non-causal*; a cause to effect relation between nodes α and β cannot be inferred from an arrow $\alpha \mapsto \beta$ if both $\alpha, \beta \in V_N$.*

5.4.1 Establishing a Causal Model via a Controlled Experiment

If sufficient data is available, a suitable Bayesian Network may be learned from the data. A *causal* model cannot be established from data alone. Additional information is needed, which is obtained through interventions on the interventional variables.

For example, the three graphs in Figure 5.5 are Markov equivalent; if the probability distribution factorises along one of these graphs, it also factorises along the others. The chains $\alpha \to \gamma \to \beta$ and $\alpha \leftarrow \gamma \leftarrow \beta$ and the fork $\alpha \leftarrow \gamma \to \beta$ are all Markov equivalent, with D-separation structure $\alpha \perp\!\!\!\perp \beta \| \gamma$. If any of these DAGs represents a *causal* network, then it is not possible to learn the causal network from the data alone.

Suppose that it is possible to intervene by controlling the variable X_γ, then if one of these graphs is the DAG for a *causal* network, it will be possible to establish which one through a controlled experiment. Figure 5.6 shows the associated structural model when the control $X_\gamma \leftarrow z$ has been applied, forcing X_γ to be independent of its ancestors. A controlled experiment, where the direct causal links between X_γ and its parent variables have been eliminated, will exhibit independence structure $X_\alpha \perp \{X_\beta, X_\gamma\}$ in the first case, $X_\alpha \perp X_\beta | X_\gamma$ in the second $\{X_\alpha, X_\gamma\} \perp X_\beta$ in the third. Once the associations $X_\alpha \perp X_\beta | X_\gamma$, $X_\alpha \not\perp X_\beta$, $X_\alpha \not\perp X_\gamma$, $X_\alpha \not\perp X_\gamma | X_\beta$, $X_\beta \not\perp X_\gamma$ and $X_\beta \not\perp X_\gamma | X_\alpha$ have been established, an additional controlled experiment, if it is possible to control the variable X_γ with interventions to force all possible values of X_γ, will determine which graph within the equivalence class is appropriate.

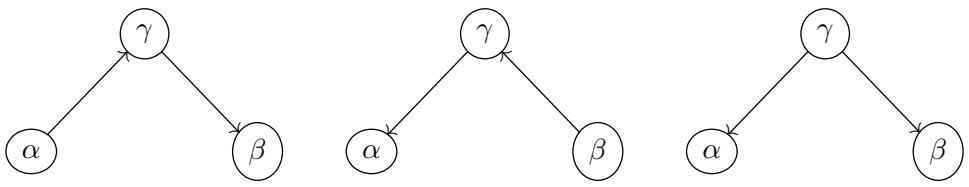

Figure 5.5: Three Markov Equivalent Graphs

If it is possible to control variables, then it is possible to learn whether or not a collider represents independent causes with a common effect. If the DAG on the left hand side of Figure 5.1 represents a causal structure, then an experiment where variable A is controlled will establish that it is not a direct cause of C, since an intervention on A leaves it separated from the rest of the network, as in Figure 5.7.

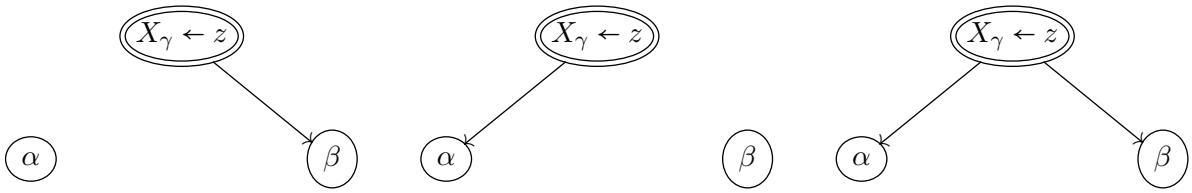

Figure 5.6: Graphs from Figure 5.5 with intervention $X_\gamma \leftarrow z$ applied

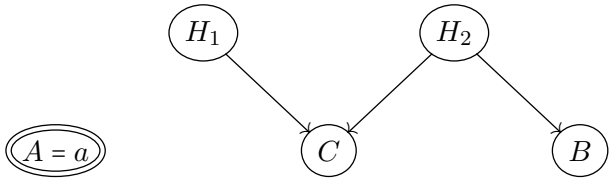

Figure 5.7: Hidden causes H_1 and H_2; intervention $A = a$

5.5 Confounding, The 'Sure Thing' Principle and Simpson's Paradox

5.5.1 Confounding

Consider the DAG given in Figure 5.8. It corresponds to the factorisation:

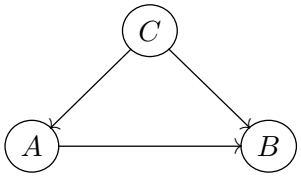

Figure 5.8: Illustration for Confounding

$$\mathbb{P}_{A,B,C} = \mathbb{P}_{B|A,C}\mathbb{P}_{A|C}\mathbb{P}_C.$$

Consider the conditional probability of B, when A is controlled; $\mathbb{P}_{B\|A}(.\|a)$. The DAG illustrating the intervention is shown in Figure 5.9. Note that

$$\mathbb{P}_{B\|A}(.\|a) = \sum_{c\in\mathcal{X}_C} \mathbb{P}_{B,C\|A}(.,c\|a).$$

and that

$$\mathbb{P}_{B,C\|A}(.,.\|a) = \mathbb{P}_{B|C\|A}(.|.\|a)\mathbb{P}_{C\|A}(.\|a) = \mathbb{P}_{B|A,C}(.|a,.)\mathbb{P}_C,$$

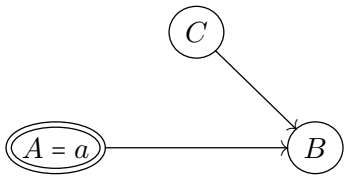

Figure 5.9: Intervention on A

where in the second term, the do-conditioning of $A \leftarrow a$ is applied first, and then C is observed. It follows that

$$\mathbb{P}_{B\|A}(.\|a) = \sum_{c \in \mathcal{X}_C} \mathbb{P}_{B|A,C}(.|a,c)\mathbb{P}_C(c).$$

This shows that to estimate $\mathbb{P}_{B\|A}(.\|a)$ from data alone (i.e. without controlling A), it is necessary to be able to estimate $\mathbb{P}_{B|A,C}$ and $\mathbb{P}_C$ from data. If C is *observable*, then the effect on the probability distribution of B of manipulating A may be estimated. But if C is a *hidden* random variable (sometimes the term *latent* is used) in the sense that no *direct* sample of the outcomes of C may be obtained, it will not be possible to estimate the probabilities used on the right hand side and hence it will not be possible to predict the effect on B of manipulating A. This is known as *confounding*.

5.5.2 Simpson's Paradox

Consider three binary variables, A, B and C. Simpson's paradox is the observation that there are situations where

$$\frac{\mathbb{P}_{B|C,A}(1|1,1)/\mathbb{P}_{B|C,A}(0|1,1)}{\mathbb{P}_{B|C,A}(1|1,0)/\mathbb{P}_{B|C,A}(0|1,0)} > 1 \qquad \text{and} \qquad \frac{\mathbb{P}_{B|C,A}(1|0,1)/\mathbb{P}_{B|C,A}(0|0,1)}{\mathbb{P}_{B|C,A}(1|0,0)/\mathbb{P}_{B|C,A}(0|0,0)} > 1,$$

but

$$\frac{\mathbb{P}_{B|A}(1|1)/\mathbb{P}_{B|A}(0|1)}{\mathbb{P}_{B|A}(1|0)/\mathbb{P}_{B|A}(0|0)} < 1.$$

For example let A denote 'treatment', B 'recovery' and C 'blood pressure'. Simpson's paradox states that even if the 'treatment' may improve the chances of recovery for those with high blood pressure and those with low blood pressure, it may nevertheless be bad for the population as a whole. It could be that although the treatment is comparatively good within the group where high blood pressure is observed after treatment and also comparatively good within the group where low blood pressure is observed after treatment, it may be bad for the population as a whole. This occurs if 'treatment' increases blood pressure and increased blood pressure reduces the chances of recovery.

This situation is illustrated by the DAG given in Figure 5.10, where A denotes treatment, B recovery and C blood pressure. Suppose that C is a hidden variable. Even if the 'treatment' variable

A can be controlled, an intervention on A does not remove any arrows from the causal diagram; there is the possibility of a Simpson's paradox, even with a controlled experiment.

If A denotes 'treatment' and B 'recovery' and C denotes a *common cause* of both A and B, as in Figure 5.8, Simpson's paradox may be resolved if A can be controlled, because controlling A breaks the causal link between C and A. This is the *sure thing* principle, considered next, which states that if the treatment improves the chances of recovery for each level of the 'common cause' variable C, then it is good for the population as a whole.

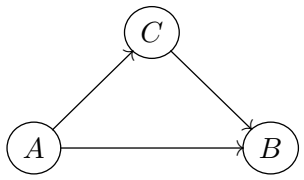

Figure 5.10: A=treatment / B=recovery / C=blood pressure

5.5.3 The Sure Thing Principle

Consider again the situation of Figure 5.8. Suppose that A is *controlled*; values for the variable A are assigned at random, so the link $C \to A$ is broken and hence the effect on B of manipulating A is not confounded by the effects of hidden variables. The following result is referred to as 'The Sure Thing Principle'. It states that when Figure 5.8 represents the causal structure and there is do-conditioning on A, then Simpson's paradox does not hold.

Proposition 5.3. *Consider three binary variables A, B, C with the network given in Figure 5.8. If*

$$\mathbb{P}_{B|C\|A}(1|1\|1) < \mathbb{P}_{B|C\|A}(1|1\|0)$$

and

$$\mathbb{P}_{B|C\|A}(1|0\|1) < \mathbb{P}_{B|C\|A}(1|0\|0)$$

then

$$\mathbb{P}_{B\|A}(1\|1) < \mathbb{P}_{B\|A}(1\|0).$$

The notation means: first A *is forced,* then C *is observed.*

Proof Firstly,

$$\mathbb{P}_{B\|A}(1\|1) = \mathbb{P}_{B|C\|A}(1|1\|1)\mathbb{P}_{C\|A}(1\|1) + \mathbb{P}_{B|C\|A}(1|0\|1)\mathbb{P}_{C\|A}(0\|1).$$

Since C is a parent of A,

$$\mathbb{P}_{C\|A}(.\|1) = \mathbb{P}_C(.).$$

It follows that

$$\mathbb{P}_{B\|A}(1\|1) = \sum_{x=0}^{1} \mathbb{P}_{B|C\|A}(1|x\|1)\mathbb{P}_{C\|A}(x\|1) = \sum_{x=0}^{1} \mathbb{P}_{B|C\|A}(1|x\|1)\mathbb{P}_C(x).$$

Similarly,

$$\mathbb{P}_{B\|A}(1\|0) = \sum_{x=0}^{1} \mathbb{P}_{B|C\|A}(1|x\|0)\mathbb{P}_C(x).$$

It now follows directly from the assumptions that

$$\mathbb{P}_{B\|A}(1\|1) < \mathbb{P}_{B\|A}(1\|0),$$

which is the stated result. □

Identifiability An effect is said to be *identifiable* if it can be estimated from data alone.

In the example above, when C is a *common cause* of both A and B, the *effect* of A on B is *identifiable*.

When C is on the *causal path* between A and B, the effect of treatment A on condition B is *not* identifiable from data alone; it is necessary to additionally *control* for the effect of B

5.6 Identifiability: Back-Door and Front-Door Criteria

In a wide variety of situations, the aim is to compute the effects of an intervention, when it is not possible to carry out a controlled experiment. The following example, introduced by Pearl, introduced the issues involved.

Example 5.3.

Consider an experiment in which soil fumigants X are to be used to increase oat crop yields Y, by controlling the eelworm population, Z. These may also have direct effects, both beneficial and adverse, on yields, besides the control of eelworms. We would like to assess the total effects of the fumigants on yields when the study is complicated by several factors. First, controlled, randomised experiments are infeasible: farmers insist on deciding for themselves which plots are to be fumigated. Secondly, the farmers' choice of treatment depends on last year's eelworm population Z_0. This is an unknown quantity, but is strongly correlated with this year's population. This presents a classic case of confounding bias, which interferes with the assessment of the treatment effects, regardless of sample size. Fortunately, through laboratory analysis of soil samples, the eelworm populations before and after treatment can be determined. Furthermore, since fumigants are only active for a short period, they do not affect the growth of eelworms surviving the treatment; eelworm growth depends on the

population of bird and other predators. This, in turn, is correlated with last year's eelworm population and hence with the treatment itself.

The situation may be represented by the causal diagram in Figure 5.11. The variables are:

- X fumigants,
- Y crop yields,
- Z_0 last year's eelworm population,
- Z_1 eelworm population before treatment,
- Z_2 eelworm population after treatment,
- Z_3 eelworm population at the end of the season,
- B population of birds and other predators.

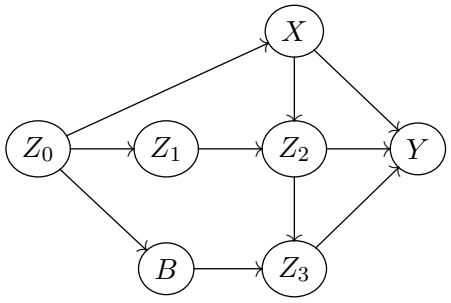

Figure 5.11: A causal diagram representing the effect of fumigants X on yields Y

In this example, the variables B and Z_0 are *hidden* variables.

The issue is whether *interventional* probabilities $\mathbb{P}_{Y\|X}(.\|X \leftarrow x)$ may be computed from information on the observables (Z_1, Z_2, Z_3, X, Y). When they can, they are said to be *identifiable*.

Definition 5.4 (Identifiable)**.** *The causal effect of X on Y is said to be* identifiable *if the quantity $\mathbb{P}_{Y\|X}$ can be computed uniquely from the probability distribution of the observable variables.*

In this section, two graphical conditions are described which ensure that causal effects can be estimated consistently from observational data. The first of these is named *back door criterion* and is equivalent to the *ignorability condition* of Rosenbaum and Rubin. The second of these is the *front-door criterion*. This involves covariates which are affected by the treatment (in this example Z_2 and Z_3).

5.6.1 Back Door Criterion

The back door criterion is defined as follows:

Definition 5.5 (Back Door Criterion)**.** *A set of nodes C satisfies the* back door criterion *relative to an ordered pair of nodes $(X, Y) \in V \times V$ if*

1. *no node of C is a descendant of X* **and**
2. *C blocks every trail (in the sense of D-separation) between X and Y which contains an edge pointing to X.*

If A and B are two disjoint subsets of nodes, C is said to satisfy the back door criterion relative to (A, B) if it satisfies the back door criterion relative to any pair $(X_i, X_j) \in A \times B$.

Example 5.4.

In Figure 5.11, the set $C = \{Z_0\}$ satisfies the back door criterion relative to (X, Y). The node Z_0 is unobservable. The set $C = \{Z_1, Z_2, Z_3\}$ does block all trails between X and Y with an arrow pointing into X, but Z_2 and Z_3 are descendants of X and therefore this set does not satisfy back door criterion. □

The name 'back door criterion' reflects the fact that the second condition requires that only trails with nodes pointing at X_i be blocked. The remaining trails can be seen as entering X_i through a back door.

Example 5.5.

Consider the back door criterion DAG, given in Figure 5.12. The sets of variables $C_1 = \{Z_3, Z_4\}$ and $C_2 = \{Z_4, Z_5\}$ satisfy the back door criterion relative to the ordered pair of nodes (X, Y), whereas $C_3 = \{Z_4\}$ does *not* satisfy the criterion relative to the ordered pair of nodes (X, Y); if Z_4 is instantiated, the Bayes ball may pass through the *collider* connection from Z_1 to Z_2.

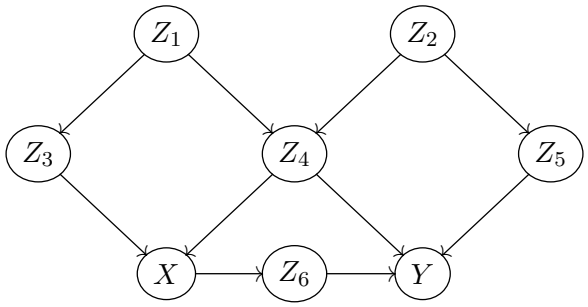

Figure 5.12: Back Door Criterion

Identifiability Consider a causal network and A a subset of the variables which satisfies the back door criterion with respect to an ordered pair (X, Y). Such a set of variables A plays a similar role to the variable C in the discussion on *confounding*; if we can observe these variables, then we can estimate the intervention probability *without* a controlled experiment; otherwise we cannot.
If a set of variables A satisfying the back door criterion with respect to (X, Y) can be chosen such that $\mathbb{P}_A$ and $\mathbb{P}_{Y|A,X}$ can be estimated from the observed data, then the distribution $\mathbb{P}_{Y\|X}$ can also be estimated from the observed data.

Identifiability If a set of variables Z satisfies the back door criterion relative to (X, Y), then the causal effect of X to Y is given by the formula

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Y|X,Z}\mathbb{P}_Z\right)^{\downarrow(X\cup Y)} \tag{5.6}$$

and the intervention of X on Y is said to be *identifiable*.

Formula (5.6) is named *adjustment for concomitants*. The word *identifiability* refers to the fact that the concomitants Z satisfying the back door criterion are observable and hence it is possible to compute, or *identify* the *intervention* probability $\mathbb{P}_{Y\|X}(y\|x)$ using the 'see' conditional probabilities $(\mathbb{P}_{X_j|\mathrm{Pa}_j})_{j=1}^d$.

5.6.2 Front Door Criterion

The *front door criterion* is defined as follows:

Definition 5.6 (Front Door Criterion)**.** *A set of variables Z satisfies the front door criterion relative to the ordered pair (X, Y) if:*

- *Z intercepts all directed paths from X to Y,*
- *there is no back-door path between X and Z,*
- *every back-door path between Z and Y is blocked by X.*

The situation is illustrated in Figure 5.13. The variable U is a hidden (latent) variable. The variable Z satisfies the front door criterion relative to (X, Y).

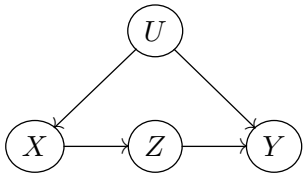

Figure 5.13: Front Door Criterion

The result is the following:

Theorem 5.7 (Front Door Criterion)**.** *Let Z satisfy the front door criterion relative to the ordered pair (X, Y). Then the causal effect on Y of an intervention on X is:*

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Z|X}\mathbb{P}_{Y|Z}\right)^{\downarrow Z}.$$

This is self evident; note that $\mathbb{P}_{Y|Z} = \left(\mathbb{P}_{Y|Z,U}\mathbb{P}_U\right)^{\downarrow(Y\cup Z)}$. In other words, if the see-conditional $\mathbb{P}_{Z|X}$ and $\mathbb{P}_{Y|Z}$ are available, then the intervention $\mathbb{P}_{Y\|X}$ may be computed. □

5.6.3 Non-Indentifiability

There are various conditions for non-identifiability of $\mathbb{P}_{Y\|X}$. These include:

1. A *necessary* condition is that there is an unblockable back-door path between X and Y; that is, a path ending with an arrow pointing into X which cannot be blocked by observable non-descendants of X. This is not a sufficient condition, as Figure 5.13 illustrates. This shows a situation where there is a non-blockable back-door path, yet $\mathbb{P}_{Y\|X}$ is identifiable (front-door criterion).

2. A *sufficient* condition for identifiability of $\mathbb{P}_{Y\|X}$ is existence of a confounding path between X and any of its children on a path from X to Y; two examples are given in Figure 5.14.

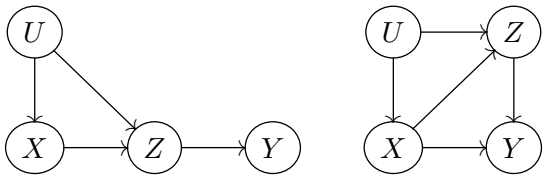

Figure 5.14: Sufficient condition for identifiability

3. Local identifiability is not a sufficient condition for global identifiability. In Figure 5.15, $\mathbb{P}_{Z_1\|X}$, $\mathbb{P}_{Z_2\|X}$, $\mathbb{P}_{Y\|Z_1}$, $\mathbb{P}_{Y\|Z_2}$ are all identifiable, but $\mathbb{P}_{Y\|X}$ is not.

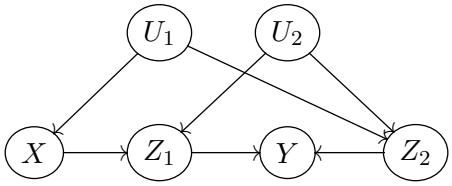

Figure 5.15: Sufficient condition for identifiability

Chapter 6

Time Series

6.1 Introduction

This chapter provides a very brief introduction to Time Series data, the decomposition into *trends*, *seasonal* (oscillating) components and *noise*, which is usually a *stationary process*. We'll discuss *Holt-Winters* filtering, a robust technique for forecasting which dates back to 1960 and which, nevertheless, still outperforms many more 'modern' techniques.

An observed *time series* is a set of observations $(x_t)_{t\in\mathbb{N}}$

Definition 6.1 (Time Series Model)**.** *A* time series model *for the observed data $\{x_t : t \in \mathcal{T}\}$ is the hypothesis that the observed data is an observation of a sequence of random variables $\{X_t : t \in \mathcal{T}\}$ and the specification of its joint probability distribution, or possibly only its expectations and covariances.*

A time series can only be observed at a finite number of times, $(x_t)_{t=1}^n$ and the n observations are a realisation of an n dimensional random vector $X = (X_1, X_2, \ldots, X_n)$. These random variables may be considered to come from an infinite sequence $\{X_t, t \in \mathbb{Z}_+ \quad \text{or} \quad \mathbb{Z}\}$, a *stochastic process*.

Example 6.1 (The binary process)**.**

A simple example of a stochastic process $\{X_t, t \in \mathbb{Z}_+\}$ is a process where the variables are i.i.d. (independent identically distributed) satisfying

$$\mathbb{P}(X_t = 1) = \mathbb{P}(X_t = -1) = \frac{1}{2}.$$

For this process, the finite dimensional marginals are well defined; for any $i_1 < \ldots < i_n$,

$$\mathbb{P}(X_{i_1} = j_1, X_{i_2} = j_2, \ldots, X_{i_n} = j_n) = 2^{-n}$$

for any $\{j_1, \ldots, j_n\} \in \{-1, 1\}^n$. □

Definition 6.2 (IID noise)**.** *A process $\{X_t, t \in \mathbb{Z}\}$ is said to be an IID noise with mean 0 and variance σ^2, written*

$$\{X_t\} \sim IID(0, \sigma^2),$$

if the random variables X_t are independent and identically distributed with $\mathbb{E}[X_t] = 0$ and $\mathbf{V}(X_t) = \sigma^2$.

Notation Througout, $\mathbf{V}(.)$ will be used to denote variance.
The binary process is clearly an example of an $\mathrm{IID}(0,1)$ noise, since the variables are independent, $\mathbb{E}[X_t] = -1 \times \frac{1}{2} + 1 \times \frac{1}{2} = 0$ and $\mathbf{V}(X_t) = \mathbb{E}[X_t^2] - \mathbb{E}[X_t^2] = \mathbb{E}[X_t^2] = 1$.

In many situations, the complete specification of the underlying stochastic process is not required; the methods will generally rely only on its means and covariances. Sometimes even less general assumptions are needed, but these will not be treated here.

Definition 6.3 (Mean function, Covariance function). *Let $\{X_t, t \in \mathcal{T}\}$ be a stochastic process with $\mathbf{V}(X_t) < \infty$ for each $t \in \mathcal{T}$. The* mean function *of $\{X_t\}$ is denoted by μ_X, or simply μ when there is no danger of ambiguity:*

$$\mu_X(t) := \mathbb{E}[X_t], \quad t \in \mathcal{T} \tag{6.1}$$

The covariance function *of $\{X_t\}$ is denoted by C_X or C when there is no danger of ambiguity and is defined as:*

$$C_X(r,s) := \mathbf{C}(X_r, X_s), \quad r,s \in \mathcal{T}. \tag{6.2}$$

The symbol $\mathbf{C}$ will be used to denote covariance.

6.2 Stationarity

A stochastic process is said to be *stationary*, if its statistical properties do not change with time. Formally, stationarity is defined in the following way.

Definition 6.4 (Stationary, Strictly Stationary, Wide sense stationary). *A time series $\{X_t, t \in \mathbb{Z}\}$ is said to be* weakly stationary*, or* wide sense stationary*, or simply* stationary *if*

1. $\mathbf{V}(X_t) < \infty$ *for all* $t \in \mathbb{Z}$,
2. $\mu_X(t) = \mu$ *for all* $t \in \mathbb{Z}$,
3. $C_X(r, r+h) = C_X(0,h)$ *for all* $r, h \in \mathbb{Z}$.

A process is said to be strictly stationary *if any finite collection $(X_{n_1}, \ldots, X_{n_k})$ has the same distribution as $(X_{n_1+t}, \ldots, X_{n_k+t})$ for any $k \geq 1$ and any $(n_1, \ldots, n_k, t) \in \mathbb{Z}$.*

Let B be the backward shift operator $(BX)_t = X_{t-1}$, with powers given by $(B^j X)_t = X_{t-j}$. Strict stationarity means that $B^h X$ has the same distribution for all $h \in \mathbb{Z}_+$.

In most practical situations, only weak stationarity is considered; usually only expectation and covariance, at most, can reasonably be assessed from data.

The third point in the definition of weak stationarity implies that $C_X(r,s)$ depends on r and s only through $r - s$. It is therefore convenient to define

$$\gamma_X(h) := C_X(h,0).$$

If only one time argument appears in γ, then the process is stationary. The value h is referred to as the *lag*.

Definition 6.5. *Let $\{X_t, t \in \mathbb{Z}\}$ be a stationary time series. The* autocovariance function (ACVF) *of $\{X_t\}$ is defined as*

$$\gamma_X(h) = \mathbf{C}(X_{t+h}, X_t).$$

The autocorrelation function (ACF) *is defined as:*

$$\rho_X(h) := \frac{\gamma_X(h)}{\gamma_X(0)}.$$

A simple example of a stationary process is the so-called *white noise*.

Definition 6.6 (White noise)**.** *A process $\{X_t, t \in \mathbb{Z}\}$ is said to be a white noise with mean μ and variance σ^2, written*

$$\{X_t\} \sim WN(\mu, \sigma^2),$$

if $\mathbb{E}[X_t] = \mu$ for all $t \in \mathbb{Z}$ and

$$\gamma(h) = \begin{cases} \sigma^2 & \text{if } h = 0, \\ 0 & \text{if } h \neq 0. \end{cases}$$

Note that IID noise is an example of white noise, but not necessarily vice versa; the underlying distribution can be different even if the mean and covariance structures are the same.

A strictly stationary time series $\{X_t, t \in \mathbb{Z}\}$ with $\mathbf{V}(X_t) < \infty$ is stationary. A stationary time series $\{X_t, t \in \mathbb{Z}\}$ does not need to be strictly stationary

From now on, the term 'stationary' will be used to denote 'weakly' or 'wide sense stationary'; the term *strictly stationary* will be used for the stronger assumption.

Example 6.2 (AR(1) process)**.**

Autoregressive (AR) processes will be considered in more detail later. A process $\{X_t, t \in \mathbb{Z}\}$ is said to be AR(1) if it stationary and satisfies:

$$X_t = \phi X_{t-1} + Z_t \qquad \{Z_t\} \sim WN(0, \sigma^2).$$

For this process, the autocovariance may be computed as follows: by squaring up both sides and using $\gamma_X(0) = \mathbf{V}(X_t)$,

$$\gamma_X(0) = \phi^2 \gamma_X(0) + \sigma^2 \Rightarrow \gamma_X(0) = \frac{\sigma^2}{1-\phi^2}.$$

for $h \geq 1$,

$$\gamma_X(h) = \mathbf{C}(X_{t+h}, X_t) = \phi\mathbf{C}(X_{t+h-1}, X_t) + \mathbf{C}(Z_{t+h}, X_t) = \phi\gamma_X(h-1)$$

so that, since $\gamma_X(-h) = \gamma_X(h)$,

$$\gamma_X(h) = \frac{\sigma^2}{(1-\phi^2)}\phi^{|h|}.$$

Its autocorrelation function (ACF) is

$$\rho_X(h) = \phi^{|h|}.$$

Note that the AR(1) process is not well defined if $|\phi| \geq 1$. □

6.3 Trends and Seasonal Components

The classical decomposition model is:

$$X_t = \mu_t + s_t + Y_t,$$

where

- μ_t is a slowly changing function (the 'trend component');
- s_t is a function with known period d (the 'seasonal component');
- Y_t is a stationary time series.

The aim is to extract the deterministic components μ_t and s_t and estimate them and then check whether or not the residual component Y_t is a stationary time series.

