

Interactively exploring high-dimensional data and models in R

Dianne Cook and Ursula Laa

7/1/23

Table of contents

Preface

Although there are many resources available for data visualization, there are few comprehensive resources on high-dimensional data visualisation. This book fills this gap by providing a comprehensive and up-to-date guide to visualising high-dimensional data and models, with R.

High-dimensional data spaces are fascinating places. You may think that there's a lot of ways to plot one or two variables, and a lot of types of patterns that can be found. You might use a density plot and see skewness or a dot plot to find outliers. A scatterplot of two variables might reveal a non-linear relationship or a barrier beyond which no observations exist. We don't as yet have so many different choices of plot types for high-dimensions, but these types of patterns are also what we seek in scatterplots of high-dimensional data. The additional dimensions can clarify these patterns, that clusters are likely to be more distinct. Observations that did not appear to be very different can be seen to be lonely anomalies in high-dimensions, that no other observations have quite the same combination of values.

What's in this book?

The book is divided into these parts:

- **Introduction:** Here we introduce you to high-dimensional spaces, how they can be visualised, and notation that is useful for describing methods in later chapters.
- **Dimension reduction:** This part covers linear and non-linear dimension reduction. It includes ways to help decide on the number of dimensions needed to summarise the high dimensional data, whether linear dimension reduction is appropriate, detecting problems that might affect the dimension reduction, and examining how well or badly a non-linear dimension reduction is representing the data.
- **Cluster analysis:** This part described methods for finding groups in data. Although it includes an explanation of a purely graphical approach, it is mostly on using graphics in association with numerical clustering algorithms.

There are explanations of assessing the suitability of different numerical techniques for extracting clusters, based on the data shapes, evaluating the clustering result, and showing the solutions in high dimensions.

- **Classification:** This part describes methods for exploring known groups in the data. You'll learn how to check model assumptions, to help decide if a method is suited to the data, examine classification boundaries and explore where errors arise.
- **Miscellaneous:** The material in this part focuses on examining data from different contexts. This includes multiple time series, longitudinal data. A key pre-processing step is to convert the data into Euclidean space.

In each of these parts an emphasis is also showing your model with your data in the high dimensional space.

Our hopes are that you will come away with understanding the importance of plotting your high dimensional data as a regular step in your statistical or machine learning analyses. There are many examples of what you might miss if you don't plot the data. Effective use of graphics goes hand-in-hand with analytical techniques. With high dimensions visualisation is a challenge but it is fascinating, and leads to many surprising moments.

Audience

High-dimensional data arises in many fields such as biology, social sciences, finance, and more. Anyone who is doing exploratory data analysis and model fitting for more than two variables will benefit from learning how to effectively visualise high-dimensions. This book will be useful for students and teachers of multivariate data analysis and machine learning, and researchers, data analysts, and industry professionals who work in these areas.

How to use the book?

The book is written with explanations followed by examples with R code. The toolbox chapter provides an overview of the primary high-dimensional visualisation methods. The remaining chapters focus on different application areas and how to use the high-dimensional visualisation to complement commonly used analytical methods.

What should I know before reading this book?

The examples assume that you already use R, and have a working knowledge of base R and tidyverse way of thinking about data analysis. It also assumes that you have some knowledge of statistical methods, and some experience with machine learning methods.

If you feel like you need build up your skills in these areas in preparation for working through this book, these are our recommended resources:

- R for Data Science by Wickham and Grolemund for learning about data wrangling and visualisation.
- Introduction to Modern Statistics by Çetinkaya-Rundel and Hardin to learn about introductory statistics.
- Hands-On Machine Learning with R by Boehmke and Greenwell to learn about machine learning.

Setting up your workflow

To get started set up your computer with the current versions of R and Rstudio Desktop.

In addition, we have made an R package to share the data and functions used in this book, called `mulgar`.¹²

```
install.packages("mulgar", dependencies=TRUE)
#| or the development version
devtools::install_github("dicook/mulgar")
```

How do I report an error?

If you encounter an error, you can report it as an issue at the Github repo for this book.

Please make a small reproducible example and report the error encountered. Reproducible examples have these components:

- a small amount of data
- small amount of code that generates the error
- copy of the error message that was generated

License

¹Mulga is a type of Australian habitat composed of woodland or open forest dominated by the mulga tree. Massive clearing of mulga led to the vast wheat fields of Western Australia. Here **mulgar** is an acronym for **MUL**tivariate **G**raphical **A**nalysis with **R**.

²Photo of mulga tree taken by L. G. Cook.

Part I

Introduction

Chapter 1

Picturing high dimensions

High-dimensional data means that we have a large number of numeric features or variables, which can be considered as dimensions in a mathematical space. The variables can be different types, such as categorical or temporal, but the handling of these variables involves different techniques.



Figure 1.1: Viewing high dimensions using low-dimensional displays is like playing shadow puppets, looking at the shadows to guess what the shape is.

Figure 1.3: How a tour can be used to explore high-dimensional data illustrated using (a) 2D data with two clusters and (b) a tour of 1D projections shown as a density plot. Imagine spinning a line around the centre of the data plot, with points projected orthogonally onto the line. With this data, when the line is at $x_1=x_2$ (0.707, 0.707) or $(-0.707, -0.707)$ the clustering is the strongest. When it is at $x_1=-x_2$ (0.707, -0.707) there is no clustering.

1.1 Getting familiar with tours

Figure ?? illustrates a tour for 2D data and 1D projections. The (grand) tour will generate all possible 1D projections of the data, and display with a univariate plot like a histogram or density plot. For this data, the `simple_clusters` data, depending on the projection, the distribution might be clustered into two groups (bimodal), or there might be no clusters (unimodal). In this example, all projections are generated by rotating a line around the centre of the plot. Clustering can be seen in many of the projections, with the strongest being when the contribution of both variables is equal, and the projection is (0.707, 0.707) or $(-0.707, -0.707)$. (If you are curious about the number 0.707, read the last section of this chapter.)

Figure 1.5: How a tour can be used to explore high-dimensional data illustrated using a tour of 2D projections of 3D data (a). The data has a donut shape with the hole revealed in a single 2D projection (b). Data usually arrives with a given number of observations, and when we plot it like this using a scatterplot, it is like shadows of a transparent object.

Figure ?? illustrates a tour for 3D data using 2D projections. The data are points on the surface of a donut shape. By showing the projections using a scatterplot the donut looks transparent and we can see through the data. The donut shape can be inferred from watching many 2D projections but some are more revealing than others. The projection shown in (b) is where the hole in the donut is clearly visible.

1.2 What's different about space beyond 2D?

The term “high-dimensional” in this book refers to the dimensionality of the Euclidean space. Figure ?? shows a way to imagine this. It shows a sequence of cube wireframes, ranging from one-dimensional (1D) through to five-dimensional (5D), where beyond 2D is a linear projection of the cube. As the dimension increases, a new orthogonal axis is added. For cubes, this is achieved by doubling the cube: a 2D cube consists of two 1D cubes, a 3D cube consists of two 2D cubes, and so forth. This is a great way to think about the space being examined by the visual methods, and also all of the machine learning methods mentioned,

in this book.

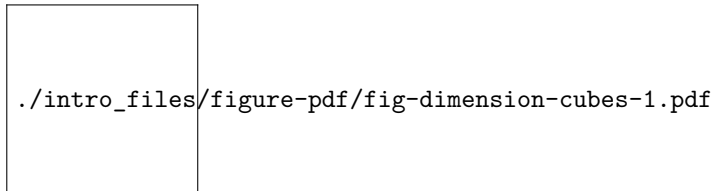


Figure 1.6: Space can be considered to be a high-dimensional cube. Here we have pictured a sequence of increasing dimension cubes, from 1D to 5D, as wireframes, it can be seen that as the dimension increase by one, the cube doubles.

Interestingly, the struggle with imagining high-dimensions this way is described in a novel published in 1884 (Abbott, 1884)¹. Yes, more than 100 years ago! This is a story about characters living in a 2D world, being visited by an alien 3D character. It also is a social satire, serving the reader strong messages about gender inequity, although this provides the means to explain more intricacies in perceiving dimensions. There have been several movies made based on the book in recent decades (e.g. Martin (1965), D. Johnson & Travis (2007)). Although purchasing the movies may be prohibitive, watching the trailers available for free online is sufficient to gain enough geometric intuition on the nature of understanding high-dimensional spaces while living in a low-dimensional world.

When we look at high-dimensional spaces from a low-dimensional space, we meet the “curse of dimensionality”, a term introduced by Bellman (1961) to express the difficulty of doing optimization in high dimensions because of the exponential growth in space as dimension increases. A way to imagine this is look at the cubes in Figure ?? : As you go from 1D to 2D, 2D to 3D, the space expands a lot, and imagine how vast space might get as more dimensions are added². The volume of the space grows exponentially with dimension, which makes it infeasible to sample enough points – any sample will be less densely covering the space as dimension increases. The effect is that most points will be far from the sample mean, on the edge of the sample space.

For visualisation, the curse manifests in an opposite manner. Projecting from high to low dimensions creates a crowding or piling of points near the center of the distribution. This was noted by Diaconis & Freedman (1984a). Figure ?? illustrates this phenomenon. As dimension increases, the points crowd the centre, even with as few as ten dimensions. This is something that we may need to correct for when exploring high dimensions with low-dimensional projections.

Figure ?? shows 2D tours of two different 5D data sets. One has clusters (a) and

¹Thanks to Barret Schloerke for directing co-author Cook to this history when he was an undergraduate student and we were starting the geozoo project.

²It is generally better to use *associated* than *correlated*. Correlated is a statistical quantity, measuring linear association. The term *associated* can be prefaced with the type of association, such as *linear* or *non-linear*.

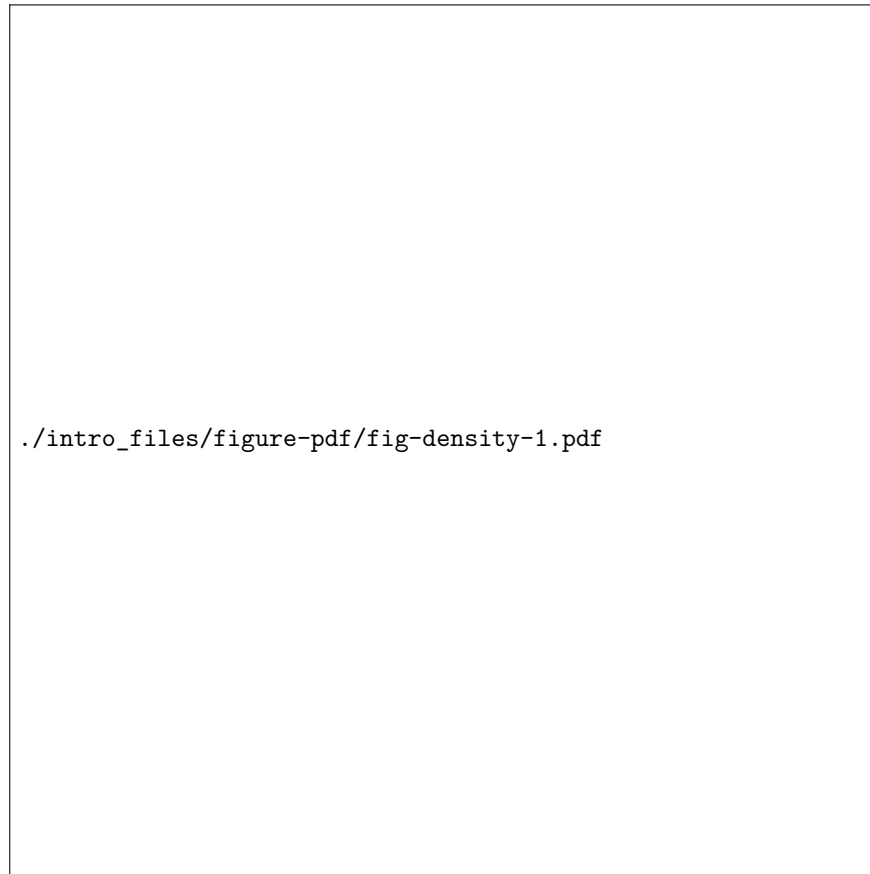


Figure 1.7: Illustration of data crowding in the low-dimensional projection as dimension increases, here from 3, 10, 100. Colour shows the number of points in each hexagon bin (pink is large, navy is small). As dimension increases the points concentrate near the centre.

the other has two outliers and a plane (b). Can you see these? One difference in the viewing of data with more than three dimensions with 2D projections is that the points seem to shrink towards the centre, and then expand out again. This is the effect of dimensionality, with different variance or spread in some directions.

Figure 1.8: Two 5D datasets shown as tours of 2D projections. Can you see clusters of points in (a) and two outliers with a plane in (b)?

1.3 What can you learn?

There are two ways of detecting structure in tours:

- patterns in a single low-dimensional projection
- movement patterns

with the latter being especially useful when displaying the projected data as a scatterplot. Figure ?? shows examples of patterns we typically look for when making a scatterplot of data. These include clustering, linear and non-linear association, outliers, barriers where there is a sharp edge beyond which no observations are seen. Not shown, but it also might be possible to observe multiple modes, or density of observations, L-shapes, discreteness or uneven spread of points. The tour is especially useful if these patterns are only visible in combinations of variables.

Figure ?? illustrates how movement patterns of points when using scatterplots to display 2D projections indicate clustering (a, b) and outliers (c, d).

This type of visualisation is useful for many activities in dealing with high-dimensional data, including:

- exploring high-dimensional data.
- detecting if the data lives in a lower dimensional space than the number of variables.
- checking assumptions required for multivariate models to be applicable.
- check for potential problems in modeling such as multicollinearity among predictors.
- checking assumptions required for probabilities calculated for statistical hypothesis testing to be valid.
- diagnosing the fit of multivariate models.

With a tour we slowly rotate the viewing direction, this allows us to see many individual projections and to track movement patterns. Look for interesting structures such as clusters or outlying points.

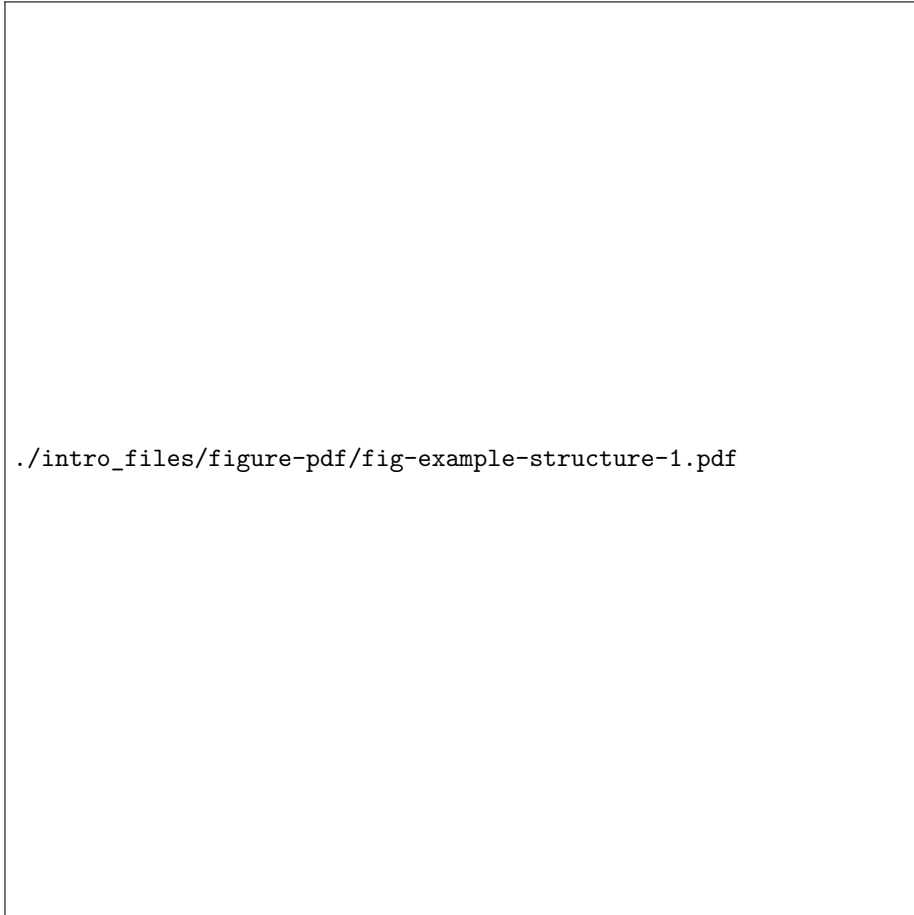


Figure 1.9: Example structures that might be visible in a 2D projection that imply presence of structure in high dimensions. These include clusters, linear and non-linear association, outliers and barriers.

