Prelims Statistics and Data Analysis, Lectures 11-16

Frank Windmeijer
With grateful acknowledgements to Christl Donnelly, Jonathan Marchini and Dino Sejdinovic

Trinity Term 2026 (version of May 20, 2026)

Contents

1 Introduction **2**
- 1.1 Using R or Python for data analysis . . . 3
- 1.2 Motivating examples . . . 4
  - 1.2.1 Single cell genomics . . . 4
  - 1.2.2 Food consumption . . . 9
  - 1.2.3 Finding structure in genetic datasets . . . 11
- 1.3 Data matrices and notation . . . 11

2 Exploratory data analysis and visualising datasets **13**
- 2.1 Crabs dataset . . . 13
- 2.2 Histograms . . . 13
- 2.3 Boxplots . . . 13
- 2.4 Pairs plots . . . 17
- 2.5 The Multivariate Normal Distribution . . . 20
  - 2.5.1 Example : the Bivariate normal distribution ($p = 2$) . 21
  - 2.5.2 Estimating parameters for the multivariate normal distribution . . . 25
  - 2.5.3 Linear transformation of the Multivariate Normal Distribution . . . 26

3 Principal Components Analysis (PCA) **27**
- 3.1 Finding the first principal component . . . 29
- 3.2 Finding the second (and subsequent) principal components . 32
- 3.3 Plotting the principal components . . . 34

3.4 Biplots 37
3.5 Variance decomposition and eigenspectrum 39
3.6 Using the covariance matrix or the correlation matrix 43
3.6.1 EU indicators dataset 43
3.7 PCA via the Singular Value Decomposition 46
3.8 PCA as minimizing reconstruction error 47
3.8.1 Using PCA for compression 50

4 Clustering Methods **51**
4.1 K-means clustering 51
4.1.1 Small example 62

1 Introduction

The first 10 lectures of the *Prelims Statistics and Data Analysis* course introduced the concept of likelihood for a probabilistic model, leading up to linear regression with several explanatory variables (multiple linear regression).

Linear regression is an example of **supervised learning** where we are interested in modelling the relationship between a set of p variables (or features) $x_1, \ldots, x_p$, measured on n observations, and a response variable Y, measured on those same n observations. For example,

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j + \epsilon, \qquad \epsilon \sim N(0, \sigma^2).$$

Interest lies in identifying which of the x_j's are important parts of the model, and also building a model that is able to make accurate predictions of Y using $x_1, \ldots, x_p$.

However, many methods in Statistics and Machine Learning are focused on **unsupervised learning**. This can broadly be described as looking for patterns and structure in datasets, that are often large and high-dimensional. In this setting we *only* have a set of variables $X_1, \ldots, X_p$, measured on n observations. Since we do not have a response variable Y, we are not interested in prediction.

The goal of unsupervised learning is to discover interesting things about the variables. Relevant questions include

1. Can we find a way to visualise the data that is informative?
2. Can we compress the dataset without losing any relevant information?
3. Can we find separate subgroups (or clusters) of observations that describe the structure of the dataset?

Unsupervised learning can be more challenging than supervised learning, since the goal is more subjective than prediction of a response variable. Often unsupervised learning is performed as part of an **exploratory data analysis**, where we seek to summarize the main characteristics of a dataset, often using visual methods.

Techniques of statistical learning (both supervised and unsupervised) are of growing importance in a number of fields. Statistical learning is closely related to the disciplines of **machine learning** and **artificial intelligence (AI)**. The video below looks at the ground-breaking development of AlphaGo.

https://www.youtube.com/watch?v=SUbqykXVx0A

Massive amounts of data are being collected in almost all walks of life. Financial institutions, businesses, governments, hospitals, and universities are all interested in utilising and making sense of data they collect.

Many mathematics students will go on to work in careers that involve carrying out or interpreting analysis of data of some form or other.

Thus, a good knowledge of statistical and machine learning techniques is a highly valuable skill set.

1.1 Using R or Python for data analysis

The main aim of this course is to teach some of the theory behind unsupervised learning. However, practical computing skills of carrying out such an analysis are essential. Therefore, exercise sheets include practical questions that will allow students to try basic examples of data visualisation and apply some of the techniques from the course to real and simulated datasets. These exercises can be carried out using either R or Python, depending upon the preference of individual tutors.

1.2 Motivating examples

1.2.1 Single cell genomics

In 2014 a study reported in the journal *Nature Biotechnology* (Pollen et al. *Nature Biotechnology* 32, 1053—1058 `https://www.nature.com/articles/nbt.2967`) collected gene expression measurements at 8,686 genes from 300 cells. One way to think about this dataset is that they measured how 'active' the gene was in each cell.

Figure 1 shows an image of the dataset where each row is one of the cells and each column is one of the genes. Viewing the data in this way it is very difficult to see any structure in the dataset. In this course we will learn about the method of Principal Components Analysis (that builds on Prelims Linear Algebra) which will allow us to find low-dimensional representations of the dataset that uncover underlying structure. For example, Figure 2 shows the "best" 2D representation of the data, and Figure 3 shows the "best" 3D representation of the data (see also the file movie.gif on the course website:

`https://courses.maths.ox.ac.uk/course/view.php?id=6033`

for a rotating version of Figure 4). What we mean by "best" here will be more clearly defined later in the course. Both of these images show structure within the dataset: some samples appear to cluster together in clear groups.

Once we have found a low-dimensional representation of the dataset that shows groupings visually, a subsequent question is then whether we can use a statistical method to infer the groupings or clusters. In other words, whether we can find a way of labelling the points into groups in a principled away. This is known as **clustering**.

Figure 4 shows the results of applying a clustering method known as K-means to the single cell dataset. You can see that each point has been labelled with a different colour.

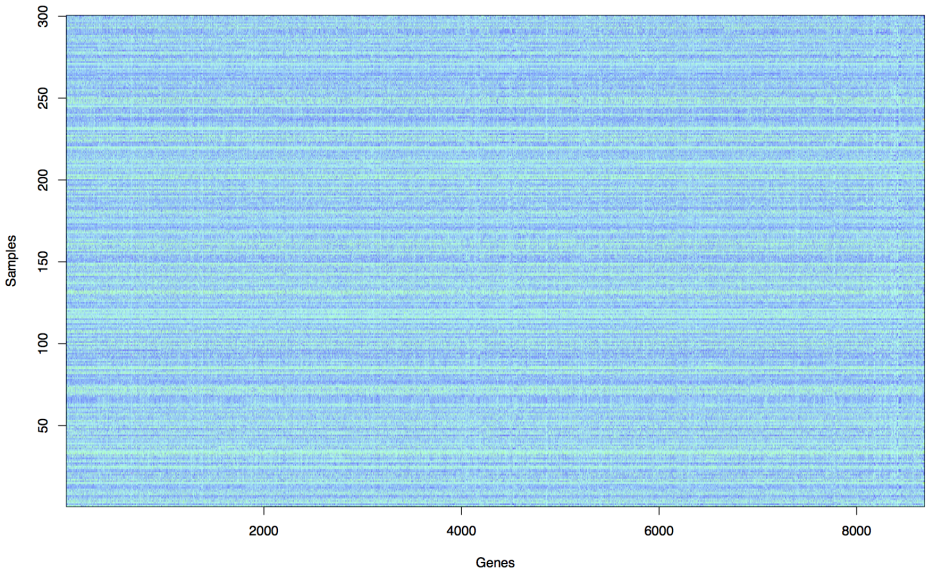


Figure 1: Single Cell dataset : gene expression measurements on 300 cells (rows) at 8,686 genes (columns).

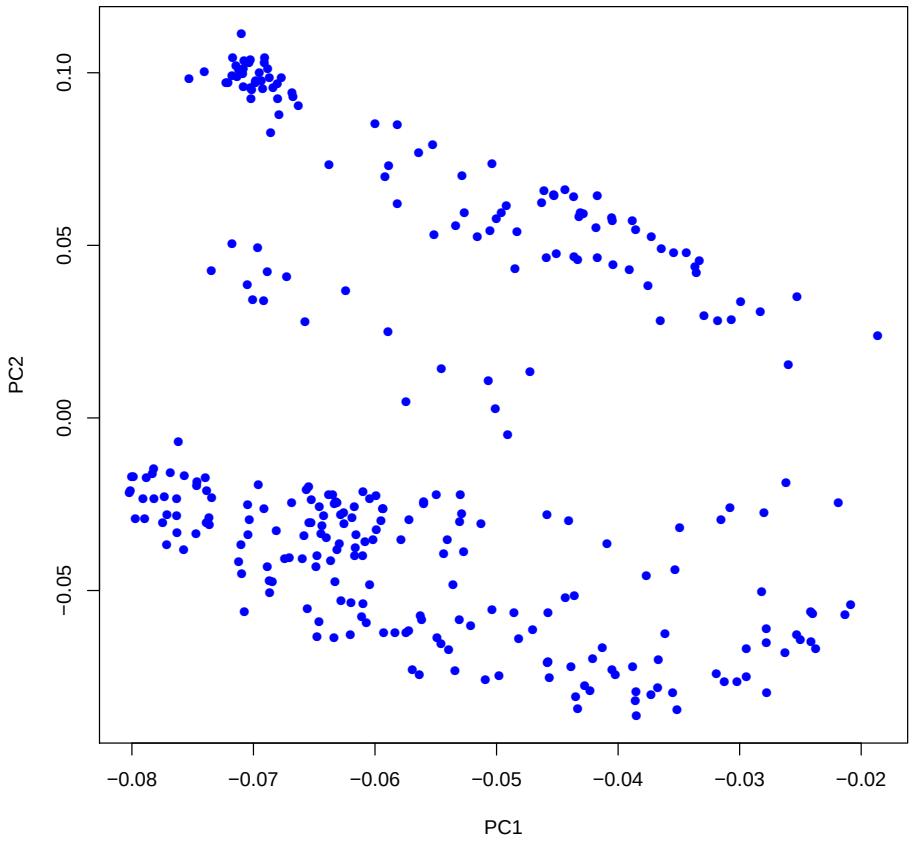


Figure 2: Plot of 1st and 2nd Principal Components for the Single Cell Genomics dataset.

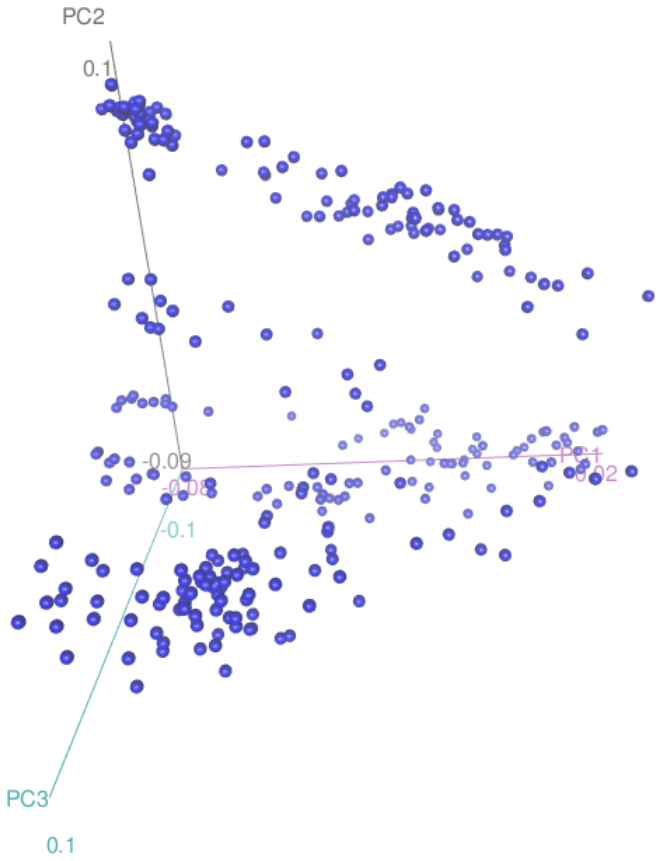


Figure 3: 3D plot of 1st, 2nd and 3rd Principal Components for the Single Cell Genomics dataset.

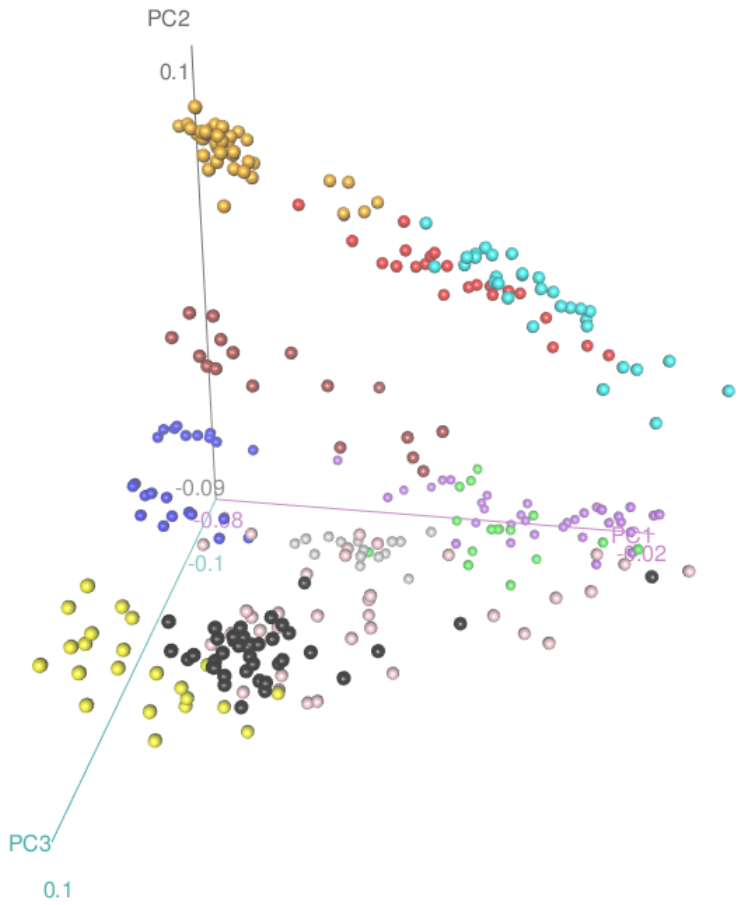


Figure 4: 3D plot of 1st, 2nd and 3rd Principal Components for the Single Cell Genomics dataset, with colouring given by the k-means clustering.

1.2.2 Food consumption

Consider the following DEFRA data showing the consumption in grams (per person, per week) of 17 different types of foodstuff measured and averaged in the four nations of the United Kingdom in 1997. We shall say that the 17 food types are the variables and the 4 nations are the observations. Looking at the data in Table 1, it is hard to spot obvious patterns.

Figure 5 shows a 2D projection of the data (i.e. the first and second principal components), and we see that Northern Ireland a major outlier. Once we go back and look at the data in the table, this makes sense: the Northern Irish eat much more grams of fresh potatoes and much fewer of fresh fruits, cheese, fish and alcoholic drinks.

	England	Wales	Scotland	N.Ireland
Cheese	105	103	103	66
Carcass meat	245	227	242	267
Other meat	685	803	750	586
Fish	147	160	122	93
Fats and oils	193	235	184	209
Sugars	156	175	147	139
Fresh potatoes	720	874	566	1033
Fresh Veg	253	265	171	143
Other Veg	488	570	418	355
Processed potatoes	198	203	220	187
Processed Veg	360	365	337	334
Fresh fruit	1102	1137	957	674
Cereals	1472	1582	1462	1494
Beverages	57	73	53	47
Soft drinks	1374	1256	1572	1506
Alcoholic drinks	375	475	458	135
Confectionery	54	64	62	41

Table 1: DEFRA data showing the consumption in grams (per person, per week) of 17 different types of foodstuff measured and averaged in the four nations of the United Kingdom in 1997.

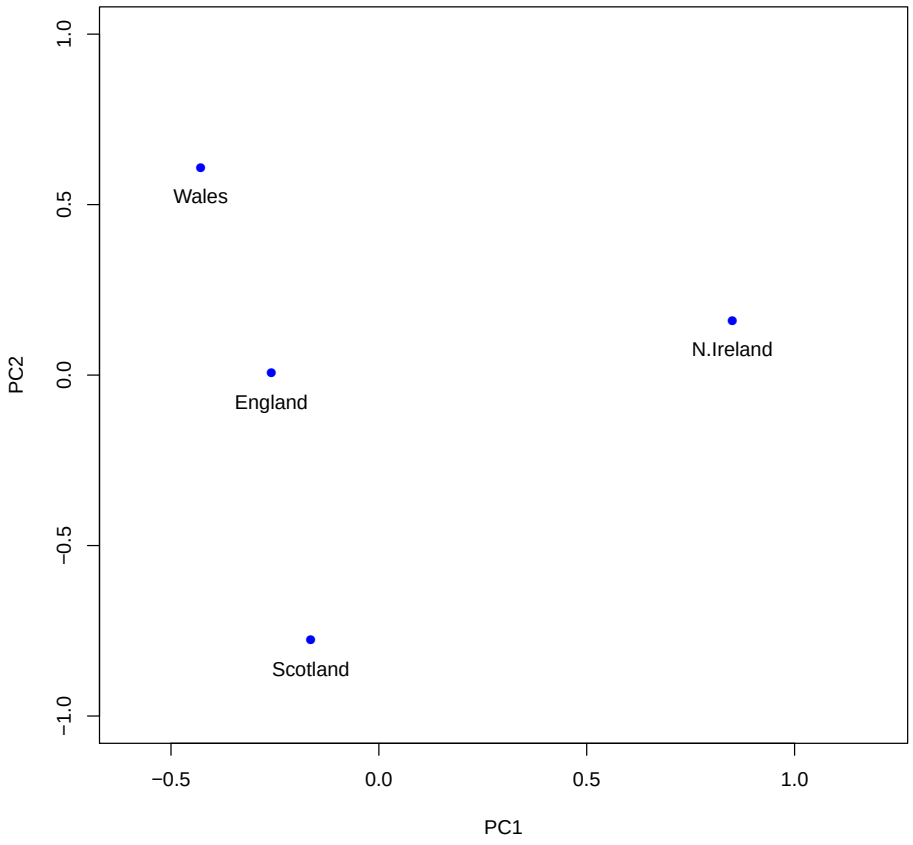


Figure 5: Plot of 1st and 2nd Principal Components for UK food dataset.