6.3.1 No Seasonal Component

Assume that

$$X_t = \mu_t + Y_t, \quad t = 1, \ldots, n$$

where, without loss of generality, $\mathbb{E}[Y_t] = 0$. There are several methods for estimating μ. Three are considered here; *least squares*, *moving average* and *differencing*.

Method 1 : Least Squares estimation of μ_t The function μ_t is modelled by a function with as few parameters as necessary for accurate modelling and the parameters are estimated by the least squares technique. For example, suppose that μ_t can be modelled by a quadratic function, $\mu_t = a_0 + a_1 t + a_2 t^2$. The parameters $(a_k)_{k=0}^2$ are estimated by $(\widehat{a}_k)_{k=0}^2$, chosen to minimise

$$\sum_{t=1}^{n}(x_t - a_0 - a_1 t - a_2 t^2)^2.$$

Method 2 : Smoothing by means of a moving average Let q be a non-negative integer and consider a smoothed version of X defined by

$$W_t := \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t+j}, \quad q+1 \le t \le n-q.$$

If it turns out that μ is approximately linear over the time interval $[t-q, t+q]$ and also that q is sufficiently large so that $\frac{1}{2q+1}\sum_{j=-q}^{q} Y_{t+j} \simeq 0$, then

$$W_t = \frac{1}{2q+1} \sum_{j=-q}^{q} \mu_{t+j} + \frac{1}{2q+1} \sum_{j=-q}^{q} Y_{t+j} \simeq \mu_t.$$

For $t \le q$ and $t > n-q$, W has to be defined in a different way. For example,

$$W_t = \begin{cases} \frac{1}{2t+1}\sum_{j=-t}^{t} X_{t+j} & t = 1, \ldots, q \\ \frac{1}{2(n-t)+1}\sum_{j=-(n-t)}^{n-t} X_{t-j} & t = n-q+1, \ldots, n. \end{cases}$$

Unless μ_t is a straight line and the stationary time series component Y is very small, it will not be possible to find a q satisfying both the conditions that μ is approximately linear over the interval $[t-q, t+q]$ (requiring small q) and such that $\frac{1}{2q+1}\sum_{t-q}^{t+q} Y_s \simeq 0$ (requiring large q).

A more general expression for a *linear filter* is:

$$\widehat{\mu}_t = \sum_j a_j X_{t+j},$$

where $\sum a_j = 1$ and $a_j = a_{-j}$. Such a filter will allow a *linear trend* $\mu_t = \alpha_0 + \alpha_1 t$ to pass without distortion since

$$\sum_j a_j(\alpha_0 + \alpha_1(t+j)) = (\alpha_0 + \alpha_1 t)\sum_j a_j + \alpha_1 \sum_j a_j j = \alpha_0 + \alpha_1 t.$$

It is possible to choose the weights $\{a_j\}$ so that a larger class of trend functions pass without distortion. For example, the *Spencer 15-point moving average*, defined as

$$\begin{cases} [a_0, a_{\pm 1}, \ldots, a_{\pm 7}] = \frac{1}{320}[74, 67, 46, 21, 3, -5, -6, -3] \\ a_j = 0 \text{ for } |j| > 7 \end{cases}$$

allows a cubic trend to pass without distortion. That is, applied to $\mu_t = at^3 + bt^2 + ct + d$,

$$\widehat{\mu}_t = \sum a_j X_{t+j} = \sum a_j \mu_{t+j} + \sum a_j Y_{t+j} \simeq \sum a_j \mu_{t+j} = \mu_t.$$

Conditions required for a filter to pass a trend which is polynomial of degree k without distortion may be computed.

Method 3: Differencing to generate stationarity The difference operator ∇ is defined by

$$\nabla X_t = X_t - X_{t-1} = (1 - B)X_t,$$

where B is the *backward shift operator*. That is, $(BX)_t = X_{t-1}$. For positive integer k, ∇^k is defined by: by:

$$\nabla^k X_t = \nabla(\nabla^{k-1} X)_t.$$

For example,

$$\nabla^2 X_t = \nabla X_t - \nabla X_{t-1} = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}.$$

Using the backward shift operator, this may be expressed as:

$$\nabla^2 X_t = (1 - B)^2 X_t = (1 - 2B + B^2)X_t = X_t - 2X_{t-1} + X_{t-2}.$$

For a linear trend $\mu_t = a + bt$,

$$\nabla X_t = \nabla \mu_t + \nabla Y_t = a + bt - a - b(t-1) + \nabla Y_t = b + \nabla Y_t.$$

For the covariance,

$$\begin{aligned} \mathbf{C}(\nabla Y_t, \nabla Y_s) &= \mathbf{C}(Y_t, Y_s) - \mathbf{C}(Y_{t-1}, Y_s) - \mathbf{C}(Y_t, Y_{s-1}) + \mathbf{C}Y_{t-1}, Y_{s-1}) \\ &= \gamma_Y(t-s) - \gamma_Y(t-s-1) - \gamma_Y(t-s+1) + \gamma_Y(t-s) \\ &= 2\gamma_Y(t-s) - \gamma_Y(t-s+1) - \gamma_Y(t-s-1). \end{aligned}$$

It follows that ∇X_t is stationary with

$$\mu_{\nabla X} = b \qquad \gamma_{\nabla X}(h) = 2\gamma_Y(h) - \gamma_Y(h+1) - \gamma_Y(h-1).$$

In general, if $\mu_t = \sum_{j=0}^{k} c_j t^j$, then

$$\nabla^k X_t = k! c_k + \nabla^k Y_t,$$

which is stationary.

6.3.2 Trend and Seasonality

Now consider the model with a seasonal component:

$$X_t = \mu_t + s_t + Y_t,$$

where $\mathbb{E}[Y_t] = 0$, $s_{t+d} = s_t$ and $\sum_{k=1}^{d} s_k = 0$. For simplicity in the representation, assume that n/d is an integer; in any reasonable modelling situation, n and d will be chosen so that n/d is an integer.

In models with a seasonal component, the data is often indexed by period and time-unit;

$$x_{j,k} = x_{k+d(j-1)}, \quad k = 1, \ldots, d, \ j = 1, \ldots, \frac{n}{d}.$$

In this notation, $x_{j,k}$ is the observation at the k:th time-unit of the j:th period.

Three methods for dealing with seasonal components will be considered; the *small trend* method, the *moving average estimation* method and the *differencing at lag d* method.

Method S1: Small trends If the trend is considered to be constant during each period, the model may be written as:

$$X_{j,k} = \mu_j + s_k + Y_{j,k}.$$

A natural way to estimate the trend is:

$$\widehat{\mu}_j = \frac{1}{d} \sum_{k=1}^{d} x_{j,k}$$

and a natural method for the seasonal component is:

$$\widehat{s}_k = \frac{d}{n} \sum_{j=1}^{n/d} (x_{j,k} - \widehat{\mu}_j).$$

Method S2: Moving average estimation For a known period d, the trend is estimated by applying a moving average to eliminate the seasonal component and to reduce the noise. For d even set $q = d/2$. The trend is estimated by:

$$\widehat{\mu}_t = \frac{0.5x_{t-q} + x_{t-q+1} + \cdots + x_{t+q-1} + 0.5x_{t+q}}{d}.$$

For for d odd, set $q = (d-1)/2$. The trend is estimated by:

$$\widehat{\mu}_t = \frac{x_{t-q} + x_{t-q+1} + \cdots + x_{t+q-1} + x_{t+q}}{d},$$

for $q + 1 \leq t \leq n - q$.

The seasonal component s_k is then estimated in the following way. Set

$$w_k = \frac{1}{\text{number of summands}} \sum_{\frac{q-k}{d} < j \leq \frac{n-q-k}{d}} (x_{k+jd} - \widehat{\mu}_{k+jd}).$$

The seasonal component satisfies $\sum_{k=1}^{d} \widehat{s}_k = 0$ and therefore the estimates are:

$$\widehat{s}_k = w_k - \frac{1}{d} \sum_{i=1}^{d} w_i, \quad k = 1, \ldots, d.$$

Method S3: Differencing at lag d Define the lag-d difference operator ∇_d by

$$\nabla_d X_t = X_t - X_{t-d} = (1 - B^d) X_t.$$

Then

$$\nabla_d X_t = \nabla_d \mu_t + \nabla_d Y_t.$$

This has no seasonal component and the methods for dealing with time series without a seasonal component may be applied.

6.4 Autocovariance and Spectral Density of a stationary time series

Recall Definition 6.4 of a weakly stationary time series. It follows directly from the definition that:

$$\begin{cases} \gamma(0) \geq 0, \\ |\gamma(h)| \leq \gamma(0) \quad \text{for all } h \in \mathbb{Z}, \\ \gamma(h) = \gamma(-h) \quad \text{for all } h \in \mathbb{Z}. \end{cases} \tag{6.3}$$

An autocovariance function is clearly non-negative definite, since $\sum_{j=1}^{n} \sum_{k=1}^{n} a_j a_k \gamma(t_j - t_k)$ is the *variance* of $\sum_{j=1}^{n} a_j X_{t_j}$.

6.5 Extracting Trend, Seasonal and Noise in R

The `stl` command may be used to decompose a time series into trend, seasonal component and noise. The computation of 'trend' is based on moving average. For illustration, consider the carbon dioxide data from Mauna Loa in the file `atmospheric-carbon-dioxide-recor.csv` .

```
> www =
"https://www.mimuw.edu.pl/~noble/courses/QPEDataScience/data/
atmospheric-carbon-dioxide-recor.csv"
> carbon = read.csv(www)
```

Delete observation 611 which is 'na':

```
> carbon = carbon[-611,]
```

(this deletes the last row, which is 'na').

```
> y = carbon$MaunaLoaCO2
> MaunLoaCo2 = ts(data = y, frequency = 12)
```

(this gets it into an appropriate format - each row represents a year)

```
> output.stl = stl(MaunLoaCo2, s.window = "periodic")
> plot(output.stl)
```

This gives a plot of the original data, the seasonal component, the trend and the 'remainder'.

```
> a <- output.stl$time.series
> acf(a)
```

The time.series part of the stl output gives a decomposition into trend, the seasonal and the noise. The acf gives the autocorrelation for each of these; the trend, seasonal and noise, while the off-diagonals show the cross autocorrelations.

The dotted blue lines indicate 'error' bars. The plot of interest is the residual (or 'remainder'). The acf indicates clear correlations between the residuals; they are not $\mathrm{WN}(0, \sigma^2)$. The plot is in Figure 6.1

To get the sample standard deviation of each column in the time series, try:

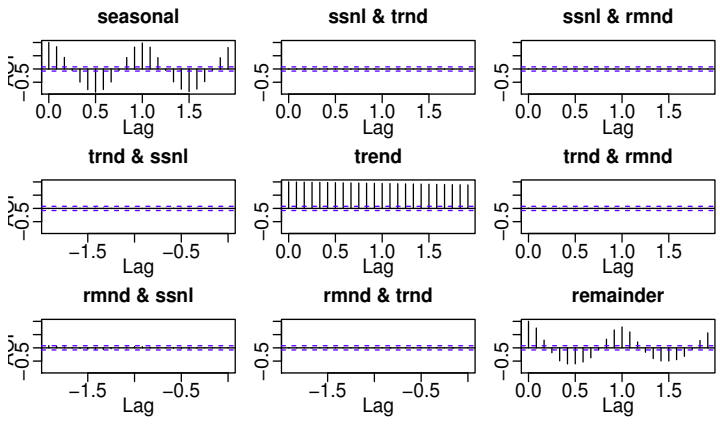

Figure 6.1: Mauna Loa: estimated acf for decomposition

```
> apply(a,2,sd)
 seasonal      trend  remainder
2.0402413 21.0085895  0.2735003
```

This indicates that the remainder is small compared with the trend and seasonal components.

6.6 Holt Winters Filtering

No trend, no seasonal component Given observations $X_1, X_2, \ldots, X_n$ from the model:

$$X_t = \mu + Z_t \qquad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2)$$

where μ is considered to be approximately constant. The method of *exponential smoothing* is to compute a *smoothed* series:

$$\widetilde{X}_t = \lambda X_t + (1-\lambda)\widetilde{X}_{t-1} \qquad \lambda \in (0,1) \tag{6.4}$$

where λ is the *smoothing parameter*. The forecast for time $t+h$ given the series up to time t is

$$\widehat{X}_{t+h|t} = \widetilde{X}_t.$$

The quantity $\widetilde{X}_t$ is the estimate of μ at time t; the assumption is that the underlying value of μ will not change between t and $t+h$.

Linear trend, no seasonal component Holt and Winters independently extended this idea (Holt (1959) and Winters (1960)) to deal with the model

$$X_t = \mu_t + Z_t \qquad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2)$$

under the assumption that the trend is approximately linear. Let $m_t = \mu_t - \mu_{t-1}$. Then the equations suggested by Holt and Winters are:

$$\begin{cases} \widetilde{X}_t = \lambda_1 X_t + (1-\lambda_1)\left(\widetilde{X}_{t-1} + \widetilde{m}_{t-1}\right) \\ \widetilde{m}_t = \lambda_2\left(\widetilde{X}_t - \widetilde{X}_{t-1}\right) + (1-\lambda_2)\widetilde{m}_{t-1} \end{cases} \tag{6.5}$$

where $\widetilde{m}_t$ is the estimate of m_t at time t. The h-step ahead forecasts are then given by:

$$\widehat{X}_{t+h|t} = \widetilde{X}_t + h\widetilde{m}_t.$$

Holt Winters with linear trend and additive seasonal component Now suppose that $\{X_t\}$ is a time series with both trend and seasonal component where the seasonal component $\{s_t\}$ has period d:

$$X_t = \mu_t + s_t + Z_t \qquad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2)$$

The Holt-Winters algorithm accommodates the seasonal component in the following way: let $Y_t = X_t - s_t$, then $\widetilde{Y}_t$ is an approximation of μ_t and

$$\begin{cases} \widetilde{Y}_t = \lambda_1(X_t - \widetilde{s}_{t-d}) + (1-\lambda_1)(\widetilde{Y}_{t-1} + \widetilde{m}_{t-1}) \\ \widetilde{m}_t = \lambda_2(\widetilde{Y}_t - \widetilde{Y}_{t-1}) + (1-\lambda_2)\widetilde{m}_{t-1} \\ \widetilde{s}_t = \lambda_3(X_t - \widetilde{Y}_t) + (1-\lambda_3)\widetilde{s}_{t-d} \end{cases}$$

The initial conditions are:

$$\begin{cases} \widetilde{Y}_{d+1} = X_{d+1} \\ \widetilde{m}_{d+1} = \frac{1}{d}(X_{d+1} - X_1) \\ \widetilde{s}_i = X_i - (X_1 + \widetilde{m}_{d+1}(i-1)) \quad i = 1, \ldots, d+1 \end{cases}$$

The predictors are:

$$\widehat{X}_{t+h|t} = \widetilde{Y}_t + h\widetilde{m}_t + \widetilde{s}_{t+h} \qquad h = 1, 2, \ldots$$

The parameters $\lambda_1, \lambda_2, \lambda_3 \in (0,1)$ may be chosen by minimising the sum of squares of the one-step prediction error on data that has already been observed:

$$\sum_{i=d+2}^{n} \left(X_i - \widehat{X}_{i|i-1}\right)^2$$

Holt Winters Seasonal Multiplicative: Exercise The same technique can be used for an underlying model:

$$X_t = (a+bt)s_t + Z_t \qquad \{Z_t\} \sim IID(0, \sigma^2).$$

If P denotes the period, then the forecasting model for time $t+\tau$ with forecasting origin t is:

$$\widehat{x}_{t,t+\tau} = (\widehat{a}_t + \tau\widehat{b}_t)\widehat{S}_{t+\tau-P}$$

where:

$$\begin{cases} \widehat{a}_t = \lambda_1\left(\frac{x_t}{\widehat{S}_{t-P}}\right) + (1-\lambda_1)\left(\widehat{a}_{t-1} + \widehat{b}_{t-1}\right) \\ \widehat{b}_t = \lambda_2(\widehat{a}_t - \widehat{a}_{t-1}) + (1-\lambda_2)\widehat{b}_{t-1} \\ \widehat{S}_t = \lambda_3\left(\frac{x_t}{\widehat{a}_t}\right) + (1-\lambda_3)\widehat{S}_{t-P}. \end{cases}$$

The justification of this is left as an exercise.

6.6.1 Illustration

The 'Air Passengers' data set is included in the data sets that come with R. Implementation of Holt-Winters can be carried out as follows: Type

```
> data(AirPassengers)
> AP <- AirPassengers
> str(AP)
 Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119
...
```

The data set is now loaded. To see what the data looks like, try

```
> plot(AP)
```

and clearly it is seasonal, with a trend.

Type

```
> ?HoltWinters
```

to obtain the syntax for the command. Note that values for λ_1, λ_2 and λ_3 may be given; if the user does not give the values, then they are computed by minimising the sum of squares of the one-step prediction errors as outlined above. To make a *multiplicative seasonal* Holt Winters, try:

```
> AP.hw <- HoltWinters(AP,seasonal="mult")
> plot(AP.hw)
> legend("topleft",c("observed","fitted"),lty=1,col=1:2)
```

This gives a plot showing both the original data and the one-step predictors. The plot is in Figure 6.2. Prediction is made quite simply using the 'predict' command, which makes the arithmetical computations from the Holt Winters object. The following shows the predictions for the next four years.

```
> AP.predict <-predict(AP.hw,n.ahead=4*12)
> ts.plot(AP,AP.predict,lty=1:2)
```

The plot is found in Figure 6.3.

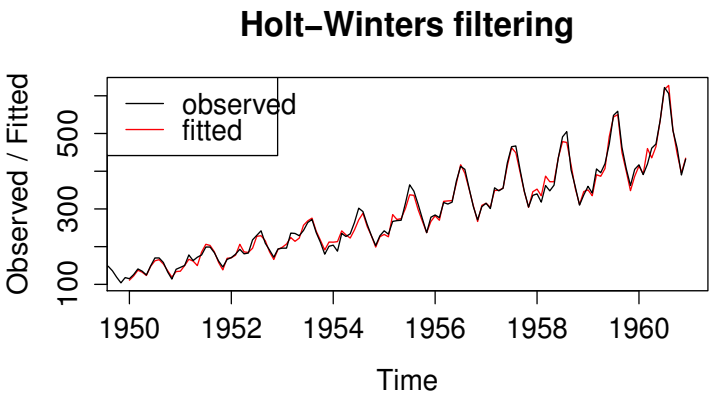

Figure 6.2: Air Passenger data with Holt Winters filtering

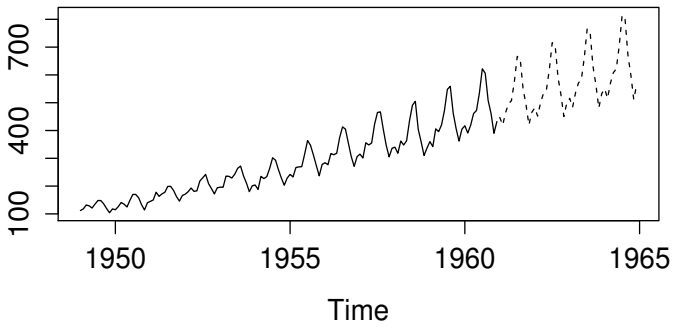

Figure 6.3: Air Passenger data: Holt Winters prediction

6.7 Linear Time Series Models

The classical decomposition of a time series is into trend, seasonal component and a stationary component. Prediction can be improved with better understanding of the stationary process. Very often, the stationary component is not $WN(0,\sigma^2)$; there are correlations. The next task is to build a suitable family of models for the stationary process. The classical models for the stationary process are *linear processes.*

6.8 Definitions and first properties

Definition 6.7 (Linear process, Strictly Linear Process). *A process* $\{X_t, t \in \mathbb{Z}\}$ *is said to be a* linear process *if it has the representation*

$$X_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j Z_{t-j}, \quad \{Z_t\} \sim WN(0,\sigma^2), \tag{6.6}$$

where $\sum_{j=-\infty}^{\infty} |\psi_j| < \infty$. *A stationary time series* $\{X_t\}$ *is* strictly linear *if it has the representation*

$$X_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j Z_{t-j}, \qquad \{Z_t\} \sim IID(0,\sigma^2).$$

where $\sum_j |\psi_j| < +\infty$.

6.8.1 The Spectral Density

The *spectral density* of a stationary process is defined as follows:

Definition 6.8 (Spectral Density). *Let* γ *be the ACVF for a stationary time series. The function* f *defined by*

$$f(\lambda) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} e^{-ih\lambda}\gamma(h), \quad -\pi \le \lambda \le \pi, \tag{6.7}$$

is the spectral density *of* γ. *It is well defined if* $\sum_{h=-\infty}^{\infty} |\gamma(h)| < \infty$.

The ACVF may be recovered from the spectral density:

$$\int_{-\pi}^{\pi} e^{ih\lambda} f(\lambda)\, d\lambda = \int_{-\pi}^{\pi} \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} e^{i(h-k)\lambda}\gamma(k)\, d\lambda = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma(k) \int_{-\pi}^{\pi} e^{i(h-k)\lambda}\, d\lambda = \gamma(h).$$

The spectral density satisfies (among other things):

$$f(0) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} \gamma(h).$$

6.9 MA(q), AR(p) and ARMA(p,q) Processes

The simplest time series model is *white noise*. A first generalisation of white noise is the *moving average* model.

Definition 6.9 (The MA(q) process)**.** *The process* $\{X_t, t \in \mathbb{Z}\}$ *is said to be a* moving average of order q *if*

$$X_t = Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q}, \quad \{Z_t\} \sim WN(0, \sigma^2), \tag{6.8}$$

where $\theta_1, \ldots, \theta_q$ *are constants.*

Definition 6.10 (The AR(p) process)**.** *The process* $\{X_t, t \in \mathbb{Z}\}$ *is said to be an* $AR(p)$ autoregressive process of order p *if it is stationary and if*

$$X_t - \phi_1 X_{t-1} - \ldots - \phi_p X_{t-p} = Z_t, \quad \{Z_t\} \sim WN(0, \sigma^2). \tag{6.9}$$

A process $\{X_t\}$ *is an* $AR(p)$ *process with mean* μ *if* $\{X_t - \mu\}$ *is an* $AR(p)$ *process.*

Definition 6.11 (The ARMA(p, q) process)**.** *A process* $\{X_t, t \in \mathbb{Z}\}$ *is said to be an* $ARMA(p, q)$ process *if it is stationary and*

$$X_t - \phi_1 X_{t-1} - \ldots - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q}, \tag{6.10}$$

where $\{Z_t\} \sim WN(0, \sigma^2)$*. A process* $\{X_t\}$ *is an* $ARMA(p, q)$ *process with mean* μ *if* $\{X_t - \mu\}$ *is an* $ARMA(p, q)$ *process.*

Clearly, an ARMA$(0, q)$ process is an MA(q) process, while an ARMA$(p, 0)$ process is an AR(p) process.

Generating Polynomials for the ARMA Process An important tool for analysis of ARMA processes is the so-called *generating polynomial*. Equations (6.10) can be written as

$$\phi(B) X_t = \theta(B) Z_t, \quad t \in \mathbb{Z},$$

where

$$\phi(z) = 1 - \phi_1 z - \ldots - \phi_p z^p,$$

$$\theta(z) = 1 + \theta_1 z + \ldots + \theta_q z^q$$

and B is the backward shift operator. The polynomials $\phi(\cdot)$ and $\theta(\cdot)$ are called *generating polynomials*.

Causal Models An important property of a time series model is that X_t depends only on information available up to and including time t. A linear time series model that satisfies this is said to be *causal.*

Definition 6.12. *An ARMA(p,q) process defined by the equations*

$$\phi(B)X_t = \theta(B)Z_t \quad \{Z_t\} \sim WN(0,\sigma^2)$$

is said to be causal *if there exists constants $\{\psi_j\}$ such that $\sum_{j=0}^{\infty}|\psi_j| < \infty$ and*

$$X_t = \sum_{j=0}^{\infty}\psi_j Z_{t-j}, \quad t \in \mathbb{Z}. \tag{6.11}$$

Another way to express this is to require that

$$\mathrm{Cov}(X_t, Z_{t+j}) = 0 \quad \text{for} \quad j = 1,2,\ldots \tag{6.12}$$

The following theorem gives conditions under which an ARMA(p,q) process is causal.

Theorem 6.13. *Let $\{X_t\}$ be an ARMA(p,q) for which $\phi(\cdot)$ and $\theta(\cdot)$ have no common zeros. Then $\{X_t\}$ is causal if and only if $\phi(z) \neq 0$ for all $|z| \leq 1$. The coefficients $\{\psi_j\}$ in Equation* (6.11) *are determined by the relation*

$$\psi(z) = \sum_{j=0}^{\infty}\psi_j z^j = \frac{\theta(z)}{\phi(z)}, \quad |z| \leq 1.$$

Proof Assume that $\phi(z) \neq 0$ if $|z| \leq 1$. Then $\frac{1}{\phi(z)}$ is analytic within the unit disc and therefore there exists a $(\xi_j)_{j=0}^{\infty}$ such that $\sum_{j=0}^{\infty}|\xi_j| < +\infty$ such that

$$\frac{1}{\phi(z)} = \sum_{j=0}^{\infty}\xi_j z^j = \xi(z), \quad |z| \leq 1.$$

The operator $\xi(B)$ may be applied to both sides of the equation $\phi(B)X_t = \theta(B)Z_t$ to give:

$$X_t = \xi(B)\theta(B)Z_t,$$

which is well defined since $\sum|\xi_j < +\infty$ and $\theta(z)$ is a polynomial of degree q.

Now assume that $\phi(z) = 0$ for some $|z| \leq 1$ and consider the power series expansion $\frac{1}{\phi(z)} = \sum\xi_j z^j$. The coefficients are not summable, hence X_t does not satisfy the definition of a linear time series model. □

If $\phi(B)X_t = \theta(B)Z_t$ and if $\phi(z) = 0$ for some z with $|z| = 1$ then there does not exist a stationary solution. Consider for example $X_t = X_{t-1} + Z_t$; $\phi(z) = 1 - z$ so that $\phi(1) = 0$. For $Z \sim WN(0,\sigma^2)$ and $X_0 = 0$, then $X_t = \sum_{j=1}^{t} Z_j$ so that $\mathrm{Var}(X_t) = \sigma^2 t$, which is clearly not stationary.

Example 6.3 (AR(1) process)**.**

Let $\{X_t\}$ be an AR(1) process:

$$X_t = Z_t + \phi X_{t-1} \quad \text{or} \quad \phi(z) = 1 - \phi z. \tag{6.13}$$

Since $1 - \phi z = 0$ gives $z = 1/\phi$ it follows that X_t is causal if $|\phi| < 1$. For $|\phi| < 1$,

$$X_t = Z_t + \phi X_{t-1} = Z_t + \phi(Z_{t-1} + \phi X_{t-2}) = Z_t + \phi Z_{t-1} + \phi^2 X_{t-2} = \sum_{j=0}^{\infty} \phi^j Z_{t-j}.$$

It now follows:

$$\gamma_X(h) = \sum_{j=0}^{\infty} \phi^{2j+|h|}\sigma^2 = \frac{\sigma^2\phi^{|h|}}{1-\phi^2}, \tag{6.14}$$

□

If $|\phi| > 1$, Equation (6.13) may be rewritten as:

$$\phi^{-1}X_t = \phi^{-1}Z_t + X_{t-1} \quad \text{or} \quad X_t = -\phi^{-1}Z_{t+1} + \phi^{-1}X_{t+1}.$$

It follows that X_t has representation

$$X_t = -\sum_{j=1}^{\infty} \phi^{-j} Z_{t+j}.$$

If $|\phi| = 1$ there does not exist a stationary solution. □

Definition 6.14. *An ARMA(p,q) process defined by the equations*

$$\phi(B)X_t = \theta(B)Z_t, \quad \{Z_t\} \sim WN(0,\sigma^2)$$

is said to be invertible *if there exists constants $\{\pi_j\}$ such that $\sum_{j=0}^{\infty}|\pi_j| < \infty$ and*

$$Z_t = \sum_{j=0}^{\infty} \pi_j X_{t-j}, \quad t \in \mathbb{Z}. \tag{6.15}$$

Theorem 6.15. *Let $\{X_t\}$ be an ARMA(p,q) for which $\phi(\cdot)$ and $\theta(\cdot)$ have no common zeros. Then $\{X_t\}$ is invertible if and only if $\theta(z) \neq 0$ for all $|z| \leq 1$. The coefficients $\{\pi_j\}$ in Equation (6.15) solve the equation:*

$$\pi(z) = \sum_{j=0}^{\infty} \pi_j z^j = \frac{\phi(z)}{\theta(z)}, \quad |z| \leq 1.$$

Proof The proof follows in the same way as the proof of Theorem 6.13. □

Example 6.4 (MA(1) process)**.**

Let $\{X_t\}$ be an MA(1) process:

$$X_t = Z_t + \theta Z_{t-1} \quad \text{or} \quad \theta(z) = 1 + \theta z.$$

Since $1 + \theta z = 0$ gives $z = -1/\theta$ it follows that X_t is invertible if $|\theta| < 1$. In that case

$$Z_t = X_t - \theta Z_{t-1} = X_t - \theta(X_{t-1} - \theta Z_{t-2}) = \sum_{j=0}^{\infty} (-1)^j \theta^j X_{t-j}.$$

The autocovariance function may be computed quite easily; for a linear stationary process with $\psi_0 = 1$, $\psi_1 = \theta$ and $\psi_j = 0$ for $j \neq 0, 1$, it follows that

$$\gamma(h) = \begin{cases} (1+\theta^2)\sigma^2 & \text{if } h = 0, \\ \theta\sigma^2 & \text{if } |h| = 1, \\ 0 & \text{if } |h| > 1. \end{cases} \tag{6.16}$$

□

6.10 Linear filters

A linear process may be regarded as a *linear filter*. Let $\{X_t\}$ be a time series. A *filter* is an operation on a time series in order to obtain a new time series $\{Y_t\}$. $\{X_t\}$ is called the *input* and $\{Y_t\}$ the *output*. A linear filter $\mathcal{C}$ is the following operation:

$$\mathcal{C}(X)_t := Y_t = \sum_{k=-\infty}^{\infty} c_{t,k} X_k. \tag{6.17}$$

We only consider the situation were $\mathbb{E}\left[X_t^2\right] < \infty$ and $\mathbb{E}\left[Y_t^2\right] < \infty$.