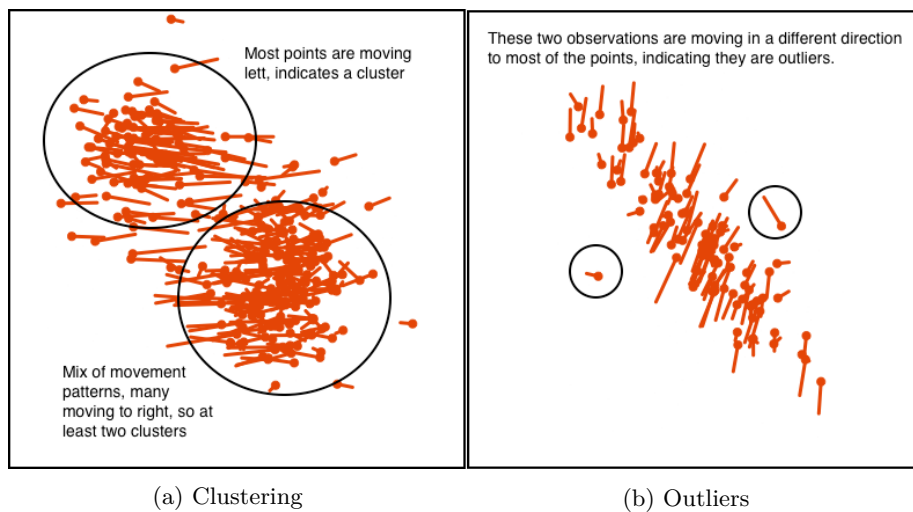


Figure 1.10: The movement of points give further clues about the structure of the data in high-dimensions. In the data with clustering, often we can see a group of points moving differently from the others. Because there are three clusters, you should see three distinct movement patterns. It is similar with outliers, except these may be individual points moving alone, and different from all others. This can be seen in the static plot, one point (top left) has a movement pattern upwards whereas most of the other observations near it are moving down towards the right.

1.4 A little history

Viewing high-dimensional data based on low-dimensional projections can probably be traced back to the early work on principal component analysis by Pearson (1901) and Hotelling (1933), which was extended to known classes as part of discriminant analysis by Fisher (1936a).

With computer graphics, the capability of animating plots to show more than a single best projection became possible. The video library (ASA Statistical Graphics Section, 2023) is the best place to experience the earliest work. Kruskal’s 1962 animation of multidimensional scaling showed the process of finding a good 2D representation of high dimensional data, although the views are not projections. Chang’s 1970 video shows her rotating a high dimensional point cloud along coordinate axes to find a special projection where all the numbers align. The classic video that must be watched is PRIM9 (Fisher et al., 1973) where a variety of interactive and dynamic tools are used together to explore high dimensional physics data, documented in Fisher et al. (1974).

The methods in this book primarily emerge from Asimov (1985)’s grand tour method. The algorithm provided the first smooth and continuous sequence of low dimensional projections, and guaranteed that all possible low dimensional projections were likely to be shown. The algorithm was refined in Buja & Asimov (1986) (and documented in detail in Buja et al. (2005)) to make it *efficiently* show all possible projections. Since then there have been numerous varieties of tour algorithms developed to focus on specific tasks in exploring high dimensional data, and these are documented in S. Lee et al. (2022).

This book is an evolution from Cook & Swayne (2007). One of the difficulties in working on interactive and dynamic graphics research has been the rapid change in technology. Programming languages have changed a little (fortran to C to java to python) but graphics toolkits and display devices have changed a lot! The tour software used in this book evolved from XGobi, which was written in C and used the X Window System, which was then rewritten in GGobi using gtk. The video library has engaging videos of these software systems. There have been several other short-lived implementations, including orca (Sutherland et al., 2000a), written in java, and cranvas (Xie et al., 2014), written in R with a back-end provided by wrapper functions to qt libraries.

Although attempts were made with these ancestor systems to connect the data plots to a statistical analysis system, these were always limited. With the emergence of R, having graphics in the data analysis workflow has been much easier, albeit at the cost of the interactivity with graphics that matches the old systems. We are mostly using the R package, `tourr` (Wickham et al., 2011a) for examples in this book. It provides the machinery for running a tour, and has the flexibility that it can be ported, modified, and used as a regular element of data analysis.

Exercises

1. Randomly generate data points that are uniformly distributed in a hypercube of 3, 5 and 10 dimensions, with 500 points in each sample, using the `cube.solid.random` function of the `geozoo` package. What differences do we expect to see? Now visualise each set in a grand tour and describe how they differ, and whether this matched your expectations?
2. Use the `geozoo` package to generate samples from different shapes and use them to get a better understanding of how shapes appear in a grand tour. You can start with exploring the conic spiral in 3D, a torus in 4D and points along the wire frame of a cube in 5D.
3. For each of the challenge data sets, `c1`, \dots , `c12` from the `mulgar` package, use the grand tour to view and try to identify structure (outliers, clusters, non-linear relationships).

Chapter 2

Notation conventions and R objects

The data can be considered to be a matrix of numbers with the columns corresponding to variables, and the rows correspond to observations. It can be helpful to write this in mathematical notation, like:

$$X_{n \times p} = [X_1 \ X_2 \ \dots \ X_p]_{n \times p} = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1p} \\ X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & & \vdots \\ X_{n1} & X_{n2} & \dots & X_{np} \end{bmatrix}_{n \times p}$$

where X indicates the the $n \times p$ data matrix, X_j indicates variable $j, j = 1, \dots, p$ and X_{ij} indicates the value j^{th} variable of the i^{th} observation. (It can be confusing to distinguish whether one is referring to the observation or a variable, because X_i is used to indicate observation also. When this is done it is usually accompanied by qualifying words such as **observation** X_3 , or **variable** X_3 .)

When there is a response variable(s), it is common to consider X to be the predictors, and use Y to indicate the response variable(s). Y could be a matrix, also, and would be $n \times q$, where commonly $q = 1$. Y could be numeric or categorical, and this would change how it is handled with visualisation.

To make a low-dimensional projection (shadow) of the data, we need a projection matrix:

$$A_{p \times d} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1d} \\ A_{21} & A_{22} & \dots & A_{2d} \\ \vdots & \vdots & & \vdots \\ A_{p1} & A_{p2} & \dots & A_{pd} \end{bmatrix}_{p \times d}$$

A should be an orthonormal matrix, which means that the $\sum_{j=1}^p A_{jk}^2 = 1, k = 1, \dots, d$ (columns represent vectors of length 1) and $\sum_{j=1}^p A_{jk} A_{jl} = 0, k, l = 1, \dots, d$ (columns represent vectors that are orthogonal to each other).

Then the projected data is written as:

$$Y_{n \times d} = XA = \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1d} \\ y_{21} & y_{22} & \dots & y_{2d} \\ \vdots & \vdots & & \vdots \\ y_{n1} & y_{n2} & \dots & y_{nd} \end{bmatrix}_{n \times d}$$

where $y_{ij} = \sum_{k=1}^p X_{ik} A_{kj}$. Note that we are using Y as the projected data here, as well as it possibly being used for a response variable. Where necessary, this will be clarified with words in the text, when notation is used in explanations later.

When using R, if we only have the data corresponding to X it makes sense to use a `matrix` object. However, if the response variable is included and it is categorical, then we might use a `data.frame` or a `tibble` which can accommodate non-numerical values. Then to work with the data, we can use the base R methods:

```
X <- matrix(c(1.1, 1.3, 1.4, 1.2, 2.7, 2.6, 2.4, 2.5, 3.5, 3.4, 3.2, 3.6), ncol=4, byrow=TRUE)
X
      [,1] [,2] [,3] [,4]
[1,]  1.1  1.3  1.4  1.2
[2,]  2.7  2.6  2.4  2.5
[3,]  3.5  3.4  3.2  3.6
```

which is a data matrix with $n = 3, p = 4$ and to extract a column (variable):

```
X[,2]
[1] 1.3 2.6 3.4
```

or a row (observation):

```
X[2,]
```

```
[1] 2.7 2.6 2.4 2.5
```

or an individual cell (value):

```
X[3,2]
```

```
[1] 3.4
```

To make a projection we need an orthonormal matrix:

```
A <- matrix(c(0.707,0.707,0,0,0,0,0.707,0.707), ncol=2, byrow=FALSE)
A
```

```
      [,1] [,2]
[1,] 0.707 0.000
[2,] 0.707 0.000
[3,] 0.000 0.707
[4,] 0.000 0.707
```

You can check that it is orthonormal by

```
sum(A[,1]^2)
```

```
[1] 0.999698
```

```
sum(A[,1]*A[,2])
```

```
[1] 0
```

and make a projection using matrix multiplication:

```
X %*% A
```

```
      [,1] [,2]
[1,] 1.6968 1.8382
[2,] 3.7471 3.4643
[3,] 4.8783 4.8076
```

The seemingly magical number 0.707 used above and to create the projection in Figure ?? arises from normalising a vector with equal contributions from each variable, (1, 1). Dividing by `sqrt(2)` gives (0.707, 0.707).

The notation convention used is:

`n` = number of observations

p = number of variables, dimension of data

d = dimension of the projection

Exercises

1. Generate a matrix A with $p = 5$ (rows) and $d = 2$ (columns), where each value is randomly drawn from a standard normal distribution. Extract the element at row 3 and column 1.
2. We will interpret A as a projection matrix and therefore it needs to be orthonormalised. Use the function `tourr::orthonormalise` to do this, and explicitly check that each column is normalised and that the two columns are orthogonal now. Which dimensions contribute most to the projection for your A ?
3. Use matrix multiplication to calculate the projection of the `mulgar::clusters` data onto the 2D plane defined by A . Make a scatterplot of the projected data. Can you identify clustering in this view?

Part II

Dimension reduction

Chapter 3

Overview

This chapter will focus on methods for reducing dimension, and how the tour can be used to assist with the common methods such as principal component analysis (PCA), multidimensional scaling (MDS), t-stochastic neighbour embedding (t-SNE), and factor analysis.

Dimension is perceived in a tour using the spread of points. When the points are spread far apart, then the data is filling the space. Conversely when the points “collapse” into a sub-region then the data is only partially filling the space, and some dimension reduction to reduce to this smaller dimensional space may be worthwhile.

When points do not fill the plotting canvas fully, it means that it lives in a lower dimension.

Let’s start with some 2D examples. You need at least two variables to be able to talk about association between variables. Figure ?? shows three plots of two variables. Plot (a) shows two variables that are strongly linearly associated¹, because when x_1 is low, x_2 is low also, and conversely when x_1 is high, x_2 is also high. This can also be seen by the reduction in spread of points (or “collapse”) in one direction making the data fill less than the full square of the plot. *So from this we can conclude that the data is not fully 2D.* The second step is to infer which variables contribute to this reduction in dimension. The axes for x_1 and x_2 are drawn extending from (0,0) and because they both extend out of the cloud of points, in the direction away from the collapse of points we can say that they are jointly responsible for the dimension reduction.

Figure ?? shows a pair of variables that are **not** linearly associated. Variable x_1 is more varied than x_3 but knowing the value on x_1 tells us nothing about

¹It is generally better to use *associated* than *correlated*. Correlated is a statistical quantity, measuring linear association. The term *associated* can be prefaced with the type of association, such as *linear* or *non-linear*.

possible values on `x3`. Before running a tour all variables are typically scaled to have equal spread. The purpose of the tour is to capture association and relationships between the variables, so any univariate differences should be removed ahead of time. Figure ?? shows what this would look like when `x3` is scaled - the points are fully spread in the full square of the plot.

```
library(ggplot2)
library(tibble)
set.seed(6045)
x1 <- runif(123)
x2 <- x1 + rnorm(123, sd=0.1)
x3 <- rnorm(123, sd=0.2)
df <- tibble(x1 = (x1-mean(x1))/sd(x1),
             x2 = (x2-mean(x2))/sd(x2),
             x3,
             x3scaled = (x3-mean(x3))/sd(x3))
dp1 <- ggplot(df) +
  geom_point(aes(x=x1, y=x2)) +
  xlim(-2.5, 2.5) + ylim(-2.5, 2.5) +
  annotate("segment", x=0, xend=2, y=0, yend=0) +
  annotate("segment", x=0, xend=0, y=0, yend=2) +
  annotate("text", x=2.1, y=0, label="x1") +
  annotate("text", x=0, y=2.1, label="x2") +
  theme_minimal() +
  theme(aspect.ratio=1)
dp2 <- ggplot(df) +
  geom_point(aes(x=x1, y=x3)) +
  xlim(-2.5, 2.5) + ylim(-2.5, 2.5) +
  annotate("segment", x=0, xend=2, y=0, yend=0) +
  annotate("segment", x=0, xend=0, y=0, yend=2) +
  annotate("text", x=2.1, y=0, label="x1") +
  annotate("text", x=0, y=2.1, label="x3") +
  theme_minimal() +
  theme(aspect.ratio=1)
dp3 <- ggplot(df) +
  geom_point(aes(x=x1, y=x3scaled)) +
  xlim(-2.5, 2.5) + ylim(-2.5, 2.5) +
  annotate("segment", x=0, xend=2, y=0, yend=0) +
  annotate("segment", x=0, xend=0, y=0, yend=2) +
  annotate("text", x=2.1, y=0, label="x1") +
  annotate("text", x=0, y=2.1, label="x3") +
  theme_minimal() +
  theme(aspect.ratio=1)
```

Now let's think about what this looks like with five variables. Figure ?? shows

Figure 3.4: Explanation of how dimension reduction is perceived in 2D, relative to variables. When an axes extends out of a direction where the points are collapsed, it means that this variable is partially responsible for the reduced dimension.

a grand tour on five variables, with (a) showing data that is primarily 2D, (b) has data that is primarily 3D and (c) is fully 5D. You can see that both (a) and (b) the spread of points collapse in some projections, with it happening more in (a). In (c) the data is always spread out in the square, although it does seem to concentrate or pile in the centre. This piling is typical when projecting from high dimensions to low dimensions. The sage tour (Laa et al., 2020a) makes a correction for this.

```
library(mulgar)
data(plane)
data(box)
render_gif(plane,
            grand_tour(),
            display_xy(),
            gif_file="gifs/plane.gif",
            frames=500,
            width=200,
            height=200)

render_gif(box,
            grand_tour(),
            display_xy(),
            gif_file="gifs/box.gif",
            frames=500,
            width=200,
            height=200)

# Simulate full cube
library(geozoo)
cube5d <- data.frame(cube.solid.random(p=5, n=300)$points)
colnames(cube5d) <- paste0("x", 1:5)
cube5d <- data.frame(apply(cube5d, 2, function(x) (x-mean(x))/sd(x)))
render_gif(cube5d,
            grand_tour(),
            display_xy(),
            gif_file="gifs/cube5d.gif",
            frames=500,
            width=200,
            height=200)
```

The next step is to determine which variables contribute. In the examples just

Figure 3.5: Different dimensional planes - 2D, 3D, 5D - displayed in a grand tour projecting into 2D. Notice that the 5D in 5D always fills out the box (although it does concentrate some in the middle which is typical when projecting from high to low dimensions). Also you can see that the 2D in 5D, concentrates into a line more than the 3D in 5D. This suggests that it is lower dimensional.

provided, all variables are linearly associated in the 2D and 2D data. You can check this by making a scatterplot matrix.

```
library(GGally)
library(mulgar)
data(plane)
ggscatmat(plane)
```

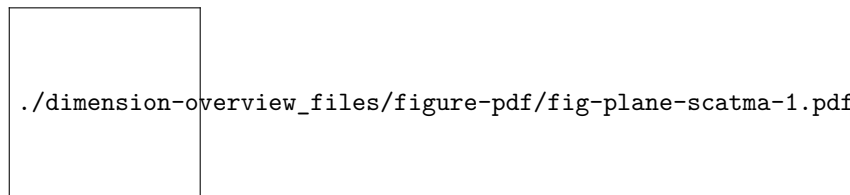


Figure 3.6: Scatterplot matrix of plane data. You can see that x1-x3 are strongly linearly associated, and also x4 and x5. When you watch the tour of this data, any time the data collapses into a line you should see only (x1, x2, x3) or (x4, x5). When combinations of x1 and x4 or x5 show, the data should be spread out.

To make an example where not all variables contribute, we have added two additional variables to the `plane` data set, which are purely noise.

```
# Add two pure noise dimensions to the plane
plane_noise <- plane
plane_noise$x6 <- rnorm(100)
plane_noise$x7 <- rnorm(100)
plane_noise <- data.frame(apply(plane_noise, 2, function(x) (x-mean(x))/sd(x)))
ggduo(plane_noise, columnsX = 1:5, columnsY = 6:7,
      types = list(continuous = "points")) +
  theme(aspect.ratio=1, axis.text = element_blank())
```

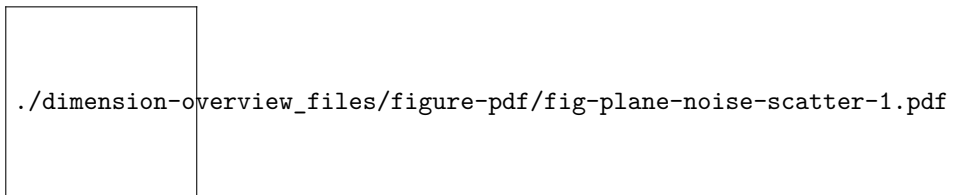


Figure 3.7: Additional noise variables are not associated with any of the first five variables.