1.2.3 Finding structure in genetic datasets

Novembre et al. (*Nature* 2008 `https://www.nature.com/articles/nature07331`) analysed genetic data from 3,000 individuals at ∼500,000 positions in the genome. The individuals were collected from different countries from around Europe as part of the Population Reference Sample (POPRES) project. Before the study it was

"not clear to what extent populations within continental regions exist as discrete genetic clusters versus as a genetic continuum, nor how precisely one can assign an individual to a geographic location on the basis of their genetic information alone."

The question of interest was to see how much structure was present in the dataset and to assess the similarities and differences between individuals from different populations. Figure 6 shows the 2D projection of the dataset. Each point on the plot is an individual and points are coloured in the plot according to which country the individual comes from. What is striking about this plot is how well the arrangement of points corresponds to the geographic locations of the samples. The plot was constructed just using the genetic data, without any knowledge of the geographic locations, yet the plot is able to uncover a map of where samples came from.

1.3 Data matrices and notation

We will assume that we have collected a dataset that consists of p variables on n observations, and that this data can be represented in an $n \times p$ matrix, denoted $\boldsymbol{X}$, where x_{ij} is the observed value of the jth variable for the ith observation.

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{np} \end{bmatrix}.$$

We will refer to $\boldsymbol{X}$ as the **data matrix**. We use bold upper case for matrices in these notes.

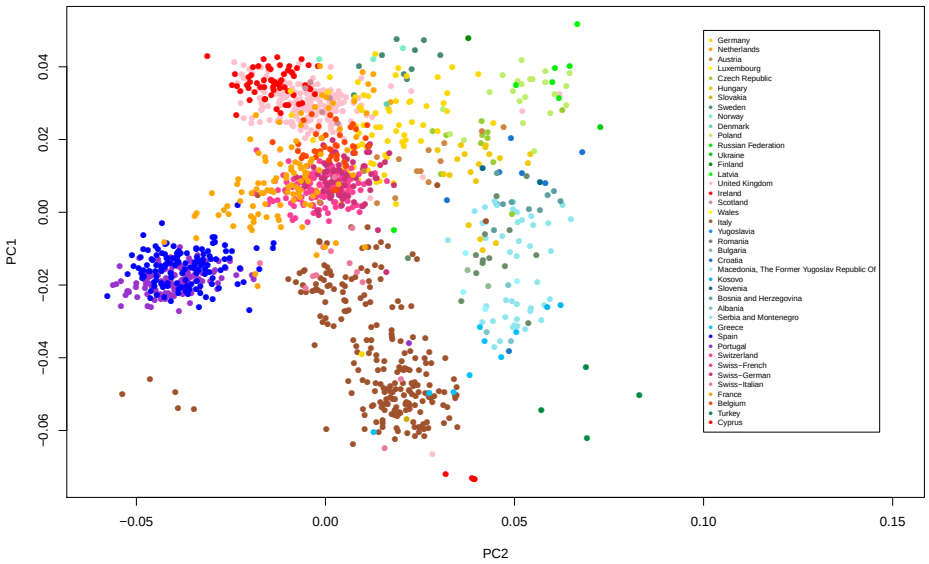


Figure 6: Plot of 1st and 2nd Principal Components for POPRES dataset

The variables (columns) are also sometimes referred to as features, attributes or dimensions. The observations (rows) are also sometimes referred to as items, individuals or samples.

We will often be interested in the rows of $\boldsymbol{X}$, which are $x_1^T, x_2^T, \ldots, x_n^T$. Here x_i is a vector of length p, containing the p variable measurements for the ith observation. That is

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix} = (x_{i1}, \ldots, x_{ip})^T$$

Vectors will be treated as column vectors by default. We assume that $x_1, \ldots, x_n$ are independent and identically distributed samples of a random vector $X = (X_1, \ldots, X_p)^T$ over $\mathbb{R}^p$.

2 Exploratory data analysis and visualising datasets

A key first step in many data analysis task is to carry out an **exploratory data analysis**. If the dataset is stored in an $n \times p$ data matrix $\boldsymbol{X}$ then we can look at data summaries and plots of various aspects of the data to help us uncover the properties of the dataset. To illustrate some simple plot types will use the 'famous' Crabs dataset.

2.1 Crabs dataset

Campbell and Mahon (*Australian Journal of Zoology*, 1974 `https://www.publish.csiro.au/ZO/ZO9740417`) studied rock crabs of the genus leptograpsus. One species, L. variegatus, had been split into two new species according to their colour: orange and blue. Preserved specimens lose their colour, so it was hoped that morphological differences would enable museum material to be classified. Data are available on 50 specimens of each sex of each species, so 200 in total. Each specimen has measurements on:

- the width of the frontal lobe (FL),
- the rear width (RW),
- the length along the carapace midline (CL),
- the maximum width (CW) of the carapace,
- the body depth (BD) in mm.

So the data matrix $\boldsymbol{X}$ has dimensions 200×5. In addition to body size measurements we have recorded variables colour/species and sex (we will later view these as labels, but will ignore for now).

2.2 Histograms

A histogram is one of the simplest ways of visualising the data from a single variable. The range of the variable is divided into bins and the frequency of observations in each bin is plotted as a bar with height proportional to the frequency. Figure 7 shows histograms of the five crab measurements.

2.3 Boxplots

A Box Plot (sometimes called a Box-and-Whisker Plot) is a relatively sophisticated plot that summarises the distribution of a given variable. These

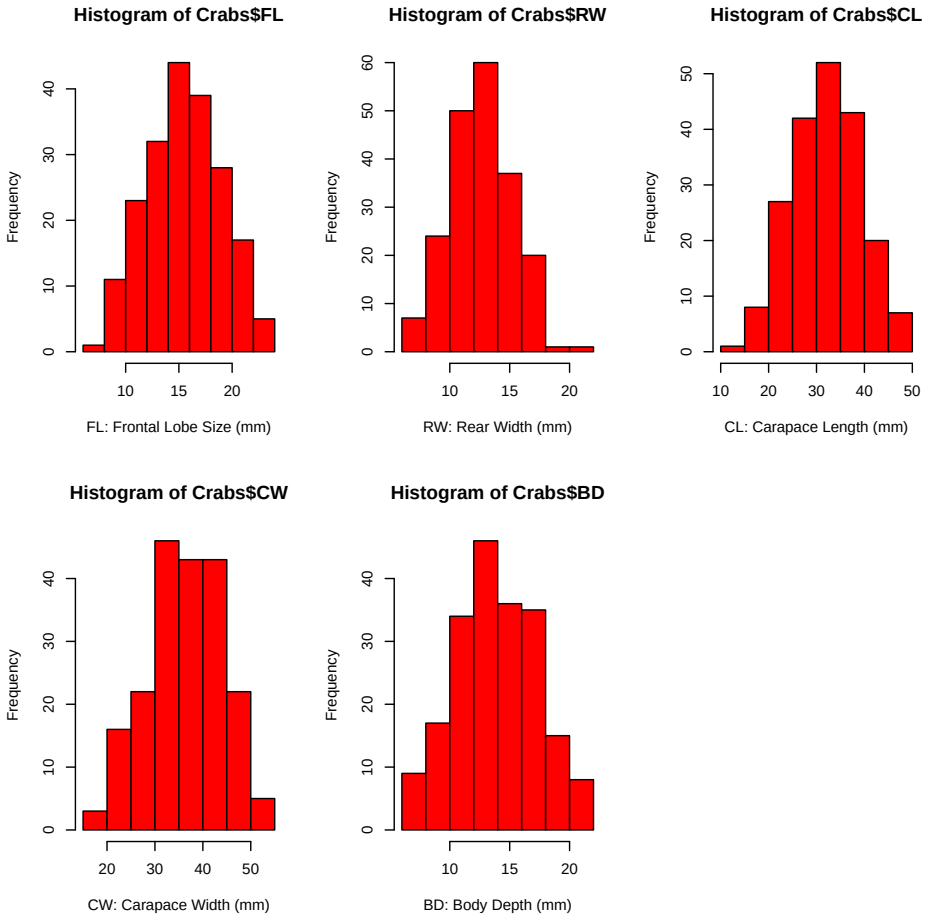


Figure 7: Histograms of the Crabs dataset

plots include the following summary statistics

Median - the 'middle' value i.e. the value for which 50% of the data fall below when arranged in numerical order.

1st quartile - the 25% value i.e. the value for which 25% of the data fall below when arranged in numerical order.

3rd quartile - the 75% value i.e. the value for which 75% of the data fall below when arranged in numerical order.

Inter Quartile Range (IQR) - the difference between the 1st and 3rd quartiles. This is a measure of 'spread' within the dataset.

A box plot consists of three main parts (shown in Figure 8)

1 A box that covers the middle 50% of the data i.e. the IQR. The edges of the box are the 1st and 3rd quartiles. A line is drawn in the box at the median value.

2 Whiskers that extend out from the box to indicate how far the data extend either side of the box. The whiskers should extend no further than α times the length of the box, i.e. the maximum length of a whisker is α times the IQR. A commonly used value of α is 1.5.

3 All points that lie outside the whiskers are plotted individually as outlying observations.

Plotting boxplots side by side allows comparisons to be made between variables. Figure 9 shows boxplots of the 5 variables in the Crabs dataset.

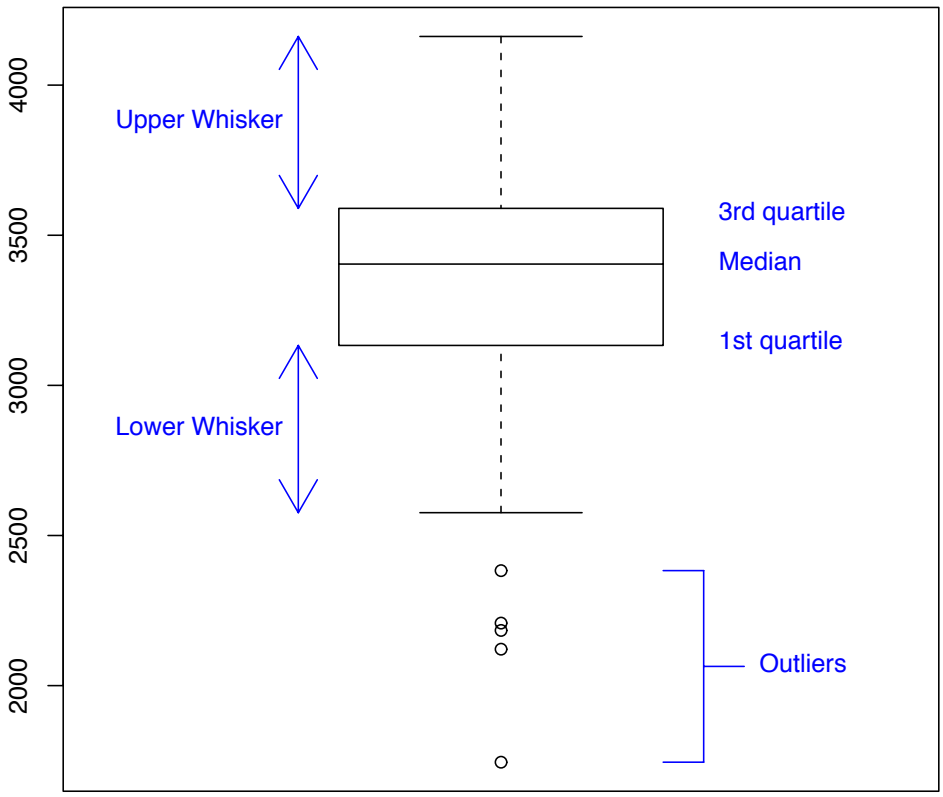


Figure 8: A Box Plot labelled with the main features of the plot.

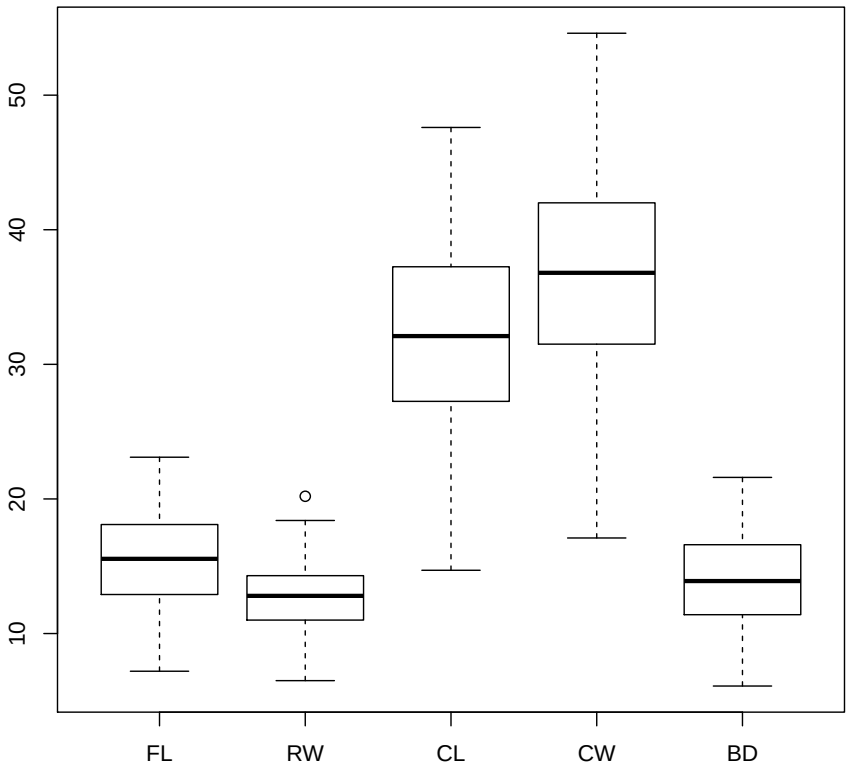


Figure 9: Box plots of the Crabs dataset showing similarities and differences between the 5 variables.

2.4 Pairs plots

Plotting pairs of variables together in a scatter plot can be helpful to see how variables co-vary (see Figure 10). However, if p is large this can be impractical as the number of pairs of variables will be $\binom{p}{2}$. We could also consider plotting triples of variables and tools exist to do that interactively.

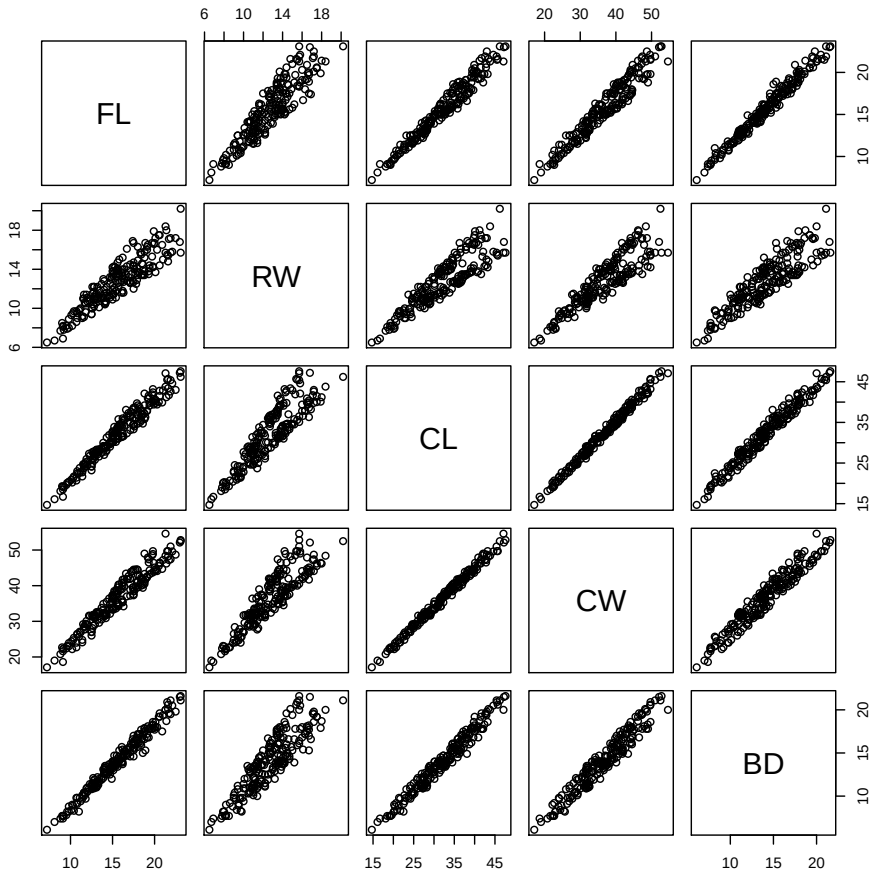


Figure 10: Pairs plots of the Crabs dataset

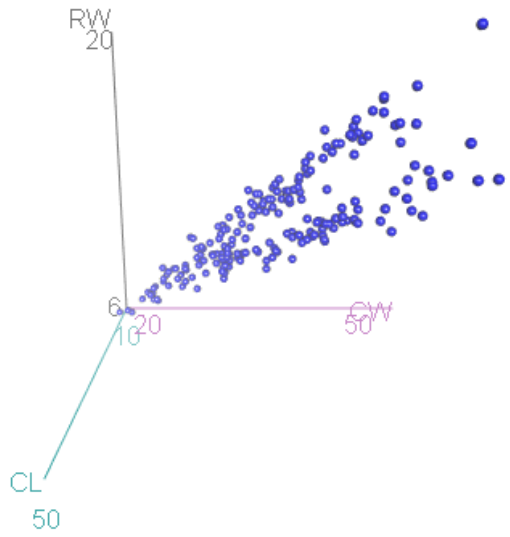


Figure 11: Screen shot from 3D interactive plotting of the Crabs variables CW, RW and CL

2.5 The Multivariate Normal Distribution

To build good models of datasets consisting of several measured variables we need probability models of multiple variables. We have seen this type of model briefly in Prelims Probability in the section on Joint Distributions. One of the simplest and most widely used models for multiple continuous random variables is the **multivariate normal distribution** (MVN).