A linear filter is said to be *time-invariant* if $c_{t,k} = c_{t-k}$, in which case it may be written as:

$$Y_t = \sum_{j=-\infty}^{\infty} c_j X_{t-j}.$$

A time-invariant linear filter (TLF) is said to by *causal* if

$$c_j = 0 \quad \text{for } j < 0,$$

When the input $\{X_t\}$ of a time invariant linear filter is stationary, then the output $\{Y_t\}$ is also stationary provided $\sum_k |c_k| < +\infty$.

Definition 6.16 (Stable Linear Filter)**.** *A TLF of the form* (6.17) *is* stable *if* $\sum_{k=-\infty}^{\infty} |c_k| < \infty$.

Definition 6.17 (Transfer function, Power function)**.** *Consider a stable linear filter and set*

$$c(z) = \sum_{j=-\infty}^{\infty} c_j z^j.$$

The function $c(e^{-i\lambda}) := \sum_{j=-\infty}^{\infty} c_j e^{-i\lambda j}$ *is known as the* transfer function*, while the function* $|c(e^{-i\lambda})|^2$ *is known as the* power transfer function.

A filter may be written as $c(B)$, in the sense that

$$Y_t = c(B) X_t$$

where B as usual denotes the backward shift operator.

A linear process is a linear filter where the input is $\mathrm{WN}(0, \sigma^2)$.

Impulse Response Function In general, for a stationary process $\{X_t : t \in \mathbb{Z}\}$, where the variables $\{X_t\}$ are functions of impulses $\{Z_t : t \in \mathbb{Z}\}$, the *impulse response function* $g(s)$ is defined as:

$$g(t; s) = \frac{\partial X_{t+s}}{\partial Z_t} \tag{6.18}$$

In the case of a causal linear filter $X_t = \sum_j c_j Z_{t-j}$, $g(t; s) = g(s) = c_s$.

The impulse response function may be extended to vectors; if $\{\underline{X}_t : t \in \mathbb{Z}\}$ is an m-vector valued process which is a function of vector impulses $\{\underline{Z}_t : t \in \mathbb{Z}\}$, then

$$g_{ij}(t,s) = \frac{\partial X_{t+s,i}}{\partial Z_{t,j}}. \tag{6.19}$$

If $\{\underline{X}_t\}$ is a linear causal vector valued process satisfying $X_{t.j} = \sum_{s\geq 0}\sum_k c_{jk;s} Z_{t-s,k}$ then

$$g_{ij}(t,s) = g_{ij}(s) = c_{ij;s}.$$

6.11 The ARIMA Process

The ARIMA process is defined as follows:

Definition 6.18 (The ARIMA(p,d,q) process)**.** *Let d be a non-negative integer. The process $\{X_t, t \in \mathbb{Z}\}$ is said to be an ARIMA(p,d,q)* process *if $\nabla^d X_t$ is a causal ARMA(p,q) process.*

A causal ARIMA(p,d,q) process $\{X_t\}$ satisfies:

$$\phi(B)X_t = \phi^*(B)(1-B)^d X_t = \theta(B)Z_t, \quad \{Z_t\} \sim \mathrm{WN}(0,\sigma^2), \tag{6.20}$$

where $\phi^*(z) \neq 0$ for all $|z| \leq 1$. The process $Y_t := \nabla^d X_t = (I-B)^d X_t$ satisfies:

$$\phi^*(B)Y_t = \theta(B)Z_t.$$

Example 6.5 (Random Walk)**.**

Consider the simple *random walk* process:

$$X_t = X_{t-1} + Z_t \qquad \{Z_t\} \sim \mathrm{WN}(0,\sigma^2) \qquad 0 < \sigma^2 < +\infty.$$

This is not a stationary process; $\mathrm{Var}(X_t) = t\sigma^2 \overset{t\to+\infty}{\longrightarrow} +\infty$; the central limit theorem gives that $\frac{X_t}{t^{1/2}} \overset{t\to+\infty}{\longrightarrow}_{(d)} N(0,\sigma^2)$. A stationary process may be obtained from X by differencing; let

$$Y_t = \nabla X_t = X_t - X_{t-1} = (I-B)X_t.$$

then Y_t is a stationary process;

$$Y_t = Z_t \sim \mathrm{WN}(0,\sigma^2).$$

It follows that the random walk $\{X_t : t \in \mathbb{Z}_+\}$ is an ARIMA(0,1,0) process. □

It is clear that, for $d \geq 1$ there are no *stationary solutions* of Equation (6.20). Furthermore, neither the mean nor ACVF of $\{X_t\}$ are determined by (6.20), since any process $X_t + Y_t$, where Y_t disappears by differencing d times, satisfies equation (6.20). For example, if Y is a random variable, then $\nabla(X_t + Y) = \nabla X_t$.

For $|\phi| < 1$, the process

$$X_t - \phi X_{t-1} = Z_t \qquad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2)$$

is a causal AR(1) process and is stationary, while for $\phi = 1$, the process is not stationary, but is an ARIMA(0,1,0) process.

Recall that a causal AR(1) process has autocorrelation function

$$\rho(h) = \phi^{|h|}, \quad |\phi| < 1.$$

and hence, for any h,

$$\lim_{|\phi| \uparrow 1} |\rho(h)| = 1.$$

Similarly it holds for any ARMA process that its ACVF decreases slowly if some of the roots of $\phi(z) = 0$ are near the unit circle. From a sample of finite length, it is difficult to distinguish between an ARIMA$(p, 1, q)$ process and an ARMA$(p + 1, q)$ where $\phi(z)$ has a root near the unit circle. An estimated ACVF that decreases slowly indicates that differencing may be advisable.

Suppose that $\{X_t\}$ is a causal and invertible ARMA(p, q) process:

$$\phi(B)X_t = \theta(B)Z_t, \quad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2),$$

where $\theta(z) \neq 0$ for all $|z| \leq 1$ and $\phi(z)$ has no roots in the unit circle. Then

$$\phi(B)\nabla X_t = \phi(B)(1 - B)X_t = \theta(B)(1 - B)Z_t, \quad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2),$$

from which it follows that ∇X_t is a causal, but non-invertible ARMA$(p, q + 1)$ process. A unit root in the moving average polynomial indicates that X_t has been *overdifferenced*.

6.11.1 Testing for Unit Roots

For given time series data, there are tests available to indicate whether or not there are unit roots present. One common test is the *Dickey Fuller* test, introduced by Dickey and Fuller (1979), which has been refined to produce the *Augmented Dickey Fuller Test* (abbrieviated to ADF). This is a relatively straightforward test. It assumes that $\{Z_t\} \sim IIDN(0, \sigma^2)$ (independent, identically distributed *normal* random variables) and works on the principles of linear regression.

The disadvantage of this test is that presence of a unit root is the *null* hypothesis. In statistics, a *null* hypothesis is never *accepted*; the result of a hypothesis test is either 'reject the null hypothesis and accept the alternative hypothesis', or 'do not reject the null hypothesis'.

Failure to reject a null hypothesis does not imply that the hypothesis is true; it simply means that there is not enough evidence to establish the alternative.

There is a test, known as the KPSS test, which states the presence of a unit root as the *alternative* hypothesis; rejecting the *null* hypothesis of no unit root *establishes* that there is a unit root.

The Dickey Fuller Test Consider the AR(1) model:

$$X_t = \phi X_{t-1} + Z_t \qquad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2).$$

Subtracting X_{t-1} from both sides gives:

$$\nabla X_t = (\phi - 1)X_{t-1} + Z_t \Rightarrow \nabla X_t = \beta X_{t-1} + Z_t \qquad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2).$$

The Dickey Fuller test simply takes a linear regression of $\{\nabla X_t\}$ against X_{t-1} and estimates the parameter β in the model, with error bounds. The test may also include a constant, and a deterministic drift; using linear regression, assuming $\{Z_t\} \sim \mathrm{IID}N(0, \sigma^2)$, one tests whether the parameter β is significant in either

$$\nabla X_t = \alpha + \beta X_{t-1} + Z_t$$

or

$$\nabla X_t = \alpha_0 + \alpha_1 t + \beta X_{t-1} + Z_t.$$

While standard multiple linear regression techniques may be used, the approach by Dickey and Fuller represents a refinement where the estimates are made in a different way and the distribution of the test statistic $DF_\tau := \frac{\widehat{\beta}}{\mathrm{sd}(\widehat{\beta})}$ turns out not to be exactly t distributed. The distribution is known as the *Dickey–Fuller distribution.*

The tests have low statistical power; they cannot distinguish between a true unit-root ($\beta = 0$) and near unit-root (β close to zero). This is called the 'near observation equivalence' problem.

The Augmented Dickey Fuller Test The testing procedure for the ADF test is the same as for the Dickey–Fuller test but it is applied to the model

$$\nabla X_t = \alpha_0 + \alpha_1 t + \beta X_{t-1} + \delta_1 \nabla X_{t-1} + \cdots + \delta_{p-1} \nabla X_{t-p+1} + Z_t \qquad \{Z_t\} \sim \mathrm{IID}N(0, \sigma^2)$$

The lag length p is determined when applying the test, using standard model building techniques from multiple linear regression analysis. The unit root test is then carried out under the null hypothesis $\beta = 0$ against the alternative hypothesis $\beta < 0$. The test statistic

$$DF_\tau = \frac{\widehat{\beta}}{\mathrm{sd}(\widehat{\beta})}$$

is computed it can be compared to the relevant critical value for the Dickey–Fuller test.

The KPSS Test The KPSS test was introduced by Kwiatkoski, Phillips, Schmidt and Shin in 1992 (Biometrics vol. 54 pp 159 - 178). It is based on the LM (Lagrange Multiplier) test in regression for omitted variables.

Assume that a time series $\{Y_t\}$ can be decomposed into a linear trend ξt, a random walk R_t and a stationary error process X_t:

$$\begin{cases} Y_t = \xi t + R_t + X_t \\ R_t = R_{t-1} + Z_t \qquad \{Z_t\} \sim IIDN(0, \sigma_Z^2) \end{cases}$$

R_0 is fixed. The hypothesis that $Y_t - \xi t$ is stationary is equivalent to the hypothesis that $\sigma^2 = 0$.

For the test statistic, it is assumed that $\{X_t\} \sim IIDN(0, \sigma_X^2)$. Let $(e_t)_{t \geq 1}$ denote the residuals from an OLS regression $Y_t = \beta_0 + \beta_1 t + \epsilon_t$, let $\widehat{\sigma}_e^2$ the estimate of the error variance from this regression and $S_t = \sum_{i=1}^{t} e_i$. The LM statistic for $Y_1, \ldots, Y_T$ is:

$$LM = \frac{\sum_{t=1}^{T} S_t^2}{\widehat{\sigma}_e^2}$$

Under the assumption that $\sigma_u^2 = 0$, the distribution (or at least the asymptotic distribution) of $\frac{1}{T^2} \sum_{t=1}^{T} S_t^2$ may be computed explicitly and $\widehat{\sigma}_e^2 \overset{T \to +\infty}{\longrightarrow} \sigma_X^2$.

Testing for unit roots using R The following gives a demonstration of a unit root test. Consider the log series of U.S. quarterly GDP from 1947.I to 2008.IV. The file is found in `q-gdp4708.txt` in the course directory. The data is plotted in Figure 6.4.

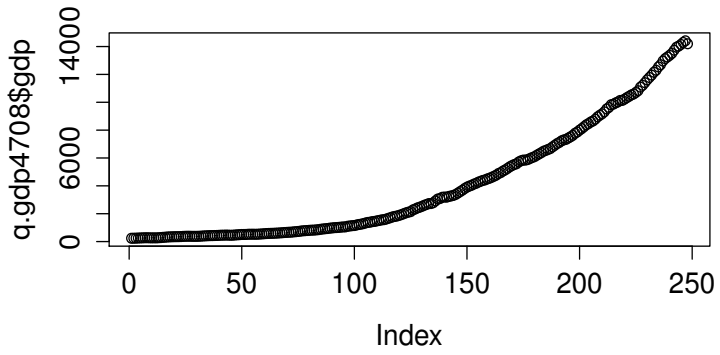

Figure 6.4: US quarterly GDP 1947 - 2008

The following indicates that the unit root test cannot be rejected. The test used is the *KPSS* test.

```
library(urca)
www2 <- "https://www.mimuw.edu.pl/~noble/courses/TimeSeries/data/q-gdp4708.txt"
q.gdp4708 <- read.table(www2, header=T)
a = ur.kpss(q.gdp4708$gdp,type = "tau")
summary(a)
```

The test statistic is larger than the 1% critical value. We may safely reject the null hypothesis of no unit root and accept the alternative of unit root. In fact, we can see that the unit root has multiplicity 2:

```
> library("forecast")
> auto.arima(q.gdp4708$gdp)
Series: q.gdp4708$gdp
ARIMA(0,2,1)

Coefficients:
          ma1
      -0.6438
s.e.   0.0685

sigma^2 estimated as 1361:  log likelihood=-1236.89
AIC=2477.79   AICc=2477.84   BIC=2484.8
```

6.12 SARIMA Processes

Seasonal series are characterised by a strong serial correlation at the seasonal lag and multiples thereof. Seasonal ARIMA models allow for randomness in the seasonal pattern from one cycle to the next.

Definition 6.19 (The SARIMA$(p,d,q)\times(P,D,Q)_s$ Process)**.** *A process $\{X_t\}$ is said to be a Seasonal ARIMA(p,d,q)×(P,D,Q) process with period s if the differenced process*

$$Y_t := (1-B)^d(1-B^s)^D X_t$$

is a causal ARMA process,

$$\phi(B)\Phi(B^s)Y_t = \theta(B)\Theta(B^s)Z_t \qquad \{Z_t\} \sim WN(0,\sigma^2)$$

where

$$\phi(z) = 1 - \phi_1 z - \ldots - \phi_p z^p$$
$$\Phi(z) = 1 - \Phi_1 z^s - \ldots - \Phi_P z^{Ps}$$
$$\theta(z) = 1 + \theta_1 z + \ldots + \theta_q z^q$$
$$\Theta(z) = 1 + \Theta_1 z^s + \ldots + \Theta_Q z^{Qs}.$$

Note that the process $\{Y_t\}$ is causal if and only if both $\phi(z) \neq 0$ and $\Phi(z) \neq 0$ for all $|z| \leq 1$.

Note The SARMA process is a *stationary* process; the mean zero SARMA process satisfies $\mathbb{E}[X_t] \equiv 0$ for all t.

Therefore, the stationary SARMA process is not suitable for the situation where the process has a *deterministic* stationary component (so that $\mathbb{E}[X_t] = s_t$, where s_t is a deterministic periodic function). What is in view here is a process where the *autocovariance* is seasonal.

The SARMA process is therefore not suitable for modelling, for example, a situation where there is a 'January effect', when trade increases in January due to January sales.

Chapter 7

Dynamic Bayesian Networks

7.1 Introduction

Dynamic Bayesian networks (DBNs) are an important tool that have proved useful for a large class of problems. The thesis of Kevin Murphy (2002) provides a comprehensive introduction to the topic.

The first mention of dynamic Bayesian networks seems to be by Dean and Kanazawa (1989). The DBN framework provides a way to extend Bayesian network machinery to model probability distributions over collections of random variables $(\underline{Z}_t)_{t\geq 0}$. The parameter $t \in \{0, 1, 2, \ldots\}$ represents time. Typically, the variables at a time slice t are partitioned into $\underline{Z}_t = (\underline{U}_t, \underline{X}_t, \underline{Y}_t)$ representing the input, hidden and output variables of the model. The term 'dynamic' refers to the fact that the system is dynamic; the basic structure remains the same over time.

Definition 7.1. *A* k - slice Dynamic Bayesian network *is a DAG corresponding to a factorisation of the probability distribution over the variables* $\{\underline{Z}_0, \underline{Z}_1, \ldots\}$ *such that for* $t \geq k$,

$$\mathbb{P}_{Z_0,\ldots,Z_t} = \mathbb{P}_{Z_0} \prod_{s=1}^{k-1} \mathbb{P}_{Z_s|Z_0,\ldots,Z_{s-1}} \prod_{s=k}^{t} \mathbb{P}_{Z_s|Z_{s-k},\ldots,Z_{s-1}}$$

where, for $t \geq k$,

$$\mathbb{P}_{Z_t|Z_{t-k-1},\ldots,Z_{t-1}} = \prod_j \mathbb{P}_{Z_t^j|Pa(Z_t^j)},$$

Z_t^j *is the* j*th node at time* t*, which could be a component of either* X_t*,* Y_t *or* U_t *and the set* $Pa(Z_t^j)$ *of parents of* Z_t^j *belongs to the collection*

$$\underline{Z}_{t-k}, \ldots, \underline{Z}_{t-1}, \{Z_t^1, \ldots, Z_t^{j-1}\}.$$

The arrows within the same time slice do not represent causality.

The requirement is that the subgraph restricted to $\{\underline{Z}_t, \ldots, \underline{Z}_{t+k-1}\}$ *is the same for each* $t \geq 0$ *and the conditional probabilities* $\mathbb{P}_{Z_t^j|Pa(Z_t^j)}$ *are the same for each* $t \geq k$*. Furthermore, for* $1 \leq i \leq j \leq k$*, and each* $s \geq j$*, the subgraph restricted to* $\{\underline{Z}_{s+i}, \ldots, \underline{Z}_{s+j}\}$ *is a subgraph of the subgraph restricted to* $\{\underline{Z}_{s+i-1}, \ldots, \underline{Z}_{s+j}\}$.

The arcs between slices are from left to right and reflecting the causal flow of time. If there is an arc from Z_{t-1}^j to Z_t^j, the node Z^j is said to be *persistent*. The arcs *within* a slice may have arbitrary direction, so long as the overall DBN is a DAG. The arcs within a time slice may be undirected, since they model correlation or constraints rather than causation. The resulting model is then a (dynamic) chain graph.

The parameters of the conditional probabilities $\mathbb{P}_{Z_t^j|\mathrm{Pa}(Z_t^j)}$ are time-invariant for $t \geq k$, i.e., the model is time-homogeneous. If parameters can change, they may be added to the state-space and treated as random variables or alternatively a hidden variable may be added that selects which set of parameters to use.

Within the engineering community, DBNs have become a popular tool, because they can express a large number of models and are often computationally tractable.

DBNs have been successfully applied to in the reconstruction of genetic networks, where genes do not remain static, but rather their expression levels fluctuate constantly. Increased expression level of a gene will result in increased levels of mRNA from that gene which will in turn influence the expression levels of other genes. DBNs have proved to be a successful way of analysing genetic expression data.

With a *Dynamic Bayesian Network*, the $n \times d$ data matrix no longer represents n *independent* instantiations of a random d-vector. Rather, the rows represent time slices of a *process* $\{\underline{X}(t) : t \in \mathbb{N}\}$.

Some assumptions (for example time homogeneity) have to be made in order to learn structure and parameters.

If the number of instantiations n available is large in comparison to d, then standard multivariate time series techniques may be used effectively. If n is small compared with d, other techniques (such as LASSO L^1 regularisation) should be used.

7.2 Multivariate Time Series

A VARMA(p,q) model (vector auto regressive moving average, lags p and q for the auto-regressive and moving average parts respectively) is a model:

$$\underline{X}(t) = \underline{\mu}_0 + t\underline{\mu}_1 + \sum_{j=1}^{p} A_j \underline{X}(t-j) + \sum_{k=1}^{q} B_k \underline{\epsilon}_{t+1-q}$$

where $\underline{\epsilon}_t \sim N(\underline{0}, \Sigma)$ are i.i.d. (the distribution is not necessarily normal, but the normality assumption, if true, leads to sharper estimation).

The MA part often leads to instability for estimation; we therefore only consider VAR(p) processes;

$$\underline{X}(t) = \underline{\mu}_0 + t\underline{\mu}_1 + \sum_{j=1}^{p} A_j \underline{X}(t-j) + \underline{\epsilon}_t$$

The package **vars** fits a vector auto regressive model:

```
> install.packages("vars")
> library(vars)
```

Within **vars**, there is a test data-set `Canada`, which contains 4 macroeconomic indicators; `prod` (labour productivity), `e` (employment), `U` (unemployment rate) and `rw` (real wages). A VAR(2) model is fitted quite simply with the command:

```
> data(Canada)
> can = VAR(Canada,p=2)
> summary(can)

VAR Estimation Results:
=========================
Endogenous variables: e, prod, rw, U
Deterministic variables: const
Sample size: 82
Log Likelihood: -175.819
Roots of the characteristic polynomial:
0.995 0.9081 0.9081 0.7381 0.7381 0.1856 0.1429 0.1429
Call:
VAR(y = Canada, p = 2)


Estimation results for equation e:
==================================
e = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

          Estimate Std. Error t value Pr(>|t|)
e.l1     1.638e+00  1.500e-01  10.918  < 2e-16 ***
prod.l1  1.673e-01  6.114e-02   2.736  0.00780 **
rw.l1   -6.312e-02  5.524e-02  -1.143  0.25692
U.l1     2.656e-01  2.028e-01   1.310  0.19444
e.l2    -4.971e-01  1.595e-01  -3.116  0.00262 **
prod.l2 -1.017e-01  6.607e-02  -1.539  0.12824
rw.l2    3.844e-03  5.552e-02   0.069  0.94499
U.l2     1.327e-01  2.073e-01   0.640  0.52418
const   -1.370e+02  5.585e+01  -2.453  0.01655 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.3628 on 73 degrees of freedom
Multiple R-Squared: 0.9985,Adjusted R-squared: 0.9984
F-statistic:  6189 on 8 and 73 DF,  p-value: < 2.2e-16
```

```
Estimation results for equation prod:
=====================================
prod = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

          Estimate Std. Error t value Pr(>|t|)
e.l1      -0.17277    0.26977  -0.640  0.52390
prod.l1    1.15043    0.10995  10.464 3.57e-16 ***
rw.l1      0.05130    0.09934   0.516  0.60710
U.l1      -0.47850    0.36470  -1.312  0.19362
e.l2       0.38526    0.28688   1.343  0.18346
prod.l2   -0.17241    0.11881  -1.451  0.15104
rw.l2     -0.11885    0.09985  -1.190  0.23778
U.l2       1.01592    0.37285   2.725  0.00805 **
const   -166.77552  100.43388  -1.661  0.10109
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.6525 on 73 degrees of freedom
Multiple R-Squared: 0.9787,Adjusted R-squared: 0.9764
F-statistic: 419.3 on 8 and 73 DF,  p-value: < 2.2e-16


Estimation results for equation rw:
===================================
rw = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

          Estimate Std. Error t value Pr(>|t|)
e.l1     -0.268833   0.322619  -0.833    0.407
prod.l1  -0.081065   0.131487  -0.617    0.539
rw.l1     0.895478   0.118800   7.538 1.04e-10 ***
U.l1      0.012130   0.436149   0.028    0.978
e.l2      0.367849   0.343087   1.072    0.287
prod.l2  -0.005181   0.142093  -0.036    0.971
rw.l2     0.052677   0.119410   0.441    0.660
U.l2     -0.127708   0.445892  -0.286    0.775
const   -33.188339 120.110525  -0.276    0.783
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.7803 on 73 degrees of freedom
Multiple R-Squared: 0.9989,Adjusted R-squared: 0.9987
F-statistic:  8009 on 8 and 73 DF,  p-value: < 2.2e-16


Estimation results for equation U:
==================================
U = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

         Estimate Std. Error t value Pr(>|t|)
e.l1     -0.58076    0.11563  -5.023 3.49e-06 ***
prod.l1  -0.07812    0.04713  -1.658 0.101682
rw.l1     0.01866    0.04258   0.438 0.662463
U.l1      0.61893    0.15632   3.959 0.000173 ***
e.l2      0.40982    0.12296   3.333 0.001352 **
prod.l2   0.05212    0.05093   1.023 0.309513
rw.l2     0.04180    0.04280   0.977 0.331928
U.l2     -0.07117    0.15981  -0.445 0.657395
const   149.78056   43.04810   3.479 0.000851 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.2797 on 73 degrees of freedom
Multiple R-Squared: 0.9726,Adjusted R-squared: 0.9696
F-statistic:   324 on 8 and 73 DF,  p-value: < 2.2e-16



Covariance matrix of residuals:
              e      prod       rw        U
e     0.131635 -0.007469 -0.04210 -0.06909
prod -0.007469  0.425711  0.06461  0.01392
rw   -0.042099  0.064613  0.60886  0.03422
U    -0.069087  0.013923  0.03422  0.07821

Correlation matrix of residuals:
```

```
             e     prod      rw       U
e     1.00000 -0.03155 -0.1487 -0.6809
prod -0.03155  1.00000  0.1269  0.0763
rw   -0.14870  0.12691  1.0000  0.1568
U    -0.68090  0.07630  0.1568  1.0000
```

The default value, which estimates $\underline{\mu}_0$ and sets $\underline{\mu}_1 = 0$ is `const`. To set $\underline{\mu}_0 = 0$ and $\underline{\mu}_1 = 0$, type:

```
> VAR(Canada,p=2,type="none")
```

To set $\underline{\mu}_0 = 0$ while estimating an unknown trend $\underline{\mu}_1$, type:

```
> VAR(Canada,p=2,type="trend")
```

To estimate both an intercept $\underline{\mu}_0$ and a trend $\underline{\mu}_1$, type:

```
> VAR(Canada,p=2,type="both")
```

The `stability` function verifies the covariance stationarity of a VAR process, using cumulative sums of residuals. This may be carried out by:

```
> var.2c=VAR(Canada,p=2,type="const")
> stab=stability(var.2c,type="OLS-CUSUM")
> plot(stab)
```

There are several tests for normality which come under `normality.test`.

```
> normality.test(var.2c)
$JB

JB-Test (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 5.094, df = 8, p-value = 0.7475


$Skewness

Skewness only (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 1.7761, df = 4, p-value = 0.7769
```

```
$Kurtosis

Kurtosis only (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 3.3179, df = 4, p-value = 0.5061
```

The function `serial.test` carries out the Portmanteau (i.e. Ljung-Box) test

```
> serial.test(var.2c,lags.pt=16,type="PT.adjusted")

Portmanteau Test (adjusted)

data:  Residuals of VAR object var.2c
Chi-squared = 231.5907, df = 224, p-value = 0.3497
```

The VARMA model is standard and is treated in any reasonable text on Time Series.

7.3 Lasso Learning

One of the most prominent applications of DBNs is to gene expression data and locating regulatory pathways. The main difficulty is that n (the number of instantiations) tends to be small compared with d (the number of genes under investigation). On the other hand, gene expression networks tend to be sparse.

One technique that has developed and is quite effective in such situations is L^1 regularisation, or LASSO learning.

LASSO and Least Angle Regression Given a set of input measurements $(x_{j,1}, \ldots, x_{j,d})$ for $j = 1, \ldots, n$ and outcome measurement $y_j : j = 1, \ldots, n$, taken as observations on independent variables, the lasso fits a linear model

$$\widehat{y_j} = \widehat{\beta}_0 + \sum_{j=1}^{d} x_j \widehat{\beta}_j.$$

The criterion it uses is:

Minimise $\sum_{j=1}^{n}(y_j - \widehat{y_j})^2$ subject to $\sum_{j=0}^{d} |\beta_j| \leq s$ for a constraint value s.

The bound s is a tuning parameter. When s is sufficiently large, the constraint has no effect and the solution is simply the usual multiple linear least squares regression of y on $x_1, \ldots, x_d$.

For smaller values of s ($s \geq 0$), the solutions are *shrunken* versions of the least squares estimates. The L^1 penalisation often forces some of the coefficient estimates $\widehat{\beta}_j$ to be zero.

The choice of s therefore plays a similar role to choosing the number of predictors in a regression model.

Cross-validation is the standard tool for estimating the best value for s.