Now we have 2D structure in 7D, but only five of the variables contribute to the 2D structure, that is, five of the variables are linearly related with each other. The other two variables (x6, x7) are not linearly related to any of the others.

We can still see the concentration of points along a line in some dimensions, which tells us that the data is not fully 7D. Then if you look closely at the variable axes you will see that the collapsing to a line only occurs when any of x1-x5 contribute strongly in the direction orthogonal to this. This does not happen when x6 or x7 contribute strongly to a projection - the data is always expanded to fill much of the space. That tells us that x6 and x7 don't substantially contribute to the dimension reduction, that is, they are not linearly related to the other variables.

```
library(ggplot2)
library(plotly)
library(htmlwidgets)

set.seed(78)
b <- basis_random(7, 2)
pn_t <- tourr::save_history(plane_noise,
                           tour_path = grand_tour(),
                           start = b,
                           max_bases = 8)
pn_t <- interpolate(pn_t, 0.1)
pn_anim <- render_anim(plane_noise,
                      frames=pn_t)

pn_gp <- ggplot() +
  geom_path(data=pn_anim$circle,
           aes(x=c1, y=c2,
              frame=frame), linewidth=0.1) +
  geom_segment(data=pn_anim$axes,
             aes(x=x1, y=y1,
                xend=x2, yend=y2,
                frame=frame),
             linewidth=0.1) +
```

```

geom_text(data=pn_anim$axes,
          aes(x=x2, y=y2,
              frame=frame,
              label=axis_labels),
          size=5) +
geom_point(data=pn_anim$frames,
           aes(x=P1, y=P2,
               frame=frame),
           alpha=0.8) +
xlim(-1,1) + ylim(-1,1) +
coord_equal() +
theme_bw() +
theme(axis.text=element_blank(),
      axis.title=element_blank(),
      axis.ticks=element_blank(),
      panel.grid=element_blank())
pn_tour <- ggplotly(pn_gp,
                   width=500,
                   height=550) %>%
animation_button(label="Go") %>%
animation_slider(len=0.8, x=0.5,
                 xanchor="center") %>%
animation_opts(easing="linear",
               transition = 0)

htmlwidgets::saveWidget(pn_tour,
                        file="html/plane_noise.html",
                        selfcontained = TRUE)

```

Figure 3.8: Grand tour of the plane with two additional dimensions of pure noise. The collapsing of the points indicates that this is not fully 7D. This only happens when any of x_1 - x_5 are contributing strongly (frame 49 x_4 , x_5 ; frame 79 x_1 ; frame 115 x_2 , x_3). If x_6 or x_7 are contributing strongly the data is spread out fully. This tells us that x_6 and x_7 are not linearly associated, but other variables are.

To determine which variables are responsible for the reduced dimension look for the axes that extend out of the point cloud.

The simulated data here is very simple, and what we have learned from the tour could also be learned from principal component analysis. However, if there are small complications, such as outliers or nonlinear relationships, that might not be visible from principal component analysis, the tour can help you to see them.

Figure ?? and ?@fig-outlier show example data with an outlier and ?@fig-

nonlinear shows data with non-linear relationships.

```
# Add several outliers to the plane_noise data
plane_noise_outliers <- plane_noise
plane_noise_outliers[101,] <- c(2, 2, -2, 0, 0, 0, 0)
plane_noise_outliers[102,] <- c(0, 0, 0, -2, -2, 0, 0)

ggscatmat(plane_noise_outliers, columns = 1:5) +
  theme(aspect.ratio=1, axis.text = element_blank())
```

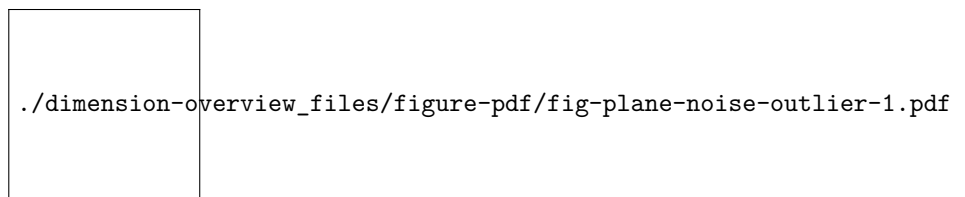


Figure 3.9: Outliers added to the plane with noise data.

```
render_gif(plane_noise_outliers,
  grand_tour(),
  display_xy(),
  gif_file="gifs/pn_outliers.gif",
  frames=500,
  width=200,
  height=200)

data(plane_nonlin)
render_gif(plane_nonlin,
  grand_tour(),
  display_xy(),
  gif_file="gifs/plane_nonlin.gif",
  frames=500,
  width=200,
  height=200)
```

Figure 3.10: Examples of different types of dimensionality issues: outliers and nonlinearity. In the left plot, you can see two points far from the others in some projections. Also the two can be seen with different movement patterns, moving faster than other points during the tour. Outliers will affect detection of reduced dimension, but it is easy to ignore with the tour. Non-linear relationships may not be captured by other techniques but are visible with the tour.

Exercises

1. Multicollinearity is when the predictors for a model are strongly linearly associated. It can adversely affect the fitting of most models, because many possible models may be equally as good. Variable importance might be masked by correlated variables, and confidence intervals generated for linear models might be too wide. Check the for multicollinearity or other associations between the predictors in:
 - a. 2001 Australian election data
 - b. 2016 Australian election data
2. Examine 5D multivariate normal samples drawn from populations with a range of variance-covariance matrices. (You can use the `mvtnorm` package to do the sampling, for example.) Examine the data using a grand tour. What changes when you change the correlation from close to zero to close to 1? Can you see a difference between strong positive correlation and strong negative correlation?

Chapter 4

Principal component analysis

Reducing dimensionality using principal component analysis (PCA) dates back to Pearson (1901) and Hotelling (1933), and Jolliffe & Cadima (2016) provides a current overview. The goal is to find a smaller set of variables, $q(< p)$, that contain as much information as the original as possible. The new set of variables, known as principal components (PCs), are linear combinations of the original variables. The PCs can be used to represent the data in a lower-dimensional space.

The process is essentially an optimisation procedure, although PCA has an analytical solution. It solves the problem of

$$\max_{a_k} \text{Var}(Xa_k),$$

where X is the $n \times p$ data matrix, $a_k(k = 1, \dots, p)$ is a 1D projection vector, called an eigenvector, and the $\text{Var}(Xa_k)$ is called an eigenvalue. So PCA is a sequential process, that will find the direction in the high-dimensional space (as given by the first eigenvector) where the data is most varied, and then find the second most varied direction, and so on. The eigenvectors define the combination of the original variables, and the eigenvalues define the amount of variance explained by the reduced number of variables.

PCA is very broadly useful for summarising linear association by using combinations of variables that are highly correlated. However, high correlation can also occur when there are outliers, or clustering. PCA is commonly used to detect these patterns also.

With visualisation we want to assess whether it is appropriate to use PCA to summarise any linear association by using combinations of variables that are

highly correlated. It can help to detect other patterns that might affect the PCA results such as outliers, clustering or non-linear dependence.

PCA is not very effective when the distribution of the variables is highly skewed, so it can be helpful to transform variables to make them more symmetrically distributed before conducting PCA. It is also possible to summarise different types of structure by generalising the optimisation criteria to any function of projected data, $f(XA)$, which is called *projection pursuit* (PP). PP has a long history (Kruskal (1964a), Friedman & Tukey (1974), Diaconis & Freedman (1984b), Jones & Sibson (1987), Huber (1985)), and there are regularly new developments (e.g. Lee & Cook (2009), Perisic & Posse (2005), Y. D. Lee et al. (2013), Loperfido (2018), Bickel et al. (2018), Zhang et al. (2023)).

4.1 Determining how many dimensions

We would start by examining the data using a grand tour. The goal is to check whether there might be potential issues for PCA, such as skewness, outliers or clustering, or even non-linear dependencies.

We'll start by showing PCA on the simulated data from Chapter ???. The scree plots show that PCA supports that the data are 2D, 3D and 5D respectively.

```
library(dplyr)
library(ggplot2)
library(mulgar)
data(plane)
data(box)
library(geozoo)
cube5d <- data.frame(cube.solid.random(p=5, n=300)$points)
colnames(cube5d) <- paste0("x", 1:5)
cube5d <- data.frame(apply(cube5d, 2, function(x) (x-mean(x))/sd(x)))
p_pca <- prcomp(plane)
b_pca <- prcomp(box)
c_pca <- prcomp(cube5d)
p_scree <- ggscree(p_pca) + theme_minimal()

b_scree <- ggscree(b_pca) + theme_minimal()
c_scree <- ggscree(c_pca) + theme_minimal()
```

The next step is to look at the coefficients for the selected number of PCs. Table ?? shows the coefficients for the first two PCs of the `plane` data. All five variables contribute, with `x1`, `x2`, `x3` contributing more to PC1, and `x4`, `x5` contributing more to PC2. Table ?? shows the coefficients for the first three PCs. Variables `x1`, `x2`, `x3` contribute strongly to PC1, PC2 has contributions from all variables except `x3` and variables `x4` and `x5` contribute strongly to PC3.

Figure 4.4: Scree plots for the three simulated data sets. The 2D in 5D is clearly recognised by PCA to be 2D because the variance drops substantially between 2-3 principal components. The 3D in 5D is possibly 3D because the variance drops from 3-4 principal components. The fully 5D data has no drop in variance, and all values are close to the typical value one would observe if the data was fully 5D.

```
library(gt)
p_pca$rotation[,1:2] %>%
  as_tibble(rownames="Variable") %>%
  gt() %>%
  fmt_number(columns = c(PC1, PC2),
              decimals = 2)
```

Table 4.1: Coefficients for the first two PCs for the plane data.

Variable	PC1	PC2
x1	0.58	-0.06
x2	-0.55	0.21
x3	0.47	-0.41
x4	0.25	0.64
x5	-0.29	-0.62

```
b_pca$rotation[,1:3] %>%
  as_tibble(rownames="Variable") %>%
  gt() %>%
  fmt_number(columns = c(PC1, PC2, PC3),
              decimals = 2)
```

Table 4.2: Coefficients for the first three PCs for the box data.

Variable	PC1	PC2	PC3
x1	-0.51	0.46	-0.11
x2	0.51	0.46	0.00
x3	-0.65	-0.09	-0.23
x4	-0.22	0.36	0.87
x5	0.02	0.66	-0.43

In each of these simulated data sets, all five variables contributed to the dimension reduction. If we added two purely noise variables to the plane data, as done in Chapter ??, the scree plot would indicate that the data is now 4D, and we would get a different interpretation of the coefficients from the PCA. We see that PC1 and PC2 are approximately the same as before, with main variables being (x1, x2, x3) and (x4, x5) respectively. PC3 and PC4 are both x6 and x7.

```
set.seed(5143)
plane_noise <- plane
```

```

plane_noise$x6 <- rnorm(100)
plane_noise$x7 <- rnorm(100)
plane_noise <- data.frame(apply(plane_noise, 2, function(x) (x-mean(x))/sd(x)))

pn_pca <- prcomp(plane_noise)
ggscree(pn_pca) + theme_minimal()

```



Figure 4.5: Additional noise variables expands the data to 4D.

```

pn_pca$rotation[,1:4] %>%
  as_tibble(rownames="Variable") %>%
  gt() %>%
  fmt_number(columns = c(PC1, PC2, PC3, PC4),
             decimals = 2)

```

Table 4.3: Coefficients for the first four PCs for the box data.

Variable	PC1	PC2	PC3	PC4
x1	0.58	0.04	-0.01	0.00

x2	-0.55	-0.18	0.03	0.07
x3	0.47	0.37	-0.05	-0.20
x4	0.24	-0.62	0.06	0.17
x5	-0.28	0.60	-0.07	-0.14
x6	0.05	0.29	0.58	0.76
x7	-0.02	-0.08	0.81	-0.58

4.1.1 Example: pisa

The `pisa` data contains simulated data from math, reading and science scores, totalling 30 variables. PCA is used here to examine the association. We might expect that it is 3D, but what we see suggests it is primarily 1D. This means that a student that scores well in math, will also score well in reading and science.

```
data(pisa)
pisa_std <- pisa %>%
  filter(CNT == "Australia") %>%
  select(-CNT) %>%
  mutate_all(mulgar::scale2)
pisa_pca <- prcomp(pisa_std)
pisa_scee <- ggscree(pisa_pca) + theme_minimal()
```

The scree plot in Figure ?? shows a big drop from one to two PCs in the amount of variance explained. A grand tour on the 30 variables can be run using `animate_xy()`:

```
animate_xy(pisa_std, half_range=1)
```

or rendered as an animated gif using `render_gif()`:

```
render_gif(pisa_std,
  grand_tour(),
  display_xy(half_range=0.9),
  gif_file="gifs/pisa_gt.gif",
  frames=500,
  width=400,
  height=400,
  loop=FALSE)
```

and we can see that the data is elliptical in most projections, sometimes shrinking to be a small circle. This pattern strongly indicates that there is one primary direction of variation in the data, with only small variation in any direction away from it. Shrinking to the small circle is analogous to how *a pencil or cigar or water bottle in 3D looks from some angles*.

Figure 4.7: Scree plot and tour of the pisa data, with 30 variables being the plausible scores for Australian students.

The coefficients of the first PC (first eigenvector) are roughly equal in magnitude (as shown below), which tells us that all variables roughly contribute. Interestingly, they are all negative, which is not actually meaningful. With different software these could easily have been all positive. The sign of the coefficients can be reversed, as long as all are reversed, which is the same as an arrow pointing one way, changing and pointing the other way.

```
round(pisa_pca$rotation[,1], 2)
```

PV1MATH	PV2MATH	PV3MATH	PV4MATH	PV5MATH	PV6MATH	PV7MATH	PV8MATH
-0.18	-0.18	-0.18	-0.18	-0.18	-0.18	-0.18	-0.18
PV9MATH	PV10MATH	PV1READ	PV2READ	PV3READ	PV4READ	PV5READ	PV6READ
-0.18	-0.18	-0.19	-0.18	-0.19	-0.19	-0.19	-0.19
PV7READ	PV8READ	PV9READ	PV10READ	PV1SCIE	PV2SCIE	PV3SCIE	PV4SCIE
-0.19	-0.19	-0.19	-0.19	-0.18	-0.18	-0.19	-0.18
PV5SCIE	PV6SCIE	PV7SCIE	PV8SCIE	PV9SCIE	PV10SCIE		
-0.19	-0.18	-0.19	-0.18	-0.19	-0.18		

The 1D structure of the pisa data means that a student that scores well in math, tends to score well in reading and science too.

4.1.2 Example: aflw

This data has player statistics for all the matches in the 2021 season. We would be interested to know which variables contain similar information, and thus might be combined into single variables. We would expect that many statistics to group into a few small sets, such as offensive and defensive skills. We might also expect that some of the statistics are skewed, most players have low values and just a handful of players are stellar. It is also possible that there are some extreme values. These are interesting features, but they will distract from the main purpose of grouping the statistics. Thus the tour is used to check for potential problems with the data prior to conducting PCA.

```
library(tourr)
data(aflw)
aflw_std <- aflw %>%
  mutate_if(is.numeric, function(x) (x-
    mean(x, na.rm=TRUE))/
    sd(x, na.rm=TRUE)))
```

To look at all of the 29 player statistics in a grand tour.


```
animate_xy(aflw_std[,7:35], half_range=0.9)
render_gif(aflw_std[,7:35],
           grand_tour(),
           display_xy(half_range=0.9),
           gif_file="gifs/aflw_gt.gif",
           frames=500,
           loop=FALSE)
```

No major surprises! There is a small amount of skewness, and there are no major outliers. Skewness indicates that most players have reasonably similar skills (bunching of points), except for some key players (the moderate outliers). The skewness could be reduced by applying a log or square root transformation to some variables prior to running the PCA. However, we elect not to do this because the moderate outliers are of interest. These correspond to talented players that we'd like to explore further with the analysis.

Below we have the conventional summary of the PCA, a scree plot showing the reduction in variance to be explained when each additional PC is considered. It is also conventional to look at a table summarising the proportions of variance explained by PCs, but with 30 variables it is easier to make some decision on the number of PCs needed based on the scree plot.

```
aflw_pca <- prcomp(aflw_std[,7:35],
                  scale = FALSE,
                  retx=TRUE)

ggscree(aflw_pca) + theme_minimal()
```

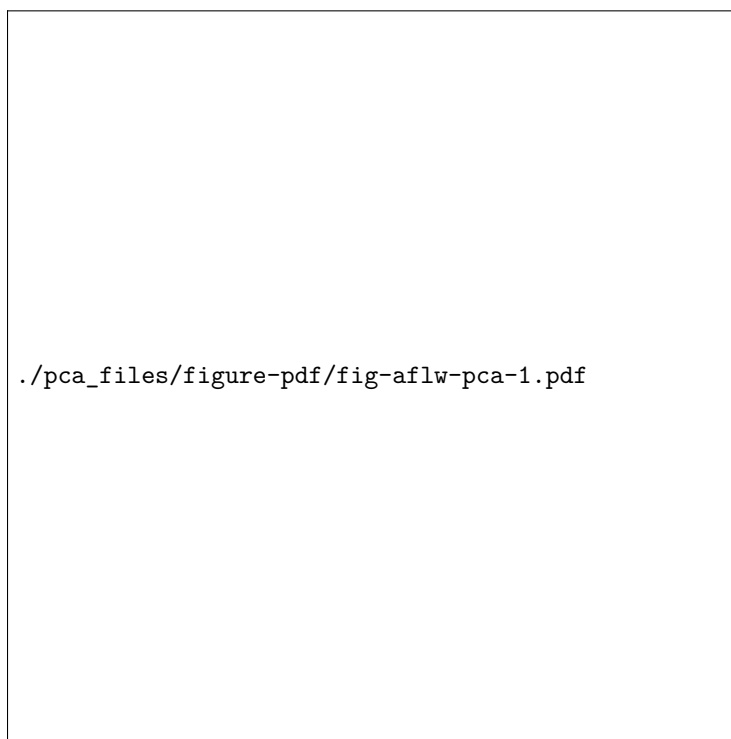


Figure 4.8: Scree plot showing decay in variance of PCs.