We have seen before that the *univariate* normal distribution has two scalar parameters μ and σ^2, and we use the notation $X \sim N(\mu, \sigma^2)$. The parameter μ denotes the mean of the distribution and σ^2 denotes the variance. The pdf of the univariate normal distribution is

$$f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \quad -\infty < x < \infty$$

The multivariate normal distribution is the generalisation of the univariate normal distribution to higher dimensions. The MVN is a distribution for a p-dimensional random column vector $X = (X_1, \ldots, X_p)^T$ that allows for non-zero correlations to exist between the elements of the vector. As such, it can be a useful distribution for modelling data that consist of multiple variables measured on the same items or observations that may (or may not) be correlated.

If $X = (X_1, \ldots, X_p)^T$ is a p-dimensional random column vector then we say X has a multivariate normal distribution with mean p-column vector $\mu = (\mu_1, \ldots, \mu_p)^T$ and $p \times p$ covariance matrix Σ, if the joint pdf of X is

$$f(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\mathbf{\Sigma}^{-1}(x-\mu)\right), \quad x \in \mathbb{R}^p$$

We use the notation $X \sim N_p(\mu, \mathbf{\Sigma})$. When $p = 1$ this reduces to the univariate normal distribution. The multivariate normal distribution is sometimes referred to as the multivariate Gaussian distribution.

The interpretation of the parameters is as follows

$$\begin{aligned} \mathbb{E}[X_j] &= \mu_j \qquad \text{for } j \in 1, \ldots, p \\ \text{var}\,(X_j) &= \mathbf{\Sigma}_{jj} \qquad \text{for } j \in 1, \ldots, p \\ \text{cov}\,(X_j, X_k) &= \mathbf{\Sigma}_{jk} \qquad \text{for } j \neq k \in 1, \ldots, p \end{aligned}$$

The correlation between two variables X_i and X_j is defined as follow

$$\text{cor}\,(X_j, X_k) = \frac{\text{cov}\,(X_j, X_k)}{\sqrt{\text{var}\,(X_j)\,\text{var}\,(X_k)}} \in [-1, 1]$$

so if $X \sim N_p(\mu, \mathbf{\Sigma})$ then

$$\text{cor}\,(X_j, X_k) = \frac{\mathbf{\Sigma}_{jk}}{\sqrt{\mathbf{\Sigma}_{jj}\mathbf{\Sigma}_{kk}}}$$

2.5.1 Example : the Bivariate normal distribution ($p = 2$)

When $p = 2$ we have $X = (X_1, X_2)^T \sim N_2(\mu, \mathbf{\Sigma})$. A special case is when $\mu = (0, 0)^T$ and the variances of X_1 and X_2 are both equal to 1, so that $\text{cor}\,(X_1, X_2) = \mathbf{\Sigma}_{ij} = \rho$. In this case the pdf of X reduces to

$$f(\boldsymbol{x}) = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{x_1^2 - 2\rho x_1 x_2 + x_2^2}{2(1-\rho^2)}\right)$$

Figure 12 shows the density of such a bivariate normal with $\rho = 0.7$ and Figure 13 shows contour plots for several multivariate normal densities with different values of ρ. Figure 14 shows a 2D density together with a sample of 200 data points **simulated** from the same distribution.

Simulation is a technique for generating pseudo-random values that have a particular distribution. Simulation is widely used in all areas of Statistics, Machine Learning and Data Analysis. We recommend the Part A course in Simulation and Statistical Programming to learn more about this subject.

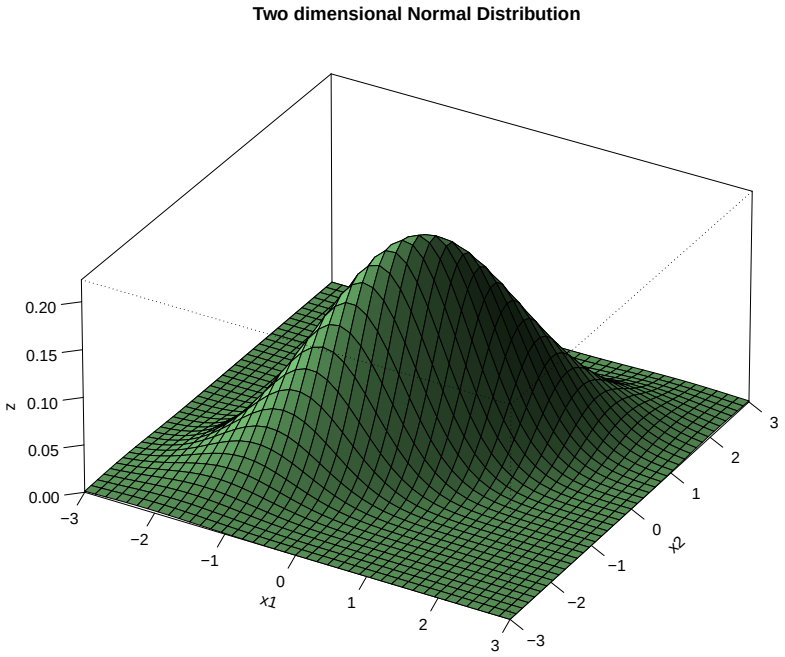


Figure 12: 2D multivariate normal density for $\mu = (0, 0)^T$, $\mathbf{\Sigma}_{11} = \mathbf{\Sigma}_{22} = 1$ and $\mathbf{\Sigma}_{12} = 0.7$

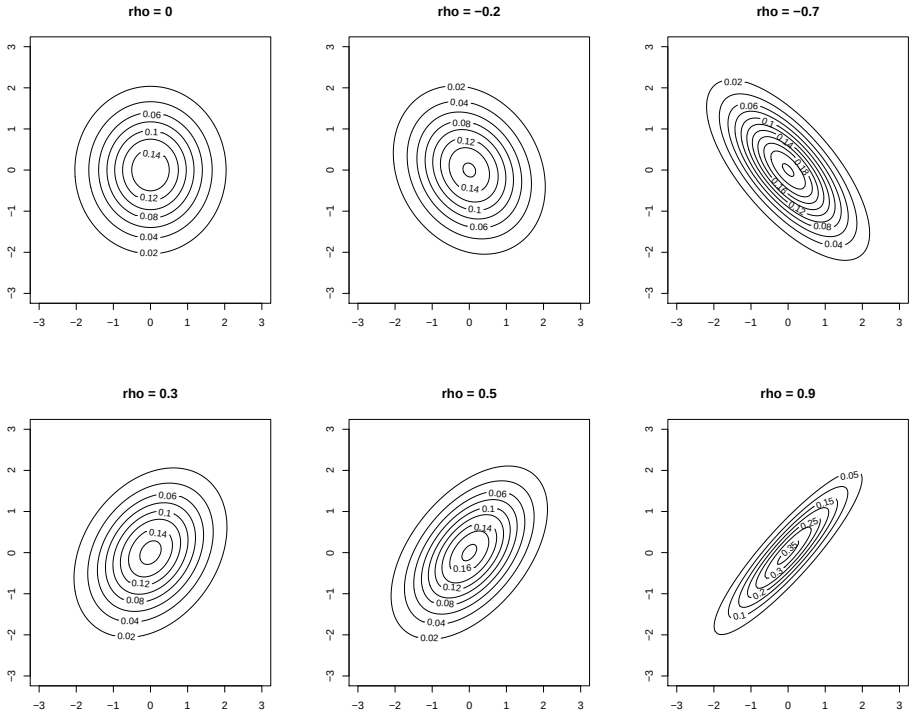


Figure 13: Contour plots for 2D multivariate normal densities with $\mu = (0, 0)^T$, $\mathbf{\Sigma}_{11} = \mathbf{\Sigma}_{22} = 1$ and $\mathbf{\Sigma}_{12} = \rho$ for different values of ρ

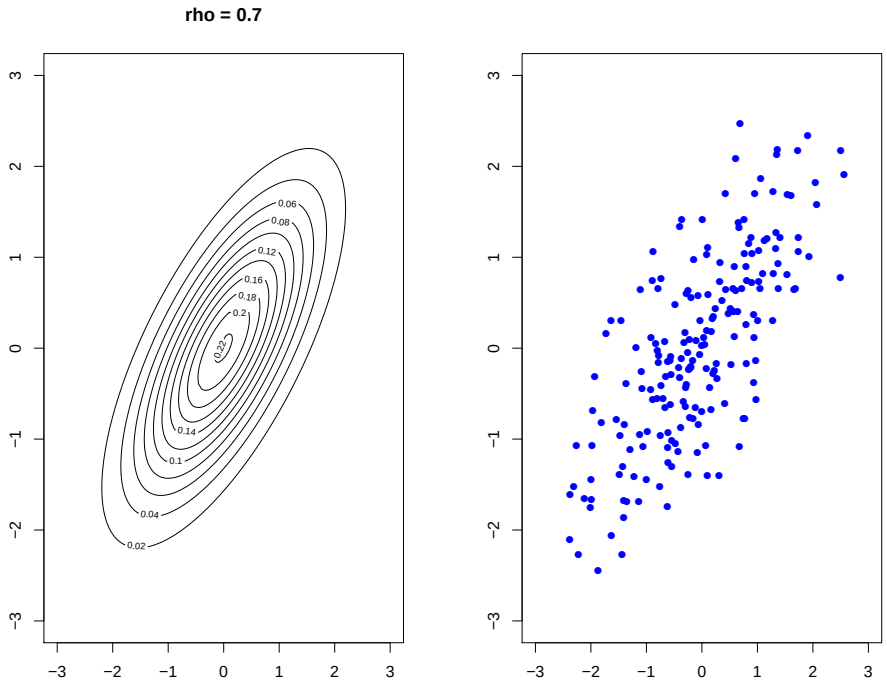


Figure 14: Contour plots for a 2D multivariate normal density with $\mu = (0,0)^T$, $\mathbf{\Sigma}_{11} = \mathbf{\Sigma}_{22} = 1$ and $\mathbf{\Sigma}_{12} = 0.7$ and a sample of 200 data points simulated from this distribution.

2.5.2 Estimating parameters for the multivariate normal distribution

Often given a sample of real data, we will want to find the MVN that best fits the data. Given a sample of n observations from a $N_p(\mu, \mathbf{\Sigma})$ distribution, it can be shown (see Exercises) that the maximum likelihood estimates of μ and $\mathbf{\Sigma}$ are

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and

$$\hat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

Note that $(x_i - \hat{\mu})$ is a $p \times 1$ column vector and so $(x_i - \hat{\mu})^T$ is a $1 \times p$ row vector and so $\hat{\mathbf{\Sigma}}$ is an $p \times p$ matrix.

This estimator of $\mathbf{\Sigma}$ is biased (remember : not all maximum likelihood estimators are unbiased). An unbiased estimator for $\mathbf{\Sigma}$ is

$$\boldsymbol{S} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

We refer to $\boldsymbol{S}$ as the **sample covariance matrix**. If we assume that the data matrix $\boldsymbol{X}$ has been 'mean centered' by subtraction of $\hat{\mu}^T$ from each row, then using matrix notation we can write

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X}$$

Note that we don't have to assume that the data are generated from a MVN distribution in order to calculate the statistic $\boldsymbol{S}$. It is a useful summary of the pairwise covariances in any collection of variables. On the Crabs data the sample covariance matrix is

$$\boldsymbol{S} = \begin{array}{c|ccccc} & FL & RW & CL & CW & BD \\ \hline FL & 12.21 & 8.15 & 24.35 & 26.55 & 11.82 \\ RW & 8.15 & 6.62 & 16.35 & 18.23 & 7.83 \\ CL & 24.35 & 16.35 & 50.67 & 55.76 & 23.97 \\ CW & 26.55 & 18.23 & 55.76 & 61.96 & 26.09 \\ BD & 11.82 & 7.83 & 23.97 & 26.09 & 11.72 \end{array}.$$

Sometimes it is useful to work with the **sample correlation matrix**, denoted $\boldsymbol{R}$, which has entries

$$\boldsymbol{R}_{jk} = \frac{\boldsymbol{S}_{jk}}{\sqrt{\boldsymbol{S}_{jj}\boldsymbol{S}_{kk}}}$$

On the Crabs data the sample correlation matrix is

$$\boldsymbol{R} = \begin{array}{c|ccccc} & FL & RW & CL & CW & BD \\ \hline FL & 1.00 & 0.91 & 0.98 & 0.96 & 0.99 \\ RW & 0.91 & 1.00 & 0.89 & 0.90 & 0.89 \\ CL & 0.98 & 0.89 & 1.00 & 1.00 & 0.98 \\ CW & 0.96 & 0.90 & 1.00 & 1.00 & 0.97 \\ BD & 0.99 & 0.89 & 0.98 & 0.97 & 1.00 \end{array}.$$

The high levels of correlation between all pairs of variables can be seen visually in Figure 10.

2.5.3 Linear transformation of the Multivariate Normal Distribution

If $X = (X_1, \ldots, X_p)^T$ is a p-dimensional random column vector such that $X \sim N_p(\mu, \boldsymbol{\Sigma})$ and $\boldsymbol{B}$ is a $m \times p$ matrix then it can be shown (see Exercises) that the mean and covariance matrix of the m-dimensional random column vector $Y = \boldsymbol{B}X$ are given by

$$\begin{aligned} \mathbb{E}[Y] &= \boldsymbol{B}\mu \\ \operatorname{cov}(Y) &= \boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{B}^T \end{aligned}$$

A further useful result (which we will not prove in this course since it is covered in Part A) is that the distribution of $Y = \boldsymbol{B}X$ is normal and given by

$$Y \sim N_m(\boldsymbol{B}\mu, \boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{B}^T)$$

3 Principal Components Analysis (PCA)

If we have a large number of p variables collected on a set of n observations it can be hard to visualise and summarize the dataset.

Principal components analysis (PCA) is a useful method that can allow us to summarize the dataset with a small number of representative variables or dimensions. In other words, we would like to find a low-dimensional representation of the data that captures as much of the information in the dataset as possible. For instance, if we can find a two-dimensional (2D) representation of the data that captures most of the information, then we can plot the observations in this 2D space, and maybe carry out further analysis within this 2D space. PCA provides a tool to do this. The mathematics underlying PCA build on ideas and results that we have learned in the Prelims Linear Algebra courses.

The question then is 'how do we define a representative variable or dimension?'

Answering this question using PCA involves the following key ideas :

1. Each of the dimensions found by PCA is a linear combination of the variables.

2. PCA seeks a small number of dimensions that are as *interesting* as possible, where interesting is measured by how variable the observations are along those dimensions. These dimensions are referred to as **principal components**, or PCs for short.

Why is variability (or variance) a good metric for determining interestingness?

In situations where the data consist of separated clusters of observations in p-dimensional space, directions that maximize variance can provide good separation of the clusters. This is illustrated in Figure 15 which shows an example of a dataset in 2 dimensions with 2 clear clusters of points. When the data points are projected onto the axis labelled A this results in clear separation of the points and a variance of 12.98. When the points are projected onto the orthogonal axis labelled B this results in no separation of the points and a much lower variance of 1.80. So in this example variance seems

to be a good way to choose a projection that separates the two clusters.

Another way to think about this is that we have found a rotation of the data points to maximize the variance. A rotation is a set of orthogonal projections. Figure 16 shows the data points before rotation (left) and after rotation (right). The points in the two clusters are well separated on the new x-axis.