Forward stepwise regression achieves the same objective as regularisation by adding in explanatory variables one at a time:

- Start with all coefficients β_j equal to zero.
- Find the predictor x_j which is most correlated to y and add it into the model. Take residuals $r = y - \widehat{y}$.
- Continue, at each stage adding to the model the predictor most correlated with r.
- Until: all predictors are in the model

The *Least Angle Regression* procedure follows the same general scheme, but does not add a predictor *fully* into the model. The coefficient of that predictor is increased only until that predictor is no longer the one most correlated with the residual r. Then some other competing predictor is included.

Least Angle Regression algorithm The algorithm proceeds as follows:

- Start with all coefficients β_j equal to zero.
- Find the predictor x_j most correlated with y.
- Increase the coefficient β_j in the direction of the sign of its correlation with y. Take residuals $r = y - \widehat{y}$. Stop when some other predictor x_k has as much correlation with r as x_j has.
- Increase (β_j, β_k) in their joint least squares direction, until some other predictor x_m has as much correlation with the residual r.
- Continue until: all predictors are in the model

It can be shown that, with one modification, this procedure gives the entire path of lasso solutions, as s is varied from 0 to infinity. The modification needed is: if a non-zero coefficient hits zero, remove it from the active set of predictors and recompute the joint direction.

Cross-Validation Cross validation is a model evaluation method where some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on "new" data. This is the basic idea for the class of model evaluation methods called cross validation.

- **Holdout** The holdout method is the simplest kind of cross validation. The data set is separated into two sets; the *training* set and the *testing* set. The function approximator fits a function using

the training set only. Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before). The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model.

- **K-fold Cross Validation** K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

- **Leave-one-out** Leave-one-out cross validation is K-fold cross validation taken to its logical extreme, with K equal to n, the number of data points in the set. That means that the function approximator is trained on all the data except for one point n separate times and a prediction is made for that point. As before the average error is computed and used to evaluate the model. The evaluation given by leave-one-out cross validation error (LOO-XVE) is good, but at first pass it seems very expensive to compute.

7.3.1 Implementation

There are several packages available in R for DBN learning. One of the most prominent is the **lars** package by Hastie and Efron (2012). Other packages available are: **glmnet** package by Friedman et. al. (2010) and **penalized** by Goeman (2012). For illustration, we use the `arth800MTS` data set from the **GeneNet** package. This describes the expression levels of 800 genes of the Arabidopsis thaliana during the diurnal cycle. We consider a subset `arth12` of 12 of the genes.

```
> library(lars)
> library(GeneNet)
> data(arth800)
> subset=c(60,141,260,333,365,424,441,512,521,578,799)
> arth12=arth800.expr[,subset]
```

Now **lars** is used to estimate a model for a target variable speified by a vector (say `y`) and a set of possible parents specified by a matrix of predictors (say `x`). The `arth800` data set consists of two time series, each of 11 points in length. That is, there are two repeated measurements for each time point. To estimate a VAR(1) process, firstly remove the two repeated measurements for the first time point

of `y` and the two repeated measurements for the last time point of `x`. They cannot be used for LASSO, since $y(t)$ needs $x(t-1)$.

```
> x = arth12[1:(nrow(arth12)-2),]
> y = arth12[-(1:2),"265768_at"]
> lasso.fit = lars(y=y,x=x,type="lasso")
> plot(lasso.fit)
```

The plot is shown in Figure 7.1.

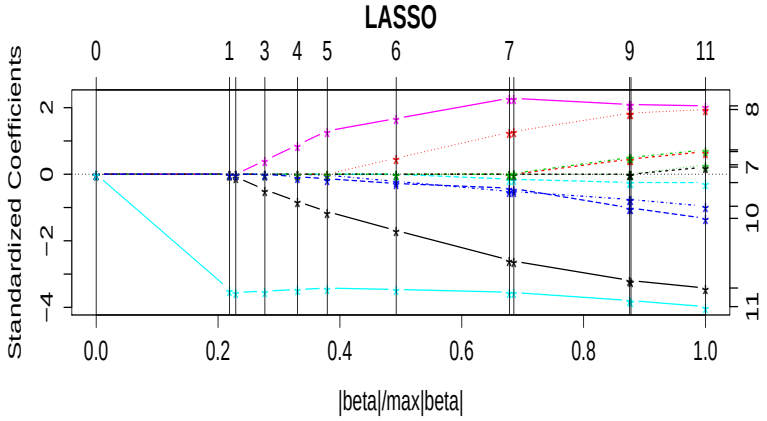

Figure 7.1: Lasso output

The figure is interpreted as follows: the aim is to predict $y(t)$ (the expression levels for gene labelled `265768_at`) by the expression levels one time unit earlier (given at time index $t-2$ because we have double measurements for each time); $x(t-2)$. The regression is carried out by evaluating the coefficients $\underline{\beta}$ which minimise $\sum_{t=3}^{22}(y(t)-\sum_{j=1}^{11}x_j(t-2)\beta_j)^2$, subject to a constraint that $\sum_{j=1}^{11}|\beta_j| \leq t$ for t increasing. For the x-axis, this is presented as $|\beta|/\max|\beta|$, where $|\beta| = \sum_{j=1}^{11}|\beta_j|$ and $\max|\beta|$ is the value of $\sum_{j=1}^{11}|\beta_j|$ for the unconstrained problem.

The values of the coefficients are denoted by different colours and the plot shows how they change as the value of t increases. The vertical lines indicate the points at which new coefficients are introduced. The coefficients may be obtained by

```
> coef(lasso.fit)
```

Structure learning (i.e. deciding which directed edges to include in the network) is carried out via *cross-validation*. The `cv.lars` function does this.

```
> lasso.cv=cv.lars(y=y,x=x,mode="fraction")
```

The *output* gives the MSE (mean squared error) as a function of $|\beta|/\max|\beta|$ (where $|\beta|$ denotes the constraint and $\max|\beta|$ denotes the value of $\sum_{j=1}^{11}|\beta_j|$ for the unconstrained problem) and the output is shown in Figure 7.2. The optimal set of arcs is chosen to minimise the mean squared error.

```
> frac=lasso.cv$index[which.min(lasso.cv$cv)]
> predict(lasso.fit,s=frac,type="coef",mode="fraction")
$s
[1] 0.1919192

$fraction
[1] 0.1919192

$mode
[1] "fraction"

$coefficients
 265768_at  263426_at  260676_at  258736_at  257710_at  255764_at
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 255070_at  253425_at  253174_at  251324_at  245319_at  245094_at
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.6420806
```

The non-zero coefficients indicate the arcs to be included on the gene `265768_at` for the optimal value `s=frac` computed by `cv.lars`.

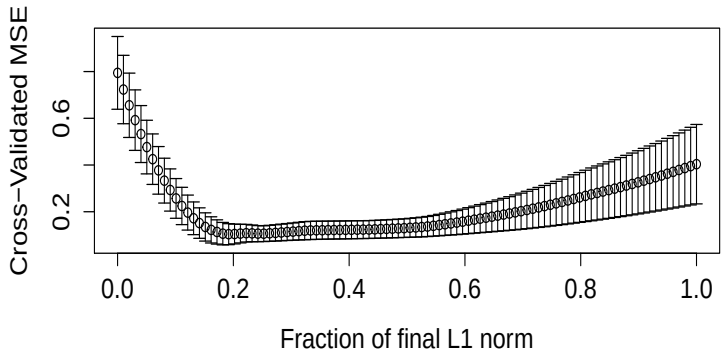

Figure 7.2: Lasso cross validation

The number of steps can be controlled by setting the `mode` argument of `predict` to `step`.

```
> predict(lasso.fit,s=3,type="coef",mode="step")$coefficients
  265768_at   263426_at   260676_at   258736_at   257710_at   255764_at   255070_at   253425_at
-0.02152962  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
  253174_at   251324_at   245319_at   245094_at
 0.00000000  0.00000000  0.00000000 -0.72966658
```

The L^1 penalty can be specified with `mode = ''lambda''`

```
> predict(lasso.fit,s=0.2,type="coef",mode="lambda")$coefficients
 265768_at  263426_at  260676_at  258736_at  257710_at  255764_at  255070_at  253425_at
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 253174_at  251324_at  245319_at  245094_at
 0.0000000  0.0000000  0.0000000 -0.6961228
```

The **lars** package also fits *least angle regression* and *stepwise regression*.

```
> lar.fit=lars(y=y,x=x,type="lar")
> lar.cv=cv.lars(y=y,x=x,type="lar")
> step.fit=lars(y=y,x=x,type="stepwise")
> step.cv=cv.lars(y=y,x=x,type="stepwise")
```

7.4 Inference for Dynamic Bayesian Networks

For a given DBN (where the network structure and the conditional probability potentials have been specified), the queries of interest are usually those of computing the marginal distribution of $X_i(t)$ conditioned on all nodes other than $X_i(t)$ at times $1, \ldots, T$. In line with standard time series problems, these problems fall into three categories:

- If $T = t$, the query is called *filtering*.
- If $T > t$ (node $X_i(t)$ is omitted), the query is called *smoothing*. It returns a smoothed value of $\widehat{X}_i(t)$; the aim of the query is noise reduction.
- If $T < t$, the query is called *prediction*.

Queries which ask for the Most Probable Explanation can be performed for filtering, smoothing and prediction with the **lars** package.

To see how it works, consider the `arth12` data set:

```
> library(GeneNet)
> data(arth800)
> subset = c(60, 141, 260, 333, 365, 424, 441, 512,
+ 521, 578, 789, 799)
> arth12 = arth800.expr[, subset]
> library(lars)
> x = arth12[1:(nrow(arth12) - 2), ]
> y = arth12[-(1:2), "265768_at"]
```

`y` contains the expression levels of gene `265768_at` at all times except for time 0 (recall that there are two measurements at each time). `x` contains the whole data set for all times except for the last one, labelled 24.

```
> lasso.fit = lars(y = y, x = x, type = "lasso")
> lasso.cv = cv.lars(y = y, x = x, mode = "fraction")
> frac = lasso.cv$index[which.min(lasso.cv$cv)]
```

`frac` contains the value of the index that minimises the cross variation. Therefore, this is the value that is used to build the model. Estimation for the expression levels of `265768_at` may be carried out quite simply by:

```
> lasso.est = predict(lasso.fit, type = "fit",
+ newx = x, s = frac,
+ mode = "fraction")$fit
> lasso.est
     0-1      0-2      1-1      1-2      2-1      2-2      4-1
7.099782 6.894064 7.166249 7.157744 7.592092 7.379432 7.990548
     4-2      8-1      8-2     12-1     12-2     13-1     13-2
8.078921 8.353137 8.333108 8.940241 8.780302 8.816387 8.758480
    14-1     14-2     16-1     16-2     20-1     20-2
8.542374 8.417818 7.446577 7.329513 6.717392 6.747178
```

The estimated expression levels at `20-1` and `20-2` are a result of *filtering*, while the others given here are a result of *smoothing*.

The values of 24-1 and 24-2 can be predicted by:

```
> lasso.pred = predict(lasso.fit, type = "fit",
+ newx = arth12[c("24-1", "24-2"), ],
+ s = frac, mode = "fraction")$fit
> lasso.pred
    24-1     24-2
6.822643 6.882054
```

The **penalized** package fits LASSO models which are compatible with **bnlearn**. Therefore, more complex conditional probability queries can be carried out using `cpquery` and `cpdist` if the model is first learned in this way.

```
> library(penalized)
> lambda = optL1(response = y, penalized = x)$lambda
> lasso.t = penalized(response = y, penalized = x,
+ lambda1 = lambda)
# nonzero coefficients: 2
> coef(lasso.t)
(Intercept)   245094_at
 14.0402894  -0.7059011
```

The only parent of gene `256768_at` is `245094_at`, which seems to act as an inhibitor.

This suggests that a model with this explanatory variable might be useful. Such a DBN can be created in the following way:

```
>dbn1 =
+  model2network("[245094_at][265768_at|245094_at]")
>xp.mean = mean(x[, "245094_at"])
>xp.sd = sd(x[, "245094_at"])
>dbn1.fit =
+  custom.fit(dbn1,
+   dist = list("245094_at" = list(coef = xp.mean,
+             sd = xp.sd), "265768_at" = lasso.t))
```

Since the data is continuous, there are two possibilities: either create a Gaussian network, or discretise the variables. The network `dbn1` is Gaussian. The mean `xp.mean` and *standard deviation* `xp.sd` need to be specified.

The regression analysis suggests that high expression levels of `245094_at` at time $t-1$ lead to low expression levels of `265768_at` at time t. The `cpquery` function can be used:

```
>cpquery(dbn1.fit, event = (‘265768_at‘ > 8),
+                  evidence = (‘245094_at‘ > 8))
[1] 0.2454624
>cpquery(dbn1.fit, event = (‘265768_at‘ > 8),
+                  evidence = (‘245094_at‘ < 8))
[1] 0.9829545
```

Note With this package, it is not permitted to condition on events of measure 0. Therefore, *intervals* must be specified *both* for `event` and `evidence`.

The function `cpdist` may be used to generate random observations. To compare the conditional distributions for both pieces of evidence, use:

```
>dist.low = cpdist(dbn1.fit, node = "265768_at",
+             evidence = (‘245094_at‘ < 8))
>dist.high = cpdist(dbn1.fit, node = "265768_at",
+              evidence = (‘245094_at‘ > 8))
```

These may be plotted and the densities compared.

Now suppose that the variables at time t are not independent of those at $t-2$ given $t-1$. It is then a good idea to construct a DBN which depends on lags 1 and 2. To check whether the introduction of $t-2$ to explain t improves the model:

```
> y = arth12[-(1:2), "245094_at"]
> colnames(x)[12] = "245094_at1"
> lambda = optL1(response = y, penalized = x)$lambda
> lasso.s = penalized(response = y, penalized = x,
+                     lambda1 = lambda)
> coef(lasso.s)
 (Intercept)    258736_at    257710_at    255070_at    245319_at
-2.659077706 -0.009220815  0.273648262 -0.444106451 -0.134050990
  245094_at1
 1.589716443
```

The assumption is that the DBN is time homogeneous. These results suggest a network structure which can be created as follows:

```
> dbn2 = empty.graph(c("265768_at", "245094_at",
+                      "258736_at", "257710_at", "255070_at",
+                      "245319_at", "245094_at1"))
> dbn2 = set.arc(dbn2, "245094_at", "265768_at")
> for (node in names(coef(lasso.s))[-c(1, 6)])
+     dbn2 = set.arc(dbn2, node, "245094_at")
> dbn2 = set.arc(dbn2, "245094_at1", "245094_at")
```

The parameters of `dbn2` may be estimated via maximum likelihood. The parameters of `265769_at` and `245094_at` may then be substituted with those from the LASSO models `lasso.t` and `lasso.s`.

7.5 Exercises

1. Consider the `Canada` data set from the **vars** package. Load the data set, make some exploratory analysis and estimate a VAR(1) process for this data set. Estimate the auto-regressive matrix A and the constant matrix B which define the VAR(1) model.

   Compare the results with the LASSO matrix when the L_1 penalty is estimated by cross-validation.

   What are your conclusions?

2. Consider the `arth800` data set from the **GeneNet** package. Load the data set. The time series expression of the 800 genes is included in a data set called `arth800.expr`. Investigate its properties.

   Compute the variances of each of the 800 variables, plot them in decreasing order and create a data set with those variables whose variance is greater than 2.

   Can you fit a VAR process using the **var** package (unlikely)? Suggest alternative approaches (such as LASSO) and apply them. Estimate a DBN with each approach and compare the DBNs. Plot the DBNs using `plot` from **G1DBN**.

Chapter 8

Discriminant Function Analysis

Suppose that we are given a learning set $\mathbf{x}$ of multivariate observations, where $\mathbf{x} \in \mathbb{R}^{n \times p}$. That is, n multivariate observations, with p variables. In addition, there is a variable $p+1$ which is a *class* variable taking values in $\mathcal{C} = \{C_1, \ldots, C_m\}$. That is, each observation comes from one of m pre-defined classes. In this set up, there are p *classification* variables and m groups or classes. Suppose that there are n_j observations from group C_j, for $j = 1, \ldots, m$. For example, for the Egyptian Skull data, there are 5 different periods and 30 skulls from each period. There is one class variable (the period) and four classification variables. The data may be represented by:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_m \end{pmatrix}$$

where m is the number of groups and

$$\mathbf{x}_j = \begin{pmatrix} x_{1j1} & \ldots & x_{1jp} \\ \vdots & & \vdots \\ x_{n_j j1} & \ldots & x_{n_j jp} \end{pmatrix}$$

and $n = \sum_{j=1}^{m} n_j$. These observations are described as *labelled observations*. There are two main goals:

- **Discrimination** Use the information in a learning set of labelled observations to construct a *classifier* (or *classification rule*) that will separate the predefined classes as much as possible.

- **Classification** Given a set of measurements on a new *unlabelled* observation, use the classifier to decide which class the observation belongs to.

There are two basic methods for discriminant analysis; the *maximum likelihood* method and *Fisher's Linear Discriminant Function* method. The maximum likelihood method may be used when the probability distribution of each population is known; the linear discriminant function method is used when the probability distribution is unknown.

8.1 The Maximum Likelihood Discriminant Rule

The *maximum likelihood rule* is used when the probability distribution, or at least the parametric family of probability distributions, is known for each population. Unknown parameters are estimated by the training data and the estimates plugged in. Then a new observation $\underline{x}$ is allocated to group j if $\widehat{L}_j(\underline{x}) = \max_m \widehat{L}_m(\underline{x})$, where $\widehat{L}_k : k = 1, \ldots, m$ is the estimated likelihood function (when the parameter estimates have been plugged in). It is assumed that the situation where there are two groups which maximise the likelihood will not arise. If it does, a classification cannot be determined.

Example 8.1 (Normal Populations, same covariance structure)**.**

Assume that $\underline{X}_j \sim N(\underline{\mu}_j, C)$ for $j = 1, \ldots, m$. That is, from group j, the observations are independent identical multivariate normal, with mean vector $\underline{\mu}_j$ and covariance matrix C. The covariance matrix is assumed to be the *same* for each group. Then

$$L_j(\underline{x}) = \frac{1}{(2\pi)^{p/2}|C|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{x} - \underline{\mu}_j)^t C^{-1}(\underline{x} - \underline{\mu}_j)\right\},$$

For an observation $\underline{x}$, finding the j that maximises $L_\cdot(\underline{x})$ is equivalent to finding the j that maximises

$$\mathcal{L}_j(\underline{x}) := \ln L_j(\underline{x}) = -\frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln|C| - \frac{1}{2}(\underline{x} - \underline{\mu}_j)^t C^{-1}(\underline{x} - \underline{\mu}_j).$$

If the parameters are unknown, they are estimated from the training examples. The expectation vectors $\underline{\mu}_j$ are estimated by the sample average $\overline{\underline{x}}_j$ for group j and the covariance matrix C is estimated by S, the *pooled* covariance matrix from *all* the observations. When classifying a new observation, the problem is then to find the j which *minimises* the *Mahalanobis distance* from the observation $\underline{x}$ to the centre of group j. Recall the definition of the Mahalanobis distance:

$$D_j^2 = (\underline{x} - \overline{\underline{x}}_j)^t S^{-1}(\underline{x} - \overline{\underline{x}}_j).$$

New observations are *classified* as belonging to group j for which D_j is smallest.

8.1.1 The Bayes Discriminant Rule

This is almost the same as likelihood, except that there is a prior probability over classes; if X_{p+1} is the class variable, then

$$\mathbb{P}(X_{p+1} = C_i) = \pi_i.$$

The *posterior probability* for class C_i given $X = (X_1 \ldots, X_p)$ is then

$$\mathbb{P}(X_{p+1} = C_i | X = x) \propto \pi_i L_i(x).$$

The observation is then classified as class $C_i; i = \text{argmax}_j \pi_j L_j(x)$ (these are estimated by plugging in the appropriate parameter estimates).

8.2 The Linear Discriminant Function

Suppose we have two classes, C_1 and C_2. Suppose we have two normal populations, $\underline{X}_1 \sim N(\underline{\mu}_1, C)$ and $\underline{X}_2 \sim N(\underline{\mu}_2, C)$. Let f_1 and f_2 denote the respective densities. Set

$$\mathbb{L}(\underline{x}) := \ln \frac{f_1(\underline{x})}{f_2(\underline{x})}$$

Then

$$\begin{aligned}
\mathbb{L}(\underline{x}) &= -\frac{1}{2}(\underline{x}-\underline{\mu}_1)^t C^{-1}(\underline{x}-\underline{\mu}_1) + \frac{1}{2}(\underline{x}-\underline{\mu}_2)^t C^{-1}(\underline{x}-\underline{\mu}_2) \\
&= (\underline{\mu}_1-\underline{\mu}_2)C^{-1}\underline{x} - \frac{1}{2}\underline{\mu}_1^t C^{-1}\underline{\mu}_1 + \frac{1}{2}\underline{\mu}_2 C^{-1}\underline{\mu}_2 \\
&= (\underline{\mu}_1-\underline{\mu}_2)C^{-1}\underline{x} - \frac{1}{2}(\underline{\mu}_1^t+\underline{\mu}_2^t)C^{-1}(\underline{\mu}_1-\underline{\mu}_2) \\
&= (\underline{\mu}_1-\underline{\mu}_2)^t C^{-1}(\underline{x}-\overline{\underline{\mu}}) = b_0 + \underline{b}^t\underline{x}
\end{aligned}$$

where

$$\overline{\underline{\mu}} = \frac{1}{2}(\underline{\mu}_1+\underline{\mu}_2).$$

This is a linear function, where

$$\begin{cases} \underline{b} = C^{-1}(\underline{\mu}_1-\underline{\mu}_2) \\ b_0 = -\frac{1}{2}\left\{\underline{\mu}_1^t C^{-1}\underline{\mu}_1 - \underline{\mu}_2^t C^{-1}\underline{\mu}_2\right\} \end{cases} \tag{8.1}$$

The function $\mathbb{L}$ is known as the *Linear Discriminant Function* (LDF). It partitions the space $\mathbb{R}^p$ into disjoint classification regions $\mathcal{R}_1$ and $\mathcal{R}_2$. If $\underline{x}$ falls into $\mathcal{R}_1$, then the observation is classified as belonging to C_1. If it falls into $\mathcal{R}_2$, then it is classified as belonging to C_2.

8.3 Misclassification Probability

Let

$$U = \underline{b}^t\underline{X} = C^{-1}(\underline{\mu}_1-\underline{\mu}_2)\underline{X}.$$

This is the part of $\mathbb{L}(\underline{X})$ that depends on $\underline{X}$.
Conditioned on the class, this is a Gaussian random variable. We now compute its mean and variance for each class. Let $\mathcal{C}$ denote the class variable, then

$$\mathbb{E}[U|\mathcal{C}\in C_i] = C^{-1}(\underline{\mu}_1-\underline{\mu}_2)\underline{\mu}_i$$

Define

$$\Delta^2 = (\underline{\mu}_1-\underline{\mu}_2)^t C^{-1}(\underline{\mu}_1-\underline{\mu}_2).$$

then

$$\mathrm{Var}(U|\mathcal{C} \in C_i) = \underline{b}^t C \underline{b} = \Delta^2$$

for $i = 1, 2$. Let M denote the event of misclassification. The *misclassification probabilities* for individuals from the respective groups is therefore:

$$\mathbb{P}(M|\mathcal{C} = C_1) = \mathbb{P}(\underline{X} \in \mathcal{R}_2|\mathcal{C} \in C_1), \qquad \mathbb{P}(M|\mathcal{C} = C_2) = \mathbb{P}(\underline{X} \in \mathcal{R}_1|\mathcal{C} \in C_2)$$

where

$$\mathbb{P}(\underline{X} \in \mathcal{R}_2|\mathcal{C} = C_1) = \mathbb{P}(\mathbb{L}(\underline{X}) < 0|\mathcal{C} = C_1) = \Phi\left(-\frac{\Delta}{2}\right)$$

and

$$\mathbb{P}(\underline{X} \in \mathcal{R}_1|\mathcal{C} = C_2) = \mathbb{P}(\mathbb{L}(\underline{X}) > 0|\mathcal{C} \in C_2) = \Phi\left(-\frac{\Delta}{2}\right).$$

The details of the computations are left as an exercise. □

A graph of $\mathbb{P}(M|\mathcal{C} = C_i)$ against Δ shows a downward sloping curve. It has value $\frac{1}{2}$ when $\Delta = 0$ (the two populations are identical) and tends to 0 as Δ increases. In other words, the greater the distance between the two population means, the less likely one is to misclassify $\underline{x}$.

8.4 Fisher's Discriminant Function

Fisher's idea was to look for appropriate linear combinations of the variables

$$Z = \sum_{k=1}^{p} a_k X_k$$

to maximise the distance between the various groups. Fisher (1936) suggested taking the linear combination that maximises the F ratio in the ANOVA table. Let $n = \sum_{j=1}^{m} n_j$, $\overline{z} = \frac{1}{n}\sum_{k=1}^{m}\sum_{l=1}^{n_k} z_{lk}$, $\overline{z}_k = \frac{1}{n_k}\sum_{l=1}^{n_k} z_{lk}$. The ANOVA is

source	$d.f.$	mean square	f
$SSB = \sum_{j=1}^{m} n_j(\overline{z}_j - \overline{z})^2$	$m-1$	$M_B = \frac{SSB}{m-1}$	$\frac{M_B}{M_E}$
$SSE = \sum_{j=1}^{m}\sum_{k=1}^{n_j}(z_{kj} - \overline{z}_j)^2$	$n-m$	$M_E = \frac{SSE}{n-m}$	
$SST = SSB + SSE = \sum_{j=1}^{m}\sum_{k=1}^{n_j}(z_{kj} - \overline{z})^2$	$n-1$		

Let T, W and B be the matrices for *Total*, *Within* (or error) and *Between* classes sums of squares defined by

$$T_{ab} = \sum_{j=1}^{m}\sum_{k=1}^{n_j}(x_{kja} - \overline{x}_a)(x_{kjb} - \overline{x}_b),$$

$$W_{ab} = \sum_{j=1}^{m} \sum_{k=1}^{n_j} (x_{kja} - \overline{x}_{ka})(x_{kjb} - \overline{x}_{kb}),$$

$$B_{ab} = \sum_{j=1}^{m} n_j (\overline{x}_{.ja} - \overline{x}_{..a})(\overline{x}_{.jb} - \overline{x}_{..b}),$$

where x_{kja} denotes observation k from sample j for variable a, $\overline{x}_a = \frac{1}{mn_j} \sum_{j=1}^{m} \sum_{k=1}^{n_j} x_{kja}$ and $\overline{x}_{ja} = \frac{1}{n_j} \sum_{k=1}^{m} x_{kja}$. As described before, T denotes 'total', B denotes 'between groups' and W as 'within groups', so W with suitable normalisation is an estimate of the error covariance. Note that

$$B = T - W.$$

Then it turns out that (this is one of the tutorial exercises) that Fisher's rule amounts to choosing a vector $\underline{a} \in \mathbb{R}^p$ that maximises the ratio

$$\frac{\underline{a}^t B \underline{a}}{\underline{a}^t W \underline{a}}.$$

Then the discriminant function is $Z = \sum_{j=1}^{p} a_j X_j$.

Definition 8.1. *The linear function Z satisfying $Z(\mathbf{x}) = \sum_{j=1}^{p} a_j x_j$ is called Fisher's* linear discriminant function. *The linear combination $\underline{a}^t \underline{x}$ is also called the* first canonical variate.

Theorem 8.2. *The vector $\underline{a}$ that maximises $\frac{\underline{a}^t B \underline{a}}{\underline{a}^t W \underline{a}}$ is the eigenvector corresponding to the largest eigenvalue of the $p \times p$ matrix $W^{-1}B$.*

Let $\underline{\overline{x}}_j$ denote the mean vector for population (or group) j. Using Fisher's linear discriminant function, the rule is to assign a p- variate observation $\mathbf{x}$ to the class for which $|\mathbf{a}^t(\mathbf{x} - \overline{\mathbf{x}}_j)|$ is lowest.

Consider two populations j and k with mean vectors $\underline{\overline{x}}_j$ and $\underline{\overline{x}}_k$ respectively and assume that the populations have been labelled such that $\underline{a}^t \underline{\overline{x}}_j \geq \underline{a}^t \underline{\overline{x}}_k$. Then, for any $\underline{x} \in \mathbb{R}^p$,

$$|\underline{a}^t(\underline{x} - \underline{x}_j)| < |\underline{a}^t(\underline{x} - \underline{x}_k)| \implies \underline{a}^t \left(\underline{x} - \frac{1}{2}(\underline{\overline{x}}_j + \underline{\overline{x}}_k) \right) > 0.$$

□

This enables the following interpretation of Fisher's linear discriminant rule. The set

$$H_{jk} = \left\{ \underline{x} \in \mathbb{R}^p | \underline{a}^t \left(\underline{x} - \frac{1}{2}(\underline{\overline{x}}_j + \underline{\overline{x}}_k) \right) = 0 \right\}$$

defines a hyperplane perpendicular to the vector $\underline{a}$. This hyperplane divides $\mathbb{R}^p$ into two disjoint half spaces; the mean $\underline{\overline{x}}_j$ lies in one and the mean $\underline{\overline{x}}_k$ lies in the other.