From the scree plot in Figure ??, we see a sharp drop from one to two, two to three and then smaller drops. After four PCs the variance drops again at six PCs and then gradually decays. We will choose four PCs to examine more closely. This explains 67.2% of the variance.

```
library(gt)
aflw_pca$rotation[,1:4] %>%
  as_tibble(rownames="Variable") %>%
  arrange(desc(PC1), desc(PC2), desc(PC3)) %>%
  gt() %>%
  fmt_number(columns = c(PC1, PC2, PC3, PC4),
              decimals = 2)
```

Table 4.4: Coefficients for the first four PCs.

Variable	PC1	PC2	PC3	PC4
disposals	0.31	−0.05	−0.03	0.07
possessions	0.31	−0.03	−0.07	0.09

kicks	0.29	-0.04	0.09	-0.12
metres	0.28	-0.03	0.10	-0.15
contested	0.28	0.01	-0.12	0.23
uncontested	0.28	-0.06	-0.01	-0.05
turnovers	0.27	-0.01	-0.01	-0.29
clearances	0.23	0.00	-0.29	0.19
clangers	0.23	-0.02	-0.06	-0.33
handballs	0.23	-0.04	-0.19	0.31
frees_for	0.21	0.02	-0.13	0.18
marks	0.21	0.03	0.32	0.02
tackles	0.20	0.01	-0.28	0.09
time_pct	0.16	-0.04	0.35	-0.02
intercepts	0.13	-0.28	0.24	0.03
rebounds_in50	0.13	-0.28	0.24	-0.06
frees_against	0.13	0.03	-0.16	-0.23
assists	0.09	0.23	0.00	0.05
bounces	0.09	0.03	0.02	-0.28
behinds	0.09	0.32	0.08	-0.02
shots	0.08	0.38	0.12	-0.03
tackles_in50	0.07	0.27	-0.18	0.03
marks_in50	0.06	0.34	0.18	0.04
contested_marks	0.05	0.16	0.34	0.15
goals	0.04	0.37	0.16	0.03
accuracy	0.04	0.34	0.10	0.06
one_pct	0.03	-0.21	0.33	0.08
disposal	0.02	-0.13	0.20	0.50
hitouts	-0.04	0.00	-0.03	0.32

When there are as many variables as this, it can be hard to digest the combinations of variables most contributing to each PC. Rearranging the table by sorting on a selected PC can help. Table ?? has been sorted according to the PC 1 coefficients.

PC 1 is primarily composed of **disposals**, **possessions**, **kicks**, **metres**, **uncontested**, **contested**, Actually almost all variables positively contribute, albeit in different amounts! It is quite common in PCA for the first PC to be a combination of all variables, although it might commonly be a closer to equal contribution, and it tells us that there is one main direction of variation in the data. For PC 1 in the AFLW data, PCA is telling us that the primary variation is through a combination of skills, and this maps to basic football playing skills, where some skills (e.g. disposals, possessions, kicks, ...) are more important.

Thus the second PC might be the more interesting. PC 2 is primarily a combination of **shots**, **goals**, **marks_in50**, **accuracy**, and **behinds** contrasted against **rebounds_in50** and **intercepts**. The negative coefficients are primary offensive

skills and the positive coefficients are defensive skills. This PC is reasonable measure of the offensive vs defensive skills of a player.

We would continue to interpret each PC by examining large coefficients to help decide how many PCs are a suitable summary of the information in the data. Briefly, PC 3 is a measure of worth of the player because `time_pct` has a large coefficient, so players that are on the field longer will contribute strongly to this new variable. It also has large (and opposite) contributions from `clearances`, `tackles`, `contested_marks`. PC 4 appears to be related to aggressive play with `clangers`, `turnovers`, `bounces` and `frees_against` featuring. So all four PCs have useful information. (Note, if we had continued to examine large coefficients on PC 5 we would find that all variables already have had reasonably large coefficients on PC 1-4, which supports restricting attention to the first four.)

Ideally, when we tour the four PCs, we'd like to be able to stop and identify players. This involves creating a pre-computed animation, with additional mouse-over. This is only feasible with a small number of observations, like the AFLW data, because all of the animation frames are constructed in a single object and passed to `plotly`. This object gets large very quickly!

```
library(plotly)
library(htmlwidgets)
set.seed(20)
b <- basis_random(4, 2)
aflw_pct <- tourr::save_history(aflw_pca$x[,1:4],
                              tour_path = grand_tour(),
                              start = b,
                              max_bases = 5)
# To reconstruct projected data plots, later
save(aflw_pct, file="data/aflw_pct.rda")
aflw_pcti <- interpolate(aflw_pct, 0.1)
aflw_anim <- render_anim(aflw_pca$x[,1:4],
                        frames=aflw_pcti,
                        obs_labels=paste0(aflw$surname,
                                          aflw$given_name))

aflw_gp <- ggplot() +
  geom_path(data=aflw_anim$circle,
            aes(x=c1, y=c2,
                frame=frame), linewidth=0.1) +
  geom_segment(data=aflw_anim$axes,
               aes(x=x1, y=y1,
                   xend=x2, yend=y2,
                   frame=frame),
               linewidth=0.1) +
  geom_text(data=aflw_anim$axes,
```

```

      aes(x=x2, y=y2,
          frame=frame,
          label=axis_labels),
      size=5) +
  geom_point(data=aflw_anim$frames,
            aes(x=P1, y=P2,
                frame=frame,
                label=obs_labels),
            alpha=0.8) +
  xlim(-1,1) + ylim(-1,1) +
  coord_equal() +
  theme_bw() +
  theme(axis.text=element_blank(),
        axis.title=element_blank(),
        axis.ticks=element_blank(),
        panel.grid=element_blank())
aflw_pctour <- ggplotly(aflw_gp,
                      width=500,
                      height=550) %>%
  animation_button(label="Go") %>%
  animation_slider(len=0.8, x=0.5,
                  xanchor="center") %>%
  animation_opts(easing="linear", transition = 0)

htmlwidgets::saveWidget(aflw_pctour,
                        file="html/aflw_pca.html",
                        selfcontained = TRUE)

```

Figure 4.9: Animation of AFLW four PCs with interactive labelling.

From Figure ?? the shape of the four PCs is similar to that of all the variables, bunching of points in the centre with a lot of moderate outliers.

```

library(plotly)
load("data/aflw_pct.rda")
aflw_pcti <- interpolate(aflw_pct, 0.1)
f18 <- matrix(aflw_pcti[,18], ncol=2)
p18 <- render_proj(aflw_pca$x[,1:4], f18,
                  obs_labels=paste0(aflw$surname,
                                    aflw$given_name))

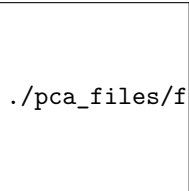
pg18 <- ggplot() +
  geom_path(data=p18$circle, aes(x=c1, y=c2)) +
  geom_segment(data=p18$axes, aes(x=x1, y=y1, xend=x2, yend=y2)) +

```

```

geom_text(data=p18$axes, aes(x=x2, y=y2, label=rownames(p18$axes))) +
geom_point(data=p18$data_prj, aes(x=P1, y=P2, label=obs_labels)) +
xlim(-1,1) + ylim(-1, 1) +
#ggtitle("Frame 18") +
theme_bw() +
theme(
  axis.text=element_blank(),
  axis.title=element_blank(),
  axis.ticks=element_blank(),
  panel.grid=element_blank())
ggplotly(pg18, width=650, height=650)

```



./pca_files/figure-pdf/fig-aflw-pcaplots-1.pdf

Figure 4.10: Frame 18 replotted so that players can be identified on mouseover.

For any particular frame, like 18 re-plotted in Figure ??, we can investigate further. Here there is a branching pattern, where the branch points in the direction of PC 1. Mouseover the players at the tip of this branch and we find players like Alyce Parker, Brittany Bonnici, Dana Hooker, Kiara Bowers. If you look up the bios of these players you'll find they all have generally good player descriptions like "elite disposals", "powerful left foot", "hard-running midfielder", "best and fairest".

In the direction of PC 2, you'll find players like Lauren Ahrens, Stacey Livingstone who are star defenders. Players in this end of PC 1, have high scores on `intercepts` and `rebounds_in50`.

Another interesting frame for inspecting PC 2 is 59. PC 2 at one end has players with high goal scoring skills, and the other good defending skills. So mousing over the other end of PC 2 finds players like Gemma Houghton and Katie Brennan who are known for their goal scoring. The branch pattern is an interesting one, because it tells us there is some combination of skills that are lacking among all players, primarily this appears to be there some distinction between defenders skills and general playing skills. It's not as simple as this because the branching is only visible when PC 1 and PC 2 are examined with PC 3.

PCA is useful for getting a sense of the variation in a high-dimensional data set. Interpreting the principal components is often useful, but it can be discom-bobulating. For the AFLW data it would be good to think about it as a guide

to the main directions of variation and to follow with a more direct engineering of variables into interesting player characteristics. For example, calculate offensive skill as an equal combination of goals, accuracy, shots, behinds. A set of new variables specifically computed to measure particular skills would make explaining an analysis easier.

PCA on the aflw data leaves us slightly dizzy, but there are some useful insights. It detects and even ranks outstanding players high on different combinations of skills. Although players tend to have a combination of skills, there appears to be a dichotomy in skill sets for top goal scorers and top defensive players.

4.2 Examining the PCA model in the data space

When you choose a smaller number of PCs (k) than the number of original variables, this is essentially producing a model for the data. The model is the lower dimensional k -D space. It is analogous to a linear regression model, except that the residuals from the model are $(p - k)$ -D.

It is common to show the model, that is the data projected into the k -D model space. When $k = 2$ this is called a “biplot”. For the `plane` and `plane_noise` data the biplots are shown in Figure ?? . This is useful for checking which variables contribute most to the new principal component variables, and also to check for any problems that might have affected the fit, such as outliers, clusters or non-linearity. Interestingly, biplots are typically only made in 2D, even if the data should be summarised by more than two PCs. Occasionally you will see the biplot made for PC j vs PC k also. With the `pca_tour()` function in the `tourr` package you can view a k -D biplot. This will display the k PCs with the axes displaying the original variables, and thus see their contribution to the PCs.

```
library(ggfortify)
library(patchwork)
plane_pca <- prcomp(plane)
p11 <- autoplot(plane_pca, loadings = TRUE,
               loadings.label = TRUE) +
  ggtitle("(a)") +
  theme_minimal() +
  theme(aspect.ratio=1)
plane_noise_pca <- prcomp(plane_noise)
p12 <- autoplot(plane_noise_pca, loadings = TRUE,
               loadings.label = TRUE) +
  ggtitle("(b)") +
  theme_minimal() +
  theme(aspect.ratio=1)
p11 + p12
```

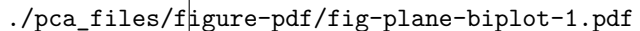


Figure 4.11: Biplots of the plane (a) and plane + noise (b) data. All five variables contribute strongly to the two principal components in (a): PC1 is primarily **x1**, **x2** and **x3** and PC2 is primarily **x4** and **x5**. In (b) the same four variables contribute in almost the same way, with variables **x6** and **x7** contributing very little. The data was constructed this way, that these two dimensions were purely noise.

It can be useful to examine this model using the tour. The model is simply a plane in high dimensions. This would be considered to be the model in the data space. The reason to do this is to check how well the model fits the data. The plane corresponding to the model should be oriented along the main direction of the points, and the spread of points around the plane should be small. We should also be able to see if there has been any strong non-linear relationship missed by the model, or outliers and clusters.

The function `pca_model()` from the `mulgar` package can be used to represent the model as a k -D wireframe plane. Figure ?? shows the models for the `plane` and `box` data, 2D and 3D respectively.

The purpose of looking at the model in the data space is to check how well the model fits the data.

```
plane_m <- pca_model(plane_pca)
plane_m_d <- rbind(plane_m$points, plane)
animate_xy(plane_m_d, edges=plane_m$edges,
           axes="bottomleft",
           edges.col="#E7950F",
           edges.width=3)
render_gif(plane_m_d,
           grand_tour(),
           display_xy(half_range=0.9,
                    edges=plane_m$edges,
                    edges.col="#E7950F",
                    edges.width=3),
           gif_file="gifs/plane_model.gif",
           frames=500,
           width=400,
           height=400,
```



```

        loop=FALSE)
box_pca <- prcomp(box)
box_m <- pca_model(box_pca, d=3)
box_m_d <- rbind(box_m$points, box)
animate_xy(box_m_d, edges=box_m$edges,
           axes="bottomleft", edges.col="#E7950F", edges.width=3)
render_gif(box_m_d,
           grand_tour(),
           display_xy(half_range=0.9,
                     edges=box_m$edges,
                     edges.col="#E7950F",
                     edges.width=3),
           gif_file="gifs/box_model.gif",
           frames=500,
           width=400,
           height=400,
           loop=FALSE)

```

Figure 4.12: PCA model overlaid on the data for the 2D in 5D, and 3D in 5D simulated data.

4.2.1 Example: pisa

The model for the `pisa` data is a 1D vector, shown in `?@fig-pisa-model`.

```

pisa_model <- pca_model(pisa_pca, d=1, s=2)

pisa_all <- rbind(pisa_model$points, pisa_std)
animate_xy(pisa_all, edges=pisa_model$edges,
           edges.col="#E7950F", edges.width=3)
render_gif(pisa_all,
           grand_tour(),
           display_xy(half_range=0.9,
                     edges=pisa_model$edges,
                     edges.col="#E7950F",
                     edges.width=5),
           gif_file="gifs/pisa_model.gif",
           frames=500,
           width=400,
           height=400,
           loop=FALSE)

```

4.2.2 Example: aflw

It is less useful to examine the PCA model for the aflw data, because the main patterns that were of interest were the exceptional players. However, we will do it anyway! `fig-aflw-model` shows the 4D PCA model overlain on the data. Even though the distribution of points is not as symmetric and balanced as the other examples, we can see that the cube structure mirrors the variation. We can see that the relationships between variables are not strictly linear, because the spread extends unevenly away from the box.

```
aflw_model <- pca_model(aflw_pca, d=4, s=1)

aflw_all <- rbind(aflw_model$points, aflw_std[,7:35])
animate_xy(aflw_all, edges=aflw_model$edges,
           edges.col="#E7950F",
           edges.width=3,
           half_range=0.8,
           axes="off")
render_gif(aflw_all,
           grand_tour(),
           display_xy(half_range=0.8,
                     edges=aflw_model$edges,
                     edges.col="#E7950F",
                     edges.width=3,
                     axes="off"),
           gif_file="gifs/aflw_model.gif",
           frames=500,
           width=400,
           height=400,
           loop=FALSE)
```

4.3 When relationships are not linear

4.3.1 Example: outliers

Figure ?? shows the scree plot for the planar data with noise and outliers. It is very similar to the scree plot on the data without the outliers (Figure ??). However, what we see from Figure ?? is that PCA loses the outliers. The animation in (a) shows the full data, and the outliers marked by colour and labels 1, 2, are clearly unusual in some projections. When we examine the tour of the first four PCs (as suggested by the scree plot) the outliers are not unusual. They are almost contained in the point cloud. The reason is clear when all the PCs are plotted, and the outliers can be seen to be clearly detected only in PC5, PC6 and PC7.

```
plane_n_o_pca <- prcomp(plane_noise_outliers)
ggscree(plane_n_o_pca) + theme_minimal()
```



Figure 4.13: Scree plot of the planar data with noise and an outlier. It is almost the same as the data without the outliers.

```
clrs <- hcl.colors(12, "Zissou 1")
p_col <- c(rep("black", 100), clrs[11], clrs[11])
p_obs_labels <- c(rep("", 100), "1", "2")

animate_xy(plane_n_o_pca$x[,1:4],
           col=p_col,
           obs_labels=p_obs_labels)
animate_xy(plane_noise_outliers,
           col=p_col,
           obs_labels=p_obs_labels)
render_gif(plane_noise_outliers,
           grand_tour(),
           display_xy(half_range=0.8,
```

```

        col=p_col,
        obs_labels=p_obs_labels),
    gif_file="gifs/plane_n_o_clr.gif",
    frames=500,
    width=200,
    height=200,
    loop=FALSE)
render_gif(plane_n_o_pca$x[,1:4],
  grand_tour(),
  display_xy(half_range=0.8,
    col=p_col,
    obs_labels=p_obs_labels),
  gif_file="gifs/plane_n_o_pca.gif",
  frames=500,
  width=200,
  height=200,
  loop=FALSE)

```

Figure 4.14: Examining the handling of outliers in the PCA of the planar data with noise variables and two outliers. PCA has lost these two extreme values.

```

library(GGally)
ggscatmat(plane_n_o_pca$x) + theme_minimal()

```

./pca_files/figure-pdf/fig-plane-o-n-pairs-1.pdf

Figure 4.15: From the scatterplot matrix we can see that the outliers are present in PC5, PC6 and PC7. That means by reducing the dimensionality to the first four PCs the model has missed some important characteristics in the data.