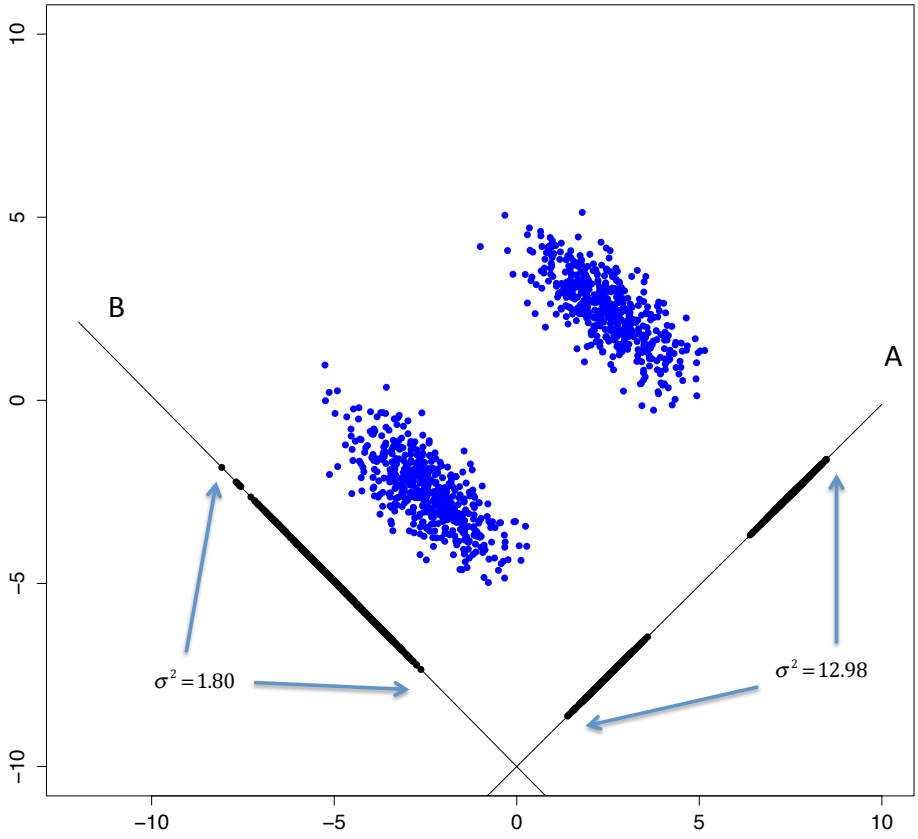


Figure 15: Example showing the variance of points projected on two (orthogonal) projections (A) and (B). The projection (A) which separates the points well has the highest variance.

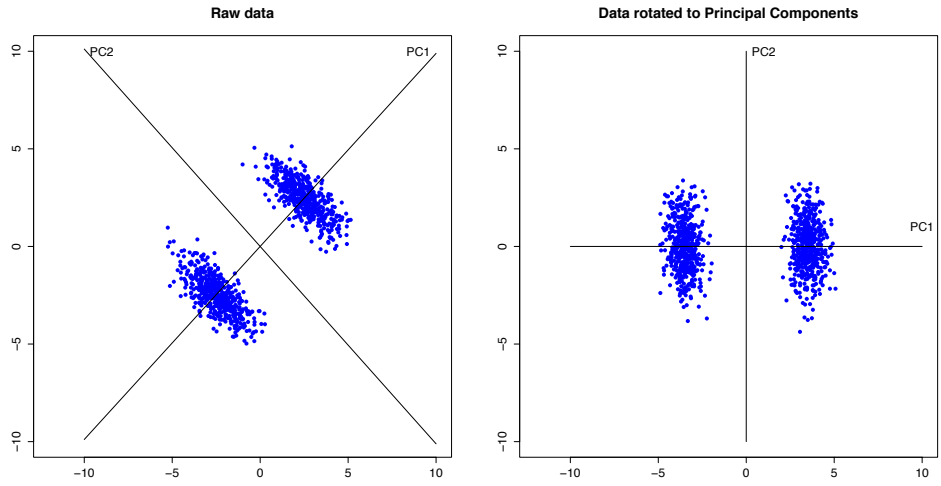


Figure 16: Example showing the variance of points projected on two (orthogonal) projections (A) and (B). The projection (A) which separates the points well has the highest variance.

3.1 Finding the first principal component

The PCA components are found one at a time. The first principal component is a linear combination of the set of p variables, denoted Z_1, such that

$$Z_1 = \alpha_{11}X_1 + \alpha_{12}X_2 + \ldots + \alpha_{1p}X_p$$

We can write

$$Z_1 = \alpha_1^T X$$

where $\alpha_1 = (\alpha_{11}, \alpha_{12}, \ldots, \alpha_{1p})^T$ and $X = (X_1, \ldots, X_p)^T$ are both column vectors. Then it can shown (Exercise Sheet 1) that

$$\text{var}\,(Z_1) = \alpha_1^T \mathbf{\Sigma} \alpha_1$$

where $\mathbf{\Sigma}$ is the $p \times p$ covariance matrix of the random variable X.

There are two problems with seeking to maximize this variance

1. Since we only have a sample of data from each of the p-variables we do not know the true covariance $\mathbf{\Sigma}$, but we can use the sample covariance matrix $\boldsymbol{S}$ instead.

2. This variance is unbounded as the α_i's increase, so we need to add a constraint on α such that

$$\sum_{j=1}^{p} \alpha_{1j}^2 = \alpha_1^T \alpha_1 = 1$$

Making these changes results in the constrained maximization problem

$$\max_{\alpha_1} \alpha_1^T \boldsymbol{S} \alpha_1 \quad \text{subject to} \quad \alpha_1^T \alpha_1 = 1$$

In other words, we try to maximize the **sample variance** of the first principal component ($\alpha_1^T \boldsymbol{S} \alpha_1$) subject to the constraint $\alpha_1^T \alpha_1 = 1$. We can solve this using Lagrange Multipliers (see Prelims Introductory Calculus course for a reminder on this technique).

Let

$$\mathcal{L}(\alpha_1, \lambda_1) = \alpha_1^T \boldsymbol{S} \alpha_1 - \lambda_1(\alpha_1^T \alpha_1 - 1)$$

We then need the vector of partial derivatives of $\mathcal{L}$ with respect to the vector α_1. This is the gradient vector $\nabla_{\alpha_1} \mathcal{L}$ seen in Intro Calculus

Vector calculus results
Let a and x are p-column vectors and $\boldsymbol{B}$ a $p \times p$ symmetric matrix with rows $B_1^T, B_2^T, \ldots, B_p^T$. Then

1. Let $y = a^T x = \sum_{j=1}^{p} a_j x_j$ be a scalar function, then

$$\nabla_x y = \Big(\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \ldots, \frac{\partial y}{\partial x_p}\Big)^T = (a_1, a_2, \ldots, a_p)^T = a$$

2. Let $z = x^T \boldsymbol{B} x = \sum_{j=1}^{p} \sum_{k=1}^{p} \boldsymbol{B}_{jk} x_j x_k$ then consider

$$\nabla_x z = \Big(\frac{\partial z}{\partial x_1}, \frac{\partial z}{\partial x_2}, \ldots, \frac{\partial z}{\partial x_p}\Big)^T$$

Since $\frac{\partial z}{\partial x_j} = 2 \sum_{k=1}^{p} \boldsymbol{B}_{jk} x_k = 2 B_j^T x$ we have that

$$\nabla_x z = 2 \boldsymbol{B} x$$

Re-writing we have

$$\mathcal{L}(\alpha_1, \lambda_1) = \alpha_1^T \boldsymbol{S} \alpha_1 - \lambda_1 \alpha_1^T \boldsymbol{I}_p \alpha_1 + \lambda_1$$

where $\boldsymbol{I}_p$ denotes the $p \times p$ identity matrix.

Using these results we have that

$$\nabla_{\alpha_1} \mathcal{L}(\alpha_1, \lambda_1) = 2\boldsymbol{S}\alpha_1 - 2\lambda_1 I_p \alpha_1 = 2\boldsymbol{S}\alpha_1 - 2\lambda_1 \alpha_1$$

Setting this equal to 0 results in the eigenvalue equation

$$\boldsymbol{S}\alpha_1 = \lambda_1 \alpha_1 \tag{1}$$

which means λ_1 and α_1 will be an eigenvalue and eigenvector of S respectively. We can find the eigenvalues of $\boldsymbol{S}$ by solving the characteristic polynomial (see Linear Algebra Recap on next page)

$$\chi_S(x) = det(\boldsymbol{S} - x\boldsymbol{I}_p) = 0$$

There will be at most p roots of this equation, so we need to determine which one to choose. If we multiply equation (1) by α_1^T then we get

$$\alpha_1^T \boldsymbol{S} \alpha_1 = \lambda_1 \tag{2}$$

Thus λ_1 is the sample variance of the first principal component which we are trying to maximize, so we choose λ_1 to be the *largest* eigenvalue of $\boldsymbol{S}$, and α_1 is the corresponding eigenvector.

Recap of results from Linear Algebra II
Let V be a vector space over $\mathbb{R}$ and $T : V \to V$ be a linear transformation.

1. A vector $v \in V$ is called an eigenvector of T if $v \neq 0$ and $Tv = \lambda v$ for some $\lambda \in \mathbb{R}$. We call $\lambda \in \mathbb{R}$ an eigenvalue of T if $Tv = \lambda v$ for some nonzero $v \in V$.

2. For $\boldsymbol{A} \in \boldsymbol{M}_n(\mathbb{R})$ the characteristic polynomial of $\boldsymbol{A}$ is defined as $det(\boldsymbol{A} - x\boldsymbol{I}_n)$. For $T : V \to V$ a linear transformation, let $\boldsymbol{A}$ be the matrix for T with respect to some basis B. The characteristic polynomial of T is defined as $det(\boldsymbol{A} - x\boldsymbol{I}_n)$.

3. Let $T : V \to V$ be a linear transformation, where $\boldsymbol{A} \in \boldsymbol{M}_n(\mathbb{R})$ is the matrix of T. Then λ is an eigenvalue of T if and only if λ is a root of the characteristic polynomial $\chi_A(x) = det(\boldsymbol{A} - x\boldsymbol{I}_n)$ of $\boldsymbol{A}$.

4. A real symmetric matrix $\boldsymbol{A} \in \boldsymbol{M}_n(\mathbb{R})$ has real eigenvalues and there exists an orthonormal basis for $\mathbb{R}^n$ consisting of eigenvectors for $\boldsymbol{A}$. In other words, there exists an orthonormal real matrix $\boldsymbol{V}$ (so $\boldsymbol{V}^T = \boldsymbol{V}^{-1}$) such that
$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$$
where $\boldsymbol{D}$ is a diagonal matrix of eigenvalues.

   We refer to this as the **eigendecomposition** of $\boldsymbol{A}$.

3.2 Finding the second (and subsequent) principal components

Now consider finding the second principal component. It will also be a linear combination of the set of p variables, denoted Z_2, such that

$$Z_2 = \alpha_2^T X = \alpha_{21}X_1 + \alpha_{22}X_2 + \ldots + \alpha_{2p}X_p$$

where $\alpha_2 = (\alpha_{21}, \alpha_{22}, \ldots, \alpha_{2p})^T$. As before we need the constraint

$$\alpha_2^T \alpha_2 = 1$$

However, we must add another constraint in order to find a distinct eigenvector. As described above we wish the linear combinations of p variables to be orthogonal projections, so we add in the further constraint that

$$\alpha_1^T \alpha_2 = 0$$

and this leads to another Lagrange multipliers problem where we seek to maximize

$$\mathcal{L}(\alpha_2, \lambda_2, m) = \alpha_2^T \boldsymbol{S} \alpha_2 - \lambda_2(\alpha_2^T \alpha_2 - 1) - m\alpha_1^T \alpha_2$$

Taking partial derivatives with respect to α_2 leads to

$$\nabla_{\alpha_2} \mathcal{L}(\alpha_2, \lambda_2, m) = 2\boldsymbol{S}\alpha_2 - 2\lambda_2\alpha_2 - m\alpha_1$$

Setting equal to 0 gives

$$\boldsymbol{S}\alpha_2 - \lambda_2\alpha_2 = \frac{1}{2}m\alpha_1 \tag{3}$$

Pre-multiplying by α_1^T, remembering that $\alpha_1^T \alpha_2 = 0$ and $\alpha_1^T \alpha_1 = 1$, gives

$$\alpha_1^T \boldsymbol{S} \alpha_2 = \frac{1}{2}m$$

However, pre-multiplying (1) by α_2^T, we get

$$\alpha_2^T \boldsymbol{S} \alpha_1 = \alpha_1^T \boldsymbol{S} \alpha_2 = 0 \tag{4}$$

since $\alpha_2^T \boldsymbol{S} \alpha_1$ is a scalar and $\boldsymbol{S}$ is symmetric, which implies that $m = 0$ and equation (3) reduces to the eigenvalue equation

$$\boldsymbol{S}\alpha_2 = \lambda_2\alpha_2 \tag{5}$$

So λ_2 is also an eigenvalue of $\boldsymbol{S}$ with associated eigenvector α_2. As before we can show that λ_2 is equal to $\alpha_2^T \boldsymbol{S} \alpha_2$ which is the sample variance of the second principal component. We must also ensure that $\alpha_1^T \boldsymbol{S} \alpha_2 = 0$, so this is achieved by setting λ_2 to be the second largest eigenvalue, with α_2 being its corresponding eigenvector.

The above process can be continued for the other principal components. This results in a sequence of principal components ordered by their variance. In other words, if we consider the eigenvalue decomposition of S (see Linear Algebra recap)

$$\boldsymbol{S} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$$

where $\boldsymbol{D}$ is a diagonal matrix of eigenvalues ordered in decreasing value, and $\boldsymbol{V}$ is a $p \times p$ matrix of the corresponding eigenvectors as orthonormal columns $(v_1, \ldots, v_p)$, then we have that

$$\alpha_j = v_j \quad \text{and} \quad \lambda_j = \boldsymbol{D}_{jj} \quad \text{for} \quad j = 1, \ldots, p$$

3.3 Plotting the principal components

If $\boldsymbol{X}$ is the $n \times p$ data matrix of the n observations on p variable, $\boldsymbol{S}$ is the $p \times p$ sample covariance matrix, with ordered eigendecomposition $\boldsymbol{S} = \boldsymbol{V}DV^T$ then the data can be transformed to the p principal component directions. If we define $\boldsymbol{Z}$ to be an $n \times p$ matrix containing the transformed data, such that $\boldsymbol{Z}_{ij}$ is the value of the jth principal component for the ith observation then we have

$$\boldsymbol{Z}_{ij} = x_i^T \alpha_j = x_i^T v_j.$$

In other words we take the vector product of the ith row of $\boldsymbol{X}$ and the jth column of $\boldsymbol{V}$. The matrix $\boldsymbol{V}$ is known as the **loadings matrix**. In matrix notation we can write this as

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{V}$$

We can then plot the columns of $\boldsymbol{Z}$ against each other to visualise the data as represented by those pairs of principal components. The matrix $\boldsymbol{Z}$ is known as the **scores matrix**. The columns of this matrix contain the projections onto the principal components.

Figure 17 shows a pairs plot of the 5 PCs for the Crabs dataset. The points have been coloured according to the 4 groups Blue Male (dark blue), Blue Female (light blue), Orange Male (orange), Orange Female (yellow). From this plot we can see that PCs 2 and 3 seem to show good discrimination of these 4 groups. This is shown more clearly in Figure 18.

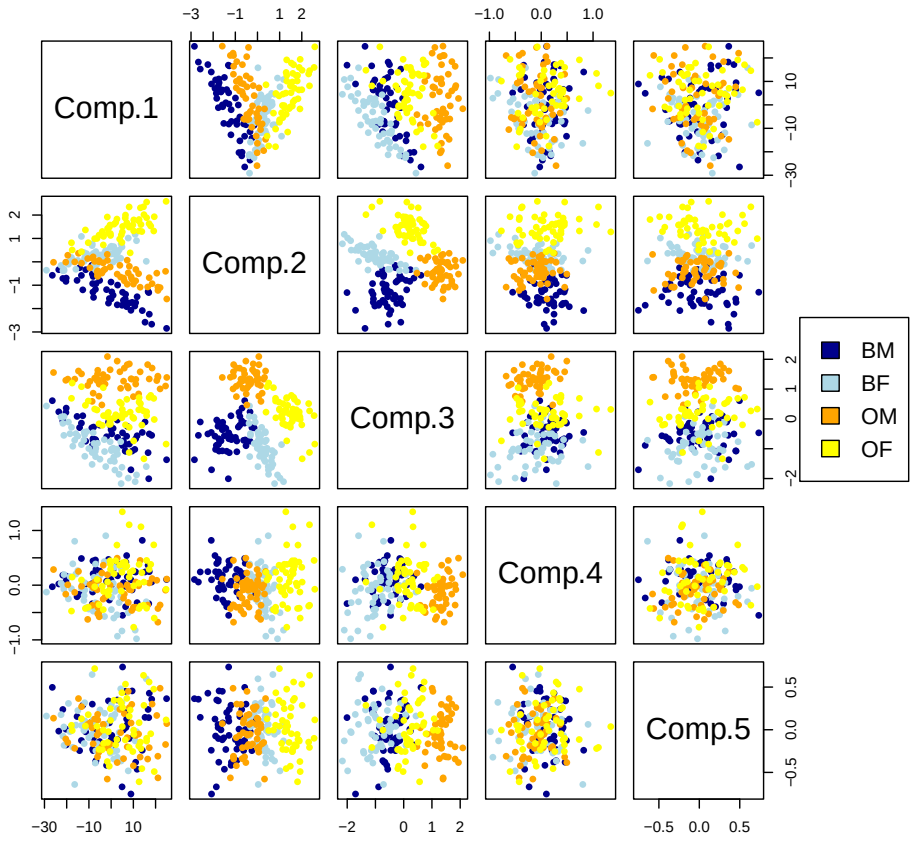


Figure 17: Pairs plot of PC components for the Crabs dataset.