By considering all pairs of populations (j, k) with $1 \leq j \leq m$ and $1 \leq k \leq m$, Fisher's linear discriminant function splits $\mathbb{R}^p$ into m disjoint regions

$$\mathbb{R}^p = \mathcal{R}_1 \cup \ldots \cup \mathcal{R}_m$$

by considering all $p(p-1)/2$ hyperplanes H_{jk}. The region $\mathcal{R}_j$ corresponds to the region where an observation $\underline{x}$ will be classified as belonging to population j. These hyperplanes are all perpendicular to the vector $\underline{a}$.

To find $\mathcal{R}_j$, drop a line from $\overline{\underline{x}}_j$, perpendicular to the vector $\underline{a}$, to the line through the origin containing the point $\underline{a}$. Denote the point of intersection by $\underline{y}_j$. From the $m-1$ hyperplanes $H_{j1}, \ldots, H_{jm}$ (there is no hyperplane H_{jj}), find the two with smallest distance from $\underline{y}_j$, on either side of that point. The region $\mathcal{R}_j$ is the region bounded by these two hyperplanes.

The following discussion shows the likelihood discrimination rule for multinormal observations, from which it follows that if there are two populations, each with the *same* covariance, then the discrimination rule is the same as for Fisher's discrimination rule.

8.5 Quadratic Discrimination

When populations are normal, but the covariance matrices are not equal, the maximum likelihood technique leads to *quadratic* discriminant functions.

Theorem 8.3. *Suppose that $\underline{X}_j \sim N(\underline{\mu}_j, C_j)$ (that is, a p-variate observation from population j has multivariate normal distribution with mean vector $\underline{\mu}_j$ and covariance matrix C_j). Suppose that $\overline{\underline{x}}_j$ and S_j are the maximum likelihood estimates of the mean and covariance matrix for population j. Then the maximum likelihood discrimination rule, where the estimates are used in place of the true parameter values, allocates a new observation $\underline{x}$ to population j if and only if*

$$(\underline{x}-\overline{\underline{x}}_j)^t(S_k^{-1}-S_j^{-1})(\underline{x}-\overline{\underline{x}}_j)+(\overline{\underline{x}}_j-\overline{\underline{x}}_k)^tS_k^{-1}(2\underline{x}-(\overline{\underline{x}}_j+\overline{\underline{x}}_k))+\ln\frac{|S_k|}{|S_j|}>0, \quad k\neq j. \tag{8.2}$$

Proof The log likelihood for population j is

$$l_j(\underline{x}) = -\frac{1}{2}\ln(2\pi)-\frac{1}{2}\ln|S_j|-\frac{1}{2}(\underline{x}-\overline{\underline{x}}_j)S_j^{-1}(\underline{x}-\overline{\underline{x}}_j).$$

The result follows from straightforward arithmetic manipulation. □

Corollary 8.4. *If $m=2$ (two populations) and $C_1=C_2=C$ and this model is used, with $S=\frac{1}{n_1+n_2}W$, then the maximum likelihood method allocates a new observation $\mathbf{x}$ to population 1 if and only if*

$$(\overline{\underline{x}}_1-\overline{\underline{x}}_2)^tW^{-1}(\underline{x}-\frac{1}{2}(\overline{\underline{x}}_1+\overline{\underline{x}}_2))>0.$$

Proof Straightforward exercise. □

8.6 Canonical Discriminant Functions

Fisher's technique may be extended quite easily to obtain more discriminant functions, to sharpen up the clasification. Let $s=\min(p,m-1)$ and let $\lambda_1>\ldots>\lambda_s$ be the first s eigenvalues of $W^{-1}B$ and let $(a_{i1},\ldots,a_{ip})^t$ denote the eigenvector corresponding to eigenvalue λ_i and set

$$Z_i(\underline{x}) = \sum_{k=1}^{p} a_{ik}x_k.$$

Then Z_i is known as the ith canonical discriminant function. It turns out (proof omitted) that the ith eigenvalue is the ratio of the within group sum of squares to the between group sum of squares for

Z_1 is the combination that gives the largest M_B/M_W ratio, subject to the constraint that $\sum a_{1k}^2 = 1$.

Z_2 is the combination that gives the largest M_B/M_W ratio, subject to constraints that $\sum_k a_{2k}^2 = 1$ and $\sum_{jk} a_{1j}S_{jk}a_{2k} = 0$; i.e. Z_2 is statistically uncorrelated with Z_1.

For $i \geq 2$, Z_i is the linear combination that gives the largest M_B/M_W ratio, subject to the constraints that $\sum_{k=1}^{p} a_{jk}^2 = 1$, $j = 1, \ldots, p$ and $\sum_{\alpha,\beta} a_{j\alpha}a_{k\beta}S_{\alpha\beta} = 0$ for all $1 \leq j < k \leq i$.

Where discriminant analysis is useful, the first few functions ought to be sufficient to show the group differences. Hopefully, sufficiently few will be required so that they can be used to represent the group differences graphically.

Important Remark The value $s = \min(p, m-1)$ is the maximum number of canonical discriminant functions *available*; this is the *rank* of B as is easily checked and hence, if $s < p$ all remaining eigenvalues $\lambda_{s+1} = \ldots = \lambda_p = 0$.

Significance Tests The Hotelling T^2 test may be used to test for a significant difference between the mean values for any pair of groups. Other tests, which are variants of this test, may be used to detect overall significant differences between the means for the m groups.

χ^2 test In addition, let $(\lambda_j)_{j=1}^{s}$ denote the eigenvalues of the matrix $W^{-1}B$. Then, approximately,

$$\phi_j^2 := \left(n - 1 - \frac{p+m}{2}\right)\ln(1+\lambda_j) \sim \chi^2_{p+m-2j}.$$

A large value substantiates the claim that there are significant differences of the mean vectors between the groups. Alternatively, $\phi_j^2 + \ldots + \phi_s^2$ may be used, the χ^2 having the d.f. $\sum_{k=j}^{s}(p+m-2k)$.

Warnings

1. The χ^2 test does not seem to be robust if assumptions $\underline{X}_j \sim N(\mu_j, C)$ are relaxed. This contrasts with univariate analysis, where the results seem to be robust when assumptions of normality are relaxed.

2. Even if the data is normal, the statistical values for λ_j may appear in the wrong order, if the variance is large. The test does not take this possibility into account. A large value for an eigenvalue further down on the list that happens by chance will give a wrong impression of the

significance of all the eigenvalues; the test has a greater chance of wrongly indicating significance than the nominal significance level.

Example 8.2 (Egyptian Skulls)**.**

The matrices W, T, $B = T - W$ can be obtained and the matrix $W^{-1}B$ calculated and its eigenvalues computed. These turn out to be $\lambda_1 = 0.437$, $\lambda_2 = 0.035$, $\lambda_3 = 0.015$, $\lambda_4 = 0.002$. The corresponding eigenvectors may be calculated, giving (up to scaling) canonical discriminant functions

$$Z_1 = 0.127X_1 - 0.037X_2 - 0.145X_3 - 0.0083X_4$$
$$Z_2 = 0.039X_1 + 0.210X_2 - 0.068X_3 - 0.077X_4$$
$$Z_3 = 0.093X_1 - 0.025X_2 + 0.015X_3 - 0.295X_4$$
$$Z_4 = 0.149X_1 - 0.000X_2 + 0.133X_3 + 0.067X_4$$

The eigenvalue λ_1 is much larger than the others; most of the sample differences are described by Z_1 alone. Large values correspond to skulls which are tall and narrow with long jaws and short nasal heights.

The *means* and *standard deviations* for the discriminant function Z_1 may be computed for the five samples. They are

	group	mean	standard deviation
I:	Earl predynastic	-0.029	0.097
II:	Late predynastic	-0.043	0.071
III:	12th and 13th dynasties	-0.099	0.075
IV:	Ptolemaic	-0.143	0.080
V:	Roman	-0.167	0.095

This discriminant function shows a clear trend in the mean. It is decreasing over time, indicating *on average* shorter broader skulls, with short jaws, but relatively larger nasal heights. But this is very much an *average* change; the standard deviation is rather large. When the 150 skulls are classified according to the group to which they are closest according to the Mahalanobis distance, rather many are wrongly classified. The following table, known as a *confusion table*, gives the number of objects which the classifier places in each class, from each class. The diagonal entries indicate the number that are correctly classified, the off-diagonals those that are incorrectly classified.

	number allocated to each group					
source group	I	II	III	IV	V	total
I	12	8	4	4	2	30
II	10	8	5	4	3	30
III	4	4	15	2	5	30
IV	3	3	7	5	12	30
V	2	4	4	9	11	30

□

Allowing for Additional Information

Suppose, for example, there are two groups and it is known that many more will fall into group 1 than into group 2. In that case, if an individual is allocated to each group, it makes sense to bias the allocation procedure in favour of group 1. The procedure of allocating an individual to the group with the smallest Mahalanobis distance is then modified, by taking into account prior probabilities of group membership.

Stepwise Discriminant Function Analysis

The standard approach to discriminant function analysis is to decide in advance the number of variables to be used. Alternatively, a *stepwise* approach may be adopted, when there are a very large number of variables, adding in the 'best' variable at each stage, until it is found that adding in extra variables does not lead to better discrimination.

The main problem with stepwise discriminant function analysis is that it introduces bias. Given enough variables, it is likely that some combination of them will produce significant discriminant functions by chance alone.

To check that the results are valid, it might then be a good idea to (for example, with the Egyptian skull data) allocate the 150 skulls to the groups I,II,III,IV,V purely at random and see if the procedure is able to detect a pattern. If it can detect a pattern with the randomised data, then there is clearly a problem.

Jackknife Classification

A particular individual will necessarily affect the statistical average of the 'correct' group for that individual. To check that the classification procedure works, it is therefore probably better to remove that individual from the computations of sample means and sample covariance matrix, and then allocate the individual based on the analysis from which that individual has been removed. When the data set is reasonably large, this does not make much difference in practise.

8.7 LDA using Multiple Regression Techniques

The results on LDA can also be obtained using linear regression techniques. This may prove to be useful when we have a large number of variables and we would like to choose a subset of them for classification purposes. We may then employ LASSO to construct the classifier.

To use regression for LDA, create a variable Y which indicates which class the observations belong to, Then regress the feature variables X on Y.

Consider two classes, n_1 resp. n_2 in each class, items $1, \ldots, n_1$ belong to class 1 and items $n_1 + 1, \ldots, n_1 + n_2$ belong to class 2. let $Y_i = y_1$ for $i = 1, \ldots, n_1$ and y_2 for $i = n_1 + 1, \ldots, n_2$.

Let $X = \begin{pmatrix} X_1 \\ \hline X_2 \end{pmatrix}$ where X_1 and X_2 are respectively the $n_1 \times p$ and $n_2 \times p$ matrices containing the values of $(X_1, \ldots, X_p)$ for the observations for populations 1 and 2 respectively.

When classification is in view, we may use centred variables. Let

$$H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^t$$

be the centring matrix and let

$$X^c = H_n X \qquad Y^c = H_n Y$$

so that the columns of X^c have mean zero and Y^c has mean 0. Therefore

$$T = X^{ct}X^c.$$

Regressing gives the OLS estimator

$$\widehat{\beta} = (X^{ct}X^c)^{-1}X^{ct}Y^c.$$

Set

$$d = \frac{1}{n_1}X_1^t\mathbf{1}_{n_1} - \frac{1}{n_2}X_2^t\mathbf{1}_{n_2},$$

The vector d is a p-vector where the entries are the differences of the sample means of the two populations for each variable.

A straightforward computation gives:

$$B = \frac{n_1 n_2}{n}dd^t$$

Let

$$S_{XX} = X_1^t H_{n_1} X_1 + X_2^t H_{n_2} X_2$$

Here

$$S_{XX;ab} = \sum_{k=1}^{n_1}(x_{k1a} - \overline{x}_{.1a})(x_{k1b} - \overline{x}_{.1b}) + \sum_{k=1}^{n_2}(x_{k2a} - \overline{x}_{.2a})(x_{k2b} - \overline{x}_{.2b});$$

For two classes, the matrix S_{XX} is the matrix W from earlier. Set

$$k = \frac{n_1 n_2}{n}$$

Then

$$X^{ct}X^c = S_{XX} + kdd^t$$

This is the identity $T = B + W$.

$$X^{ct}Y^c = k(y_1 - y_2)d$$

$$Y^{ct}Y^c = k(y_1 - y_2)^2.$$

It follows that

$$\widehat{\beta} = k(y_1 - y_2)(S_{XX} + kdd^t)^{-1}d = k(y_1 - y_2)S_{XX}^{-1}(I_p + kdd^tS_{XX}^{-1})^{-1}d.$$

Recall the matrix result:

$$(A + uv^t)^{-1} = A^{-1} - \frac{(A^{-1}u)(v^tA^{-1})}{1 + v^tA^{-1}u}.$$

Set $A = I_p$, $u = kd$, $v = S_{XX}^{-1}d$, then

$$(I_p + kdd^tS_{XX}^{-1})^{-1} = I_p - \frac{kdd^tS_{XX}^{-1}}{1 + kd^tS_{XX}^{-1}d} = \frac{I_p}{1 + kd^tS_{XX}^{-1}d}$$

from which

$$\widehat{\beta} = \frac{k(y_1 - y_2)}{n - 2 + T^2}\widehat{\Sigma}_{XX}^{-1}d$$

where $\widehat{\Sigma}_{XX} = \frac{1}{n-2}S_{XX}$ and

$$T^2 = kd^t\widehat{\Sigma}_{XX}^{-1}d = \frac{n_1n_2}{n}(\overline{X}_1 - \overline{X}_2)^t\widehat{\Sigma}_{XX}^{-1}(\overline{X}_1 - \overline{X}_2)$$

is the *Hotelling* T^2 *statistic* for testing $\mu_1 = \mu_2$.

Recall the formulae for linear discriminant analysis (8.1) Note that $D^2 = d^t\widehat{\Sigma}_{XX}^{-1}d$ is proportional to the estimate of Δ and

$$\widehat{\beta} \propto \widehat{\Sigma}_{XX}^{-1}(\overline{X}_1 - \overline{X}_2) = \widehat{b}.$$

Variable Selection High dimensional data contains highly correlated variables. The equivalence between LDA and linear regression means that exactly the same techniques may be employed for making a selection; stepwise regression or other techniques that have not yet been encountered, such as LARS (least angle regression) and LASSO.

8.7.1 Logistic Discrimination

Consider two classes. Starting from

$$\log\frac{L_1(x)}{L_2(x)} = b_0 = b^tx$$

where

$$b = \Sigma_{XX}^{-1}(\mu_1 - \mu_2)$$

$$b_0 = -\frac{1}{2}(\mu_1^t\Sigma_{XX}^{-1}\mu_1 + \mu_2^t\Sigma_{XX}^{-1}\mu_2)$$

and using $\mathbb{P}(C_1|x) \propto L_1(x)$, $\mathbb{P}(C_2|x) \propto L_2(x)$ so that $\mathbb{P}(C_2|x) = 1 - \mathbb{P}(C_1|x)$, it follows that

$$\text{logit}(p(C_1|x) = b_0 + b^t x$$

which is of the form of a logistic regression model. The logistic approach to discrimination assumes this linear model, estimates the parameters by logistic regression and assigns the observation to whichever category has the higher estimated likelihood.

8.8 Implementation in R

Implementation in R is straightforward, using (for example) the MASS library. This is illustrated using the `skulls.dat` data set.

```
>www<-"https://www.mimuw.edu.pl/~noble/courses/QPEDataScience/data/
skulls.dat"
> skulls <- read.table(www,header=T)
> View(skulls)
> library("MASS")
> fit <- lda(Year ~ MB + BH + BL + NH, data=skulls, na.action="na.omit",
CV=TRUE)
```

'lda' stands for 'linear discriminant analysis'. The variable 'Year' is to be explained in terms of MB, BH, BL and NH. The 'na.action' refers to how R should treat a value that is not a number. The command 'CV = TRUE' generates the predictions. These are jacknifed (i.e. 'leave one out'). The `.$class` item gives the classes assigned to the skulls.

```
> head(fit$class)
[1] -1850 -4000 -3300 -4000 -1850 -200
Levels: -4000 -3300 -1850 -200 150
```

From the first 11 skulls, it is clear that they are not perfectly classified. To assess the accuracy of prediction, the following may help:

```
> ct <- table(skulls$Year, fit$class)
> ct

       -4000 -3300 -1850 -200 150
  -4000     9    10     5    4   2
  -3300    11     7     5    4   3
  -1850     6     4    12    2   6
  -200      3     3     7    5  12
  150       2     4     4   10  10
> diag(prop.table(ct, 1))
```

```
    -4000     -3300     -1850      -200       150
0.3000000 0.2333333 0.4000000 0.1666667 0.3333333
> sum(diag(prop.table(ct)))
[1] 0.2866667
```

Note that 'leave one out' is a more reliable method and this has substantially affected the accuracy of the prediction. The last item gives the total percentage correct.

Quadratic discriminant analysis may be carried out by substituting the `lda` command for `qda`. Quadratic discriminant analysis in this example does not give as good classification.

```
> fit <- qda(Year ~ MB + BH + BL + NH, data=na.omit(skulls), CV = TRUE,
prior=c(1,1,1,1,1)/5)
> ct <- table(skulls$Year, fit$class)
> ct

        -4000 -3300 -1850 -200 150
  -4000     8    12     4    4   2
  -3300    11     5     4    6   4
  -1850     4     5     6   11   4
  -200      2     3     2   14   9
  150       3     4     5   11   7
 > sum(diag(prop.table(ct)))
[1] 0.2666667
```

If one wants to obtain Fisher's canonical discriminant functions, this is not possible with the 'jacknifed' method; one needs to define a training data set. In this case, it is the whole data set. try

```
> fit2 <- lda(Year~MB + BH + BL + NH, data = skulls,CV=FALSE)
> fit2
Call:
lda(Year ~ MB + BH + BL + NH, data = skulls, CV = FALSE)

Prior probabilities of groups:
-4000 -3300 -1850  -200   150
  0.2   0.2   0.2   0.2   0.2

Group means:
            MB       BH       BL       NH
-4000 131.3667 133.6000 99.16667 50.53333
-3300 132.3667 132.7000 99.06667 50.23333
-1850 134.4667 133.8000 96.03333 50.56667
```

```
-200  135.5000 132.3000 94.53333 51.96667
150   136.1667 130.3333 93.50000 51.36667

Coefficients of linear discriminants:
           LD1         LD2         LD3           LD4
MB  0.12667629  0.03873784  0.09276835  0.1488398644
BH -0.03703209  0.21009773 -0.02456846 -0.0004200843
BL -0.14512512 -0.06811443  0.01474860  0.1325007670
NH  0.08285128 -0.07729281 -0.29458931  0.0668588797

Proportion of trace:
   LD1    LD2    LD3    LD4
0.8823 0.0809 0.0326 0.0042
```

These coefficients give the discriminant functions listed above. Discriminant analysis requires *training* data, which is used to construct the classifier, followed by data to be classified. Once the classifier has been constructed, classification is made using:

```
 > pred <- predict(fit2,skulls[,1:4])
```

The classes to which the objects are assigned are found in `pred$class`.

```
> ct2 <- table(skulls$Year,pred$class)
> ct2

       -4000 -3300 -1850 -200 150
  -4000    12     8     4    4   2
  -3300    10     8     5    4   3
  -1850     4     4    15    2   5
  -200      3     3     7    5  12
  150       2     4     4    9  11
```

8.9 Worked Example: Diabetes

Consider the diabetes data in `diabetes.txt`. Ignore the first 4 columns; they do not contain useful information. The three primary variables are `glucose area` (measure of glucose intolerance), `insulin area` (measure of insulin response to oral glucose), `SSPG` (steady state plasma glucose - a measure of insulin resistance). In addition, `relative weight` and `fasting plasma glucose` were measured. The

three clinical classifications (the target variable) are: overt diabetic (Class 1, 33 individuals), chemical diabetic (Class 2, 36) and normal (Class 3, 76).

Draw a scatterplot matrix of all 5 variables with different colours or symbols representing the three classes of diabetes. Do these pairwise plots suggest multivariate Gaussian distributions for each class with equal covariance matrices? Carry out an LDA and draw a 2D scatterplot of the first two discriminating functions. Using the leave-one-out cross validation procedure, find the confusion table (how many from each class have been classified according to each class) and identify those observations which have been wrongly classified according to the LDA rule. Do the same for QDA.

Hint You'll find the package **GGAlly** useful for the scatterplot matrix. For adding colours, the package seems to like factors, so apply `as.factor()` to the column that represents the different colours (different classes of diabetes).

You'll see clearly how using two discriminant functions helps with the classification in this example.

Solution Firstly, we need to get the data.

```
> www <-
"https://www.mimuw.edu.pl/~noble/courses/QPEDataScience/data/diabetes.
txt"
> diabetes <-read.table(www,header=F)
```

This data doesn't have column headers, so let's add them:

```
>
colnames(diabetes)<-c("one","two","three","four","glucArea","InsArea",
"SSPG","relweight","FPG","type")
```

Later, we'll discover that **ggplot2** and **GGally** need the class variable to be a *factor* and not a *numerical* variable. We can see the type of each variable as follows:

```
> sapply(diabetes,class)
      one       two     three      four  glucArea   InsArea      SSPG
relweight       FPG
"integer" "integer" "integer" "integer" "numeric" "integer" "integer"
"integer" "integer"
     type
"integer"
```

To use **GGally**, we need the *class* variable to be a *factor* and we can do this as follows:

```
> diabetes$type <-as.factor(diabetes$type)
```

Now activate the libraries with the appropriate graph packages;

```
> library(GGally, lib.loc = "/usr/lib/R/site-library")
Loading required package: ggplot2
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
> library(ggplot2, lib.loc = "/usr/lib/R/site-library")
```

Now we would like to visually examine pairs of variables to see if there are good pairs that will help with classification. Scatterplots are useful here. For example, `Glucose area` versus `Insulin area` where the points are coloured according to `type` of diabetes is done as follows:

```
> ggplot(diabetes,aes(x=glucArea,y=InsArea,color=type))+geom_point()
```

Look at the plot. We can see that even with these two variables, we can make a reasonable job of classification; type 3 diabetes can be determined by the `InsArea` variable, although the difference between types 1 and 2 is not so obvious from this pair of variables.

From `GGally`, we can get scatter plots of all pairs of variables, coloured by `type` as follows:

```
> ggpairs(data=diabetes,columns=5:9,aes(colour=type))
```

To do LDA, we need the package **MASS**

```
> library(MASS)
> discrim <-
lda(type~glucArea+InsArea+SSPG+relweight+FPG,data=diabetes,na.action=
"na.omit",CV=FALSE)
> discrim
Call:
lda(type ~ glucArea + InsArea + SSPG + relweight + FPG, data =
diabetes,
    CV = FALSE, na.action = "na.omit")

Prior probabilities of groups:
        1         2         3
0.2275862 0.2482759 0.5241379

Group means:
   glucArea   InsArea      SSPG relweight      FPG
1 0.9839394 217.66667 1043.7576  106.0000 318.8788
2 1.0558333  99.30556  493.9444  288.0000 208.9722
3 0.9372368  91.18421  349.9737  172.6447 114.0000
```

```
Coefficients of linear discriminants:
                    LD1          LD2
glucArea  -1.3624356881 -3.784142444
InsArea    0.0336487883  0.036633317
SSPG      -0.0125763942 -0.007092017
relweight  0.0001022245 -0.006173424
FPG       -0.0042431866  0.001134070

Proportion of trace:
   LD1    LD2
0.8812 0.1188
```

Note that if we do not assign prior probabilities, it simply takes the proportion of each class in the training set.

Here we see that there are *two* discriminant functions and we make a scatterplot as follows:

```
> discrim.lda <-predict(discrim,diabetes[,5:9])
```

This applies the discriminant classifier to the data. This is (of course) unsound, because we've used the same data to construct the classifier as we are using to do the classification!

The loadings for the discriminant functions for this data set are found under $\$x$. We add the two discriminant function loadings to the data frame:

```
> diabetes$lda1 <- discrim.lda$x[,1]
> diabetes$lda2<-discrim.lda$x[,2]
> ggplot(diabetes,aes(x=lda1,y=lda2,color=type))+geom_point()
```

This puts the scores of the discriminant functions as additional columns of the data frame. We can clearly see that, using *both* discriminant functions, we can determine, with good precision, the type of diabetes from these variables.

To see how well the classifier is doing (hoping that using the same data to learn and to test doesn't make too much difference):

```
> pred<-predict(discrim,diabetes[,5:9])
> tab <- table(diabetes$type,pred$class)
> tab

    1  2  3
  1 27  5  1
  2  0 31  5
  3  0  3 73
```

Of course, to get a more accurate idea of the performance, use `CV = TRUE` to ensure that, for each observation, the observation is not used in the construction of the classifier.

8.10 Recursive Partitioning and Tree-Based Methods

Recursive partitioning is the process for constructing a *decision tree*, where for each node we decide to split into two child nodes, or not to split. It is the key to the nonparametric statistical method of *classification and regression trees* (CART) introduced by Breiman, Friedman, Olshen and Stone, 1984.

The algorithm asks a series of hierarchical *Boolean* questions. For a continuous variable X_j, whether or not $X_j \leq \theta_i$ for some threshold value θ_i. For a categorical variable X_k with state space $\{\theta_1, \ldots, \theta_K\}$, whether or not $X_k \in S$, where S is a strict subset of $\{\theta_1, \ldots, \theta_K\}$.

Let Y be the variable to be predicted and $X_1, \ldots, X_r$ the collection of predictors. The output (Y) is a *class* variable; $Y \in \mathcal{C} = \{C_1, \ldots, C_L\}$. If $X_1, \ldots, X_r$ are continous variables, then the input space is $\mathbb{R}^r$ and, following the answers to successive questions, the input space is partitioned into a number of non-overlapping hyper-rectangles. To each hyper-rectangle is associated a class from $\mathcal{C}$, which may be the maximum likelihood estimator of Y based on the answers to the questions.

8.10.1 Classification Trees

A *classification tree* is the result of asking an ordered sequence of questions, where the next question in the sequence depends on the answers to the previous questions of the sequence. The sequence terminates in a prediction of the class.

The starting point is the *root node* and consists of the entire learning set $\mathcal{L}$. A node is a subset of the variables, which can be *terminal* or *non-terminal*. A *non-terminal* node is a node which splits into two child nodes. The binary split is determined by a Boolean condition on the value of a single variable, where the condition is either satisfied ("yes") or not satisfies ("no") by the observed value of the variable. A *terminal* node is a node that does not split.

All observations in $\mathcal{L}$ that have reached a particular (parent) node and satisfy the condition drop down to one of the two child nodes; the remaining observations drop down to the other child node.

In this way, each observation in $\mathcal{L}$ drops down to one of the terminal nodes.

There may be more than one terminal node with the same class label. A single-split tree with only two terminal nodes is called a *stump*. The set of all terminal nodes is a *partition* of the data; each datum will belong to exactly one terminal node.

Example Suppose we have two input variables X_1 and X_2.

- Q(root): $X_2 \leq \theta_1$? yes/no
- Q(yes): $X_1 \leq \theta_2$? yes/no
- Q(no): $X_2 \leq \theta_3$? yes/no
- Q(no)(yes): $X_1 \leq \theta_4$? yes/no

DRAW PICTURE OF THE TREE - IT HAS 5 TERMINAL NODES.

The space is split into 5 regions: Assume $(X_1, X_2) \in [0,1]^2$ and $\theta_i \in [0,1]$ for $i = 1,2,3,4$, then the 5 rectangles are $R_1 = [0,\theta_2] \times [0,\theta_1]$, $R_2 = [\theta_1,1] \times [0,\theta_1]$, $R_3 = [0,\theta_4] \times [\theta_1,\theta_3]$, $R_4 = [\theta_4,1] \times [\theta_1,\theta_2]$, $R_5 = [0,1] \times [\theta_3,1]$.

DRAW A PICTURE OF $[0,1] \times [0,1]$ PARTITIONED INTO RECTANGLES.

It is clear that categorical variables and ordinal variables can also be included; ordinal variables (which take values in a set $1,\ldots,N$ which represent an ordering are included in exactly the same way; the questions are of the form $X \leq \theta$ for some value of θ. For categorical variables, if a variable has M distinct categories represented in the data at the node, labelled $l_1,\ldots,l_M$, the set $\mathcal{S}$ of splits is simply the number of ways of partitioning into two non-empty subsets. There are $2^{M-1}-1$ ways of doing this. For example, if $M = 4$, there are $2^3 - 1 = 7$ possible splits: $(\{l_1\},\{l_2,l_3,l_4\})$, $(\{l_2\},\{l_1,l_3,l_4\})$, $(\{l_3\},\{l_1,l_2,l_4\})$, $(\{l_4\},\{l_1,l_2,l_3\})$, $(\{l_1,l_2\},\{l_3,l_4\})$, $(\{l_1,l_3\},\{l_2,l_4\})$, $(\{l_1,l_4\},\{l_2,l_3\})$.