4.3.2 Example: Non-linear associations

Figure ?? shows the tour of the full 5D data containing non-linear relationships in comparison with a tour of the first three PCs, as recommended by the scree plot (Figure ??). The PCs capture some clear and very clean non-linear relationship, but it looks like it has missed some of the complexities of the relationships. The scatterplot matrix of all 5 PCs (Figure ??) shows that PC4 and PC5 contain interesting features: more non-linearity, and curiously an outlier.

```
data(plane_nonlin)
plane_nonlin_pca <- prcomp(plane_nonlin)
ggscree(plane_nonlin_pca) + theme_minimal()
```



Figure 4.16: Scree plot of the non-linear data suggests three PCs.

```
animate_xy(plane_nonlin_pca$x[,1:3])
render_gif(plane_nonlin_pca$x[,1:3],
  grand_tour(),
  display_xy(half_range=0.8),
  gif_file="gifs/plane_nonlin_pca.gif",
  frames=500,
  width=200,
  height=200)
```

Figure 4.17: Comparison of the full data and first three principal components. Some of the non-linearity is clearly visible in the reduced dimension space, but the full data has more complexities.

```
ggscatmat(plane_nonlin_pca$x)
```

./pca_files/figure-pdf/fig-plane-nonlin-pairs-1.pdf

Figure 4.18: From the scatterplot matrix we can see that there is a non-linear relationship visible in PC1 and PC2, with perhaps a small contribution from PC3. However, we can see that when the data is reduced to three PCs, it misses catching all on the non-linear relationships and also interestingly it seems that there is an unusual observation also.

One of the dangers of PCA is that interesting and curious details of the data only emerge in the lowest PCs, that are usually discarded. The tour, and examining the smaller PCs, can help to discover them.

Exercises

1. Make a scatterplot matrix of the first four PCs of the **aflw** data. Is the branch pattern visible in any pair?
2. Construct five new variables to measure these skills offense, defense, playing time, ball movement, errors. Using the tour, examine the relationship between these variables. Map out how a few players could be characterised based on these directions of skills.
3. Symmetrise any **aflw** variables that have skewed distributions using a log or square root transformation. Then re-do the PCA. What do we learn that is different about associations between the skill variables?
4. Examine the **bushfires** data using a grand tour on the numeric variables, ignoring the **cause** (class) variable. Would it be important to transform any variables to remove skewness? Do so, if necessary. How many principal components would be recommended? Examine the PCA model with the data.
5. Use the **pca_tour** to examine the three PCs. How do all of the variables contribute to this reduced space?

Project

Linear dimension reduction can optimise for other criteria, and here we will explore one example: the algorithm implemented in the **dobin** package finds a basis in which the first few directions are optimized for the detection of outliers in

the data. We will examine how it performs for the `plane_noise_outliers` data (the example where outliers were hidden in the first four principal components.)

1. Start by looking up the documentation of `dobin::dobin`. How many parameters does the method depend on?
2. We first apply the function to the `plane_noise_outliers` data using default values for all parameters.
3. Recall that the outliers were added in rows 101 and 102 of the data. Make a scatter plots showing the projection onto the first, second and third component, using color to highlight the outliers. Are they visible as outliers with three components?
4. Adjust the `frac` parameter of the `dobin` function to `frac = 0.99` and repeat the graphical evaluation from point 3. How does it compare to the previous solution?
5. XXXX show the tour of the full data with the model (first three components from `dobin`)

Chapter 5

Non-linear dimension reduction

5.1 Background

Non-linear dimension reduction (NLDR) aims to find a low-dimensional representation of the high-dimensional data that shows the main features of the data. In statistics, it dates back to Kruskal (1964a)’s work on multidimensional scaling (MDS). Some techniques only require an interpoint similarity or distance matrix as the main ingredient, rather than the full data. We’ll focus on when the full data is available here, so we can also compare structure perceived using the tour on the high-dimensional space, relative to structure revealed in the low-dimensional embedding.

There are many methods available for generating non-linear low dimensional representations of the data. MDS is a classical technique that minimises the difference between two interpoint distance matrices, the distance between points in the high-dimensions, and in the low-dimensional representations. A good resource for learning about MDS is Borg & Groenen (2005).

```
library(mulgar)
library(Rtsne)
library(uwot)
library(ggplot2)
library(patchwork)
set.seed(42)
cnl_tsne <- Rtsne(clusters_nonlin)
cnl_umap <- umap(clusters_nonlin)
n1 <- ggplot(as.data.frame(cnl_tsne$Y), aes(x=V1, y=V2)) +
```

```

geom_point() +
ggtitle("(a) t-SNE") +
theme_minimal() +
theme(aspect.ratio=1)
n2 <- ggplot(as.data.frame(cnl_umap), aes(x=V1, y=V2)) +
geom_point() +
ggtitle("(b) UMAP") +
theme_minimal() +
theme(aspect.ratio=1)
n1 + n2

```

./nlldr_files/figure-pdf/fig-nlldr-clusters-1.pdf

Figure 5.1: Two non-linear embeddings of the nonlinear clusters data: (a) t-SNE, (b) UMAP. Both suggest four clusters, with two being non-linear in some form.

Figure ?? show two NLDR views of the `clusters_nonlin` data set from the `mulgar` package. Both suggest that there are four clusters, and that some clusters are non-linearly shaped. They disagree on the type of non-linear pattern, where t-SNE represents one cluster as a wavy-shape and UMAP both have a simple parabolic shape. Popular methods in current use include t-SNE (Maaten & Hinton, 2008), UMAP (McInnes et al., 2018) and PHATE (Moon et al., 2019).

```

library(tourr)
render_gif(clusters_nonlin,
  grand_tour(),
  display_xy(),
  gif_file = "gifs/clusters_nonlin.gif",
  frames = 500,
  width = 300,
  height = 300)

```

The full 4D data is shown with a grand tour in `?@fig-clusters-nonlin`. The four clusters suggested by the NLDR methods can be seen. We also get a better sense of the relative size and proximity of the clusters. There are two small spherical clusters, one quite close to the end of the large sine wave cluster. The fourth cluster is relatively small, and has a slight curve, like a bent rod. The t-SNE representation is slightly more accurate than the UMAP representation. We would expect that the wavy cluster is the sine wave seen in the tour.

NLDR can provide useful low-dimensional summaries of high-dimensional structure but you need to check whether it is a sensible and accurate representation by comparing with what is perceived from a tour.

5.2 Linking NLDR representation with tour view

NLDR can produce useful low-dimensional summaries of structure in high-dimensional data, like those shown in Figure ???. However, there are numerous pitfalls. The fitting procedure can produce very different representations depending on the parameter choices, and even the random number seeding the fit. (You can check this by changing the `set.seed` in the code above, and by changing from the default parameters.) Also, it may not be possible to represent the high-dimensional structures faithfully in low dimensions. For these reasons, one needs to connect the NLDR view with a tour of the data, to help assess its usefulness and accuracy. For example, with this data, we would want to know which of the two curved clusters in the UMAP representation correspond to the sine wave cluster.

5.2.1 Using liminal

Figure ??? shows how the NLDR plot can be linked to a tour view, using the `liminal` package, to better understand how well the structure of the data is represented. Here we see learn that the smile in the UMAP embedding is the small bent rod cluster, and that the unbrow is the sine wave.

```
library(liminal)
umap_df <- data.frame(umapX = cnl_umap[, 1],
                      umapY = cnl_umap[, 2])

limn_tour_link(
  umap_df,
  clusters_nonlin,
  cols = x1:x4
)
```

Figure 5.2: Two screenshots from `liminal` showing which clusters match between the UMAP representation and the tour animation. The smile corresponds to the small bent rod cluster. The unbrow matches to the sine wave cluster.

5.2.2 Using detourr

`?@fig-detourr-clusters-nonlin` shows how the linking is achieved using `detourr`. It uses a shared data object, as made possible by the `crosstalk`

package, and the UMAP view is made interactive using `plotly`.

```
library(detourr)
library(dplyr)
library(crosstalk)
library(plotly)
umap_df <- data.frame(umapX = cnl_umap[, 1],
                     umapY = cnl_umap[, 2])
cnl_df <- bind_cols(clusters_nonlin, umap_df)
shared_cnl <- SharedData$new(cnl_df)

detour_plot <- detour(shared_cnl, tour_aes(
  projection = starts_with("x"))) |>
  tour_path(grand_tour(2),
            max_bases=50, fps = 60) |>
  show_scatter(alpha = 0.7, axes = FALSE,
              width = "100%", height = "450px")

umap_plot <- plot_ly(shared_cnl,
                    x = ~umapX,
                    y = ~umapY,
                    color = I("black"),
                    height = 450) %>%
  highlight(on = "plotly_selected",
            off = "plotly_doubleclick") %>%
  add_trace(type = "scatter",
            mode = "markers")

bscols(
  detour_plot, umap_plot,
  widths = c(5, 6)
)
```

5.3 Example: fake_trees

Figure ?? shows a more complex example, using the `fake_trees` data. We know that the 10D data has a main branch, and 9 branches (clusters) attached to it, based on our explorations in the earlier chapters. The t-SNE view, where points are coloured by the known branch ids, is very helpful for seeing the linear branch structure.

What we can't tell is that there is a main branch from which all of the others extend. We also can't tell which of the clusters corresponds to this branch. Linking the plot with a tour helps with this. Although, not shown in the sequence of snapshots in Figure ??, the main branch is actually the dark blue

cluster, which is separated into three pieces by t-SNE.

```
library(liminal)
library(Rtsne)
data(fake_trees)
set.seed(2020)
tsne <- Rtsne::Rtsne(dplyr::select(fake_trees, dplyr::starts_with("dim")))
tsne_df <- data.frame(tsneX = tsne$Y[, 1],
                     tsneY = tsne$Y[, 2])

limn_tour_link(
  tsne_df,
  fake_trees,
  cols = dim1:dim10,
  color = branches
)
```

Figure 5.3: Three snapshots of using the `liminal` linked views to explore how t-SNE has summarised the `fake_trees` data in 2D.

The t-SNE representation clearly shows the linear structures of the data, but it makes some inaccurate breaks of some of them.

Exercises

1. Using the `penguins_sub` data generate a 2D representation using t-SNE. Plot the points mapping the colour to species. What is most surprising? (Hint: Are the three species represented by three distinct clusters?)
2. Re-do the t-SNE representation with different parameter choices. Are the results different each time, or could they be considered to be equivalent?
3. Use `liminal` to link the t-SNE representation to a tour of the penguins. Highlight the points that have been placed in an awkward position by t-SNE from others in their species. Watch them relative to the others in their species in the tour view, and think about whether there is any rationale for the awkward placement.
4. Use UMAP to make the 2D representation, and use `liminal` to link with a tour to explore the result.
5. Repeat 1-4 using `detourr`.
6. CHALLENGE: Generate the linked plots (t-SNE, tour) using `plotly` and `crosstalk` alone. (Code from Chapter ?? might help here.)

Part III

Cluster analysis

Chapter 6

Overview

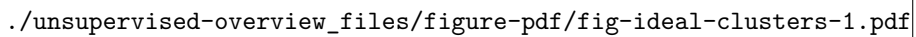
Unsupervised classification, or cluster analysis, organizes observations into similar groups. Cluster analysis is a commonly used, appealing, and conceptually intuitive statistical method. Some of its uses include market segmentation, where customers are grouped into clusters with similar attributes for targeted marketing; gene expression analysis, where genes with similar expression patterns are grouped together; and the creation of taxonomies for animals, insects, or plants. Clustering can be used as a way of reducing a massive amount of data because observations within a cluster can be summarized by its centre. Also, clustering effectively subsets the data thus simplifying analysis because observations in each cluster can be analyzed separately.

6.1 What are clusters?

Organizing objects into groups is a common task to help make sense of the world around us. Perhaps this is why it is an appealing method of data analysis. However, cluster analysis is more complex than it initially appears. Many people imagine that it will produce neatly separated clusters like those in Figure ??(a), but it almost never does. Such ideal clusters are rarely encountered in real data, so we often need to modify our objective from *find the natural clusters in this data*. Instead, we need to organize the *cases into groups that are similar in some way*. Even though this may seem disappointing when compared with the ideal, it is still often an effective means of simplifying and understanding a dataset.

Knowing what shapes are in your data helps with clustering it.

At the heart of the clustering process is the work of discovering which variables are most important for defining the groups. It is often true that we only require a subset of the variables for finding clusters, whereas another subset (called nuisance variables) has no impact. In the bottom left plot of Figure ??, it is clear that the variable plotted horizontally is important for splitting this data



./unsupervised-overview_files/figure-pdf/fig-ideal-clusters-1.pdf

Figure 6.1: Different structures in data impact cluster analysis. When there are well-separated groups (a), it is simple to group similar observations. Even when there are not, partitioning observations into groups may still be useful. There may be nuisance observations or (b) nuisance variables (c) that affect the interpoint distance calculations and distract the clustering algorithm, and there may oddly shaped clusters (d) which are hard to numerically describe.

into two clusters, whereas the variable plotted vertically is a nuisance variable. Nuisance is an apt term for these variables, because they can radically change the interpoint distances and impair the clustering process.

Dynamic graphical methods help us to find and understand the cluster structure in high dimensions. With the tools in our toolbox, primarily tours, along with linked scatterplots and parallel coordinate plots, we can see clusters in high-dimensional spaces. We can detect gaps between clusters, the shape and relative positions of clusters, and the presence of nuisance variables. We can even find unusually shaped clusters, like those in the bottom right plot in Figure ?? . In simple situations we can use graphics alone to group observations into clusters, using a “spin and brush” method. In more difficult data problems, we can assess and refine numerical solutions using graphics.

This part of the book discusses the use of interactive and dynamic graphics in the clustering of data. Section ?? introduces cluster analysis, focusing on interpoint distance measures. Chapter ?? describes an example of a purely graphical approach to cluster analysis, the spin and brush method. In the example shown in that section, we were able to find simplifications of the data that had not been found using numerical clustering methods, and to find a variety of structures in high-dimensional space. Chapter ?? describes methods for reducing the interpoint distance matrix to an intercluster distance matrix using hierarchical algorithms, Chapter ?? covers model-based clustering, and Chapter ?? described clustering with self-organising maps. Each of these chapters shows how graphical tools can be used to assess the results of numerical methods. Chapter ?? summarizes the chapter and revisits the data analysis strategies used in the examples. Additional references that provide good companions to the material presented in these chapters are Venables & Ripley (2002a), Boehmke & Greenwell (2019), Hennig et al. (2015), Giordani et al. (2020), Kassambara (2017), and the CRAN Task View (Leisch & Gruen, 2023). Chapter ?? summarizes the chapter and revisits the data analysis strategies used in the examples.

6.2 The importance of defining similar

Before we can begin finding groups of cases that are similar¹, we need to decide how to define or measure whether they are close together or far apart. Consider a dataset with three cases (a_1, a_2, a_3) and four variables (V_1, V_2, V_3, V_4), described in matrix format as

mathtools

¹Both *similarity* and *dissimilarity* measures are used for defining how similar cases are. It can be confusing! They measure similar in opposite directions. With a dissimilarity measure, a smaller number means the cases are closer, as in a distance metric. A similarity measure usually ranges between 0 and 1, with 1 indicating that the cases are closer, for example, correlation.

which is plotted in Figure ?? . The Euclidean distance between two cases (rows of the matrix) with p elements is defined as

$$d_{\text{Euc}}(a_i, a_j) = \|a_i - a_j\| \quad i, j = 1, \dots, n,$$

where $\|x_i\| = \sqrt{x_{i1}^2 + x_{i2}^2 + \dots + x_{ip}^2}$. For example, the Euclidean distance between cases 1 and 2 in the above data, is

$$\begin{aligned} d_{\text{Euc}}(a_1, a_2) &= \sqrt{(7.3 - 7.4)^2 + (7.6 - 7.2)^2 + (7.7 - 7.3)^2 + (8.0 - 7.2)^2} \\ &= 1.0 \end{aligned}$$

For the three cases, the interpoint Euclidean distance matrix is

$$d_{\text{Euc}} = \begin{bmatrix} 0.0 & & \\ 1.0 & 0.0 & \\ 6.3 & 5.5 & 0.0 \end{bmatrix} \begin{matrix} a_1 \\ a_2 \\ a_3 \end{matrix}$$

Cases a_1 and a_2 are more similar to each other than they are to case a_3 , because the Euclidean distance between cases a_1 and a_2 is much smaller than the distance between cases a_1 and a_3 and between cases a_2 and a_3 .