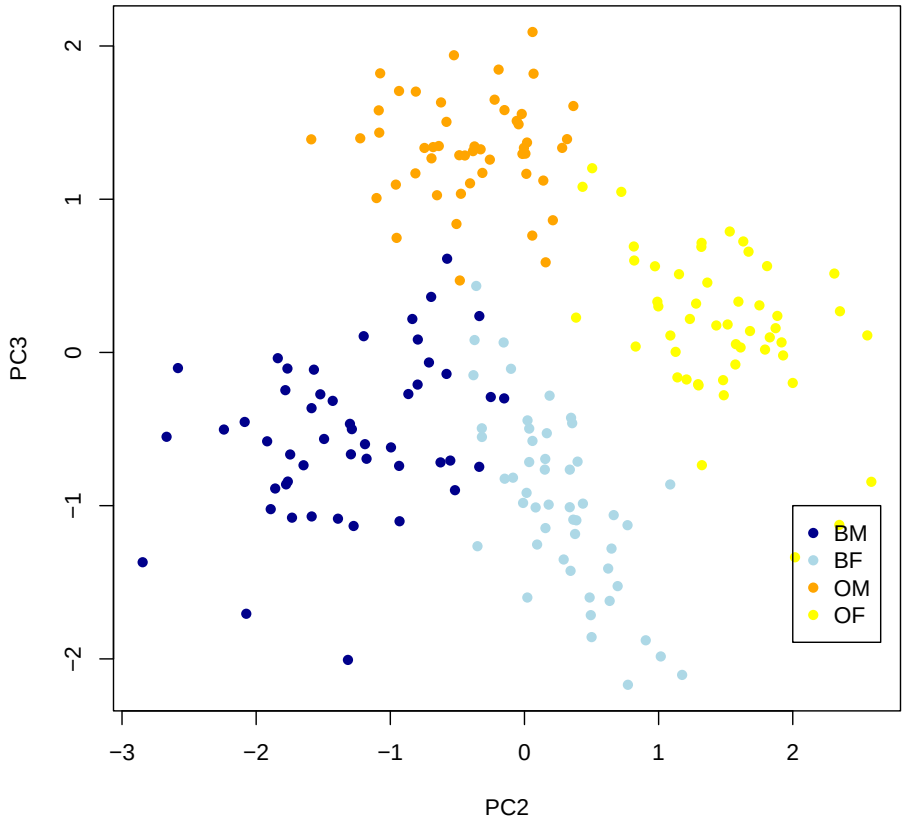


Figure 18: Plot of PCs 2 and 3 for the Crabs dataset.

3.4 Biplots

The columns of the loadings matrix $\boldsymbol{V}$ contains the linear projections for each of the PCs. It can be interesting to look at these loadings to see which variables contribute to each PC. For example, the loadings matrix for the Crabs dataset is as follows

	$PC1$	$PC2$	$PC3$	$PC4$	$PC5$
FL	0.28	0.32	0.50	0.73	0.12
RW	0.19	0.86	−0.41	−0.14	−0.14
CL	0.59	−0.19	0.17	−0.14	−0.74
CW	0.66	−0.28	−0.49	0.12	0.47
BD	0.28	0.15	0.54	−0.63	0.43

So for example, this means that the first, second and third PCs are

$$Z_1 = 0.28X_1 + 0.19X_2 + 0.59X_3 + 0.66X_4 + 0.28X_5$$

$$Z_2 = 0.32X_1 + 0.86X_2 - 0.19X_3 - 0.28X_4 + 0.15X_5$$

$$Z_3 = 0.50X_1 - 0.41X_2 + 0.17X_3 - 0.49X_4 + 0.54X_5$$

Notice how the loadings for the 1st PC are all positive. This is quite usual, especially when the units of observation are biological samples (such as crabs), where the 1st PC is essentially a linear combination of features that is measuring size. Often it is the case that this variable will account for a large amount of variance, but tends not to be able to split samples into distinct groups. It is often the PCs with a mixture of positive and negative loadings that provide the *contrast* that separates groups.

It can be useful to represent the loadings information on the plots of the PCs scores. Typically, we will plot one PC versus another. What we would like to see is which variables contribute to each PC. From the above we can see that variable FL has weight 0.32 in PC2 and -0.50 in PC3. We represent that on the PC plot using a vector (0.32, 0.50). Similarly, we can represent the contribution of variable RW on the PC plot using a vector (0.86, -0.41) etc. Often we need to scale the vectors by a constant factor to appear on the plot in a visually appealing way. Such a plot which shows the PCs scores together with vectors showing the PC loadings is called a **Biplot**.

Figure 19 shows the Biplot for PCs 2 and 3 for the Crabs dataset. It shows that PC2 is contrast between variables RW, FL, BD and CW, CL,

whereas PC3 is a contrast between variables CW, RW and CL, FL, BD. Also, PC2 separates the Orange Females from the Blue Males well, whereas PC2 separates the Blue Females from the Orange Males well.

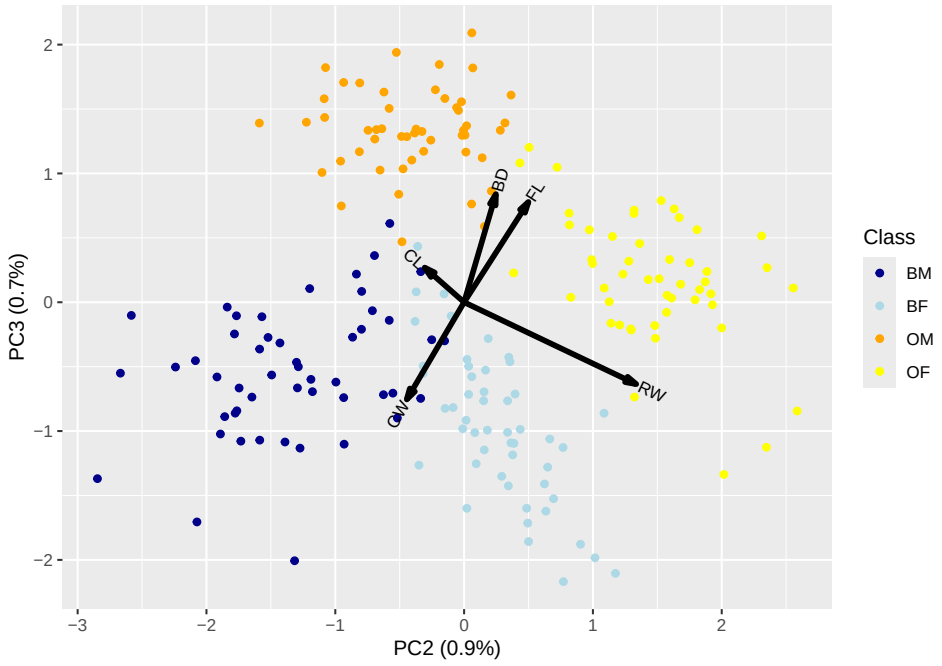


Figure 19: BiPlot of PCs 2 and 3 for the Crabs dataset.

3.5 Variance decomposition and eigenspectrum

The total amount of variance in the original data matrix $\boldsymbol{X}$ is the sum of the diagonal entries in $\boldsymbol{S}$. In other words, we have that $\widehat{\text{var}}(X_j) = \boldsymbol{S}_{jj}$ so that

$$\text{Total variance } = \sum_{j=1}^{p} \boldsymbol{S}_{jj} = tr(\boldsymbol{S})$$

but since $tr(AB) = tr(BA)$ where A and B are two $p \times p$ matrices we have that

$$\text{Total variance } = tr(\boldsymbol{S}) = tr(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T) = tr(\boldsymbol{D}\boldsymbol{V}^T\boldsymbol{V}) = tr(\boldsymbol{D})$$

since $\boldsymbol{V}$ is an orthonormal matrix of eigenvectors, so that $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$.

So we can see that the eigenvalues tell us interesting information about how important each component is within the dataset. It is common practice to plot the decreasing sequence of eigenvalues to visualise the structure in the dataset. Such plots are sometimes referred to as **eigenspectrum plots** or **variance scree plots**, and usually they are scaled so each bar is percentage of the total variance. That is we plot

$$\frac{100\boldsymbol{D}_{jj}}{tr(\boldsymbol{D})} \quad \text{for } j \in 1, \ldots, p$$

Figure 20 shows the scree plot for a simulated dataset with 3 clear clusters. The 1st PC (middle) clearly separates the groups and the scree plot (right) shows that the 1st PC accounts for almost all the variance in the dataset.

Figure 21 shows the scree plot for a simulated dataset with 4 clear clusters. The 1st and 2nd PCs (middle) clearly separate the groups and the scree plot (right) shows that the 1st and 2nd PCs account for almost all the variance in the dataset.

The scree plot is sometimes used to decide on a set of PCs to carry forward for further analysis. For example, we might choose to take the PCs that account for the top $\theta\%$ of the variance. However, care is sometimes needed here. Figure 22 shows the scree plot for the Crabs dataset and shows that the 1st PC accounts for almost all the variance. As discussed before, this PC is most likely measuring the size of the crabs.

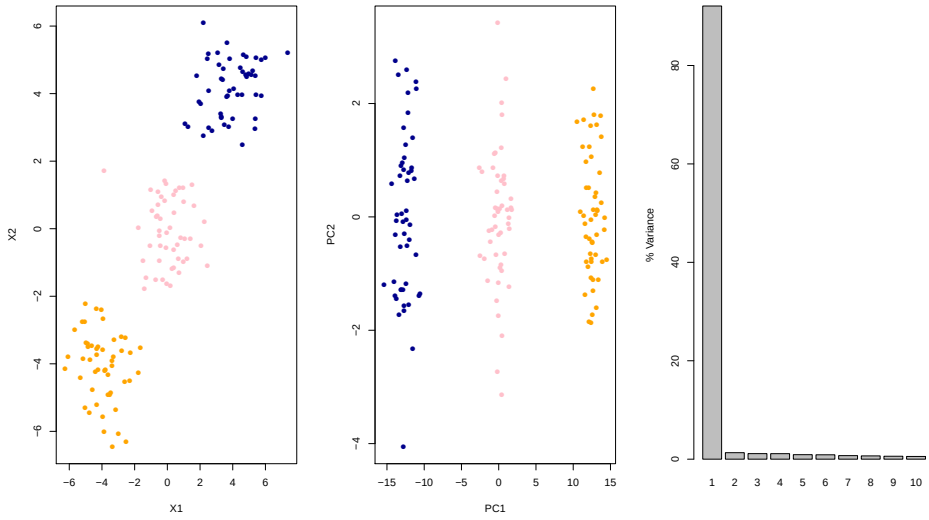


Figure 20: Example with 3 clear clusters (left). The 1st PC (middle) clearly separates the groups. The scree plot (right) shows that the 1st PC accounts for almost all the variance (98.24%)in the dataset, whereas we know that it is the 2nd and 3rd PCs that are the most useful in separating the groupings of Orange/Blue and Male/Female crabs.

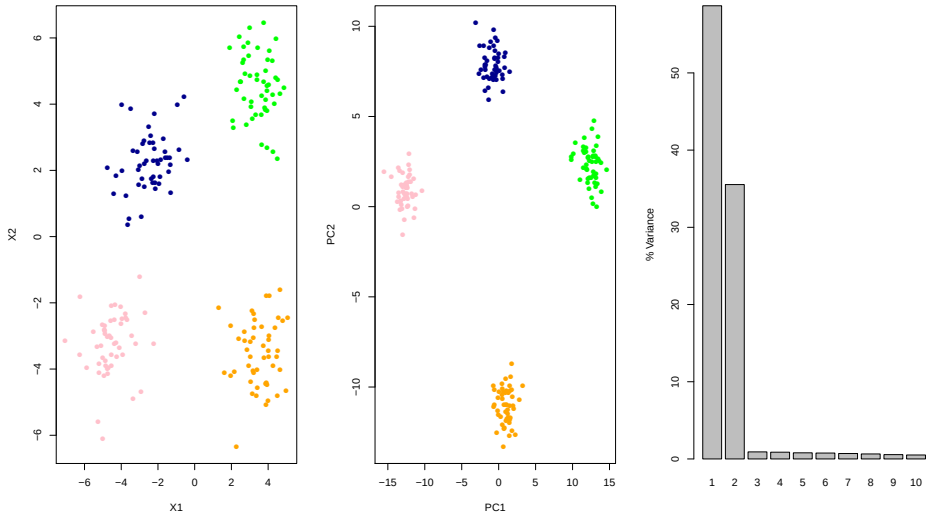


Figure 21: Example with 4 clear clusters (left). The 1st and 2nd PCs (middle) clearly separate the groups. The scree plot (right) shows that the 1st and 2nd PCs accounts for almost all the variance in the dataset.

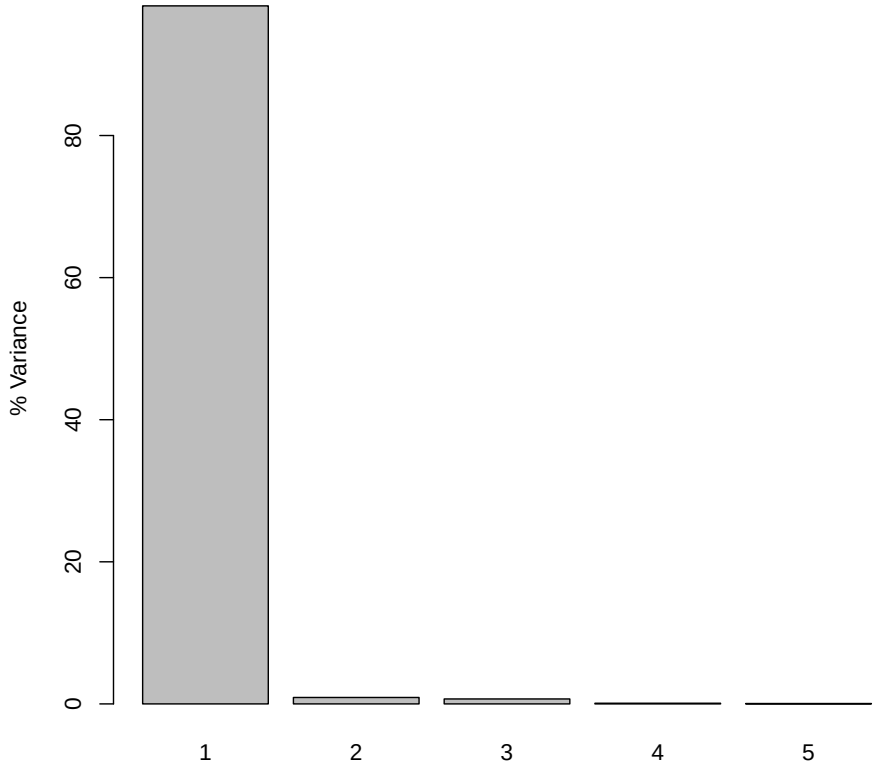


Figure 22: Scree plot for Crabs dataset

3.6 Using the covariance matrix or the correlation matrix

A key practical problem when applying PCA is deciding exactly what data the method should be applied to. Should we apply the method to the raw data, or should we first transform it in some way? You should always ask yourself this question when starting out on a new data analysis.

The first principal component maximises the variance of a linear combination of variables. If the variables have very different levels of variability then maximizing the projection with the largest variance may tend to dominate the first principal component. This might be unsatisfactory when looking for structure in the dataset. For this reason, it can sometimes make sense to standardize the variables before PCA, so that each variable has the same variance. This can be achieved by applying PCA to the **sample correlation matrix** $\boldsymbol{R}$, rather than the sample covariance matrix $\boldsymbol{S}$.

Remember that

$$\operatorname{cor}(X_j, X_k) = \frac{\operatorname{cov}(X_j, X_k)}{\sqrt{\operatorname{var}(X_j)\operatorname{var}(X_k)}} \in [-1, 1]$$

This means that the relationship between $\boldsymbol{R}$ and $\boldsymbol{S}$ is

$$\boldsymbol{R}_{jk} = \frac{\boldsymbol{S}_{jk}}{\sqrt{\boldsymbol{S}_{jj}\boldsymbol{S}_{kk}}}$$

If we let $\boldsymbol{W}$ be a diagonal matrix with entries $\boldsymbol{S}_{jj}$ for $j \in 1, \ldots, p$. In other words, $\boldsymbol{W}$ is the same as $\boldsymbol{S}$, but with all off-diagonal entries set to 0. Then in matrix notation we can write

$$\boldsymbol{R} = \boldsymbol{W}^{-1/2}\boldsymbol{S}\boldsymbol{W}^{-1/2}$$

It can be shown (see Exercises) that the PCA components derived from using $\boldsymbol{S}$ are not the same as those derived from using $\boldsymbol{R}$, and knowledge of one of these sets of components does *not* enable the other set to be derived.