Cleveland Heart Disease Data The data file `cleveland.data` from the UCI repository

`www.ics.uci.edu/mlearn/databases/heart-disease`

contains data obtained from a heart disease study conducted by the Cleveland Clinic Foundation. The response variable is `diag` (diagnosis of heart disease: `buff` = healthy, `sick` = heart disease). There were 303 patients in the study. There are 13 input variables: `age` (in years), `gender` (male / female), `cp` (chest pain type: `angina` = typical angina, `abnang` = atypical angina, `notang` = non-anginal pain, `asympt` = asymptomatic), `trestbps` (resting blood pressure), etc .....

Choosing the Quesions Each question splits the population of the node in two. When we are learning a classification tree (i.e. a list of questions), we choose the question which gives the greatest Kullback Leibler information.

So, if we have two classes, $\mathcal{C} = \{C_0, C_1\}$, where the class index is the value of Y, p the proportion for which $Y = 1$ and $1-p$ the proportion for which $Y = 0$, let p_{11} be the proportion of those who answer 'yes' and $Y = 1$, p_{10} those who answer 'yes' and $Y = 0$, and p_{01} those who answer 'no' and $Y = 1$, p_{00} those who answer 'no' and $Y = 0$. The Shannon Information Gain is:

$$\sum_{i=0}^{1}\sum_{j=0}^{1} p_{ij} \log \frac{p_{ij}}{p_{i+}p_{+j}}$$

where $p_{i+} = p_{i0} + p_{i1}$ and $p_{+j} = p_{0j} + p_{1j}$. The question which gives the greatest Shannon Information Gain for each node is asked, until no question will give an appreciable increase in SIG. There are other possible criteria for choosing the question; SIG has good properties, which we'll discuss next.

8.11 Shannon Entropy and Information

We now to show how the negative of *Shannon entropy* gives a convincing approach to the amount of information given by the answer to a question if we know the probability distribution and why, when

assessing the amount of information gained, the *Kullback-Leibler* divergence is a useful quantity.

In the following, we consider the *parameter space* $\Theta = \mathcal{C} = \{C_1, \ldots, C_L\}$, the set of possible classes.

Definition 8.5 (Shannon Entropy)**.** *For a distribution with density π over a parameter space Θ, the negative of the* Shannon entropy *is defined as:*

$$\mathcal{E}(\pi) := -\int_\Theta \pi(\theta)\log\pi(\theta)d\theta.$$

We follow Lindley by taking the negative of this quantity, which we call the *information* in the distribution:

$$\mathcal{I}(\pi) = -\mathcal{E}(\pi) = \int_\Theta \pi(\theta)\log\pi(\theta)d\theta.$$

The negative sign in Shannon's definition is due to the fact that he is considering the *opposite* of information; Shannon's entropy is a measure of *disorder*.

Shannon gives reasons why this is a good measure and we follow Lindley's description of Shannon's motivational arguments.

In the discussion here, Θ is finite and π_Θ is a probability mass function. In the absence of any prior information about classes, we can take $\pi(\theta) = \frac{1}{L}$ (uniform distribution) for each $\theta \in \{1, \ldots, L\}$ (the class labels). A priori, $\mathcal{I}(p) = \sum_\Theta \pi(\theta)\log\pi(\theta))$, then the amount of information, say I, can be measured by the amount of additional information required before the value of θ is known.

This information could be provided in two stages:

Stage 1 Let $\Theta_1 \subset \Theta$ be a non-empty, strict subset of Θ where $\sum_{\theta\in\Theta_1}\pi_\Theta(\theta) \neq 0$ or 1. Suppose the experimenter is told whether $\theta \in \Theta_1$ or $\theta \in \Theta\backslash\Theta_1$. The prior distribution over $(\Theta_1, \Theta\backslash\Theta_1)$ is $(\Pi, 1-\Pi)$, where $\Pi = \sum_{\theta\in\Theta_1}\pi_\Theta(\theta)$.

In the second stage, suppose the experimenter is told the value of θ; the information provided is I_2 if $\theta \in \Theta_1$, or I_3 if $\theta \in \Theta\backslash\Theta_1$. The distributions over Θ_1 and $\Theta\backslash\Theta_1$ are $\frac{\pi_\Theta(\theta)}{\Pi}$ and $\frac{\pi_\Theta(\theta)}{1-\Pi}$ respectively.

Shannon requires that the information provided in the first stage and that the average amount in the second stage add up to the total information; that is:

$$I = I_1 + \Pi I_2 + (1-\Pi)I_3.$$

This requirement is the fundamental postulate of Shannon.

Shannon proves that (apart from arbitrary multiplicative constant) $I(\pi) = \sum_{\theta\in\Theta}\pi_\Theta(\theta)\log\pi_\Theta(\theta)$ is the *only* function satisfying this property (together with a mild continuity property).

We can see that I, thus defined, has this property;

$$\begin{aligned}
I_1 &= \Pi\log\Pi + (1-\Pi)\log(1-\Pi)\\
I_2 &= \sum_{\theta\in\Theta_1}\frac{\pi_\Theta(\theta)}{\Pi}\log\frac{\pi_\Theta(\theta)}{\Pi} = \frac{1}{\Pi}\left(\sum_{\theta\in\Theta_1}(\log\pi(\theta) - \log\Pi\right)\\
I_3 &= \sum_{\theta\in\Theta\backslash\Theta_1}\frac{\pi_\Theta(\theta)}{1-\Pi}\log\frac{\pi_\Theta(\theta)}{1-\Pi} = \frac{1}{1-\Pi}\sum_{\Theta\backslash\Theta_1}\pi_\Theta(\theta)\left(\log\pi_\Theta(\theta) - \log(1-\Pi)\right)
\end{aligned}$$

and the identity $I = I_1 + \Pi I_2 + (1-\Pi)I_3$ follows directly. Shannon also shows that this is the *only* function of π_Θ for which this is satisfied for arbitrary π_Θ and $\Theta_1 \subset \Theta$.

After the experiment has been performed, a result x observed and the distribution over Θ updated to $\pi_{\Theta|X}(.|x)$, the information is:

$$\mathcal{I}(\pi_{\Theta|X}(.|x)) = \int_\Theta \pi_{\Theta|X}(\theta|x) \log \pi_{\Theta|X}(\theta|x) d\theta$$

and the information *gain* is:

$$\mathcal{K}(x) = \mathcal{I}(\pi_{\Theta|X}(.|x)) - \mathcal{I}(\pi_\Theta).$$

We assume that, given a true parameter value θ, the outcome x of an experiment is governed by a probability distribution $p_{X|\Theta}(.|\theta)$.

The information difference depends on the observation x. If we are choosing between different experiments (in this case questions to be asked), then clearly we do not know the outcome before we carry out the experiment! We therefore average the information difference over all outcomes for an experiment to get a suitable measure:

$$\begin{aligned}
\int \mathcal{K}(x) p_X(x) dy &= \int p_X(x) \int (\pi_{\Theta|X}(\theta|x) \log \pi_{\Theta|X}(\theta|x) - \pi_\Theta(\theta) \log \pi_\Theta(\theta)) d\theta dx \\
&= \int \int \left(p_X(x) \frac{\pi_\Theta(\theta) p_{X|\theta}(x|\theta)}{p_X(x)} \log \frac{\pi_{\Theta|X}(\theta|x) p_X(x)}{p_X(x)} - \pi_\Theta(\theta) \log \pi_\Theta(\theta) \right) d\theta dx \\
&= \int \int \pi_\Theta(\theta) p_{X|\Theta}(x|\theta) \log \frac{\pi_{\Theta|X}(\theta|x) p_X(x)}{p_Y(x) \pi_\Theta(\theta)} d\theta dx = \mathbb{D}_{KL}(\pi_{\Theta|X} p_X \| \pi_\Theta p_X). \quad (8.3)
\end{aligned}$$

(Here $\pi_\Theta p_{X|\Theta}$ is the joint distribution over parameter space / state space).

This is the Kullback-Leibler divergence between the *joint* distribution $\pi_\Theta p_{X|\Theta}$ over $\Theta \times \mathcal{X}$ and the product distribution $\pi_\Theta p_X$ over $\Theta \times \mathcal{X}$ (if the parameter and observation were independent, the Kullback-Leibler divergence would be zero; the experiment would provide no information).

The Kullback-Leibler divergence has several important properties, which indicate that it is useful for measuring the gain of information from an experiment. Firstly, if f and g are two probability distributions over a state space $\mathcal{X}$, then $\mathbb{D}_{KL}(f\|g) \geq 0$, where the inequality is strict if f and g differ on a set of positive f probability. This follows from Jensen's inequality; if ϕ is a convex function and X a random variable with well defined expected value, then $\mathbb{E}[\phi(X)] \geq \phi(\mathbb{E}[X])$. The function $\phi(x) = -\log x$ is convex. Applying this to Kullback Leibler, this gives:

$$\begin{aligned}
\mathbb{D}_{KL}(f\|g) &= \int f(x) \log \frac{f(x)}{g(x)} dx = -\int f(x) \log \frac{g(x)}{f(x)} dx \\
&\geq -\log \int f(x) \frac{g(x)}{f(x)} dx = -\log \int f(x) dx = -\log 1 = 0.
\end{aligned}$$

Another property is the *additive* property, which was Shannon's basic reason for introducing the entropy functional. Let $\mathcal{E}$ denote an experiment which takes place in two parts, $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$, where $\mathcal{E}_2$ is performed after $\mathcal{E}_1$. Let $\mathcal{K}_{\mathcal{E}_1}$ denote the average information provided by the whole experiment, $\mathcal{K}_{\mathcal{E}_1}$ the information provided by the first part and $\mathcal{K}_{\mathcal{E}_2|\mathcal{E}_1}$ the additional information provided by the second, then

$$\mathcal{K}_{\mathcal{E}} = \mathcal{K}_{\mathcal{E}_1} + \mathcal{K}_{\mathcal{E}_2|\mathcal{E}_1}.$$

This follows quite easily; $\mathcal{K}_{\mathcal{E}_2|\mathcal{E}_1}$ is defined as the average information gain from the second part. Now, using $X = (X_1, X_2)$ to denote answers to two successive questions (or more generally two parts of an experiment) and $x = (x_1, x_2)$ to denote the two outcomes:

$$\begin{aligned}
\mathcal{K}_{\mathcal{E}_2|\mathcal{E}_1} &= \int_{\mathcal{X}_1} p_{X_1}(x_1) \int_{\mathcal{X}_2} \int_{\Theta} p_{X_2|\Theta,X_1}(x_2|\theta,x_1)\pi_{\Theta|X_1}(\theta|x_1) \log \frac{p_{X_2|\Theta,X_1}(x_2|\theta,x_1)\pi_{\Theta|X_1}(\theta|x_1)}{\pi_{\Theta|X_1}(\theta|x_1)p_{X_2|X_1}(x_2|x_1)} d\theta dx_2 dx_1 \\
&= \int_{\mathcal{X}} \int_{\Theta} p_{X|\Theta}(x|\theta)\pi_{\Theta}(\theta) \log \frac{p_{X|\Theta}(x|\theta)\pi_{\Theta}(\theta)p_{X_1}(x_1)}{p_{X_1|\Theta}(x_1|\theta)\pi_{\Theta}(\theta)p_X(x)} d\theta dx.
\end{aligned}$$

The last line comes from taking $p_{X|\Theta} = p_{X_1,X_2|\Theta} = p_{X_2|X_1,\Theta}p_{X_1|\Theta}$ and $\pi_{\Theta|X_1} = \frac{\pi_{\Theta}p_{X_1|\Theta}}{p_{X_1}}$. From this:

$$\mathcal{K}_{\mathcal{E}_1} + \mathcal{K}_{\mathcal{E}_2|\mathcal{E}_1} = \int_{\mathcal{X}} \int_{\Theta} \pi_{\Theta}(\theta)p_{X|\Theta}(x|\theta) \left(\log \frac{\pi_{\Theta}(\theta)p_{X|\Theta}(x|\theta)p_{X_1}(x_1)}{\pi_{\Theta}(\theta)p_{X_1|\theta}(x_1|\theta)p_X(x)} + \log \frac{\pi_{\Theta}(\theta)p_{X_1|\Theta}(x_1|\theta)}{\pi_{\Theta}(\theta)p_{X_1}(x_1)} \right) d\theta dx = \mathcal{K}_{\mathcal{E}}.$$

We now consider the concept of *independent experiments*; two experiments $\mathcal{E}_1$ and $\mathcal{E}_2$, whose outcomes are observations of random variables X_1 and X_2, where both distributions have the same parameter space Θ, are said to be *independent* if $p_{X_1,X_2|\Theta} = p_{X_1|\Theta}p_{X_2|\Theta}$. That is, for any given parameter value θ, X_1 and X_2 are conditionally independent *conditioned on the value of the parameter*. Suppose $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ where $\mathcal{E}_1$ is performed first and $\mathcal{E}_2$ is then performed. Let $\mathcal{E}_2(x_1)$ indicate the experiment $\mathcal{E}_2$, given that $\mathcal{E}_1$ gave outcome x_1; independence means that $\mathcal{E}_2(x_1) = \mathcal{E}_2$, which does not depend on x_1.

If $\mathcal{E}_1$ and $\mathcal{E}_2$ are independent, then $\mathcal{K}_{\mathcal{E}_2|\mathcal{E}_1} \leq \mathcal{K}_{\mathcal{E}_2}$, with equality if and only if $X_1 \perp X_2$ (i.e. they are *marginally* independent; $p_{X_1,X_2} = p_{X_1}p_{X_2}$). This is seen by a simple computation:

$$\begin{aligned}
\mathcal{K}_{\mathcal{E}_2} - \mathcal{K}_{\mathcal{E}_2|\mathcal{E}_1} &= \int_{\Theta} \int_{\mathcal{X}} \pi_{\Theta}(\theta)p_{X|\Theta}(x|\theta) \left(\log \frac{\pi_{\Theta}(\theta)p_{X_2|\Theta}(x_2|\theta)}{\pi_{\Theta}(\theta)p_{X_2}(x_2)} - \log \frac{\pi_{\Theta|X_1}(\theta|x_1)p_{X_2|\Theta,X_1}(x_2|\theta,x_1)}{p_{X_2|X_1}(x_2|x_1)\pi_{\Theta|X_1}(\theta|x_1)} \right) dx d\theta \\
&= \int_{\Theta} \int_{\mathcal{X}} \pi_{\Theta}(\theta)p_{X|\Theta}(x|\theta) \left(\log \frac{\pi_{\Theta}(\theta)p_{X_2|\Theta}(x_2|\theta)}{\pi_{\Theta}(\theta)p_{X_2}(x_2)} - \log \frac{\pi_{\Theta}(\theta)p_{X|\Theta}(x|\theta)p_{X_1}(x_1)}{p_X(x)\pi_{\Theta}(\theta)p_{X_1|\Theta}(x_1|\theta)} \right) dx d\theta \\
&= \int_{\Theta} \int_{\mathcal{X}} \pi_{\Theta}(\theta)p_{X|\Theta}(x|\theta) \log \frac{p_{X_1|\Theta}(x_1|\theta)p_{X_2|\Theta}(x_2|\theta)}{p_{X|\Theta}(x|\theta)} \frac{p_X(x)}{p_{X_1}(x_1)p_{X_2}(x_2)} dx d\theta \\
&= \int_{\mathcal{X}} p_X(x) \log \frac{p_X(y)}{p_{X_1}(x_1)p_{X_2}(x_2)} dx \geq 0.
\end{aligned}$$

The expression in the last line is a Kullback-Leibler divergence, which is 0 if and only if $p_{X_1,X_2} = p_{X_1}p_{X_2}$.

This tells us (among other things) that if Experiment 2 is an independent repeat of Experiment 1, then the repetition is less informative, on average, than the original experiment.

Indeed, if we consider $\mathcal{E}_1, \mathcal{E}_2, \ldots$ a sequence of independent identical experiments and $\mathcal{E}^{(n)} = (\mathcal{E}_1, \ldots, \mathcal{E}_n)$, let $\mathcal{K}_n := \mathcal{K}_{\mathcal{E}^{(n)}}$, then $\mathcal{K}_n$ is a *concave* increasing function of n.

8.11.1 Tree-Growing Procedure

Some basic questions have to be answered:

1. How do we choose the Boolean conditions for splitting at each node? The choice of SIG is motivated by the fact that the sum of information gains from a sequence of questions is the same as the information gain if the multiple question were posed. This is a versatile choice, but not the only one.

2. Choice of criterion for when to split a parent node into two child nodes or when to decide if it is a terminal node.

3. Assigning a class to a node.

Node Impurity Functions Ideally, we would like all elements of a terminal node to belong to the same class, but this is not to be expected. Impurity is a measure of the amount of mixing in terminal nodes. Suppose that Y takes values in $\{1, \ldots, K\}$ (there are K possible classes). For node τ, we define the *node impurity function* as:

$$i(\tau) = \phi(p(1|\tau), \ldots, p(K|\tau))$$

where $p(i|\tau)$ is the proportion of class i observations in node τ. This is an estimate of $\mathbb{P}(X \in \Pi_i|\tau)$, probability that the observation is in class i given that the questions thus far place the observation at node τ.

The Shannon Information Gain is obtained by using

$$i(\tau) = -\sum_{k=1}^{K} p(k|\tau) \log p(k|\tau)$$

There are other possibilities; for example,

$$i_G(\tau) = \sum_{k \neq k'} p(k|\tau)p(k'|\tau) = 1 - \sum_{k} (p(k|\tau))^2.$$

i_G is the so-called Gini index. If classification is binary, then the entropy is

$$i(\tau) = -p \log p - (1-p) \log(1-p)$$

and the Gini index is:

$$i_G(\tau) = 2p(1-p).$$

8.12 Assigning classes to nodes: Estimating the Misclassification Rate

Suppose we have reached a node τ. The misclassification rate is:

$$R(\tau) = 1 - \max_k p(k|\tau).$$

For two classes, this is:

$$R(\tau) = 1 - \max(p, 1-p) = \min(p, 1-p).$$

For a tree, the mis-classification is based on the *terminal* nodes. If $\widetilde{\mathcal{T}}$ denotes the set of terminal nodes, then the true misclassification rate for the tree $\mathcal{T}$ is:

$$R(\mathcal{T}) = \sum_{\tau \in \widetilde{\mathcal{T}}} R(\tau)P(\tau)$$

where $P(\tau)$ is the probability that an observation is placed in (terminal) node τ. We use estimates (based on the learning set where classifications are known) to estimate $P(\tau)$ and $R(\tau)$ for each terminal node τ.

8.13 Pruning the Tree

The tree is grown according to a greedy algorithm; for each node, choose the question which gives the greatest increase in score for that node. This can lead to a tree that is too large. For tree pruning, we use a regularisation approach, starting at the terminal nodes and removing them if they do not represent sufficient gain over the parents. For a node τ, which is terminal in the current tree, we consider:

$$R_\alpha(\tau) = R(\tau) + \alpha$$

where $R(\tau)$ denotes the estimated mis-specification. Then

$$R_\alpha(\mathcal{T}) = R(\mathcal{T}) + \alpha|\widetilde{\mathcal{T}}|.$$

The term $\alpha|\widetilde{\mathcal{T}}|$ is a penalty on the tree size. For each α, we choose the subtree $\mathcal{T}_\alpha$ which minimises $R_\alpha(\mathcal{T})$. This gives $\mathcal{T}(\alpha)$. The tree $\mathcal{T}(\alpha)$ is not necessarily unique.

The chosen value of α determines the tree size. Although $\alpha \in [0, +\infty)$, the number of possible sub-trees of $\mathcal{T}$ is finite. We can consider α_1 the lowest value of α such that $\mathcal{T}(\alpha) \neq \mathcal{T}$ and let $\mathcal{T}_1 = \mathcal{T}(\alpha)$, α_2 the next lowest yielding $\mathcal{T}_2 = \mathcal{T}(\alpha_2)$ and so on. This gives a finite sequence of trees $\mathcal{T} \supset \mathcal{T}_1 \supset \mathcal{T}_2 \subset \ldots$.

Suppose a node τ in an optimal tree $\mathcal{T}$ has two terminal child nodes τ_L and τ_R, then $R(\tau) \geq R(\tau_L) + R(\tau_R)$ (we're using R to denote the estimates used to generate the tree). Now let $\mathcal{T}_1, \mathcal{T}_2, \ldots$ denote the trees obtained by reducing $\mathcal{T}$ as α is increased. Let (τ_1, τ_2) denote the terminal nodes of $\mathcal{T}$ which are not in $\mathcal{T}_1$ (in case of ambiguity, we take a specific sequence of trees) and let $\tau \in \mathcal{T}_1$ denote the terminal node in $\mathcal{T}_1$ which is a non-terminal node in $\mathcal{T}$. For a node τ in a tree $\mathcal{T}$, we denote by $\mathcal{T}_\tau$ the subtree with root τ, going down to the terminal nodes of $\mathcal{T}$.

As long as $R_\alpha(\tau) \geq R_\alpha(\mathcal{T}_\tau)$, the subtree $\mathcal{T}_\tau$ has lower cost than terminating the tree at τ and hence it is retained.

Therefore, when

$$\alpha < \frac{R(\tau) - R(\mathcal{T}_\tau)}{|\widetilde{T}_\tau| - 1}$$

we retain $\mathcal{T}_\tau$. We can set

$$g_1(\tau) = \frac{R(\tau) - R(\mathcal{T}_{1,\tau})}{|\widetilde{T}_{1,r}| - 1} \qquad \tau \notin \mathcal{T}(\alpha_1)$$

where $\mathcal{T}_{1,\tau} = \mathcal{T}_\tau$ and $g_1(\tau)$ gives the critical value for α; when $g_1(\tau) \geq \alpha_1$ for each τ, we do not prune the terminal nodes.

The *weakest link* node $\widetilde{\tau}_1$ is the node in $\mathcal{T}_1$ that satisfies

$$g(\widetilde{\tau}_1) = \min_{\tau \in \mathcal{T}_1} g(\tau).$$

As α increases, $\widetilde{\tau}_1$ is the first node for which $R_\alpha(\tau) = R_\alpha(\mathcal{T}_\tau)$, so $\alpha_2 = g_1(\widetilde{\tau}_1)$. Recursively,

$$g_3(\tau) = \frac{R(\tau) - R(\mathcal{T}_{2,\tau}}{|\widetilde{\mathcal{T}}_{2,\tau}| - 1} \qquad \tau \in \mathcal{T}(\alpha_2), \quad \tau \notin \widetilde{\mathcal{T}}(\alpha_2)$$

and so on.

8.13.1 Choosing the best pruned subtree

Choosing the subtree requires good estimates of the misclassification rate. There are two approaches: for large data sets, using an independent test set is straightforward and computationally efficient. For small data sets, cross validation is recommended. Randomly assign the data into two sets of equal size, the learning set and the test set. Construct the tree using the learning set; estimate the misclassification rate using the test set.

At each stage, dropping down a level, let the chance of misclassification be p^*. We can consider each observation dropped down as a Bernoulli trial, from which we can compute the estimate of misclassification, together with a standard error.

Cross Validation Divide the data into V sets of approximately equal size, call them $D_1, \ldots, D_V$. Create V learning sets $\mathcal{L}_v = D \backslash D_v$. Use $\mathcal{L}_v$ to learn the classification tree $\mathcal{T}^v$. Fix the value of the complexity parameter α and let $\mathcal{T}^v(\alpha)$ be the best pruned subtree of $\mathcal{T}^v$, $v = 1, \ldots, V$. Drop each

observation of the vth test set down the tree $\mathcal{T}^v(\alpha)$ and let n_{ij}^v denote the number of observations class j that are classified as being of class i from test set v. Then $n_{ij}(\alpha) = \sum_{v=1}^{V} n_{ij}^v(\alpha)$. Set

$$R^{CV/V}(\mathcal{T}(\alpha)) = \frac{1}{n}\sum_{i=1}^{K}\sum_{j=1;j\neq i}^{K} n_{ij}(\alpha)$$

Examples

The R commands for the examples are found in the script `23CART.R` on the course page.

Example: Iris Data

For the Iris data, construct a decision tree to predict the species of iris based on petal and sepal length and width. You'll find the code in the accompanying R-script. The package **rpart** is useful.

Find the different classification rules that the tree produces.

Predict the class of a new observation with sepal length, sepal width, petal length, petal width equal to $(6.5, 3.0, 5.2, 2.0)$.

Example 2: Diabetes Data

We'll use the `PimaIndiansDiabetes2` data set in the **mlbench** package for predicting the probability of being diabetes positive based on multiple clinical variables.

Firstly, randomly split the data into a training set (80% for building a predictive model) and test set (20% for evaluating the model). Make sure to set seed so that the results can be reproduced.

Now create a fully grown tree showing all predictor variables in the data set.

Now use the test set to make predictions and evaluate accuracy of the model.

Now prune the tree. Check whether the pruning has made the model substantially worse for prediction and accuracy.

Example 3: Boston Housing

Refer to the R script which accompanies the tutorial.

We first load the libraries which contain good scripts for constructing trees.

The data is found in the MASS package. The variable of interest is `medv` (median value of owner-occupied homes in $1000's.

Carry out the diagnostics suggested in the script to get an idea of correlations between the variables and which explanatory variables may be useful.

Now randomly split the data into training and testing (described in the script).

Run a regression. How well does the fitted model predict new data?

Regression Tree (CART method) The `rpart` package has good routines for this. The data are recursively split into terminal nodes or leaves of the tree. To obtain a prediction for a new sample, we would follow the if-then statements defined by the tree using values of the new sample's predictors until reaching a terminal node. The model formula in the terminal node would then be used to generate the prediction. In simple (traditional) trees, the model is a simple numeric value (yes/no, or a given numeric value). In other cases, the terminal node may be defined by a more complex function of the predictors (terminal nodes have models within them).

Basic implementation is done by Growing, Examining, Pruning.

Grow a Tree To grow a traditional tree, we can use the `rpart()` function in the rpart package.

tree.fit <- rpart(formula, data=, method=,control=)

where +formula is in the format outcome predictor1+predictor2+predictor3+etc. +data= specifies the data frame +method= "class" for a classification tree; "anova" for a regression tree +control= optional parameters for controlling tree growth. For example, control=rpart.control(minsplit=30,cp=0.001) requires that the minimum number of observations in a node be 30 before attempting a split and that a split must decrease the overall lack of fit by a factor of 0.001 (cost complexity factor) before being attempted.

Examine the Tree A collection of functions helps us evaluate and examine the model.

+printcp(tree.fit) displays table of fits across cp (complexity parameter) values +rsq.rpart(tree.fit) plots approximate R-squared and relative error for different splits (2 plots). Labels are only appropriate for the "anova" method. +plotcp(tree.fit) plots the cross-validation results across cp values +print(tree.fit) print results +summary(tree.fit) detailed results including surrogate splits +plot(tree.fit) plot decision tree +text(tree.fit) label the decision tree plot +post(tree.fit, file=) create postscript plot of decision tree (there may be better ways to get good looking tree plots)

First we look at what the error looks like across the range of complexity parameters (depth of tree). The command is:

```
printcp(rtree.fit) # display the results
```

(as used in the script).

A detailed summary of the tree is obtained by

```
summary(rtree.fit)
```

which gives a lot of information. We can also look at the predictors used in the tree and their relative importance in the prediction. We see specifically that `rm` (average number of rooms per dwelling) and `lstat` (lower status of the population, percent) are driving much of the prediction.

This particular tree methodology can also handle missing data. When building the tree, missing data are ignored. For each split, a variety of alternatives (called surrogate splits) are evaluated. A surrogate split is one whose results are similar to the original split actually used in the tree. If a surrogate split approximates the original split well, it can be used when the predictor data associated with the original split are not available. In practice, several surrogate splits may be saved for any particular split in the tree.

Plotting the tree may be done as follows:

```
plot(rtree.fit, uniform=TRUE,
    main="Regression Tree for Median Home Value")
```

Prune the Tree Prune back the tree to avoid overfitting the data. Hastie et al. (2008) suggest selecting the tree size associated with the numerically smallest error. That is, the size of the tree is selected by examining the error using cross-validation, specifically the minimum of the xerror column (cross-validation error) printed by printcp( ).

Pruning is easily done using the function prune(fit, cp=) by examining the cross-validated error results from printcp(), selecting the complexity parameter associated with minimum error, and placing it into the prune( ) function. Alternatively, this can be automated using

```
tree.fit$cptable[which.min(tree.fit\$cptable[,''xerror'']),''CP''].
```

In this case the pruned tree is not that much smaller than the original tree.

There are, of course other approaches for pruning. Breiman et al. (1984) suggest using the cross-validation approach and applying a one-standard-error rule on the optimization criteria for identifying the simplest tree. That is, find the smallest tree that is within one standard error of the tree with smallest absolute error, which is the leftmost cp value for which the mean lies below the horizontal line placed 1 SE above the minmum of the curve by the minline in the plotcp() function.

Test of Prediction Finally, for comparison with the regression model, we examine the R2 of the original and pruned trees.