There are many different ways to calculate similarity. Similarity measures based on correlation distance have become common. Correlation distance is typically used where similarity of structure is more important than similarity in magnitude.

As an example, see the parallel coordinate plot of the sample data at the right of Figure ??. Cases a_1 and a_3 are widely separated, but their shapes are similar (low, medium, medium, high). Case a_2 , although overlapping with Case a_1 , has a very different shape (high, medium, medium, low). The correlation between two cases is defined as

$$\rho(a_i, a_j) = \frac{(a_i - c_i)'(a_j - c_j)}{\sqrt{(a_i - c_i)'(a_i - c_i)}\sqrt{(a_j - c_j)'(a_j - c_j)}}$$

./unsupervised-overview_files/figure-pdf/unnamed-chunk

./unsupervised-overview_files/figure-pdf/unnamed-chunk-2-1.pdf

Figure 6.2: The scatterplot matrix (left) shows that cases a_1 and a_2 have similar values. The parallel coordinate plot (right) allows a comparison of other structure, which shows the similarity in the trend of the profiles on cases a_1 and a_3 .

When c_i, c_j are the sample means \bar{a}_i, \bar{a}_j , then ρ is the Pearson correlation coefficient. If, indeed, they are set at 0, as is commonly done, ρ is a generalized correlation that describes the angle between the two data vectors. The correlation is then converted to a distance metric; one equation for doing so is as follows:

$$d_{\text{Cor}}(a_i, a_j) = \sqrt{2(1 - \rho(a_i, a_j))}$$

The above distance metric will treat cases that are strongly negatively correlated as the most distant.

The interpoint distance matrix for the sample data using d_{Cor} and the Pearson correlation coefficient is

$$d_{\text{Cor}} = \begin{bmatrix} 0.0 & & \\ 3.6 & 0.0 & \\ 0.1 & 3.8 & 0.0 \end{bmatrix} \begin{matrix} a_1 \\ a_2 \\ a_3 \end{matrix}$$

By this metric, cases a_1 and a_3 are the most similar, because the correlation distance is smaller between these two cases than the other pairs of cases.

Note that these interpoint distances differ dramatically from those for Euclidean distance. As a consequence, the way the cases would be clustered is also be very different. Choosing the appropriate distance measure is an important part of a cluster analysis.

After a distance metric has been chosen and a cluster analysis has been performed, the analyst must evaluate the results, and this is actually a difficult task. A cluster analysis does not generate p -values or other numerical criteria, and the process tends to produce hypotheses rather than testing them. Even the most determined attempts to produce the “best” results using modeling and validation techniques may result in clusters that, although seemingly significant, are useless for practical purposes. As a result, cluster analysis is best thought of as an exploratory technique, and it can be quite useful despite the lack of formal validation because of its power in data simplification.

Defining an appropriate distance metric is a vitally important decision.

The context in which the data arises is the key to assessing the results. If the clusters can be characterized in a sensible manner, and they increase our knowledge of the data, then we are on the right track. To use an even more pragmatic criterion, if a company can gain an economic advantage by using a particular clustering method to carve up their customer database, then that is the method they should use.

Exercises

Use the following data to answer these questions:

```
      x1  x2  x3
a1 0.13 0.21 0.09
a2 0.91 0.95 0.85
a3 0.62 0.73 0.65
a4 0.21 0.92 0.43
```

1. Compute the Euclidean distance between cases **a1**, **a2**, **a3**, **a4**.
2. Compute the correlation distance (as defined above) between cases **a1**, **a2**, **a3**, **a4**.
3. Which two points have the (a) biggest (b) smallest Mahalanobis (statistical) distance, assuming that the covariance matrix is:

```
      x1  x2  x3
x1 1.0 0.8 0.8
x2 0.8 1.0 0.8
x3 0.8 0.8 1.0
```

(The function `mahalanobis` will calculate this in R. Technically this gives distance between each case and the mean vector.)

4. Is the ordering of distance between cases the same if Manhattan distance is used instead of Euclidean?
5. Compute the Chebychev distance between cases **a1**, **a2**, **a3**, **a4**.
6. Compute Bray-Curtis distance between cases **a1**, **a2**, **a3**, **a4**.
7. Make a plot of the data, and write a paragraph describing how the different distance metrics agree and disagree on how close or far the cases are from each other.

Chapter 7

Spin and brush approach

When the data to be clustered is purely numeric, and if the clusters are well-separated, a purely graphical spin and brush approach to cluster analysis works well. This is true even when there are nuisance variables and/or cases, marked differences in variance structures between groups or when groups have non-linear boundaries. It does not work very well when there are clusters that overlap, or when there are no distinct clusters but rather we simply wish to partition the data. In these situations it may be better to begin with a numerical solution and to use visual tools to evaluate it, perhaps making refinements subsequently. Several examples of the spin and brush approach are documented in the literature, such as Cook et al. (1995a) and Wilhelm et al. (1999).

The spin and brush approach is simply:

1. Run the (grand) tour.
2. Stop when you see a separated cluster of points.
3. Paint the cluster a chosen colour.
4. Repeat 1-2 until the data is grouped, and when no other separated cluster is visible in any projection. You may need to re-paint some points if they appear to be grouped incorrectly in a different projection, or paint more points that after spinning most likely belong to an existing group.

It can help to be able to remove a cluster, to de-clutter the display, in order to find more clusters.

7.1 Using `plotly`

Figure ?? shows a fixed length grand tour produced using `plotly`, with a brush tool made available to colour points. The code is messy, but follows that of `animation-tour-basic.R` from Sievert (2020). There are key pieces in the code:

1. Create the sequence of tour projections, and the data object containing data projections and projection coordinates. Include an observation id in the data object.
2. Specify that the `id` variable is to be used to mark observations across animation frames using `highlight_key`.
3. Draw them using `plotly`, with a `frame` parameter which specifies the animation sequence.
4. Add a brush colour palette and specific brushing control using `highlight` to the `plotly` plot.

The result is an HTML object which can be saved for sharing. The drawback of this approach is that the results of the user actions cannot be saved, so you cannot recover the clusters corresponding to the colours.

```
# Following https://github.com/plotly/plotly.R/blob/master/demo/animation-tour-basi
# TURN INTO A FUNCTION TO MAKE IT EASIER
library(tourr)
library(plotly)
load("data/penguins_sub.rda")
p_mat <- as.matrix(penguins_sub[,1:4])
tour <- new_tour(p_mat,
                grand_tour(), NULL)

tour_dat <- function(step_size) {
  step <- tour(step_size)
  proj <- center(p_mat %*% step$proj)
  data.frame(x = proj[,1], y = proj[,2],
             species = penguins_sub$species,
             id = 1:nrow(penguins_sub))
}

proj_dat <- function(step_size) {
  step <- tour(step_size)
  data.frame(
    x = step$proj[,1], y = step$proj[,2], measure = colnames(p_mat)
  )
}

steps <- c(0, rep(1/15, 150))
stepz <- cumsum(steps)

# tidy version of tour data
tour_dats <- lapply(steps, tour_dat)
tour_datz <- Map(function(x, y) cbind(x, step = y),
                 tour_dats, stepz)
```

```

tour_data <- dplyr::bind_rows(tour_datz)

tour_data <- highlight_key(tour_data, ~id)

# tidy version of tour projection data
proj_dats <- lapply(steps, proj_dat)
proj_datz <- Map(function(x, y) cbind(x, step = y), proj_dats, stepz)
proj_data <- dplyr::bind_rows(proj_datz)
proj_data$x <- proj_data$x*3
proj_data$y <- proj_data$y*3

ax <- list(
  title = "",
  range = c(-3, 3),
  zeroline = FALSE
)

# Set colors
clrs <- grDevices::hcl.colors(6, palette="Zissou 1")

# for nicely formatted slider labels
options(digits = 2)

p_b_s <- proj_data %>%
  plot_ly(x = ~x, y = ~y, frame = ~step,
    color = I("gray80"),
    width=600, height=600) %>%
  config(displaylogo = FALSE,
    modeBarButtonsToRemove = c("sendDataToCloud", "editInChartStudio", "zoom2d", "zoomIn2d"))
  add_segments(xend = 0, yend = 0) %>%
  add_text(text = ~measure) %>%
  add_markers(color = I("black"), data = tour_data, text = ~id, ids = ~id, hoverinfo = "text")
  layout(xaxis = ax, yaxis = ax) %>%
  hide_legend() %>%
  animation_opts(50, transition = 0, redraw = FALSE) %>%
  highlight(on = "plotly_selected",
    off = "plotly_doubleclick",
    color = clrs,
    persistent = TRUE,
    dynamic = TRUE,
    opacityDim = 0.5)
htmlwidgets::saveWidget(p_b_s,
  file="html/penguins_brush_and_spin.html",
  selfcontained = TRUE)

```

Figure 7.1: Controls to brush and spin to discover the three clusters in the penguins data. Use the lasso brush to colour points persistently when you see a cluster. Spin and colour again as necessary to identify all the clusters.

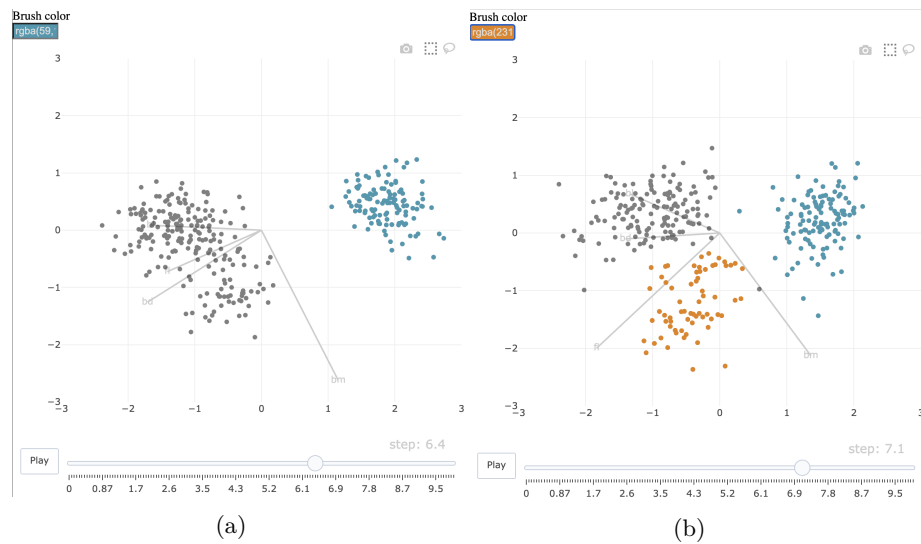


Figure 7.2: Screenshots of the spin and brush approach being used on the penguins data.

7.2 Using detourr

Spin-and-brush can also be achieved using the `detourr` package, and the results can be saved for further analysis. The code is very easy, and then all the controls are interactive.

```
library(detourr)
grDevices::hcl.colors(3, palette="Zissou 1")
detour(penguins_sub[,1:4],
       tour_aes(projection = bl:bm)) |>
  tour_path(grand_tour(2), fps = 60,
            max_bases=20) |>
  show_scatter(alpha = 0.7,
              axes = FALSE)
```

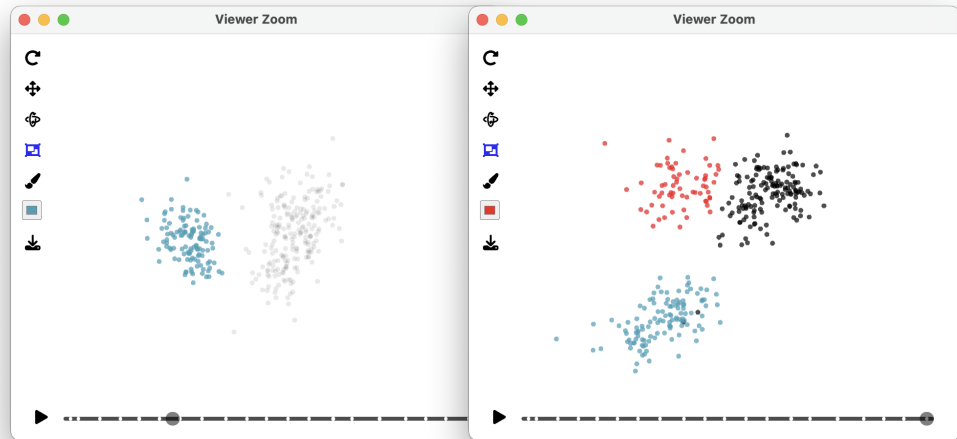
- `tour_aes(projection = bl:bm)` is `ggplot`-style syntax for specifying the variables `bl:bm` to include in the tour.
- `tour_path(grand_tour(2), fps = 60, max_bases=20)` specifies 2D grand tour path, with a longer than default path set by `max_bases=20` and the `fps` argument sets the smoothness.
- Brush interaction is set by choosing the square icon (4th from top), so when the cursor is moved over the window points are selected.
- You can choose specific colours to brush, from the colour palette by using hexcolours to match your favourite palette. Here we've used colours from the Zissou palette.
- The paintbrush icon sets the selected points to the current colour.
- Save the final colour labels using the download icon.

Figure ?? shows the stages of spin and brush on the penguins data using `detourr`. The final results can be examined and used for later analysis. Because this data came with a class variable, the penguin species, it is interesting to see how close the spin-and-brush clustering approach came to recovering these:

```
library(readr)
load("data/penguins_sub.rda")
detourr_penguins <- read_csv("data/detourr_penguins.csv")
table(penguins_sub$species, detourr_penguins$colour)
```

	000000	3e9eb6	f5191c
Adelie	143	0	3
Chinstrap	6	0	62
Gentoo	2	117	0

It's quite close! All but two of the 119 Gentoo penguins were identified as a cluster (labelled as "3e9eb6" from the chosen light blue hex colour), and all but three of the 146 Adelie penguins were identified as a cluster, (labelled as



(a) First cluster cluster

(b) Second cluster

Figure 7.3: Screenshots of the spin and brush approach using `detourr` on the penguins data.

“000000” which is the unbrushed black group). Most of the Chinstrap species were recovered also (labelled as “f5191c” for the red hex colour).

Exercises

1. Use the spin and brush approach to identify the three clusters in the `mulgar::clusters` data set.
2. Use the spin and brush approach to identify the six clusters in the `mulgar::multiclust` data set. (The code below using `detourr` could be useful.)

```
library(detourr)

# Use a random starting basis because the first two variables make it too easy
strt <- tourr::basis_random(10, 2)
detour(multiclust,
  tour_aes(projection = -group)) |>
  tour_path(grand_tour(2), start=strt, fps = 60) |>
  show_scatter(alpha = 0.7, axes = FALSE)
```

3. Use the spin and brush technique to identify the branches of the `fake_trees` data available in the `liminal` package (originally from

PHATE). The result should look something like this:

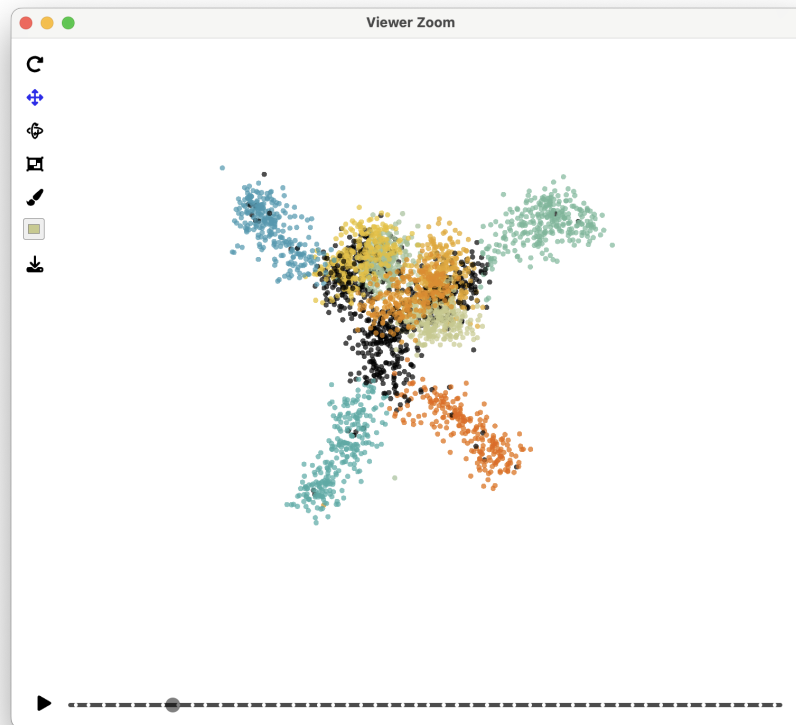


Figure 7.4: Example solution after spin and brush on `fake_trees` data.