3.6.1 EU indicators dataset

Table 2 shows data on six economic indicators for the 27 European Union countries collected in 2012. Looking at the table we can see clearly that the scale of each of the variables is quite different. The last row of the table shows that variables BOP and PRC have much higher variances than the other variables.

Country	CPI	UNE	INP	BOP	PRC	UN%
Belgium	116.03	4.77	125.59	908.60	6716.50	-1.60
Bulgaria	141.20	7.31	102.39	27.80	1094.70	3.50
CzechRep.	116.20	4.88	119.01	-277.90	2616.40	-0.60
Denmark	114.20	6.03	88.20	1156.40	7992.40	0.50
Germany	111.60	4.63	111.30	499.40	6774.60	-1.30
Estonia	135.08	9.71	111.50	153.40	2194.10	-7.70
Ireland	106.80	10.20	111.20	-166.50	6525.10	2.00
Greece	122.83	11.30	78.22	-764.10	5620.10	6.40
Spain	116.97	15.79	83.44	-280.80	4955.80	0.70
France	111.55	6.77	92.60	-337.10	6828.50	-0.90
Italy	115.00	5.05	87.80	-366.20	5996.60	-0.50
Cyprus	116.44	5.14	86.91	-1090.60	5310.30	-0.40
Latvia	144.47	12.11	110.39	42.30	1968.30	-3.60
Lithuania	135.08	11.47	114.50	-77.40	2130.60	-4.30
Luxembourg	118.19	3.14	85.51	2016.50	10051.60	-3.00
Hungary	134.66	6.77	115.10	156.20	1954.80	-0.10
Malta	117.65	4.15	101.65	359.40	3378.30	-0.60
Netherlands	111.17	3.23	103.80	1156.60	6046.00	-0.40
Austria	114.10	2.99	116.80	87.80	7045.50	-1.50
Poland	119.90	6.28	146.70	-74.80	2124.20	-1.00
Portugal	113.06	9.68	89.30	-613.40	4073.60	0.80
Romania	142.34	4.76	131.80	-128.70	1302.20	3.20
Slovenia	118.33	5.56	105.40	39.40	3528.30	1.80
Slovakia	117.17	9.19	156.30	16.00	2515.30	-2.10
Finland	114.60	5.92	101.00	-503.70	7198.80	-1.30
Sweden	112.71	6.10	100.50	1079.10	7476.70	-2.30
UnitedKingdom	120.90	6.11	90.36	-24.30	6843.90	-0.80
Variance	111.66	9.95	357.27	450057.15	5992520.48	7.12

Table 2: CPI, consumer price index (index = 100 in 2005); UNE, unemployment rate in 15–64 age group; INP, industrial production (index = 100 in 2005); BOP, balance of payments (€/capita); PRC, private final consumption expenditure (€/capita); UN%, annual change in unemployment rate.

Figure 23 shows plots of the 1st and 2nd PCs for the EU indicators dataset. The Left plot used the covariance matrix $\boldsymbol{S}$. The Right plot used the correlation matrix $\boldsymbol{R}$. Points are labelled with the abbreviated country name. There is a clear difference between the two.

When using the covariance matrix $\boldsymbol{S}$ the loadings of the 1st and 2nd PCs are

$$\begin{aligned} Z_1 &= -0.003CPI - 0.0004UNE - 0.0039INP + 0.121BOP + 0.993PRC - 0.00003UN\% \\ Z_2 &= 0.004CPI - 0.001UNE + 0.009INP + 0.992BOP - 0.121PRC - 0.0014UN\% \end{aligned}$$

so it is the variables BOP and PRC that are dominating these PCs.

When using the correlation matrix $\boldsymbol{R}$ the loadings of the 1st and 2nd PCs are

$$\begin{aligned} Z_1 &= -0.51CPI - 0.37UNE - 0.29INP + 0.36BOP - 0.62PRC - 0.02UN\% \\ Z_2 &= -0.17CPI + 0.34UNE - 0.53INP - 0.49BOP + 0.12PRC + 0.56UN\% \end{aligned}$$

and the weightings for the variables are quite different.

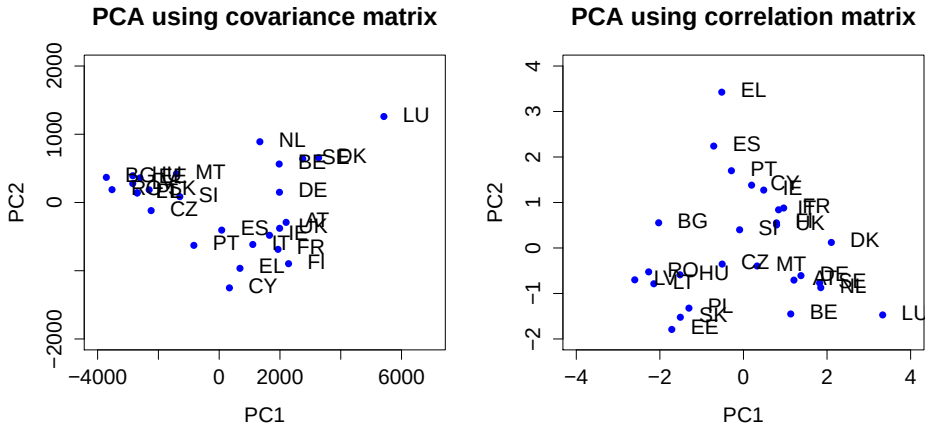


Figure 23: Plots of the 1st and 2nd PCs for the EU indicators dataset. (Left) Using the covariance matrix $\boldsymbol{S}$. (Right) Using the correlation matrix $\boldsymbol{R}$. Points are labelled with the abbreviated country name.

3.7 PCA via the Singular Value Decomposition

The Crabs data has the property that the number of observations $n = 200$ is much larger than the number of variable $p = 5$. This is not always the case for datasets that we might work with. For example, earlier we saw that the Single Cell dataset had $n = 300$ and $p = 8,686$, the UK Foods dataset had $n = 4$ and $p = 17$ and the POPRES dataset of human genetic data had $n = 3,000$ and $p = 500,000$. A key step in carrying out PCA is calculating an eigen decomposition of the $p \times p$ sample covariance matrix $\boldsymbol{S}$.

Typical algorithms for finding the eigendecomposition of a $p \times p$ matrix scale like $O(p^3)$. Also, when p is large this can be a very large matrix to store and work with on a computer. Also, since n points in a p-dimensional space defines a linear subspace whose dimension is at most $n - 1$, we would find that $p - n + 1$ eigenvalues are zero.

There is a faster way of obtaining the scores matrix $\boldsymbol{Z} = \boldsymbol{XV}$ that contains the projections onto the PCs. We need to use the following result

Theorem (Singular Value Decomposition) If $\boldsymbol{X}$ is a $n \times p$ matrix of real numbers then there exists a factorization, called the Singular Value Decomposition (SVD) of $\boldsymbol{X}$, with the following form

$$\boldsymbol{X} = \boldsymbol{P\Lambda Q}^T$$

where

- $\boldsymbol{P}$ is a $n \times n$ matrix such that $\boldsymbol{PP}^T = \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}_n$
- $\boldsymbol{Q}$ is a $p \times p$ matrix such that $\boldsymbol{QQ}^T = \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}_p$
- $\boldsymbol{\Lambda}$ is $n \times p$ diagonal matrix with min(n, p) non-negative real numbers on the diagonal.

The diagonal entries of $\boldsymbol{\Lambda}$ are known as **singular values** of $\boldsymbol{X}$.

NOTE: This theorem is given without proof and Prelims students are not expected to be able to prove it.

Using the SVD we can write the following

$$\begin{aligned} \boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X} &= \frac{1}{n-1}\boldsymbol{Q\Lambda}^T\boldsymbol{P}^T\boldsymbol{P\Lambda Q}^T \quad (6) \\ &= \frac{1}{n-1}\boldsymbol{Q\Lambda}^T\boldsymbol{\Lambda Q}^T \quad (7) \end{aligned}$$

Note that $\boldsymbol{\Lambda}^T\boldsymbol{\Lambda}$ is a $p \times p$ diagonal matrix with entries that the squares of the entries in $\boldsymbol{\Lambda}$. This has the same form as the eigendecomposition of $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$ where $\boldsymbol{V} = \boldsymbol{Q}$ and $\boldsymbol{D} = \frac{1}{n-1}\boldsymbol{\Lambda}^T\boldsymbol{\Lambda}$.

Therefore

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{V} = \boldsymbol{X}\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{\Lambda}$$

which implies that we need to calculate $\boldsymbol{P}$ and $\boldsymbol{\Lambda}$. This can be achieved using the eigendecomposition of the $n \times n$ matrix $\boldsymbol{X}\boldsymbol{X}^T$ since

$$\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{P}^T = \boldsymbol{P}(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T)\boldsymbol{P}^T$$

In other words, we can eigendecompose $\boldsymbol{X}\boldsymbol{X}^T$ and obtain $\boldsymbol{P}$ and $\boldsymbol{\Lambda}$ from that. Calculating the eigendecomposition of $\boldsymbol{X}\boldsymbol{X}^T$ scales like $O(n^3)$ which is much less than the eigendecomposition of $\boldsymbol{X}^T\boldsymbol{X}$ which scales like $O(p^3)$. Also, it is much easier to store and work with an $n \times n$ matrix than a $p \times p$ matrix when $n << p$.

3.8 PCA as minimizing reconstruction error

There is another way to derive principal components analysis that uses the idea that we might try to find the low-dimensional approximation that is as *close* as possible to the original data.

If $\boldsymbol{X}$ is our $n \times p$ data matrix then a rank-1 approximation to the matrix can be constructed as

$$\widetilde{\boldsymbol{X}} = z_1 w_1^T$$

where z_1 is an n-column vector and w_1 is p-column vector. Together the product $z_1 w_1^T$ is an $n \times p$ matrix, and the idea is that we find the best pair of vectors z_1 and w_1 to minimize the distance.
We can measure the distance between $\boldsymbol{X}$ and $\widetilde{\boldsymbol{X}}$ by sum of the squared differences between the two matrices.
The ith row of $\boldsymbol{X}$ is x_i. The ith row of $\widetilde{\boldsymbol{X}}$ is $z_{i1}w_1$.

$$\begin{aligned} J(z_1, w_1) &= \frac{1}{n}\sum_{i=1}^{n}(x_i - z_{i1}w_1)^T(x_i - z_{i1}w_1) \quad (8) \\ &= \frac{1}{n}\sum_{i=1}^{n}[x_i^T x_i - 2z_{i1}w_1^T x_i + w_1^T z_{i1}^T z_{i1} w_1] \quad (9) \end{aligned}$$

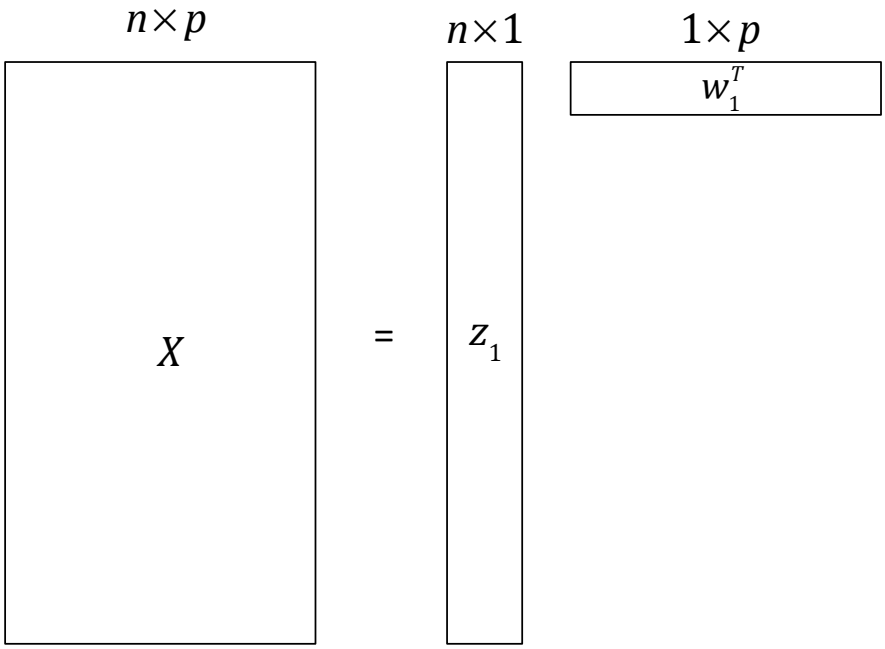


Figure 24: Figure illustrating a rank-1 approximation to the data matrix $\boldsymbol{X}$.

$$= \frac{1}{n}\sum_{i=1}^{n}[x_i^T x_i - 2z_{i1}w_1^T x_i + z_{i1}^2(w_1^T w_1)] \tag{10}$$

$$\tag{11}$$

Since we can arbitrarily scale z_1 and w_1 so that the product $\widetilde{\boldsymbol{X}} = z_1 w_1^T$ stays the same, we can add the constraint that $w_1^T w_1 = 1$. Taking derivatives wrt z_{i1} and equating to zero gives

$$\frac{\partial}{\partial z_{i1}} J(z_1, w_1) = \frac{1}{n}[-2w_1^T x_i + 2z_{i1}] = 0 \Rightarrow z_{i1} = w_1^T x_i = x_i^T w_1$$

or in matrix form

$$z_1 = \boldsymbol{X} w_1$$

Plugging this back in gives

$$J(w_1) = \frac{1}{n}\sum_{i=1}^{n} x_i^T x_i - \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$$

$$
\begin{aligned}
&= \frac{1}{n}\sum_{i=1}^{n} x_i^T x_i - \frac{1}{n}\sum_{i=1}^{n}(w_1^T x_i)(w_1^T x_i)^T \\
&= constant - \frac{1}{n}\sum_{i=1}^{n} w_1^T x_i x_i^T w_1
\end{aligned}
$$

Now if we assume that the columns of $\boldsymbol{X}$ have been mean centered then

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$$

which gives

$$J(w_1) = constant - w_1^T\hat{\boldsymbol{\Sigma}}w_1 \tag{12}$$

To minimize this we need to maximize $w_1^T\hat{\boldsymbol{\Sigma}}w_1$ subject to the constraint $w_1^T w_1 = 1$. We can replace $\hat{\boldsymbol{\Sigma}}$ with $\boldsymbol{S}$ since $\boldsymbol{S} \propto \hat{\boldsymbol{\Sigma}}$ and solve using Lagrange multipliers

$$\max_{w_1} w_1^T\boldsymbol{S}w_1 \quad \text{subject to} \quad w_1^T w_1 = 1$$

which leads to the eigenvalue equation

$$\boldsymbol{S}w_1 = \lambda_1 w_1 \tag{13}$$

This is the same equation we had before (see eqn 1), so w_1 and λ_1 are an eigenvector and eigenvalue of $\boldsymbol{S}$ respectively. Since $w_1^T\boldsymbol{S}w_1 = \lambda_1$ is what we are trying to maximize we choose λ_1 to be the largest eigenvalue of $\boldsymbol{S}$ and w_1 its associated eigenvector. Thus the best rank-1 approximation to $\boldsymbol{X}$ is $z_1 w_1^T = \boldsymbol{X}w_1 w_1^T$.

This approach can be extended to find the best rank-2 approximation as

$$\boldsymbol{X} \approx z_1 w_1^T + z_2 w_2^T$$

using the constraints $w_2^T w_2 = 1$ and $w_2^T w_1 = 0$ and assuming that we have already estimated w_1 and z_1. It can be shown that w_2 is the eigenvector associated with the second largest eigenvalue of $\boldsymbol{S}$.

A general extension to the best rank-q approximation to $\boldsymbol{X}$ then follows from that, and can be written as

$$\boldsymbol{X} \approx \sum_{i=1}^{q} z_i w_i^T \tag{14}$$

3.8.1 Using PCA for compression

The data matrix $\boldsymbol{X}$ has a total of np elements, and the best rank-q approximation in equation 14 has $q(n+p)$ elements. Also, the derivation assumed the $\boldsymbol{X}$ had been mean centered so really we need to store the column mean vector $\hat{\mu}$, which adds another p values.

So if we can find a relatively small value of q such $q(n+p)+p << np$ for which the distance between the rank-q approximation and $\boldsymbol{X}$ is small then the approximation can be used to compactly store the majority of the information about the dataset $\boldsymbol{X}$. This is an example of **data compression** and can be useful when storing or transmitting a dataset.

4 Clustering Methods

We have seen the PCA can provide low dimensional representations of a dataset that show groupings of observations when visualised. However, PCA does not involve a direct labelling of observations into different groups. **Clustering** refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a dataset.