We see here the tradeoff between "overfit" to training data and potential generalisability to new data. More formal evlauations would be done using cross-validation. But the smaller pruned tree is still doing pretty well (almost as well as the multiple regression).

Example 4: Boston Housing: Regression Tree

Randomly split the data into training set (80% for building a predictive model) and test set (20% for evaluating the model). Make sure to set seed for reproducibility.

Create the regression tree. Here, the best `cp` value is the one that minimises the prediction error RMSE (root mean squared error).

The prediction error is measured by the RMSE, which corresponds to the average difference between the observed known values of the outcome and the predicted value by the model. RMSE is computed as RMSE = mean((observeds – predicteds)^2) `% > % sqrt()`. The lower the RMSE, the better the model.

Plot the final tree model.

Example 5: Conditionnal inference tree

The conditional inference tree (ctree) uses significance test methods to select and split recursively the most related predictor variables to the outcome. This can limit overfitting compared to the classical rpart algorithm.

At each splitting step, the algorithm stops if there is no dependence between predictor variables and the outcome variable. Otherwise the variable that is the most associated to the outcome is selected for splitting.

The conditional tree can be easily computed using the caret workflow, which will invoke the function ctree() available in the party package.

Use the data `PimaIndiansDiabetes2`. First split the data into training (80%) and test set (20%)

Build conditional trees using the tuning parameters maxdepth and mincriterion for controlling the tree size. caret package selects automatically the optimal tuning values for your data, but here we'll specify maxdepth and mincriterion.

Now make predictions using the test data.

Chapter 9

Choice experiments

9.1 Designing Experiments and Modelling Data

In this section, we describe the *random utility model* and its variants for describing data from DCEs (Discrete Choice Experiments).

9.1.1 How are choice sets constructed and data analysed

The distinctive feature of choice experiments, compared to typical experiments in economics, namely that we manipulate several dimensions at the same time, also has important consequences for statistical inference. Our objective is to identify the impact of different attributes, how the fact that an option is characterised by a specific level of a specific attribute affects the probability that it is selected. Because different choice sets differ in terms of several attributes, we cannot look at the simple summary statistics, which would be informative if there were only one change. Consider the simplest possible scenario of binary choices between a policy and the status quo. If Policy A, characterised by some combination of levels of various attributes, is preferred over the status quo 40% of the time, and Policy A', which differs from Policy A on only one attribute, is chosen over the status quo 70% of the time, we could conclude that the change from A to A' made the policy more attractive. Because there is only one difference between the two, it is this attribute that has made the difference. Such direct inference is generally not possible in choice experiments, because several attributes are manipulated simultaneously (and, typically, their levels change in more than one option). To estimate the effect of each attribute, parametric assumptions must thus be made.

Below, we describe the *mixed logit model*, which is the workhorse for analysing DCE data. This approach requires the researcher to be a bit more econometrics-savvy. Various useful software packages make the analysis manageable. We recommend the `mlogit` package for R, which is versatile and user-friendly for the random parameter model and which has proved useful for analysing discrete choice experiment data. Naturally, experimenters often team with an expert when modelling choice-experimental data.

It is fair to say that many experimental economists have a natural dislike of parametric methods. One reason is that they are aware of artefacts of arbitrary modelling choices aimed at obtaining attrac-

tive publishable results. The practice of pre-registration of the methods of analysis and presentation of several specifications may, to some extent, alleviate these concerns. One seemingly positive aspect of the parametric approach is that the researcher explicitly models noise and this consideration informs design.

9.2 Utility Models

We now describe the *random utility model* and some of its variants. In particular, the *random parameters model* has shown itself to be a powerful tool in experimental economics. We begin by presenting the standard (deterministic) utility model, building up to the random utility model and then dealing with the random parameter model and other variations.

Data for utility models concern some individuals who make one choice, or a sequence of choices, each choice being from a set of mutually exclusive and exhaustive *alternatives*; exactly one of these alternatives is chosen. These choices are influenced by known covariates, where the dependence can be either

- *both* on the alternative *and* the choice situation, or
- *only* on the alternative or
- *only* on the choice situation.

Consider a family choosing a destination for their vaccation.

- Examples of *choice situation* specific variables would be: the length of vaccation and the season.
- Examples of *individual* specific variables would be: family income and family size.
- Examples of *alternative* specific variables would be: distance to destination, cost of vaccation.

The unit of observation is therefore the choice situation; it is also the individual if only one choice situation per individual is observed. The structure of such data can therefore be characterised by three indices: the *alternative*, the *choice situation*, and the *individual*. If we use a two-parameter notation, we include the individual-specific variables in the choice situation.

9.2.1 Non-Random Utility Model

For the standard (non-random) utility model, one has to consider three sets of covariates at most:

- Covariates, denoted x_{ij} specific to the choice situation / alternative combination (i, j), with generic coefficients β and covariates t_j specific to the alternative j with a generic coefficient ν.
- Choice situation specific covariates z_i with alternative specific coefficients γ_j.
- Alternative and choice situation specific covariates w_{ij} with alternative specific coefficients δ_j.

These covariates enter the (non-random) utility as follows: for choice situation i, alternative j it is written as:

$$V_{ij} = \alpha_j + \beta x_{ij} + \nu t_j + \gamma_j z_i + \delta_j w_{ij}. \tag{9.1}$$

Since *comparisons* are in view, the absolute value of the utility is irrelevant; only utility differences are useful for modelling the choice of alternative. For two alternatives j and k,

$$V_{ij} - V_{ik} = (\alpha_j - \alpha_k) + \beta(x_{ij} - x_{ik}) + (\gamma_j - \gamma_k)z_i + (\delta_j w_{ij} - \delta_k w_{ik}) + \nu(t_j - t_k).$$

Clearly, the only relevant coefficients of choice situation specific covariates are alternative specific, otherwise they would disappear when differences are taken. Since only *differences* are of interest, we can choose one parameter value as base-line (say 1) and set $\gamma_1 = 0$.

9.2.2 Random Utility Models

For the *random* utility model, a fourth consideration needs to be included, the random component, which we denote by ϵ. In a random utility model, the utility for subject n, choice situation i, denoted U_{nij} may be written as

$$U_{nij} = V_{nij} + \epsilon_{nij}$$

where V_{nij} is a function of observable covariates and unknown parameters, which are to be estimated, taking the form of (9.1). The quantity ϵ_{nij} is a random deviation which is a function of all the *unobserved* or *latent* variables that determine the utility together with the observed covariates.

Alternative j, in choice situation j is optimal for individual n if $U_{nil} < U_{nij}$ for all $l \neq j$, which means that $\epsilon_{nil} < (V_{nij} - V_{nil}) + \epsilon_{nij} \qquad \forall l \neq j$. Suppressing the notation (by omitting the n, i denoting individual n in choice situation i), the probability of choosing alternative j is therefore $\mathbb{P}\left(\bigcup_{l\neq j}\{\epsilon_l < V_j - V_l + \epsilon_j\}\right)$. Let us denote:

$$P_j(z) = \mathbb{P}\left(\bigcup_{l\neq j}\{\epsilon_l < V_j - V_l + z\}\middle|\, \epsilon_j = z\right)$$

namely, $P_j(z)$ is the conditional probability of choosing j given that $\epsilon_j = z$. Let F_{-j} denote the joint cumulative density of all the ϵ's except ϵ_j. Then, if there are J alternatives labelled $1, \ldots, J$,

$$P_j(z) = F_{-j}(V_j - V_1 + z, \ldots, V_j - V_J + z)$$

(where the $V_j - V_j + z$ term is omitted). Let us denote the marginal density of ϵ_j by f_j. The *unconditional* probability of choosing alternative j is therefore:

$$\mathbb{P}_j = \int F_{-j}(V_j - V_1 + z, \ldots, V_j - V_J + z) f_j(z) dz.$$

9.2.3 Distribution of the Error Terms

The *multinomial logit model* was developed by McFadden. The errors for each different individual, choice situation, and alternative combination are taken to be *independent*, so that (suppressing the notation for individual and choice situation and assuming J alternatives):

$$P_j(z) = \prod_{l \neq j} F_l(V_j - V_l + z) \qquad \mathbb{P}_l = \int \prod_{l \neq j} F_l(V_j - V_l + z) f_j(z) dz.$$

Several distributions have been considered for the error. For models considered here, the errors follow the cumulative distribution function of a Gumbel distribution, which turns out to be computationally convenient and flexible for modelling. The mean of this distribution is not zero, but this does not matter; the errors are i.i.d. and it is *differences* that are in view.

The c.d.f. of a Gumbel is:

$$F(z) = \mathbb{P}(X \leq z) = \exp\left\{-\exp\left\{-\frac{z-\mu}{\theta}\right\}\right\}$$

and the density is:

$$f(z) = \frac{1}{\theta} \exp\{-\frac{x-\mu}{\theta}\} \exp\left\{-\exp\left\{-\frac{x-\mu}{\theta}\right\}\right\}$$

where μ is the *location* parameter and θ the *scale* parameter. The expectation is $\mathbb{E}[X] = \mu + \theta\gamma$, where γ is the *Euler-Mascheroni* constant and $\mathrm{Var}(X) = \frac{\pi^2\theta^2}{6}$.

Because the errors are not mean zero, we may consider the V_{ij}'s (from Equation (9.1)) to have an *intercept* term c_i, so that

$$V_{ij} = c_i + \alpha_j + \beta x_{ij} + \nu t_j + \gamma_j z_i + \delta_j w_{ij}.$$

which (of course) disappears when the difference between two alternatives $V_{ij} - V_{ik}$ is considered. When the model contains such a parameter, the mean of ϵ_{ij} is not identified and, without the loss of generality, we can take $\mu_{ij} = 0$ for all $j = 1, \ldots, J_i$ (where choice situation i has J_i alternatives). A natural choice for normalisation is to impose that one of the values of θ_{ij} for $j \in \{1, \ldots, J_i\}$ is equal to 1. With the hypothesis that the errors are identically distributed, we therefore take $\theta_{ij} = 1$ for all $j = 1, \ldots, J_i$ and all choice situations i. With these choices (and suppressing the individual / choice situation indices):

$$P_j(z) = \prod_{l \neq j} \exp\{-\exp\{-(V_j - V_l + z)\}\} \qquad \mathbb{P}_j = \int_{-\infty}^{\infty} \prod_{l \neq j} \exp\{-\exp\{-(V_j - V_l + z)\}\} e^{-z} e^{-e^{-z}} dz.$$

From this, the simple and elegant closed form may be computed which corresponds to the logit transform of the deterministic part of the utility:

$$\mathbb{P}_j = \frac{e^{V_j}}{\sum_{l=1}^{J} e^{V_l}}.$$

Clearly, the ratio of two probabilities $\mathbb{P}_l$ and $\mathbb{P}_m$ is given by:

$$\frac{\mathbb{P}_l}{\mathbb{P}_m} = \exp\{V_l - V_m\}$$

and depends only on the properties of the two alternatives. This is known as the IIA property; independence of irrelevant alternatives.

Marginal Effects Now let us return to the two-parameter notation, (i, j) denotes individual i (presented with a choice situation) and alternative j within the choice situation. The *marginal effects* are the derivatives of the probabilities with respect to the covariates. These can be *choice situation specific* (z_i) or *alternative specific* (x_{ij}). Straightforward computation gives:

$$\begin{cases} \frac{\partial}{\partial x_{ij}}\mathbb{P}_{ij} = \beta\mathbb{P}_{ij}(1-\mathbb{P}_{ij}) \\ \frac{\partial}{\partial x_{ik}}\mathbb{P}_{ij} = -\beta\mathbb{P}_{ij}\mathbb{P}_{ik} \qquad k \neq j \\ \frac{\partial}{\partial z_i}\mathbb{P}_{ij} = (\gamma_j - \sum_{l=1}^{J_i}\gamma_l\mathbb{P}_{il})\mathbb{P}_{ij} \end{cases}$$

These can be written as:

$$\begin{cases} \frac{\partial}{\partial x_{ij}}\log\mathbb{P}_{ij} = \beta(1-\mathbb{P}_{ij}) \\ \frac{\partial}{\partial x_{ik}}\log\mathbb{P}_{ij} = -\beta\mathbb{P}_{ik} \qquad k \neq j \\ \frac{\partial}{\partial z_i}\log\mathbb{P}_{ij} = (\gamma_j - \sum_{l=1}^{J_i}\gamma_l\mathbb{P}_{il}). \end{cases}$$

9.3 Logit models: relaxing the i.i.d. hypothesis

Thus far, we assumed that the error terms are i.i.d. (identically and independently distributed), which implies that they are uncorrelated and homoscedastic. Extensions of the basic multinomial logit model have been proposed the idea behind which is to relax one of these two hypothesis while maintaining the hypothesis of a Gumbel distribution.

9.3.1 Heteroscedastic logit model

The heteroskedastic logit model was proposed by Bhat (1995). The probability that $U_l > U_j$, conditioned on ϵ_l is:

$$P_l(z) := \mathbb{P}\left(\epsilon_j < V_l - V_j + z | \epsilon_l = z\right) = \exp\left\{-\exp\left\{\frac{V_l - V_j + z}{\theta_j}\right\}\right\},$$

from which the following conditional and unconditional probabilities follow:

$$\begin{cases} P_l(z) = \prod_{j\neq l}\exp\left\{-\exp\left\{\frac{V_l - V_j + z}{\theta_j}\right\}\right\} \\ \mathbb{P}_l = \int_{-\infty}^{\infty}\prod_{j\neq l}\exp\left\{-\exp\left\{\frac{V_l - V_j + t}{\theta_j}\right\}\right\}\frac{1}{\theta_l}\exp\left\{-\frac{t}{\theta_l}\right\}\exp\left\{-\exp\left\{-\frac{t}{\theta_l}\right\}\right\}dt \end{cases}$$

There is no closed form for this integral, but it is one-dimensional and can be computed efficiently by the Gauss-Laguerre quadrature method (the method used by the `mlogit` package for R).

9.3.2 The nested logit model

The nested logit model was first proposed by McFadden in 1978. It is a generalisation of the multinomial logit model that is based on the idea that some alternatives may be joined in several groups (called nests). The error terms may then present some correlation in the same nest, whereas error terms of different nests are still uncorrelated.

Denoting the nests by $m = 1, \ldots, M$ and B_m the set of alternatives belonging to nest m, the cumulative distribution of the errors is:

$$\exp\left\{-\sum_{m=1}^{M}\left(\sum_{j\in B_m} e^{-\epsilon_j/\lambda_m}\right)^{\lambda_m}\right\}.$$

The marginal distributions of the ϵ's are still univariate extreme value (Gumbel), but there is now some correlation within nests. $1-\lambda_m$ is a measure of the correlation, i.e. $\lambda_m = 1$ implies no correlation. In the special case where $\lambda_m = 1$ for all m, the errors are i.i.d. Gumbel errors and the nested logit model reduces to the multinomial logit model. It can then be shown that the probability of choosing alternative j that belongs to nest l is:

$$\mathbb{P}_j = \frac{e^{V_j/\lambda_l}\left(\sum_{k\in B_l} e^{V_k/\lambda_l}\right)^{\lambda_l-1}}{\sum_{m=1}^{M}\left(\sum_{k\in B_m} e^{V_k/\lambda_m}\right)^{\lambda_m}}$$

This model is a random utility model if $\lambda_j \in (0,1)$ for each j.

Let us now write the deterministic part of the utility of alternative j as the sum of two terms: the first one (Z_j) being specific to the *alternative* and the second one (W_l) to the nest it belongs to:

$$V_j = Z_j + W_l.$$

We can then rewrite the probabilities as:

$$\mathbb{P}_j = \frac{\exp\{(Z_j+W_l)/\lambda_l\}}{\sum_{k\in B_l}\exp\{(Z_k+W_l)/\lambda_l\}} \times \frac{\left(\sum_{k\in B_l}\exp\{(Z_k+W_l)/\lambda_l\}\right)^{\lambda_l}}{\sum_{m=1}^{M}\left(\sum_{k\in B_m}\exp\{(Z_k+W_m)\lambda_m\}\right)^{\lambda_m}}$$

Let $I_l := \log\sum_{k\in B}\exp\{Z_k/\lambda_l\}$, where log (as throughout) denotes natural logarithm. This is often called the *log-sum*, the *inclusive* value or the *inclusive utility*. The probability of choosing alternative j may then be written as:

$$\mathbb{P}_j = \frac{\exp\{Z_j/\lambda_l\}}{\sum_{k\in B_l}\exp\{Z_k/\lambda_l\}} \times \frac{\exp\{W_l+\lambda_l I_l\}}{\sum_{m=1}^{M}\exp\{W_m+\lambda_m I_m\}} = \mathbb{P}_{j|l}\times\mathbb{P}_l.$$

The first term $\mathbb{P}_{j|l}$ is the conditional probability of choosing alternative j if nest l is chosen, which is often referred to as the lower model. The second term $\mathbb{P}_l$ is the marginal probability of choosing nest l and is referred as the upper model. $W_l+\lambda_l I_l$ can be interpreted as the expected utility of choosing the best alternative in l, W_l being the expected utility of choosing an alternative in this nest (whatever this alternative is) and $\lambda_l I_l$ the expected *extra* utility gained by being able to choose the best alternative in

the nest. The inclusive values link the two models. It is then straightforward to show that IIA applies *within* nests, but not for two alternatives in *different* nests.

Maximising *directly* the likelihood function of the nested model leads to an efficient estimator; other methods (for example estimating the two components separately) are less efficient.

9.3.3 The random parameters (or mixed) logit model

As we shall see in the case studies and examples, the random utility model is often not sufficiently flexible for the analysis of discrete choice data; the additional flexibility required is given by the *Random Parameter Logistic Model*, introduced by Train (abbreviated RPL).

Derivation of the model A mixed logit model or random parameters logit model is a logit model for which the parameters are assumed to vary from one individual to another; the parameter choices $\beta_1, \ldots, \beta_n$ for n individuals are a random sample of size n from a distribution with density $f_\theta(\beta)$. Here $\{f_\theta : \theta \in \Theta\}$ is a suitable parametric family. The value of θ is chosen by the user, to satisfy standard model fitting criteria. This model therefore, to some extent, can take the heterogeneity of the population into account.

The probabilities For the standard logit model, the probability that individual i chooses alternative l is:

$$\mathbb{P}_{il} = \frac{\exp\{\beta' x_{il}\}}{\sum_j \exp\{\beta' x_{ij}\}}.$$

where β is the parameter vector (assumed to be the same for each individual). Suppose now that these coefficients are individual-specific. The probabilities are then:

$$\mathbb{P}_{il} = \frac{\exp\{\beta_i' x_{il}\}}{\sum_j \exp\{\beta_i' x_{ij}\}}.$$

One idea could be to estimate the parameters for every individual. These parameters can only be identified and estimated with any degree of accuracy if a large number of choice situations per individual is available, which does not occur very often in practice.

The random parameter model takes the β_i's to be random draws from a distribution $f_\theta(\beta)$ where $\{f_\theta : \theta \in \Theta\}$ is a suitable parametric family. The probability that individual i will choose alternative l, for a given value of β_i is:

$$\mathbb{P}_{il|\beta_i} = \frac{\exp\{\beta_i' x_{il}\}}{\sum_j \exp\{\beta_i' x_{ij}\}}. \tag{9.2}$$

To get the *unconditional* probability, we have to integrate out this conditional probability, using the density function of β. Suppose (for example) that $V_{il} = \alpha + \beta_i x_{il}$ i.e., there is only one individual-specific coefficient and that the density of β_i is $f_\theta(\beta)$, θ being the vector of the parameters of the distribution of β. The unconditional probability is then:

$$\mathbb{P}_{il} = \mathbb{E}_\theta\left[\mathbb{P}_{il|\beta}\right] = \int_\beta \mathbb{P}_{il|\beta} f_\theta(\beta) d\beta = \int_\beta \frac{\exp\{\beta' x_{il}\}}{\sum_j \exp\{\beta' x_{ij}\}} f_\theta(\beta) d\beta. \tag{9.3}$$

As before, for the one-dimensional integral, the `mlogit` package uses standard numerical integration methods.

If $V_{il} = \beta' x_{il}$ where β_i is a vector of length K and $f_\theta(\beta)$ is the joint density of the K individual-specific coefficients, the unconditional probability is:

$$\mathbb{P}_{il} = \mathbb{E}_\theta\left[\mathbb{P}_{il|\beta}\right] = \int_{\beta_1} \dots \int_{\beta_K} \mathbb{P}_{il|\beta} f_\theta(\beta) d\beta_1 \dots d\beta_K.$$

This K-dimension integral cannot, in general, be estimated easily using standard quadrature methods and the only practical method available to date is to use simulations. More precisely, R draws of the parameters are taken from the distribution of β, the probability, conditioned on choice of β, is computed for every draw and the unconditional probability, which is the expected value of the conditional probabilities is estimated by the average of these R conditional probabilities.

Individual parameters The expected value of a random coefficient $\mathbb{E}[\beta]$ is simply estimated by the mean of the R draws on its distribution: $\overline{\beta} = \sum_{r=1}^R \beta_r$. Individual parameters are obtained by first computing the probabilities of the observed choice of i for every value of β_r:

$$\mathbb{P}_{ir} = \frac{\sum_j y_{ij} e^{\beta_i x_{ij}}}{\sum_j e^{\beta_i x_{ij}}}$$

where y_{ij} is a dummy equal to one if i has chosen alternative j. The expected value of the parameter for an individual is then estimated by using these probabilities to weight the R values of β:

$$\widehat{\beta_i} = \frac{\sum_r \mathbb{P}_{ir} \beta_r}{\sum_r \mathbb{P}_{ir}}.$$

Panel data If there are repeated observations for the same individuals, the longitudinal dimension of the data can be taken into account in the mixed logit model, assuming that the random parameters of individual i are the same for all his choice situations. Denoting y_{itl} a dummy equal to 1 if i choose alternative l for the tth choice situation, the probability of the observed choice is:

$$\mathbb{P}_{it} = \prod_j \frac{\sum_j y_{itj} e^{\beta_i x_{itl}}}{\sum_j e^{\beta_i x_{itj}}}.$$

The joint probability for the T observations of individual i is then:

$$\mathbb{P}_i = \prod_t \prod_j \frac{\sum_j y_{itj} e^{\beta_i x_{itj}}}{\sum_j e^{\beta_i x_{itj}}}$$

and the log-likelihood is simply $\sum_i \log \mathbb{P}_i$.

9.3.4 Latent Class Model

The *latent class model* (LCM) for discrete choice analysis is an alternative method to the RPL model. The LCM for discrete choice analysis assumes that there are a finite number of categories, for each category there is a 'true' parameter vector β and each individual belongs to one of these categories. This makes it less flexible than the RPL, where each individual can have different parameters, but is clearly more useful when it is important to locate the sources of the heterogeneity for individual preferences.

The LCM groups respondents in a finite number of classes (the number of classes may be chosen by analysing with different numbers of classes and then using one of the standard selection criteria, such as AIC or BIC), Membership of a specific class is based on the subject's answers to the DCE questions posed and also other characteristics (e.g. socio-demographic factors). The LCM assumes that the preferences of respondents are homogeneous within each class; they may be heterogeneous across classes. Grouping respondents with homogeneous preferences in a finite number of classes is relevant for decision-makers because it helps them to understand the sources of heterogeneity between individuals.

The LCM works as follows: We place a prior probability of H_{iq}, that individual i is from class q, where $q \in \{1, \ldots, Q\}$ and there are Q classes. The probability that individual i in choice set t chooses option j *given* that the individual is from class q is $\mathbb{P}_{it|q}(j)$ where $j \in \{1, \ldots J\}$ (choice set has J alternatives). Here

$$\mathbb{P}_{it|q}(j) = \frac{\exp\{x'_{it.j}\beta_q\}}{\sum_{j=1}^{J} \exp\{x'_{it.j}\beta_q\}}$$

The log-likelihood function for *all* the respondents is:

$$\log L = \sum_{i=1}^{N} \log \left\{ \sum_{q=1}^{Q} H_{iq} \left(\prod_{t=1}^{T} \mathbb{P}_{it|q}(j) \right) \right\}$$

A convenient and standard choice of prior H is a multinomial logit:

$$H_{iq} = \frac{\exp\{z'_i\theta_q\}}{\sum_{p=1}^{Q} \exp\{z'_i\theta_p\}} \quad q = 1, \ldots Q, \quad \theta_Q = 0.$$

Here, z_i denotes a set of observable characteristics (e.g., socio-demographics such as age, income and sex) that enter the model for class membership.

The parameters to be estimated are now the β_q parameters and also the θ_q parameters. Once these have been estimated, the Bayes rule may be used to obtain respondent-specific (posterior) estimates of the class probability $\widehat{H}_{q|i}$, conditioned on their estimated choice probabilities:

$$\widehat{H}_{q|i} = \frac{\widehat{\mathbb{P}}_{i|q}\widehat{H}_{iq}}{\sum_{p=1}^{Q} \widehat{\mathbb{P}}_{i|p}\widehat{H}_{ip}}.$$

These respondent-specific (posterior) estimates of the class probability may then used in a *beta regression* analysis to profile the members of each class. To determine the number of classes, the Consistent Akaike Information Criterion (CAIC), and the Bayesian Information Criterion (BIC) may be used.

After deciding on the number of classes and classifying respondents, each class may be characterised using, among other things, information on the attitudes and socio-demographic characteristics of respondents. Those variables may then be regressed against respondent-specific (posterior) estimates of the class probability $\widehat{H}_{q|i}$. Since the dependent variable is in form of probability, a Beta regression model for each segment may be used.

9.3.5 Estimating Willingness to Pay

The parameters β do not give directly willingness-to-pay values, since they do not have the correct scaling, although WTP can be computed when the price coefficient is known. WTP for an attribute is commonly expressed as the negative ratio of the (non-price) attribute coefficient to the price coefficient:

$$\text{WTP}(\text{non-price attribute}) = -\frac{\beta_{\text{non-price attribute}}}{\beta_{\text{price}}} \tag{9.4}$$

This value represents the marginal WTP of the respondents. For the attributes coded as continuous, the calculated value represents, for a respondent with parameter vector β, the amount of increase in the attribute for which the respondent is willing to pay one unit of money (e.g. \$1). In the case of categorical attributes, the calculated value represents the respondent's willingness to pay WTP for the level of interest of the attribute with respect to the baseline level.
After deciding on the number of classes and classifying respondents, each class may be characterised using, among other things, information on the attitudes and socio-demographic characteristics of respondents. Those variables may then be regressed against respondent-specific (posterior) estimates of the class probability $\widehat{H}_{q|i}$. Since the dependent variable is in form of probability, a Beta regression model for each segment may be used.

9.4 Optimal design

When considering the question of design, we simplify the utility to:

$$U_{ij} = x'_{ij}\beta + \epsilon_{ij}$$

where i indexes individual / choice situation and j indexes alternative within the choice situation and the ϵ_{ij}'s are i.i.d. extreme-value (Gumbel) error terms.. Also, we consider k covariates, so that we take each x_{ij} as a k-vector and

$$x'_{ij}\beta = \sum_{l=1}^{k} x_{ij;k}\beta_k.$$

so that β is a k-vector of parameters. Let $\mathbf{X}$ denote a *stacked choice design matrix.* By this we mean a matrix whose elements are

$$(x'_{ij})_{i=1,\ldots,S;j=1,\ldots J}$$

where each x_{js} is a $k \times 1$ vector with the attribute levels of alternative j in choice set i.
With these notations, the multinomial logit probability that alternative j in choice set i is chosen is:

$$\mathbb{P}_{ij} = \frac{\exp\{\sum_{l=1}^{k} x_{ij;l}\beta_l\}}{\sum_{s=1}^{J}\exp\{\sum_{l=1}^{k} x_{is;l}\beta_l\}}.$$

The information matrix can be computed quite easily; the parameters β are the canonical parameters of an exponential family, hence (suppressing the notation for choice situation and using y to denote alternative)

$$I_{ij}(\beta) = \frac{\partial^2}{\partial\beta_i\partial\beta_j}\log\sum_y \exp\left\{\sum_{l=1}^{k} x_{yl}\beta_l\right\} = \sum_y \mathbb{P}_y x_{yj}x_{yk} - \left(\sum_y x_{yj}\mathbb{P}_y\right)\left(\sum_y x_{yk}\mathbb{P}_y\right)$$

so that, for a single individual/choice situation,

$$I(\beta) = x'\mathbf{P}x - x'\mathbb{P}\mathbb{P}'x$$

where $\mathbf{P}$ is the diagonal matrix $\mathbf{P} = \text{diag}(\mathbb{P}(1),\ldots,\mathbb{P}(J))$, $\mathbb{P}$ is the vector $\mathbb{P} = (\mathbb{P}(1),\ldots,\mathbb{P}(J))'$ of probabilities for each alternative within the choice situation. x is the matrix with elements x_{yj} denoting the value of covariate j in the alternative y.

Hence, if we have n respondents (without any individual-specific covariates) and S choice situations, the total information from an experiment with stacked design matrix $\mathbf{X}$ and parameter vector β is:

$$I(\mathbf{X},\beta) = n\sum_{i=1}^{S} x'_i(\mathbf{P}_i - \mathbb{P}_i\mathbb{P}'_i)x_i \tag{9.5}$$

where each x_i is a $J \times k$ matrix: $x_{ij;l}$ is the value of covariate l for alternative j within choice situation i.