You can use the download button to save the data with the colours. Tabulate the `branches` id variable in the original data with the `colour` groups created from brushing, to see how closely you have recovered the original classes.

```
library(detourr)
library(liminal)
library(mulgar)
data("fake_trees")

# Original data is 100D, so need to reduce dimension using PCA first
ft_pca <- prcomp(fake_trees[,1:100],
                 scale=TRUE, retx=TRUE)

ggscree(ft_pca)
detour(as.data.frame(ft_pca$x[,1:10]),
```

```
tour_aes(projection = PC1:PC10)) |>  
tour_path(grand_tour(2), fps = 60, max_bases=50) |>  
show_scatter(alpha = 0.7, axes = FALSE)  
  
ft_sb <- read_csv("data/fake_trees_sb.csv")  
table(fake_trees$branches, ft_sb$colour)
```


Chapter 8

Hierarchical clustering

8.1 Overview

Hierarchical cluster algorithms sequentially fuse neighboring points to form ever-larger clusters, starting from a full interpoint distance matrix. *Distance between clusters* is described by a “linkage method”, of which there are many. For example, single linkage measures the distance between clusters by the smallest interpoint distance between the members of the two clusters, complete linkage uses the maximum interpoint distance, and average linkage uses the average of the interpoint distances. Wards linkage, which usually produces the best clustering solutions, defines the distance as the reduction in the within-group variance. A good discussion on cluster analysis and linkage can be found in Boehmke & Greenwell (2019), on Wikipedia or any multivariate textbook.

Hierarchical clustering is summarised by a dendrogram, which sequentially shows points being joined to form a cluster, with the corresponding distances.

Here we will take a look at hierarchical clustering, using Wards linkage, on the `simple_clusters` data. The steps taken are to:

1. Plot the data to check for presence of clusters and their shape.
2. Compute the hierarchical clustering.
3. Plot the dendrogram to help decide on an appropriate number of clusters, using the `dendro_data` function from the `ggdendro` package.
4. Show the dendrogram overlaid on the data, calculated by the `hierfly` function in `mulgar`.
5. Plot the clustering result, by colouring points in the plot of the data.

```
library(ggplot2)
library(mulgar)
```

```

library(ggdendro)
library(dplyr)
library(patchwork)
library(tourr)
library(plotly)
library(htmlwidgets)
library(colorspace)
library(GGally)

data(simple_clusters)

# Compute hierarchical clustering with Ward's linkage
cl_hw <- hclust(dist(simple_clusters[,1:2]),
               method="ward.D2")
cl_ggd <- dendro_data(cl_hw, type = "triangle")

# Compute dendrogram in the data
cl_hfly <- hierfly(simple_clusters, cl_hw, scale=FALSE)

# Show result
simple_clusters <- simple_clusters %>%
  mutate(cbw = factor(cutree(cl_hw, 2)))

```

Figure ?? illustrates the hierarchical clustering approach for a simple simulated data set (a) with two well-separated clusters in 2D. The dendrogram (b) is a representation of the order that points are joined into clusters. The dendrogram strongly indicates two clusters because there are two branches branches representing the last join are much longer than all of the other branches.

Although, the dendrogram is usually a good summary of the steps taken by the algorithm, it can be misleading. The dendrogram might indicate a clear clustering (big differences in heights of branches) but the result may be awful. You need to check this by examining the result on the data, called model-in-the-data space by Wickham et al. (2015).

Plot (c) shows the dendrogram in 2D, overlaid on the data. The segments show how the points are joined to make clusters. In order to represent the dendrogram this way, new points (represented by a “+” here) need to be added corresponding to the centroid of groups of points that have been joined. These are used to draw the segments between other points and other clusters. We can see that the longest (two) edges stretches across the gap between the two clusters. This corresponds to the top of the dendrogram, the two long branches where we would cut it to make the two-cluster solution. This two-cluster solution is shown in plot (d).

```

# Plot the data
pd <- ggplot(simple_clusters, aes(x=x1, y=x2)) +
  geom_point(colour="#3B99B1", size=2, alpha=0.8) +
  ggtitle("(a)") +
  theme_minimal() +
  theme(aspect.ratio=1)

# Plot the dendrogram
ph <- ggplot() +
  geom_segment(data=cl_ggd$segments,
              aes(x = x, y = y,
                  xend = xend, yend = yend)) +
  geom_point(data=cl_ggd$labels, aes(x=x, y=y),
            colour="#3B99B1", alpha=0.8) +
  ggtitle("(b)") +
  theme_minimal() +
  theme_dendro()

# Plot the dendrogram on the data
pdh <- ggplot() +
  geom_segment(data=cl_hfly$segments,
              aes(x=x, xend=xend,
                  y=y, yend=yend)) +
  geom_point(data=cl_hfly$data,
            aes(x=x1, y=x2,
                shape=factor(node),
                colour=factor(node),
                size=1-node), alpha=0.8) +
  xlab("x1") + ylab("x2") +
  scale_shape_manual(values = c(16, 3)) +
  scale_colour_manual(values = c("#3B99B1", "black")) +
  scale_size(limits=c(0,17)) +
  ggtitle("(c)") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none")

# Plot the resulting clusters
pc <- ggplot(simple_clusters) +
  geom_point(aes(x=x1, y=x2, colour=clw),
            size=2, alpha=0.8) +
  scale_colour_discrete_divergingx(palette = "Zissou 1",
                                  nmax=5, rev=TRUE) +
  ggtitle("(d)") +
  theme_minimal() +

```

```
theme(aspect.ratio=1, legend.position="none")  
  
pd + ph + pdh + pc + plot_layout(ncol=2)
```

./hierarchical-clustering_files/figure-pdf/fig-hc-sim-1.pdf

Figure 8.1: Hierarchical clustering on simulated data: (a) data, (b) dendrogram, (c) dendrogram on the data, and (d) two cluster solution. The extra points corresponding to nodes of the dendrogram are indicated by + in (c).

Clustering algorithms are all prone to being confused by different patterns occurring in data. For hierarchical clustering the choice for defining distance once points have been joined into clusters can produce starkly different results. Plotting the dendrogram on the data provides a good way to assess the solution.

8.2 Common patterns which confuse clustering algorithms

Figure ?? shows two examples of structure in data that will confuse hierarchical clustering: nuisance variables and nuisance cases. We usually do not know that these problems exist prior to clustering the data. Discovering these iteratively as you conduct a clustering analysis is important for generating useful results.

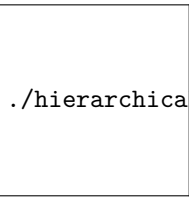
```
# Nuisance observations
set.seed(20190514)
x <- (runif(20)-0.5)*4
y <- x
d1 <- data.frame(x1 = c(rnorm(50, -3),
                       rnorm(50, 3), x),
                 x2 = c(rnorm(50, -3),
                       rnorm(50, 3), y),
                 cl = factor(c(rep("A", 50),
                              rep("B", 70))))

d1 <- d1 %>%
  mutate_if(is.numeric, function(x) (x-mean(x))/sd(x))
pd1 <- ggplot(data=d1, aes(x=x1, y=x2)) +
  geom_point() +
  ggtitle("Nuisance observations") +
  theme_minimal() +
  theme(aspect.ratio=1)

# Nuisance variables
set.seed(20190512)
d2 <- data.frame(x1=c(rnorm(50, -4),
                     rnorm(50, 4)),
                 x2=c(rnorm(100)),
                 cl = factor(c(rep("A", 50),
                              rep("B", 50))))

d2 <- d2 %>%
  mutate_if(is.numeric, function(x) (x-mean(x))/sd(x))
pd2 <- ggplot(data=d2, aes(x=x1, y=x2)) +
  geom_point() +
  ggtitle("Nuisance variables") +
  theme_minimal() +
  theme(aspect.ratio=1)

pd1 + pd2 + plot_layout(ncol=2)
```



./hierarchical-clustering_files/figure-pdf/fig-problems-1.pdf

Figure 8.2: Two examples of data structure that causes problems for hierarchical clustering. Nuisance observations can cause problems because the close observations between the two clusters can cause some chaining in the hierarchical joining of observations. Nuisance variables can cause problems because observations across the gap can seem closer than observations at the end of each cluster.

If an outlier is a point that is extreme relative to other observations, an “inlier” is a point that is extreme relative to a cluster, but inside the domain of all of the observations. Nuisance observations are inliers, cases that occur between larger groups of points. If they were excluded there might be a gap between clusters. These can cause problems for clustering when distances between clusters are measured, and can be very problematic when single linkage hierarchical clustering is used. Figure ?? shows how nuisance observations affect single linkage but not Wards linkage hierarchical clustering.

```
# Compute single linkage
d1_hs <- hclust(dist(d1[,1:2]),
               method="single")
d1_ggds <- dendro_data(d1_hs, type = "triangle")
pd1s <- ggplot() +
  geom_segment(data=d1_ggds$segments,
              aes(x = x, y = y,
                  xend = xend, yend = yend)) +
  geom_point(data=d1_ggds$labels, aes(x=x, y=y),
             colour="#3B99B1", alpha=0.8) +
  theme_minimal() +
  ggtitle("(a) Single linkage dendrogram") +
  theme_dendro()

# Compute dendrogram in data
d1_hflys <- hierfly(d1, d1_hs, scale=FALSE)

pd1hs <- ggplot() +
  geom_segment(data=d1_hflys$segments,
              aes(x=x, xend=xend,
                  y=y, yend=yend)) +
  geom_point(data=d1_hflys$data,
```

```

      aes(x=x1, y=x2,
          shape=factor(node),
          colour=factor(node),
          size=1-node), alpha=0.8) +
scale_shape_manual(values = c(16, 3)) +
scale_colour_manual(values = c("#3B99B1", "black")) +
scale_size(limits=c(0,17)) +
ggtitle("(b) Dendrogram in data") +
theme_minimal() +
theme(aspect.ratio=1, legend.position="none")

# Show result
d1 <- d1 %>%
  mutate(cls = factor(cutree(d1_hs, 2)))
pc_d1s <- ggplot(d1) +
  geom_point(aes(x=x1, y=x2, colour=cls),
             size=2, alpha=0.8) +
  scale_colour_discrete_divergingx(palette = "Zissou 1",
                                   nmax=4, rev=TRUE) +
  ggtitle("(c) Two-cluster solution") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none")

# Compute Wards linkage
d1_hw <- hclust(dist(d1[,1:2]),
               method="ward.D2")
d1_ggdw <- dendro_data(d1_hw, type = "triangle")
pd1w <- ggplot() +
  geom_segment(data=d1_ggdw$segments,
              aes(x = x, y = y,
                  xend = xend, yend = yend)) +
  geom_point(data=d1_ggdw$labels, aes(x=x, y=y),
            colour="#3B99B1", alpha=0.8) +
  ggtitle("(d) Ward's linkage dendrogram") +
  theme_minimal() +
  theme_dendro()

# Compute dendrogram in data
d1_hflyw <- hierfly(d1, d1_hw, scale=FALSE)

pd1hw <- ggplot() +
  geom_segment(data=d1_hflyw$segments,
              aes(x=x, xend=xend,
                  y=y, yend=yend)) +

```

```

geom_point(data=d1_hflyw$data,
           aes(x=x1, y=x2,
              shape=factor(node),
              colour=factor(node),
              size=1-node), alpha=0.8) +
scale_shape_manual(values = c(16, 3)) +
scale_colour_manual(values = c("#3B99B1", "black")) +
scale_size(limits=c(0,17)) +
ggtitle("(e) Dendrogram in data") +
theme_minimal() +
theme(aspect.ratio=1, legend.position="none")

# Show result
d1 <- d1 %>%
  mutate(clw = factor(cutree(d1_hw, 2)))
pc_d1w <- ggplot(d1) +
  geom_point(aes(x=x1, y=x2, colour=clw),
            size=2, alpha=0.8) +
  scale_colour_discrete_divergingx(palette = "Zissou 1",
                                   nmax=4, rev=TRUE) +
  ggtitle("(f) Two-cluster solution") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none")

pd1s + pd1hs + pc_d1s +
  pd1w + pd1hw + pc_d1w +
  plot_layout(ncol=3)

```

./hierarchical-clustering_files/figure-pdf/fig-d1-s-1.pdf

Figure 8.3: The effect of nuisance observations on single linkage (a, b, c) and Ward's linkage hierarchical clustering (d, e, f). Single linkage clustering gets so distracted by the nuisance cases that the two cluster result is one big cluster, and a singleton cluster. Conversely, Ward's clusters this data nicely into two useful groups.

Nuisance variables are ones that do not contribute to the clustering, such as `x2` here. When we look at this data we see a gap between two elliptically shape clusters, with the gap being only in the horizontal direction, `x1`. When we

8.2. COMMON PATTERNS WHICH CONFUSE CLUSTERING ALGORITHMS85

compute the distances between points, in order to start clustering, without knowing that `x2` is a nuisance variable, points across the gap might be considered to be closer than points within the same cluster. Figure ?? shows how nuisance variables affects complete linkage but not Wards linkage hierarchical clustering. (Wards linkage can be affected but it isn't for this data.) Interestingly, the dendrogram for complete linkage looks ideal, that it suggests two clusters. It is not until you examine the resulting clusters in the data that you can see the error, that it has clustered across the gap.

```
# Compute complete linkage
d2_hc <- hclust(dist(d2[,1:2]),
               method="complete")
d2_ggdc <- dendro_data(d2_hc, type = "triangle")
pd2c <- ggplot() +
  geom_segment(data=d2_ggdc$segments,
              aes(x = x, y = y,
                  xend = xend, yend = yend)) +
  geom_point(data=d2_ggdc$labels, aes(x=x, y=y),
             colour="#3B99B1", alpha=0.8) +
  ggtitle("(a) Complete linkage dendrogram") +
  theme_minimal() +
  theme_dendro()

# Compute dendrogram in data
d2_hflyc <- hierfly(d2, d2_hc, scale=FALSE)

pd2hc <- ggplot() +
  geom_segment(data=d2_hflyc$segments,
              aes(x=x, xend=xend,
                  y=y, yend=yend)) +
  geom_point(data=d2_hflyc$data,
             aes(x=x1, y=x2,
                 shape=factor(node),
                 colour=factor(node),
                 size=1-node), alpha=0.8) +
  scale_shape_manual(values = c(16, 3)) +
  scale_colour_manual(values = c("#3B99B1", "black")) +
  scale_size(limits=c(0,17)) +
  ggtitle("(b) Dendrogram in data") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none")

# Show result
d2 <- d2 %>%
  mutate(clc = factor(cutree(d2_hc, 2)))
```

```

pc_d2c <- ggplot(d2) +
  geom_point(aes(x=x1, y=x2, colour=clc),
             size=2, alpha=0.8) +
  scale_colour_discrete_divergingx(palette = "Zissou 1",
                                   nmax=4, rev=TRUE) +
  ggtitle("(c) Two-cluster solution") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none")

# Compute Wards linkage
d2_hw <- hclust(dist(d2[,1:2]),
               method="ward.D2")
d2_ggdw <- dendro_data(d2_hw, type = "triangle")
pd2w <- ggplot() +
  geom_segment(data=d2_ggdw$segments,
              aes(x = x, y = y,
                  xend = xend, yend = yend)) +
  geom_point(data=d2_ggdw$labels, aes(x=x, y=y),
             colour="#3B99B1", alpha=0.8) +
  ggtitle("(d) Ward's linkage dendrogram") +
  theme_minimal() +
  theme_dendro()

# Compute dendrogram in data
d2_hflyw <- hierfly(d2, d2_hw, scale=FALSE)

pd2hw <- ggplot() +
  geom_segment(data=d2_hflyw$segments,
              aes(x=x, xend=xend,
                  y=y, yend=yend)) +
  geom_point(data=d2_hflyw$data,
             aes(x=x1, y=x2,
                 shape=factor(node),
                 colour=factor(node),
                 size=1-node), alpha=0.8) +
  scale_shape_manual(values = c(16, 3)) +
  scale_colour_manual(values = c("#3B99B1", "black")) +
  scale_size(limits=c(0,17)) +
  ggtitle("(e) Dendrogram in data") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none")

# Show result
d2 <- d2 %>%

```

```

mutate(clw = factor(cutree(d2_hw, 2)))
pc_d2w <- ggplot(d2) +
  geom_point(aes(x=x1, y=x2, colour=clw),
             size=2, alpha=0.8) +
  scale_colour_discrete_divergingx(palette = "Zissou 1",
                                   nmax=4, rev=TRUE) +
  ggtitle("(f) Two-cluster solution") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none")

pd2c + pd2hc + pc_d2c +
  pd2w + pd2hw + pc_d2w +
  plot_layout(ncol=3)

```

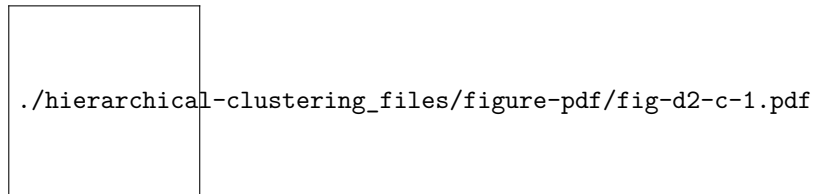


Figure 8.4: Complete linkage clustering on nuisance variables in comparison to Ward's linkage. The complete linkage dendrogram looks quite reasonably suggesting a two-cluster solution, but when it is plotted amongst the data that it is clearly not a good two-cluster solution.