Clustering involves partitioning observations into groups, where observations in the same group are similar to each other, while observations in different groups are quite different from each other. To make this more concrete, we need to define exactly what we mean by *similar* or *different*. There will not just be one way of defining this, and the choice can depend on the dataset being analyzed, as dictated by domain specific knowledge.

Some specific examples where clustering is needed are

1. Marketing and Advertising - given a database of existing customers, can we identify subgroups of customers who might be receptive to a particular form of advertising, or more likely to buy a particular product.

2. Image segmentation - in medical imaging we may wish to automatically detect tumours in an image, by clustering pixels of an image into different groups, and then identifying outlying groups.

There are a large number of different clustering methods. In this course, we will learn about two of the most popular : **k-means clustering** and **hierarchical clustering**.

In k-means clustering we seek to partition the observations into a pre-specified number of clusters. In hierarchical clustering, we don't pre-specify the number of clusters. Instead, we build a tree-like representation of the dataset, called a **dendrogram**, that allows us to partition the observations into as many clusters as we want.

4.1 K-means clustering

Figure 25 shows a simple simulated dataset with 3 clear clusters of points in 2 dimensions. To start with we will use this simple example to illustrate the method.

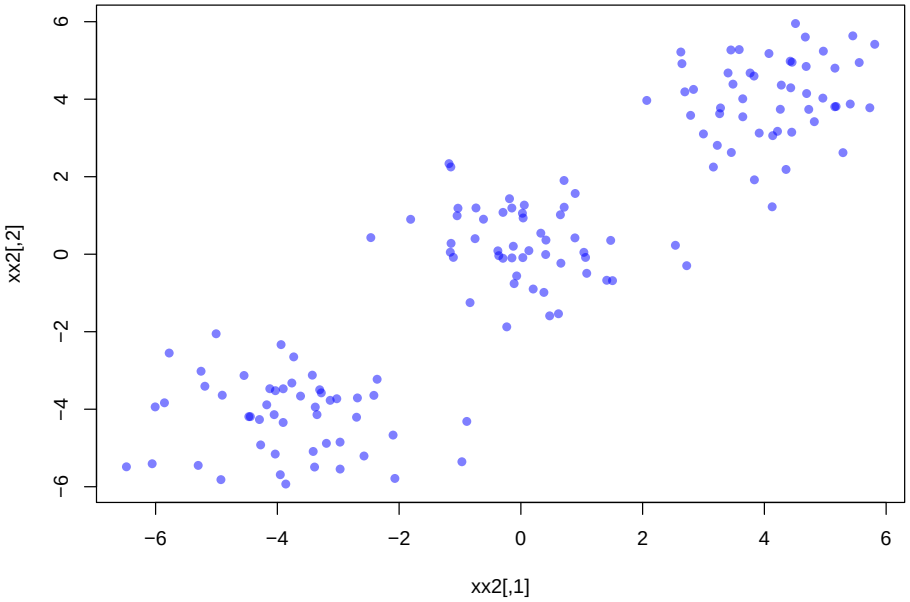


Figure 25: Simple example with 3 clear clusters of points.

To perform K-means clustering we must first decide upon the number of clusters K. The algorithm will assign each observation to exactly one of the K clusters.

We start by assuming the our dataset is stored in the $n \times p$ data matrix $\boldsymbol{X}$ and with (i, j)th entry x_{ij}. As before, each row is an observation, and each column is a variable.

Let $C = (C_1, C_2, \ldots, C_K)$ denote sets containing indices of the n observations in each cluster. We will refer to C as a *clustering* of the observations. Let $c(i)$ be the cluster assignment of the ith observation. For example, if observation i is assigned to the kth cluster then $i \in C_k$ and $c(i) = k$. These sets satisfy two properties

1. $C_1 \cup C_2 \cup \ldots C_K = \{1, \ldots, n\}$
2. $C_i \cap C_j = \emptyset$ for all $i \neq j$

We want to choose a clustering which has the property that the differences between the observations *within* each cluster are as small as possible. If we define $W(C_k)$ to be a measure of how different the observations are within cluster k then we want to solve the problem

$$\min_{C_1,C_2,\ldots,C_k} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

To move forward we need to define the within-cluster variation measure $W(C_k)$. The most common choice, especially when dealing with real valued variables involves squared Euclidean distance

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

In other words we take pairs of observations in the cluster C_k and measure the squared Euclidean distance between those observations, and then sum over all such pairs, and then divide by the size of the cluster. So we need to optimize

$$\min_{C_1,C_2,\ldots,C_k} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

One option would be to search over all possible clusterings of the n observations. The number of possible clusterings becomes large very quickly. If we let $S(n,k)$ denote the number of ways to partition a set of n objects into k non-empty subsets then

$$S(n,k) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{n}{j} j^n$$

These numbers are known as Stirling numbers of the second kind and get very large quickly. For example,

$$S(100,4) = 66955751844038698560793085292692610900187911879206859351901$$

This clearly illustrates that it is very challenging to search over all these possibilities. However, if we can find a clustering that is "good" but not the best, then this may be (and often is) good enough for our purposes. The following algorithm can be shown to provide a good *local optimum* (i.e. a pretty good solution).

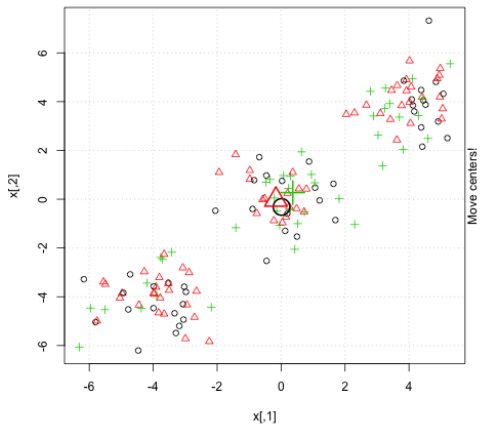
K-means algorithm

1. Choose K.
2. Randomly assign each observation to one of the clusters $C_1, C_2, \ldots, C_K$. These are the initial cluster assignments.
3. Iterate the following 2 steps until the cluster assignments stop changing
   (a) For each cluster compute the cluster mean. The kth cluster mean denoted μ_k is the mean of the all the x_i in cluster k i.e.

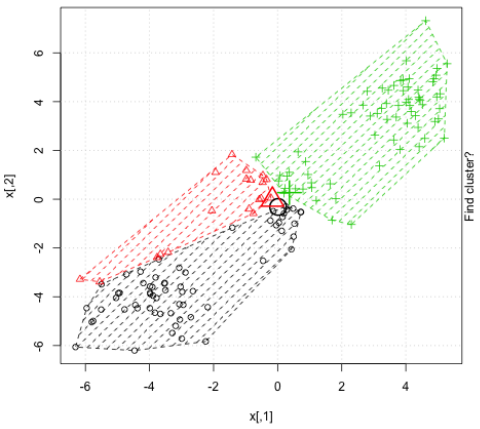
$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

   (b) Re-assign all observations to the cluster whose mean is closest, where closest is defined using Euclidean distance.

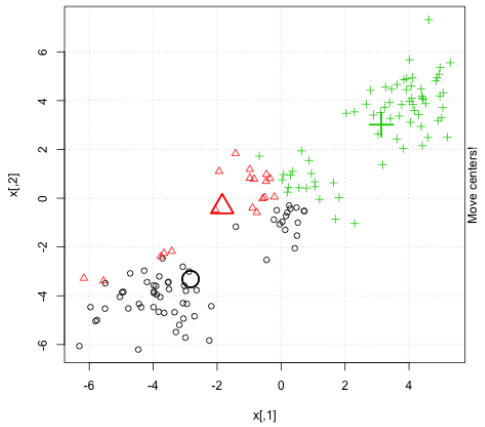
Figure 26 shows the results of running this algorithm on the simple example with 3 clear clusters.



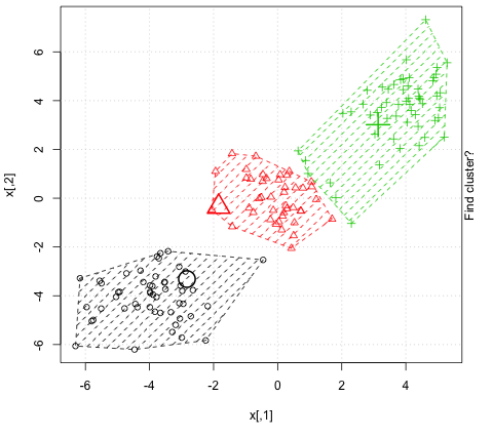
(a) Randomly allocate points to 3 clusters



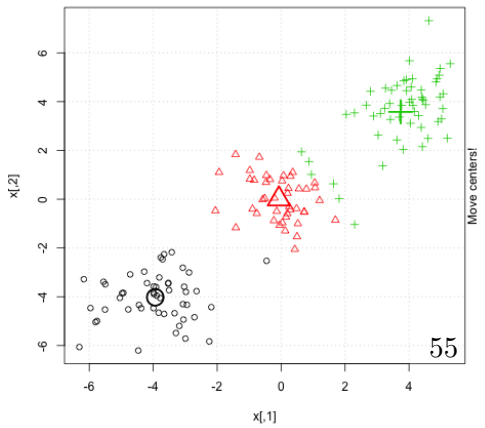
(b) Calculate cluster means



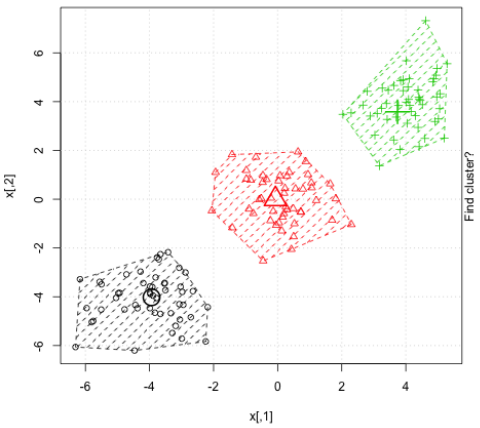
(c) Allocate points to new clusters



(d) Calculate cluster means



(e) Allocate points to new clusters



(f) Calculate cluster means

Figure 26: Example of k-means on toy dataset with 3 clear clusters

Convergence of the K-means algorithm

It can be shown (see Exercises) that for a cluster C_k

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2 \tag{15}$$

so the problem can be re-written as

$$\min_{C_1,C_2,\ldots,C_k}\left\{\sum_{k=1}^{K}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2\right\} \tag{16}$$

We call

$$\sum_{k=1}^{K}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2 \tag{17}$$

the **objective function** that we are trying to minimize.

Also, given a different set of cluster means, denoted $v_1,\ldots,v_K$ it can be shown (see Exercises) that

$$\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-v_{kj})^2 = \sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2 + |C_k|\sum_{j=1}^{p}(\mu_{kj}-v_{kj})^2 \tag{18}$$

Let $C^{(t)}$ denote the clustering at iteration t.
Let $\mu_k^{(t)}$ be the mean of observations in cluster k at iteration t.

Now suppose that we have just updated the cluster assignments for the next iteration i.e. we have determined what $C^{(t+1)}$ is and so the current value of the objective function (using $C^{(t+1)}$ and $\mu_k^{(t)}$) is

$$\sum_{k=1}^{K}\sum_{i\in C_k^{(t+1)}}\sum_{j=1}^{p}(x_{ij}-\mu_{kj}^{(t)})^2$$

But since $\mu_k^{(t)}$ may not be the same as the cluster means for the assignments $C^{(t+1)}$ (since the cluster assignments can change) when we update the cluster means to $\mu_k^{(t+1)}$ we must have that

$$\sum_{k=1}^{K}\sum_{i\in C_k^{(t+1)}}\sum_{j=1}^{p}(x_{ij}-\mu_{kj}^{(t+1)})^2 \leq \sum_{k=1}^{K}\sum_{i\in C_k^{(t+1)}}\sum_{j=1}^{p}(x_{ij}-\mu_{kj}^{(t)})^2$$

using Equation 18. So updating the cluster means to $\mu_k^{(t+1)}$ never increases the objective function.

Similarly, in Step 3(b) the cluster means are fixed and we update the assignments. We can re-write the objective function in (17) as a sum over the n observations

$$\sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij}-\mu_{c_i j})^2$$

this is clearly minimized by assigning each observation to the cluster with minimum Euclidean distance to the cluster mean.

Multiple starts

The algorithm does not always give the same solution since the start point is random. Figure 27 shows an example dataset simulated to have points in 5 clusters that are quite close together and so hard to separate. The true cluster means are shown as red triangles in the plot.

Figure 28 shows the results of two different runs of k-means with $K = 5$. In the first run (left), the solution involves points being allocated to just 4 clusters. One of the 5 clusters becomes empty during the run as no points are closest to its cluster center. In the second run (right), this doesn't happen and we get 5 clear clusters, which almost perfectly cluster the points.

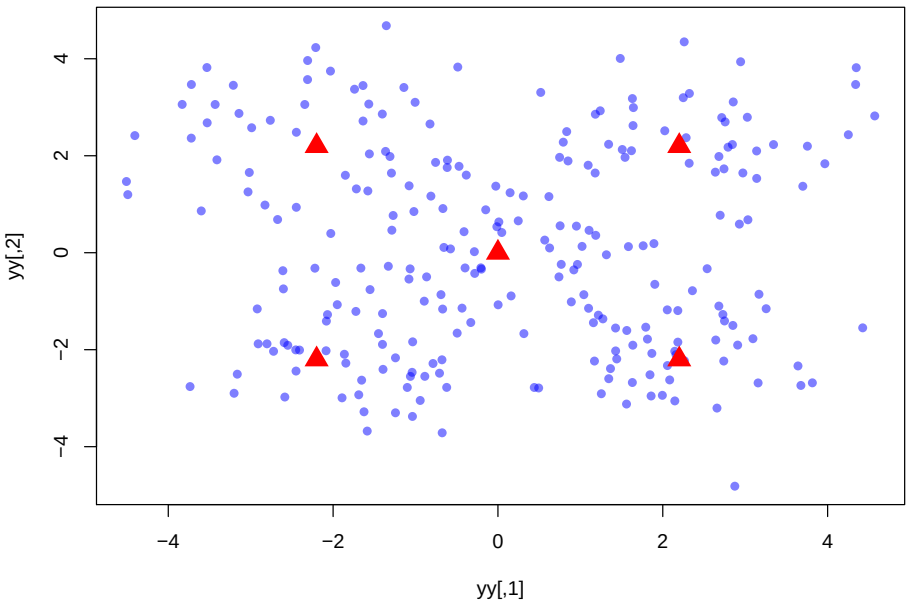
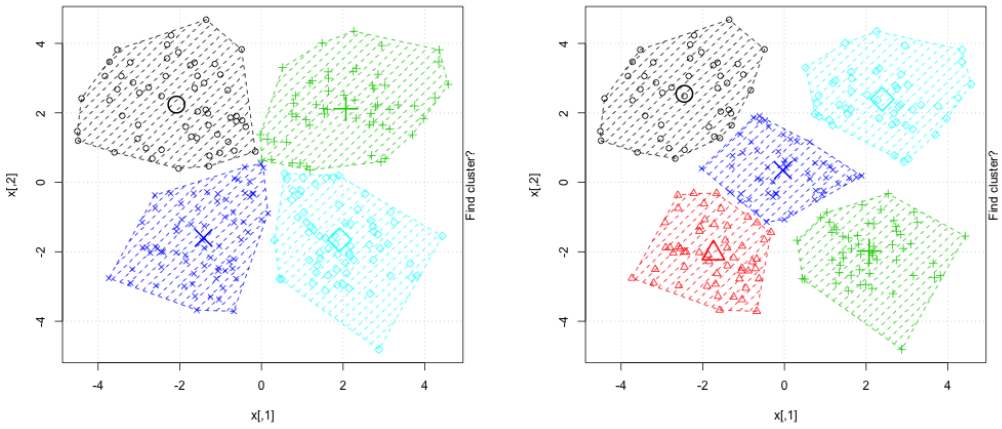


Figure 27: Example dataset with 5 close clusters of points.



(a) Solution 1 (b) Solution 2

Figure 28: Two different solutions from k-means with 5 clusters.

Figure 29 shows the results of applying k-means to the Single Cell Genomics dataset seen at the start of these course notes. The dataset was first reduced down to the 11 PCs and then k-means was run with $K = 11$. The Figure shows the k-means labelling of the points in the first 3 PC dimensions.

See also the file movie.gif on the course website:

`https://courses.maths.ox.ac.uk/course/view.php?id=6033`

for a rotating 3D view of this figure.

Choosing the number of clusters

The choice of K is a key first step in the algorithm. In some situations we may be interested in making statements about what value of K we think is best. This is an example of **model choice**, which occurs in many others parts of Data Analysis and Statistical Inference. A natural first thought might be to choose K by fitting the model to the dataset for many different values of K and then picking the value that results in a minimum of the objective. However, this is a flawed strategy as choosing $K = n$ and assigning each observation to its own cluster minimizes the objective function. This is almost never a realistic or meaningful solution to the problem. This is an example of **over-fitting**, in which we use a model that has too many parameters (in case the $K = n$ cluster means) relative to the number of observations.

Better solutions involve working with an objective function that penalises models with an increasing number of parameters. Such approaches to model choice will be covered in later courses on Statistics and Data Analysis.