9.5 Implementation of Random Utility using mlogit

Format of Data For choice experiments, data can be given in two formats, the *wide* format or the *long* format. The data set `Train` from **mlogit** is an example of a data set in wide format; we give the first three rows of it to give an idea.

```
> data("Train",package="mlogit")
> Train$choiceid <- 1:nrow(Train)
> head(Train,3)
  id choiceid choice price_A time_A change_A comfort_A price_B time_B change_B comfort_B
1  1        1      A    2400    150        0         1    4000    150        0         1
```

```
2  1        2      A    2400    150        0         1    3200    130        0         1
3  1        3      A    2400    115        0         1    4000    115        0         0
```

The `id` column gives the identity of the individual, `choiceid` the label of the experimental run for that individual, `choice` the choice that was made, followed by a list of attributes for the two alternatives (Alternative A and Alternative B) where for a train ride, the price, the time taken, the number of changes and the level of comfort are given.

The data set is from a *stated preference* survey in the Netherlands.

The data set `ModeCanada` from **mlogit** is an example of a data set in *long* format;

```
> data("ModeCanada",package = "mlogit")
> head(ModeCanada)
  case   alt choice dist  cost ivt ovt freq income urban noalt
1    1 train      0   83 28.25  50  66    4     45     0     2
2    1   car      1   83 15.77  61   0    0     45     0     2
3    2 train      0   83 28.25  50  66    4     25     0     2
4    2   car      1   83 15.77  61   0    0     25     0     2
5    3 train      0   83 28.25  50  66    4     70     0     2
6    3   car      1   83 15.77  61   0    0     70     0     2
```

In this data set, there are four modes of transport (air, train, bus, car). The variables are distance (`dist`), monetary cost (`cost`), in-vehicle-time (`ivt` - amount of time spent inside the vehicle), out-of-vehicle time (`ovt` - how much of the total time was not in the vehicle), frequency (`freq`), income (`income`), whether or not the trip has a large city at the origin (`urban`) and number of alternatives available (`noalt`). The *alternative specific* variables are `cost`, `ivt`, `ovt` and `freq`, while the choice situation specific variables are `dist`, `income`, `urban` and `noalt`.

The package **mlogit** uses objects of class `Formula` from the package **Formula** (by Zeileis and Croissant). For example, if we want to explain the choice by:

- `cost`, an alternative specific variable with a generic coefficient,
- `income` and `urban`, choice specific variables,
- `ivt`, an alternative specific variable where we require alternative specific coefficients

the formula is:

```
> library("Formula")
> f <- Formula(choice ~ cost | income + urban | ivt)
```

Now suppose we're interested in a subset of the data, where `noalt` (number of alternatives available) takes the value 4; i.e. all four forms of transport are available. This is done using the `dfidx` command, forcing the data into an appropriate data frame, as follows:

```
MC = dfidx(ModeCanada, subset = noalt == 4, alt.levels = c("train","air","bus","car"))
```

and, the first 10 entries of the data frame `MC` are:

```
> head(MC)
~~~~~~~
 first 10 observations out of 11116
~~~~~~~
   choice dist   cost ivt ovt freq income urban noalt      idx
1       0  377  58.25 215  74    4     45     0     4 109:rain
2       1  377 142.80  56  85    9     45     0     4  109:air
3       0  377  27.52 301  63    8     45     0     4  109:bus
4       0  377  71.63 262   0    0     45     0     4  109:car
5       0  377  58.25 215  74    4     70     0     4 110:rain
6       1  377 142.80  56  85    9     70     0     4  110:air
7       0  377  27.52 301  63    8     70     0     4  110:bus
8       0  377  71.63 262   0    0     70     0     4  110:car
9       0  377  58.25 215  74    4     35     0     4 111:rain
10      1  377 142.80  56  85    9     35     0     4  111:air

~~~ indexes ~~~~
   case   alt
1   109 train
2   109   air
3   109   bus
4   109   car
5   110 train
6   110   air
7   110   bus
8   110   car
9   111 train
10  111   air
indexes:  1, 2
```

We may want to consider the *total time*, which is the sum of `ivt` and `ovt`. We can do this as follows:

```
> MC$time = with(MC,ivt+ovt)
```

Fitting the model is now very easy; we give an example here.

```
> m1.MC1 = mlogit(choice~cost+freq+ovt|income|ivt,MC)
> summary(m1.MC1)

Call:
mlogit(formula = choice ~ cost + freq + ovt | income | ivt, data = MC,
    method = "nr")

Frequencies of alternatives:choice
    train       air       bus       car
0.1666067 0.3738755 0.0035984 0.4559194

nr method
9 iterations, 0h:0m:0s
g'(-H)^-1g = 0.00014
successive function values within tolerance limits
```

```
Coefficients :
                  Estimate Std. Error  z-value  Pr(>|z|)
(Intercept):air -3.2741952  0.6244152  -5.2436 1.575e-07 ***
(Intercept):bus -2.5758571  1.0845227  -2.3751 0.0175439 *
(Intercept):car -1.4300823  0.3013764  -4.7452 2.083e-06 ***
cost            -0.0333389  0.0070955  -4.6986 2.620e-06 ***
freq             0.0925297  0.0050976  18.1517 < 2.2e-16 ***
ovt             -0.0430036  0.0032247 -13.3356 < 2.2e-16 ***
income:air       0.0381466  0.0040831   9.3426 < 2.2e-16 ***
income:bus      -0.0509401  0.0181702  -2.8035 0.0050553 **
income:car       0.0101536  0.0031648   3.2083 0.0013353 **
ivt:train       -0.0014504  0.0011875  -1.2214 0.2219430
ivt:air          0.0595097  0.0100727   5.9080 3.463e-09 ***
ivt:bus         -0.0067835  0.0044334  -1.5301 0.1259938
ivt:car         -0.0064603  0.0018985  -3.4029 0.0006668 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-Likelihood: -1874.3
McFadden R^2:  0.35443
Likelihood ratio test : chisq = 2058.1 (p.value = < 2.22e-16)
```

9.5.1 The Random Parameters Model

We return to the `Train` data set. This is given in wide format and has to be coerced into a suitable data frame in long format. The `id` variable with the individual index nests the choice situation variable `choiceid`. We'll use `dfidx` to put the data into an appropriate format, which we call `Tr`.

```
Tr = dfidx(Train,shape="wide",choice="choice",varying=4:11,sep = "_",
           idx = list(c("choiceid","id")),idnames=c("chid","alt"),
           opposite = c("price","comfort","time","change"))
```

Next, `price` (in guilders) is converted to euros and time (in minutes) is converted to hours.

```
> Tr$price = Tr$price/100 * 2.20371
> Tr$time = Tr$time / 60
> head(Tr,3)
~~~~~~~
 first 3 observations out of 5858
~~~~~~~
  choice     price time change comfort idx
1   TRUE -52.88904 -2.5      0      -1 1:A
2  FALSE -88.14840 -2.5      0      -1 1:B
```

```
3   TRUE -52.88904 -2.5      0      -1 2:A

~~~ indexes ~~~~
  chid id alt
1    1  1   A
2    1  1   B
3    2  1   A
indexes:  1, 1, 2
```

Firstly, we estimate the multinomial model:

```
> Train.ml = mlogit(choice~price+time+change+comfort|-1,Tr)
> summary(Train.ml)

Call:
mlogit(formula = choice ~ price + time + change + comfort | -1,
    data = Tr, method = "nr")

Frequencies of alternatives:choice
      A       B
0.50324 0.49676

nr method
5 iterations, 0h:0m:0s
g'(-H)^-1g = 0.00014
successive function values within tolerance limits

Coefficients :
         Estimate Std. Error z-value  Pr(>|z|)
price   0.0673580  0.0033933 19.8506 < 2.2e-16 ***
time    1.7205514  0.1603517 10.7299 < 2.2e-16 ***
change  0.3263409  0.0594892  5.4857 4.118e-08 ***
comfort 0.9457256  0.0649455 14.5618 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-Likelihood: -1724.2
```

Dividing coefficients by the price coefficient gives estimated monetary values that the customers put on the various features.

```
> coef(Train.ml)[-1]/coef(Train.ml)[1]
    time   change  comfort
25.54337  4.84487 14.04028
```

so that the values are 26 euros per hour of travelling, 5 euros for a change and 14 euros to travel in a more comfortable class.

Let us now consider a random parameters model, with three random parameters, `time`, `change` and `comfort`. The *uncorrelated* mixed logit model is estimated by:

```
> Train.mxlu = mlogit(choice~price+time+change+comfort|-1,Tr,
+                     panel=TRUE,rpar=c(time="n",change="n",comfort="n"),
+                     R=100,correlation=FALSE,halton=NA,method="bhhh")
> names(coef(Train.mxlu))
[1] "price"      "time"       "change"     "comfort"    "sd.time"    "sd.change"
"sd.comfort"
```

There are three additional parameters, the standard deviations of the distribution of the three random parameters.

We can introduce correlation by setting `correlation = TRUE`;

```
> Train.mxlc = mlogit(choice~price+time+change+comfort|-1,Tr,
+                     panel=TRUE,rpar=c(time="n",change="n",comfort="n"),
+                     R=100,correlation=TRUE,halton=NA,method="bhhh")
> names(coef(Train.mxlc))
 [1] "price"                "time"                 "change"               "comfort"
      "chol.time:time"
 [6] "chol.time:change"     "chol.change:change"   "chol.time:comfort"
"chol.change:comfort"  "chol.comfort:comfort"
```

where the additional parameters come from the Choleski decomposition of the covariance matrix of the three random parameters;

$$C = \begin{pmatrix} c_{11} & 0 & 0 \\ c_{12} & c_{22} & 0 \\ c_{13} & c_{23} & c_{33} \end{pmatrix}$$

where CC' is the covariance matrix.

The random parameters are obtained using `rpar`. For example, the marginal for the time parameter may be summarised as follows:

```
> marg.ut.time = rpar(Train.mxlc,"time")
> summary(marg.ut.time)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    -Inf 1.283749 4.893752 4.893752 8.503756      Inf
```

These parameters are in the *preference space*, but parameters in the WTP (willingness to pay) space are easier to interpret. Divide the marginal utility by the price covariate, taken as a random parameter:

```
> wtp.time = rpar(Train.mxlc,"time",norm="price")
> summary(wtp.time)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
     -Inf  8.753119 33.367588 33.367588 57.982056       Inf
```

The standard errors of the parameters of the covariance matrix may be computed using `vcov`:

```
> vcov(Train.mxlc,what="rpar")
              time     change  comfort
time    28.6460389 -0.2787999 5.557933
change  -0.2787999  3.1047367 1.232467
comfort  5.5579334  1.2324667 7.895535
```

Chapter 10

Bayesian Nonparametric Models

10.1 Introduction

Let us consider two problems that have arisen so far:

- How many classes should I choose for a clustering problem?
- How many factors should I use in a factor analysis?

The answer, so far, was of the style: 'for an agglomerative clustering (e.g. using the Ward algorithm), look at the dendrogram; there is often a clear distance where the procedure stabilises and this gives a reasonable clustering.'

Similarly with factor analysis; we construct the factors and then we consider their properties; how much of the observation is explained by the factors and how much is 'noise'. In many situations it should be clear *by inspection* which of the factors that we constructed using PCA are important.

For *Partition Around Medians* (or Partition Around Means) it is advisable to run the algorithm for k clusters for several different choices of k.

Bayesian Nonparametric (BNP) models provide one approach to the problem. Rather than comparing different models with varying complexity, the BNP approach is to fit a *single* model that can adapt its complexity to the data.

BNP models allow the complexity to *grow* as more data is observed.

Example: Clustering A model where the data naturally fits into k different clusters can be thought of as a *mixture model*, where we have a mixture of k different underlying populations. For `pam` (for example), we need to specify the number of clusters in advance. The BNP model estimates the number of clusters needed to model the observed data and allows future data to exhibit previously unseen clusters.

A BNP model expresses a *generative process* of the data that includes hidden variables.The model specifies the joint probability distribution of the hidden and observed variables.

Given a data set, data analysis is performed by *posterior inference*, computing the conditional distribution of the hidden variables given the observed data. We (in some sense) 'reverse' the generative process; we find the distribution of the hidden variables that is likely to have generated the given data. Its *complexity*, i.e. the number of mixture components, or number of factors, is part of the posterior distribution. We do not need to specify these in advance; they are determined as part of the data analysis.

We'll look at two Bayesian nonparametric models. In this lecture, we'll consider the *Chinese Restaurant Process* which is used for cluster assignment and in the next we'll consider the *Indian Buffet Process*, which is used for *Latent Factor models*.

10.2 Mixture Models and Clustering

In a *mixture model*, each data point is assumed to belong to a cluster. In posterior inference, we infer a grouping (or clustering) of the data. This amounts to inferring both the *identities* of the clusters and the *assignments of data* to them.

Example: Cognitive Response Times Several cognitive processes contribute to producing behavioural responses. The question is how to decompose observed RT's into their underlying components.

10.2.1 Finite Mixture Model

Assume there are K clusters, each associated with a parameter $\theta_k : k = 1, \dots K$. For example, a datum from cluster k is an observation from a $N(\theta_k, \Sigma)$ population. Each observation y_i is assumed to be *generated* by *first* choosing a cluster c_i with probability $\mathbb{P}(c_i)$ and then choosing y_i from the distribution parametrised by θ_{c_i}.

Bayesian mixture models have a *prior* over the clusters $\mathbb{P}(c_i)$ and a prior over cluster parameters; the parameters for different clusters are chosen independently of each other and $\theta \sim G_0$ for a distribution G_0.

The generative process defines a joint distribution;

$$\mathbb{P}(\underline{y}, \underline{c}, \underline{\theta}) = \prod_{k=1}^{K} G_0(\theta_k) \prod_{n=1}^{N} F(y_N|\theta_{c_n})\mathbb{P}(c_n).$$

Observations $\underline{y} = (y_1, \dots, y_N)$, cluster assignments $\underline{c} = (c_1, \dots, c_N)$, cluster parameters $\underline{\theta} = (\theta_1, \dots, \theta_K)$.

Given a data set, we are usually interested in the cluster assignments, to be inferred from the observations $\underline{y}$ and we can get this by Bayes rule:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$

which gives:

$$\mathbb{P}(\underline{c}|\underline{y}) = \frac{\mathbb{P}(\underline{y}|\underline{c})\mathbb{P}(\underline{c})}{\sum_c \mathbb{P}(\underline{y}|\underline{c})\mathbb{P}(\underline{c})}.$$

The *likelihood* is obtained by marginalising over the settings of θ

$$\mathbb{P}(\underline{y}|\underline{c}) = \int_\theta \left(\prod_{n=1}^{N} F(y_n|\theta_{c_n}) \prod_{k=1}^{K} G_0(\theta_k) \right) d\underline{\theta}$$

We would like to be able to calculate this explicitly. If G_0 is *conjugate* to F, then we can do this. For example, we could take G_0 as a Gaussian distribution; when new clusters are established, their cluster means are drawn independently from a Gaussian distribution. If $F(.|\theta)$ (the distribution of the observation given the cluster parameters θ) is also Gaussian, the expression may be computed.
The denominator cannot be computed, since this involves summing over *every single* partition of the data into K groups. Approximate methods (such as Markov chain Monte Carlo) are therefore used to establish properties of the posterior distribution and hence discover good clusterings.

10.2.2 The Chinese Restaurant Process

The *Chinese Restaurant Process* is a method for generating clusters and cluster memberships. The term is attributable to Jim Pitman from Berkeley. He always found in San Fransisco that (a) no matter how busy it was, his favourite Chinese Restaurant could always accommodate new customers and that (b) tables could always be expanded to accommodate new customers who wanted to sit at a table which contained their friends and colleagues who were already in the restaurant.

Let c_n denote the table assignment of customer n and let $\mathbf{c}_{1:n-1} = (c_1, \ldots, c_{n-1})$. Suppose that K_n denotes the number of tables occupied after the nth customer has arrived. Then

$$\mathbb{P}(c_n = k|\mathbf{c}_{1:n-1}) = \begin{cases} \frac{m_k}{n-1+\alpha} & k \leq K_{n-1} \\ \frac{\alpha}{n-1+\alpha} & k = K_N \end{cases}$$

where m_k denotes the number of customers sitting at table k when there are $n-1$ customers.

Now,

$$p(c_1, \ldots, c_N) = p(c_1)p(c_2|c_1)\ldots p(c_N|c_1, \ldots, c_{N-1})$$

so that

$$p(c_1, \ldots, c_N) = \frac{\alpha^{K_N} \prod_{j=1}^{K_N} (m_{j,N} - 1)!}{\prod_{n=1}^{N} (\alpha - 1 + n)}$$

where $m_{j,N}$ denotes the total number of customers at table j after N arrivals.

If we only observe different tables and we do not examine the order in which tables were occupied, this distribution is *exchangeable*.

The RT Example The CRP allows us to place a prior distribution over *partitions* of RTs into the hypothetical cognitive processes that generated them, without committing in advance to the number of such processes.

Each process k is associated with a set of parameters θ_k specifying the distribution over the RTs obtained from the posterior, which may be obtained by Gibbs sampling.

Data Analysis using CRP When we analyse data with a CRP, we form an approximation of the joint posterior over the (hidden) class variables and the distributional parameters for each cluster. In practise, there are two uses for this posterior.

1. Examine the likely partitioning of the data. This gives us a sense of how the data are grouped and how many groups the CRP model chooses.
2. Form predictions with the *posterior predictive distribution*. With the CRP, the posterior predictive distribution is:

$$\mathbb{P}(y_{n+1}|y_1,\ldots,y_n) = \sum_{c_1,\ldots,c_{n+1}} \mathbb{P}(y_{n+1}|c_{n+1},\theta)\mathbb{P}(c_{n+1}|c_1,\ldots,c_n)\mathbb{P}(c_1,\ldots,c_n,\underline{\theta}|y_1,\ldots,y_n)d\underline{\theta}.$$

10.3 Implementation in R

There is a package **dirichletprocess** which is useful.

```
>install.packages("dirichletprocess",dependencies=TRUE)
>library(dirichletprocess)
```

and now let us try it out.

We'll consider the data set `faithful`, which is contained in the basic R installation. There are 272 bi-variate observations. The variables are the time between eruptions and the length of the eruption for the geyser 'Old Faithful'.

Making a plot of the waiting times might suggest that there are two clusters and that each cluster may be Gaussian, so the data could arise as a mixture of two Gaussians. We'll see how this is recovered.

```
?faithful
```

On the right you will see a description of the data.

```
its <- 500
faithfulTransformed <- scale(faithful$waiting)
dp <- DirichletProcessGaussian(faithfulTransformed)
dp <- Fit(dp, its)
plot(dp)
```

On the right hand side, you see the plot. Clearly, it has estimated *two* clusters from the data. The density is therefore estimated as the composition of two Gaussians. It has estimated the cluster sizes and has assigned data to two clusters. From this, it has computed the sample mean and sample variance of each cluster, the Gaussian density for each cluster and has fitted the corresponding population density.

Now let us try a multivariate example; let us use both variables in the `faithful` data set.

```
faithfulTrans <- scale(faithful)
dp <-  DirichletProcessMvnormal(faithfulTrans)
dp <- Fit(dp, 1000)
plot(dp)
```

The `Fit` command may take a long time to execute.

The `plot` shows the clustering that the CRP Gauss model has assigned; points from one cluster in red and the other in blue.

10.4 Binomial Clusters: different 'success' probabilities

We now consider data that is either 'success' or 'failure'. A *Bernoulli trial* $\mathrm{Be}(\theta)$ is a random variable X which takes the value 1 ('success') or 0 ('failure') and $\mathbb{P}(X = 1) = \theta$, $\mathbb{P}(X = 0) = 1 - \theta$.

Suppose we have n independent identically distributed $\mathrm{Be}(\theta)$ variables and $Y = X_1 + \ldots + X_n$ is the total number of 'success', then

$$\mathbb{P}(Y = k) = \binom{n}{k}\theta^k(1-\theta)^{n-k} \qquad k = 0, 1, \ldots, n.$$

We now consider a situation where the 'success' probability θ is unknown. One way of modelling the uncertainty is to place a *prior* distribution over θ and one of the most convenient families of distributions is the *Beta* distribution. A random variable Θ has $\mathrm{Beta}(\alpha, \beta)$ distribution if its density is

$$f_\Theta(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}\mathbf{1}_{[0,1]}(\theta).$$

Here

$$\mathbb{E}[\Theta] = \frac{\alpha}{\alpha+\beta}, \qquad \mathrm{Var}(\Theta) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

so that prior information can be modelled. The parameters α and β are chosen such that the 'guess' for θ is $\frac{\alpha}{\alpha+\beta}$ and the more certainty the user has concerning this guess, the larger the choice of $\alpha+\beta$.

The Update If we have a prior distribution $\text{Beta}(\alpha, \beta)$ over the parameter θ and we observe y from $\text{Binomial}(n, \theta)$, the *posterior* distribution over θ is given by $\text{Beta}(y+\alpha, n-y+\beta)$.

Note that its expected value is: $\frac{y+\alpha}{n+\alpha+\beta}$.

We can envisage situations where the data comes from several *different* populations, each with its own 'success' probability. Suppose that $Y_i \sim \text{Binomial}(n_i, \theta_i)$ for $i = 1, \ldots, N$.

A model could be

$$\theta_1, \ldots, \theta_N \sim i.i.d.\text{Beta}(\alpha, \beta)$$
$$Y_i|\theta_i \sim \text{Binomial}(n_i, \theta_i)$$

In the example below, the θ_is do not seem to be i.i.d. beta(α, β). They would appear to come from a *bimodal* distribution, which is not the pattern of a Beta. The Beta is either unimodal, or else takes its maxima at the end points of 1 and / or 0.

A Dirichlet Process model could be:

$$F \sim \text{DP}(\alpha, G_0)$$
$$\alpha_i, \beta_i \sim F$$
$$\theta_i \sim \text{Beta}(\alpha_i, \beta_i)$$
$$y_i \sim \text{Binomial}(n_i, \theta_i)$$

where there are K different (α_i, β_i) values, each corresponding to a cluster; the clusters generated by the Chinese Restaurant Process with parameter α and the distribution from which the (α_i, β_i) values for each cluster are chosen is F, determined by sampling parameters from G_0.

Example: Tumour risk in rats The data set `rats` is contained in the package **dirichletprocess** and the data is from Gelman, Carlin, Stern, and Rubin (2014). In this example, there are 71 different experiments, and during each experiment a number of rats are inspected for tumours, with the number of rats which have tumours in each experiment being the observed data. The first column is the number of rats which have tumours in each experiment, and the second is the number of rats. A naive approach would model each experiment as a Binomial draw with unknown θ_i (the 'success' probability) and known N_i. A Beta distribution is the conjugate prior for the Binomial distribution and would be used as the prior on θ:

$$y_i|\theta_i, N_i \sim \text{Binomial}(N_i, \theta_i)$$
$$\theta_i \sim \text{Beta}(\alpha, \beta).$$

The empirical distribution of However, Figure 4a shows the empirical distribution of $\widehat{\theta_i} := \frac{y_i}{N_i}$ would suggest otherwise; the empirical distribution suggests bi-modality, something that a *single* Beta distribution cannot capture. Hence this choice of prior over θ_i is dubious. An alternative procedure is to

instead use a nonparametric prior, which is a *mixture* of Beta distributions. Since these parameters are constrained to lie between 0 and 1, a *Dirichlet process* mixture of Beta distributions might be reasonable. This leads to the following model:

$$
\begin{aligned}
&y_i|\theta_i, N_i \sim \text{Binomial}(N_i, \theta_i),\\
&\theta_i \sim \text{Beta}(\alpha_i, \beta_i)\\
&\alpha_i, \beta_i \sim F,\\
&F \sim DP(\alpha, G_0)
\end{aligned}
$$

for some parameters α and G_0. These can follow the default implementations in the package **dirichletprocess**; for reasonable choices, the results should not depend heavily on these. This can be implemented as follows:

```
> library(dirichletprocess)
> numSamples = 200
> thetaDirichlet <- matrix(nrow=numSamples, ncol=nrow(rats))
> dpobj <- DirichletProcessBeta(rats$y/rats$N,maxY=1,g0Priors = c(2,
150),mhStep=c(0.25, 0.25),hyperPriorParameters = c(1, 1/150))
 > dpobj <- Fit(dpobj, 10)
 > clusters <- dpobj$clusterParameters
> a <- clusters[[1]] * clusters[[2]]
> b <- (1 - clusters[[1]]) * clusters[[2]]
> for(i in seq_len(numSamples)){
+ posteriorA <- a[dpobj$clusterLabels] + rats$y
+ posteriorB <- b[dpobj$clusterLabels] + rats$N - rats$y
+ thetaDirichlet[i, ] <- rbeta(nrow(rats), posteriorA, posteriorB)
+ dpobj <- ChangeObservations(dpobj, thetaDirichlet[i, ])
+ dpobj <- Fit(dpobj, 5)
+ clusters <- dpobj$clusterParameters
+ a <- clusters[[1]] * clusters[[2]]
+ b <- (1 - clusters[[1]]) * clusters[[2]]
+ }
```

Note the reason why the observations are changing is because the DP mixture model is applied to the θ_i parameters, which are resampled (and hence have different values) during each MCMC iteration.

```
> library(ggplot2)
> ggplot(rats, aes(x=y/N)) +
+ geom_density(fill="black") #Plot the emperical distribution
> ggplot(rats, aes(x=y/N)) +
+ geom_density(fill="black") #Plot the emperical distribution
> posteriorFrame <- PosteriorFrame(dpobj, ppoints(1000))
```

```
> ggplot() +
+ geom_ribbon(data=posteriorFrame,aes(x=x,
ymin=X5.,ymax=X95.),alpha=0.2) +
+ geom_line(data=posteriorFrame, aes(x=x, y=Mean)) +
+ xlim(c(0, 0.35)) #Plot the resulting prior distribution
```

Plotting the resulting estimation reveals that the DP is a more suitable prior than the Beta distribution. This confirms what we saw from the empirical distribution that the data is bi-modal.

Inconsistency The CRP method for clustering has its uses and, as we have seen can give greater accuracy in modelling. There are, though, problems with it. Miller and Harrison (JMLR volume 15 (2014) pp 3333 - 3370) point to inconsistency; let $N(n)$ denote the number of clusters chosen to maximise the posterior, then this does not necessarily converge to the true value.

10.5 Latent Factor Models and Dimensionality Reduction

Mixture models assume that each observation is assigned to one of K components. Latent factor models weaken this assumption; each observation is influenced by each of K components in a different way.

Latent factor models provide *dimensionality reduction*; the number of components is usually smaller than the dimension of the data. Each observation is associated with a vector of *component activations* (latent factors) that describe how much the each component contributes to it.

The most popular of these models, Factor Analysis (FA), Principal Component Analysis (PCA) and *Independent Component Analysis* (ICA) all assume that the number of factors (K) is known. The Bayesian Nonparametric approach described here allows the number of factors to grow as more data is added.

As with the BNP mixture model, the posterior distribution provides both the properties of the latent factors and how many are exhibited in the data.

In classical factor analysis, the data matrix $\mathbf{x}$ is an $N \times M$ matrix; N M-variate observations. Observation y_n is expressed as

$$y_n = Gx_n + \epsilon_n$$

where y_n is the observed M-vector for observation n, x_n is an $M \times K$ factor loading matrix expressing how the latent factor k influences observation dimension m, x_n is a K-dimensional vector expressing the activity of each latent factor and ϵ_n is a vector of independent Gaussian noise terms.

The factor loadings can be decomposed as $G_{mk} = z_{mk}w_{mk}$ where z_{mk} is a binary 'mask' variable, equal to 1 if factor k is 'on' and equal to 0 if factor k is 'off'. w_{mk} is a continuous weight variable.

Now let us consider the Bayesian approach to inferring the latent factors, mask variables and weights. We place priors over them and use Bayes rule to compute the posterior $\mathbb{P}(X, Z, W|Y)$.

Just as the CPR, the infinite-capacity distribution over Z has been furnished with a similarly colourful culinary metaphor, the *Indian Buffet Process*.

A customer (a dimension) enters a buffet which has an infinite number of dishes to choose from, arranged in a line. The probability that customer m samples dish k (i.e. $z_{mk} = 1$) is proportional to its popularity h_k (the number of previous customers who have sampled dish k). When the customer has considered all previous sampled dishes, he chooses an additional Poisson(α/N) dishes that have never been sampled before. When all M customers have navigated the buffet, the resulting binary matrix Z is a draw from the IBP.

The IBP plays the same role for latent factor models as the CRP plays for mixture models; it functions as an infinite capacity prior over the space of latent variables allowing an unbounded number of latent factors.

We then proceed with

$$\mathbb{P}(X, W, Z|Y) = \mathbb{P}(Y|X, W, Z)\mathbb{P}(X)\mathbb{P}(W)\mathbb{P}(Z).$$

Exact inference is (of course) impossible; Markov chain Monte Carlo methods have to be used.