Hierarchical clustering

Hierarchical clustering is an alternative approach that avoids having to specify the number of clusters in advance. An added advantage is that the method results in a tree-like (hierarchical) representation of the dataset, that can be helpful when visualising structure in the dataset, especially when the data is high-dimensional, i.e. p is large.

In this course we will describe **agglomerative clustering** approaches. The methods are called agglomerative because they iteratively fuse pairs obser-

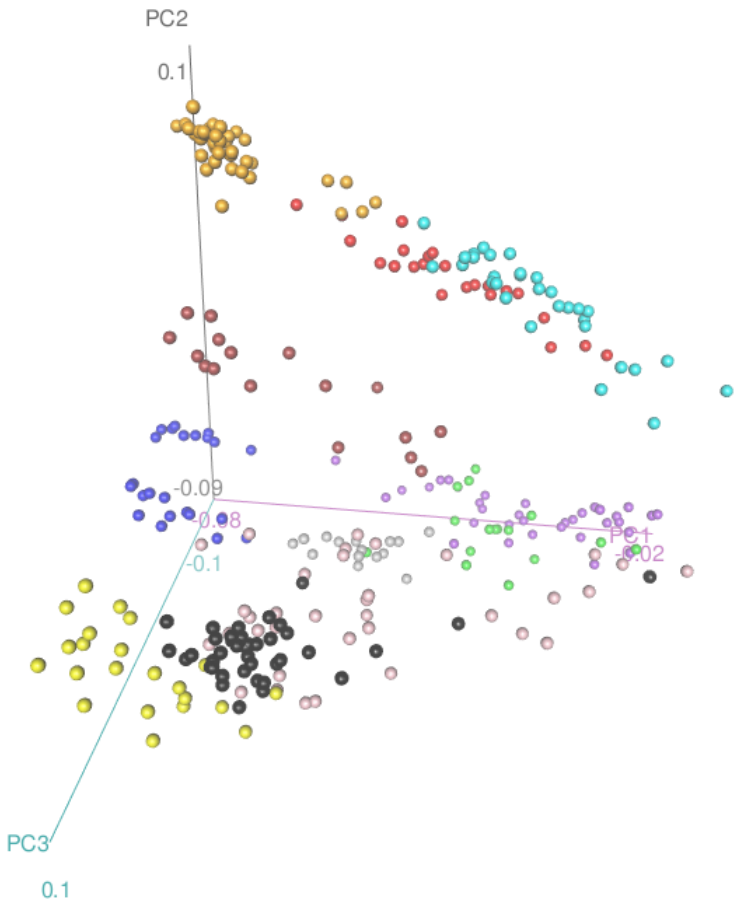


Figure 29: 3D plot of 1st, 2nd and 3rd Principal Components for the Single Cell Genomics dataset. Points are coloured according to a run of the k-means algorithm with $K = 11$ working on the first 11 PCs from a PCA of the dataset.

vations together to form clusters.

1. Begin with n observations and a measure of all the $\binom{n}{2}$ pairwise dissimilarities, denoted $d_{ii'}$ for $i \neq i' \in (1, \ldots, n)$. These dissimilarities can be represented in a lower diagonal matrix, denoted $\boldsymbol{D}^{(n)}$.

$$\boldsymbol{D}^{(n)} = \begin{bmatrix} d_{21} & & & & & \\ d_{31} & d_{32} & & & & \\ \vdots & \vdots & \ddots & & & \\ d_{i1} & d_{i2} & \dots & d_{i(i-1)} & & \\ \vdots & \vdots & \ddots & \vdots & \ddots & \\ d_{n1} & d_{n2} & \dots & d_{nj} & \dots & d_{n(n-1)} \end{bmatrix}$$

It is common to use Euclidean distance to measure dissimilarity (but other options exist). Treat each observation as its own cluster.

2. For $i = n, n-1, \dots, 2$

   (a) Find the pair of clusters with the smallest dissimilarity. Fuse these two clusters.

   (b) Compute the new dissimilarity matrix between the new fused cluster and all other $i-1$ remaining clusters and create an updated matrix of dissimilarities $\boldsymbol{D}^{(n-1)}$.

Linkage methods

A key step in the algorithm is the choice of how to compute the new dissimilarity (or distance) between the new fused cluster and all other $i-1$ remaining clusters. There are several commonly used options, with differing properties in terms of type of clusters they tend to produce. Let G and H be two groups of observations then we can define the function $d(G, H)$ between groups G and H to be some function of all the pairwise dissimilarities $d_{ii'}$ where $i \in G$ and $i' \in H$.

Single Linkage (SL) takes the intergroup distance to be the **closest** of all the pairs of observations between the two groups

$$d_{SL}(G, H) = \min_{i \in G, i' \in H} d_{ii'}$$

This is sometimes called the **nearest neighbour** method.

Complete Linkage (CL) takes the intergroup distance to be the **furthest** of all the pairs of observations between the two groups

$$d_{CL}(G, H) = \max_{i \in G, i' \in H} d_{ii'}$$

Group Average (GA) takes the intergroup distance to be the **average** of all the pairs of observations between the two groups

$$d_{GA}(G,H) = \frac{1}{|G||H|}\sum_{i\in G}\sum_{i'\in H} d_{ii'}$$

4.1.1 Small example

Consider the following small dataset with $n = 6$ and $p = 2$ variables

$$\boldsymbol{X} = \begin{bmatrix} 0.27 & 2.42 \\ 0.88 & 1.09 \\ 5.77 & 6.76 \\ 5.96 & 4.71 \\ 2.64 & 0.94 \\ 3.13 & 4.49 \end{bmatrix}.$$

Using Euclidean distance to measure dissimilarities results in the following matrix of dissimilarities, $d_{ii'}$ is the (i,i')th entry of the matrix. Note : we only show the lower triangle of the matrix and only record values to 2dp. We use the notation $\boldsymbol{D}^{(k)}$ to denote the dissimilarities of k clusters of observations. The smallest entry is coloured in **red**.

$\boldsymbol{D}^{(6)} =$

	1	2	3	4	5
2	**1.46**				
3	7.01	7.49			
4	6.13	6.24	2.06		
5	2.79	1.77	6.61	5.02	
6	3.53	4.08	3.48	2.84	3.59

We will apply single linkage clustering to this dataset.

The smallest entry in the matrix is $d_{2,1}$, so we merge observations 1 and 2 into a new cluster (denoted (1,2)) and compute the new dissimilarities

$\boldsymbol{D}^{(5)} =$

	$(1,2)$	3	4	5
3	7.01			
4	6.13	2.06		
5	**1.77**	6.61	5.02	
6	3.53	3.48	2.84	3.59

Next we merge clusters (1,2) and 5 into a new cluster (denoted (1,2,5)) and compute the new dissimilarites

$$\boldsymbol{D}^{(4)} = \begin{array}{c|ccc} & (1,2,5) & 3 & 4 \\ \hline 3 & 6.61 & & \\ 4 & 5.02 & \mathbf{2.06} & \\ 6 & 3.53 & 3.48 & 2.84 \end{array}$$

Next we merge clusters 3 and 4 into a new cluster (denoted (3,4)) and compute the new dissimilarites

$$\boldsymbol{D}^{(3)} = \begin{array}{c|cc} & (1,2,5) & (3,4) \\ \hline (3,4) & 5.02 & \\ 6 & 3.53 & \mathbf{2.84} \end{array}$$

Next we merge clusters (3,4) and 6 into a new cluster (denoted (3,4,6) and compute the new dissimilarites

$$\boldsymbol{D}^{(2)} = \begin{array}{c|c} & (1,2,5) \\ \hline (3,4,6) & \mathbf{3.53} \end{array}$$

Finally, we merge the two remaining clusters into a single cluster containing all the observations. Notice how the sequence of dissimilarities which dictate which clusters merge (ie. the numbers in red) are an increasing sequence of values.

Dendrograms

The results of an agglomerative clustering of a dataset can be represented as dendrogram, which is a tree-like diagram that allows us to visualise the way in which the observations have been joined into clusters.

A dendrogram is read from bottom to top. At the bottom we have a set of *leaves*. Each *leaf* of the dendrogram represents a data point. As we move up the tree leaves fuse into branches. The first two leaves to merge are the two observations that merge into a cluster in the first step of the algorithm. It is normal that this merge point occurs on the y-axis at a value which is the dissimilarity between the two merging clusters. The next two leaves to merge are the two observations that merge into a cluster in the second step of the algorithm, and so on. Figure 30 shows the resulting dendrogram from the mini-example of 6 observations described above.

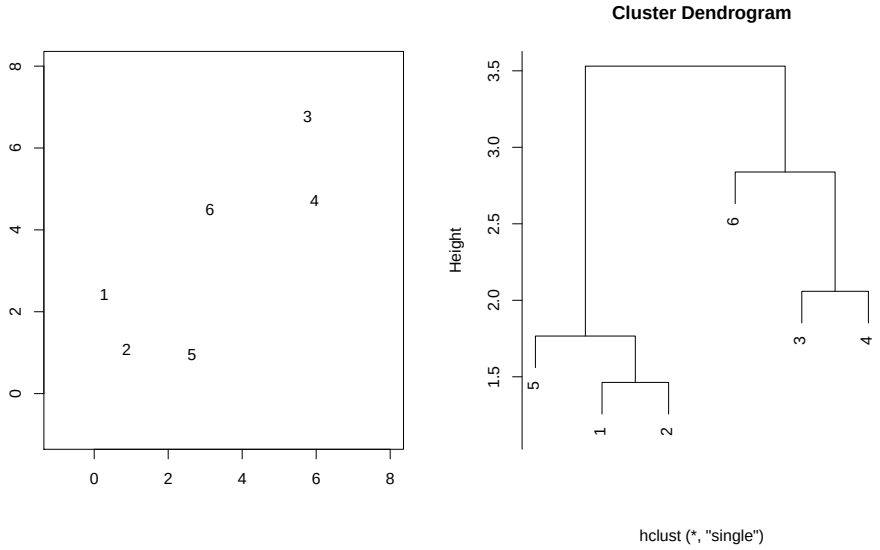


Figure 30: Small example of hierarchical clustering. The left plot shows the raw dataset consisting of 6 observations in 2 dimensions. The right plot shows the dendrogram of using agglomerative clustering with single linkage.

Differences between Linkage methods

Different linkage methods can lead to different dendrograms. Figure 31 shows a new dataset of 45 points, with 15 points each simulated from 3 different bivariate normal distributions with different means. The 3 sets of points are coloured according to their true cluster. Points are also labelled with a number.

Figure 32 shows the results of building dendrograms using Single Linkage, Complete Linkage and Average Linkage, and illustrates how these methods can differ.

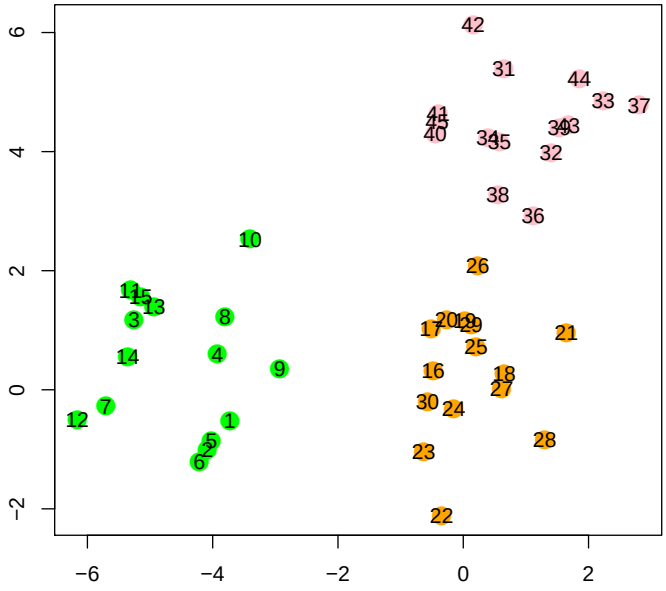


Figure 31: Example dataset with 45 observations, with 15 each from a different bivariate normal distribution.

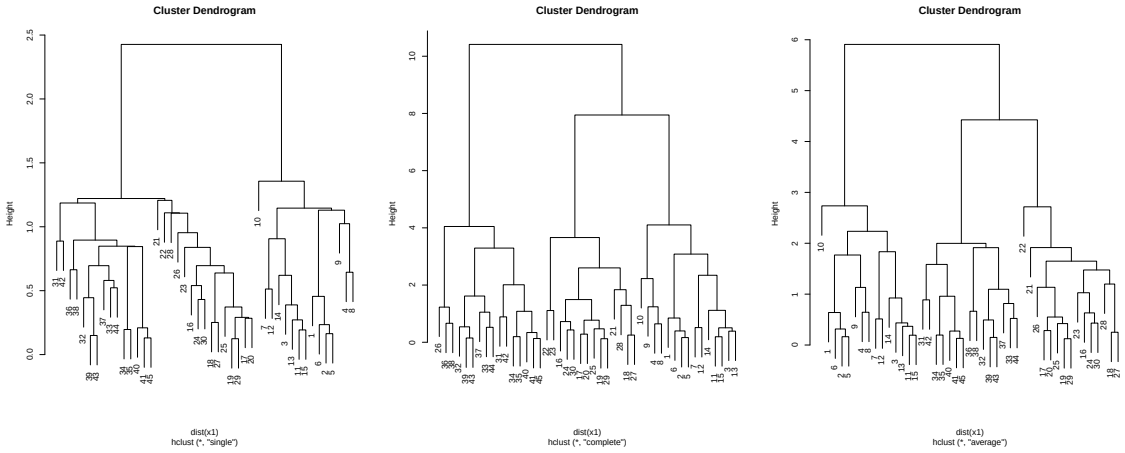


Figure 32: Dendrograms using different Linkage methods applied to the dataset in Figure 31.

Figure 33 shows the results of building a dendrogram using Complete Linkage on the EU indicators dataset (see Table 2)

Figure 33: Dendrogram produced from hierarchical clustering of the EU indicators dataset (see Table 2)

Using dendrograms to cluster observations

Dendrograms can be used to cluster observations into a discrete number of clusters or groups. To do this we make a horizontal *cut* across the dendrogram, as shown in the top plots in Figure 34. Here the cut point has been chosen to produce 3 clusters in each case. The leaves of the dendrograms have been coloured to indicate cluster membership of the observations. The clustering is also shown in 2D plots of the original observations, below each dendrogram. At this cut-point the Single Linkage method seems to have performed worse than Complete Linkage and Average Linkage, as it has a cluster that is just a single observation. Complete Linkage and Average Linkage produce very similar clusterings, and differ only in their assignment of 1 observation. It should be noted that the clustering produced by Single Linkage with 4 clusters produces very sensible results compared to the true clustering.

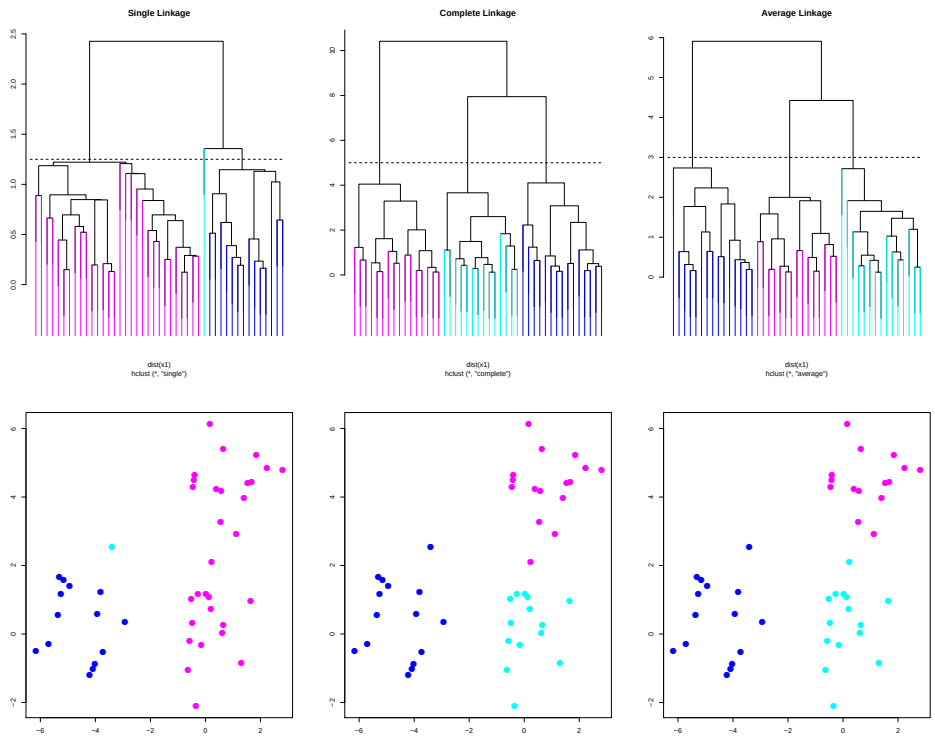


Figure 34: (Top row) Dendrograms using different Linkage methods applied to the dataset in Figure 31. Each dendrogram has been cut to produce 3 clusters. (Bottom row) The corresponding clustering of points in the original 2D space produced by cutting the dendrogram to have 3 clusters.