Two dendrograms might look very similar but the resulting clustering can be very different.

8.3 Dendrograms in high-dimensions

The first step with any clustering with high dimensional data is also to check the data. You typically don't know whether there are clusters, or what shape they might be, or if there are nuisance observations or variables. A pairs plot like in Figure ?? is a nice complement to using the tour (**?@fig-penguins-gt**) for this. Here you can see three elliptical clusters, with one is further from the others.

```

load("data/penguins_sub.rda")
ggscatmat(penguins_sub[,1:4]) +
  theme_minimal() +
  xlab("") + ylab("")

```

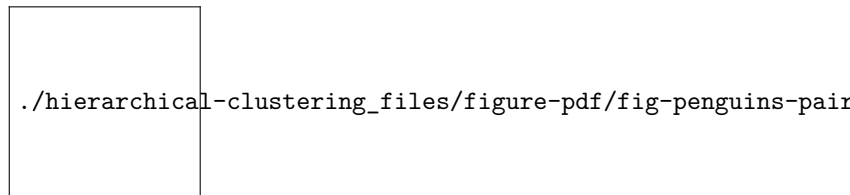


Figure 8.5: Make a scatterplot matrix to check for the presence of clustering, shape of clusters and presence of nuisance observations and variables. In the penguins it appears that there might be three elliptically shaped clusters, with some nuisance observations.

```
set.seed(20230329)
b <- basis_random(4,2)
pt1 <- save_history(penguins_sub[,1:4],
                    max_bases = 500,
                    start = b)
save(pt1, file="data/penguins_tour_path.rda")
animate_xy(penguins_sub[,1:4],
            tour_path = planned_tour(pt1),
            axes="off", rescale=FALSE,
            half_range = 3.5)

load("data/penguins_tour_path.rda")
render_gif(penguins_sub[,1:4],
            planned_tour(pt1),
            display_xy(half_range=0.9, axes="off"),
            gif_file="gifs/penguins_gt.gif",
            frames=500,
            loop=FALSE)
```

The process is the same as for the simpler example. We compute and draw the dendrogram in 2D, compute it in p -D and view with a tour. Here we have also chosen to examine the three cluster solution for single linkage and wards linkage clustering.

```
p_dist <- dist(penguins_sub[,1:4])
p_hcw <- hclust(p_dist, method="ward.D2")
p_hcs <- hclust(p_dist, method="single")

p_clw <- penguins_sub %>%
  mutate(cl = factor(cutree(p_hcw, 3)))
p_cls <- penguins_sub %>%
```

```

mutate(cl = factor(cutree(p_hcs, 3)))

p_w_hfly <- hierfly(p_clw, p_hcw, scale=FALSE)
p_s_hfly <- hierfly(p_cls, p_hcs, scale=FALSE)

# Generate the dendrograms in 2D
p_hcw_dd <- dendro_data(p_hcw)
pw_dd <- ggplot() +
  geom_segment(data=p_hcw_dd$segments,
              aes(x = x, y = y,
                  xend = xend, yend = yend)) +
  geom_point(data=p_hcw_dd$labels, aes(x=x, y=y),
             alpha=0.8) +
  theme_dendro()

p_hcs_dd <- dendro_data(p_hcs)
ps_dd <- ggplot() +
  geom_segment(data=p_hcs_dd$segments,
              aes(x = x, y = y,
                  xend = xend, yend = yend)) +
  geom_point(data=p_hcs_dd$labels, aes(x=x, y=y),
             alpha=0.8) +
  theme_dendro()

load("data/penguins_tour_path.rda")
glyphs <- c(16, 46)
pchw <- glyphs[p_w_hfly$data$node+1]
pchs <- glyphs[p_s_hfly$data$node+1]

animate_xy(p_w_hfly$data[,1:4],
           #col=colw,
           tour_path = planned_tour(pt1),
           pch = pchw,
           edges=p_w_hfly$edges,
           axes="bottomleft")

animate_xy(p_s_hfly$data[,1:4],
           #col=colw,
           tour_path = planned_tour(pt1),
           pch = pchs,
           edges=p_s_hfly$edges,
           axes="bottomleft")

```

```

render_gif(p_w_hfly$data[,1:4],
  planned_tour(pt1),
  display_xy(half_range=0.9,
    pch = pchw,
    edges = p_w_hfly$edges,
    axes = "off"),
  gif_file="gifs/penguins_hflyw.gif",
  frames=500,
  loop=FALSE)

render_gif(p_s_hfly$data[,1:4],
  planned_tour(pt1),
  display_xy(half_range=0.9,
    pch = pchs,
    edges = p_s_hfly$edges,
    axes = "off"),
  gif_file="gifs/penguins_hflys.gif",
  frames=500,
  loop=FALSE)

# Show three cluster solutions
clrs <- hcl.colors(3, "Zissou 1")
w3_col <- clrs[p_w_hfly$data$cl[p_w_hfly$data$node == 0]]
render_gif(p_w_hfly$data[p_w_hfly$data$node == 0, 1:4],
  planned_tour(pt1),
  display_xy(half_range=0.9,
    col=w3_col,
    axes = "off"),
  gif_file="gifs/penguins_w3.gif",
  frames=500,
  loop=FALSE)

s3_col <- clrs[p_s_hfly$data$cl[p_w_hfly$data$node == 0]]
render_gif(p_s_hfly$data[p_w_hfly$data$node == 0, 1:4],
  planned_tour(pt1),
  display_xy(half_range=0.9,
    col=s3_col,
    axes = "off"),
  gif_file="gifs/penguins_s3.gif",
  frames=500,
  loop=FALSE)

```

Figure ?? shows results for single linkage and wards linkage clustering of the penguins data. Plots (a) and (b) show the 2D dendrograms. The 2D dendrograms are very different. Wards linkage produces a clearer indication of clusters, with a

suggestion of three, or possibly four or five clusters. The dendrogram for single linkage suggests two clusters, and has the classical waterfall appearance that is often seen with this type of linkage. (If you look carefully, though, you will see it is actually a three cluster solution. At the very top of the dendrogram there is another branch connecting one observation to the other two clusters.)

Plots (c) and (d) show the dendrograms in 4D overlaid on the data. The two are starkly different. The single linkage clustering is like pins pointing to (three) centres, with some long extra edges.

Plots (e) and (f) show the three cluster solutions, with Wards linkage almost recovering the clusters of the three species. Single linkage has two big clusters, and the singleton cluster. Although the Wards linkage produces the best result, single linkage does provide some interesting and useful information about the data. That singleton cluster is an outlier, an unusually-sized penguin. We can see it as an outlier just from the tour in `?@fig-penguins-gt` but single linkage emphasizes it, bringing it more strongly to our attention.

Figure 8.8: Dendrograms for Wards and single linkage of the penguins data, shown in 2D (top) and in 4D (middle), and the three-cluster solution of each.

Single linkage on the penguins has a very different joining pattern to Wards! While Wards provides the better result, single linkage provides useful information about the data, such as emphasizing the outlier.

Figure ?? provides HTML objects of the dendrograms, so that they can be directly compared. The same tour path is used, so the sliders allow setting the view to the same projection in each plot.

```
load("data/penguins_tour_path.rda")
# Create a smaller one, for space concerns
pt1i <- interpolate(pt1[,1:5], 0.1)
pw_anim <- render_anim(p_w_hfly$data,
                      vars=1:4,
                      frames=pt1i,
                      edges = p_w_hfly$edges,
                      obs_labels=paste0(1:nrow(p_w_hfly$data),
                                         p_w_hfly$data$c1))

pw_gp <- ggplot() +
  geom_segment(data=pw_anim$edges,
              aes(x=x, xend=xend,
                  y=y, yend=yend,
                  frame=frame)) +
  geom_point(data=pw_anim$frames,
```

```

      aes(x=P1, y=P2,
          frame=frame,
          shape=factor(node),
          label=obs_labels),
      alpha=0.8, size=1) +
xlim(-1,1) + ylim(-1,1) +
scale_shape_manual(values=c(16, 46)) +
coord_equal() +
theme_bw() +
theme(legend.position="none",
      axis.text=element_blank(),
      axis.title=element_blank(),
      axis.ticks=element_blank(),
      panel.grid=element_blank())

pwg <- ggplotly(pw_gp, width=450, height=500,
               tooltip="label") %>%
  animation_button(label="Go") %>%
  animation_slider(len=0.8, x=0.5,
                  xanchor="center") %>%
  animation_opts(easing="linear", transition = 0)
htmlwidgets::saveWidget(pwg,
                        file="html/penguins_cl_ward.html",
                        selfcontained = TRUE)

# Single
ps_anim <- render_anim(p_s_hfly$data, vars=1:4,
                      frames=pt1i,
                      edges = p_s_hfly$edges,
                      obs_labels=paste0(1:nrow(p_s_hfly$data),
                                          p_s_hfly$data$c1))

ps_gp <- ggplot() +
  geom_segment(data=ps_anim$edges,
              aes(x=x, xend=xend,
                  y=y, yend=yend,
                  frame=frame)) +
  geom_point(data=ps_anim$frames,
             aes(x=P1, y=P2,
                 frame=frame,
                 shape=factor(node),
                 label=obs_labels),
             alpha=0.8, size=1) +
xlim(-1,1) + ylim(-1,1) +

```



```

scale_shape_manual(values=c(16, 46)) +
coord_equal() +
theme_bw() +
theme(legend.position="none",
      axis.text=element_blank(),
      axis.title=element_blank(),
      axis.ticks=element_blank(),
      panel.grid=element_blank())

psg <- ggplotly(ps_gp, width=450, height=500,
               tooltip="label") %>%
  animation_button(label="Go") %>%
  animation_slider(len=0.8, x=0.5,
                  xanchor="center") %>%
  animation_opts(easing="linear", transition = 0)
htmlwidgets::saveWidget(psg,
                        file="html/penguins_cl_single.html",
                        selfcontained = TRUE)

```

Figure 8.9: Animation of dendrogram from Wards (top) and single (bottom) linkage clustering of the penguins data.

Viewing the dendrograms in high-dimensions provides insight into how they have joined points to clusters. It helps to make sense of the result, and whether it is a useful result or not.

Chapter 9

k-means clustering

One of the simplest and efficient techniques for clustering data is the *k*-means algorithm. The algorithm begins with a choice for *k*, the number of clusters to divide the data into. It is seeded with *k* initial means, and sequentially iterates through the observations, assigning them to the nearest mean, and re-calculating the *k* means. It stops at a given number of iterations or when points no longer change clusters. The algorithm will tend to segment the data into roughly equal sized, or spherical clusters, and thus will work well if the clusters are separated and equally spherical in shape.

A good place to learn ore about the *k*-means algorithm is Chapter 20 of Boehmke & Greenwell (2019). The algorithm has been in use for a long time! It was named *k*-means by MacQueen (1967), but developed by Lloyd in 1957 (as described in Lloyd (1982)) and separately by Forgy (1965), and perhaps others as it is a very simple procedure.

The key elements to examine in a k-means clustering algorithm result are:

- means
- boundaries

9.1 Examining results in 2D

Figure ?? shows the results of *k*-means clustering on the 2D `simple_clusters` data and two variables of the penguins data. We can see that it works well when the clusters are spherical, but for the penguins data it fails because the shape of the clusters is elliptical. It actually makes a mistake that would not be made if we simply visually clustered: cluster 3 has grouped points across a gap, a divide that visually we would all agree should form a separation.

```

library(mulgar)
library(ggplot2)
library(dplyr)
library(colorspace)
library(patchwork)
data("simple_clusters")
load("data/penguins_sub.rda")

set.seed(202305)
sc_bl_bd_km <- kmeans(simple_clusters[,1:2], centers=2,
                      iter.max = 50, nstart = 5)
sc_bl_bd_km_means <- data.frame(sc_bl_bd_km$centers) %>%
  mutate(cl = factor(rownames(sc_bl_bd_km$centers)))
sc_bl_bd_km_d <- simple_clusters[,1:2] %>%
  mutate(cl = factor(sc_bl_bd_km$cluster))

sc_bl_bd_km_p <- ggplot() +
  geom_point(data=sc_bl_bd_km_d,
             aes(x=x1, y=x2, colour=cl),
             shape=16, alpha=0.4) +
  geom_point(data=sc_bl_bd_km_means,
             aes(x=x1, y=x2, colour=cl),
             shape=3, size=5) +
  scale_color_discrete_divergingx("Zissou 1") +
  ggtitle("(a)") +
  theme_minimal() +
  theme(aspect.ratio = 1,
        legend.position = "bottom",
        legend.title = element_blank())

p_bl_bd_km <- kmeans(penguins_sub[,1:2], centers=3,
                     iter.max = 50, nstart = 5)
p_bl_bd_km_means <- data.frame(p_bl_bd_km$centers) %>%
  mutate(cl = factor(rownames(p_bl_bd_km$centers)))
p_bl_bd_km_d <- penguins_sub[,1:2] %>%
  mutate(cl = factor(p_bl_bd_km$cluster))

p_bl_bd_km_p <- ggplot() +
  geom_point(data=p_bl_bd_km_d,
             aes(x=bl, y=bd, colour=cl),
             shape=16, alpha=0.4) +
  geom_point(data=p_bl_bd_km_means,
             aes(x=bl, y=bd, colour=cl),
             shape=3, size=5) +

```

```

    scale_color_discrete_divergingx("Zissou 1") +
    ggtitle("(b)") +
    theme_minimal() +
    theme(aspect.ratio = 1,
          legend.position = "bottom",
          legend.title = element_blank())

sc_bl_bd_km_p + p_bl_bd_km_p + plot_layout(ncol=2)

```

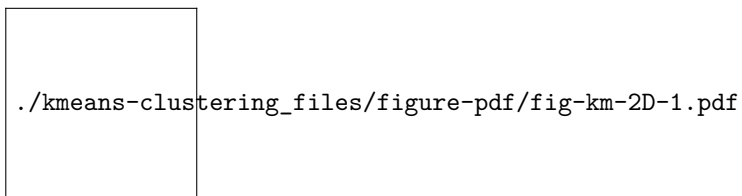


Figure 9.1: Examining k -means clustering results for simple clusters (a) and two variables of the penguins data (b). The means are indicated by a +. The results are perfect for the simple clusters but not for the penguins data. The penguin clusters are elliptically shaped which is not captured by k -means. Cluster 3 has observations grouped across a gap in the data.

9.2 Examining results in high dimensions

This approach extends to high-dimensions. One colours observations by the cluster label, and overlays the final cluster means. If we see gaps in points in a single cluster it would mean that k -means fails to see important cluster structure. This is what happens with the 4D penguins data as shown in ?@fig-p-km.

```

p_km <- kmeans(penguins_sub[,1:4], centers=3,
              iter.max = 50, nstart = 5)
p_km_means <- data.frame(p_km$centers) %>%
  mutate(cl = factor(rownames(p_km$centers)))
p_km_d <- penguins_sub[,1:4] %>%
  mutate(cl = factor(p_km$cluster))

library(tourr)
p_km_means <- p_km_means %>%
  mutate(type = "mean")
p_km_d <- p_km_d %>%
  mutate(type = "data")
p_km_all <- bind_rows(p_km_means, p_km_d)

```

```

p_km_all$type <- factor(p_km_all$type, levels=c("mean", "data"))
p_pch <- c(3, 20)[as.numeric(p_km_all$type)]
p_cex <- c(3, 1)[as.numeric(p_km_all$type)]
animate_xy(p_km_all[,1:4], col=p_km_all$c1,
           pch=p_pch, cex=p_cex, axes="bottomleft")
render_gif(p_km_all[,1:4],
           grand_tour(),
           display_xy(col=p_km_all$c1,
                     pch=p_pch,
                     cex=p_cex,
                     axes="bottomleft"),
           gif_file="gifs/p_km.gif",
           width=400,
           height=400,
           frames=500)

```

Exercises

1. Compute a k -means clustering for the `fake_trees` data, varying k to about 20. Choose your best k , and examine the solution using the first 10 PCs on the data. It should capture the data quite nicely, although it will break up each branch into multiple clusters.
2. Compute a k -means clustering of the first four PCs of the `aflw` data. Examine the best solution (you choose which k), and describe how it divides the data. By examining the means, can you tell if it extracts clusters of offensive vs defensive vs midfield players? Or does it break the data into high skills vs low skills?

Chapter 10

Model-based clustering

Model-based clustering Fraley & Raftery (2002) fits a multivariate normal mixture model to the data. It uses the EM algorithm to fit the parameters for the mean, variance–covariance of each population, and the mixing proportion. The variance-covariance matrix is re-parametrized using an eigen-decomposition

$$\Sigma_k = \lambda_k D_k A_k D_k', \quad k = 1, \dots, g \text{ (number of clusters)}$$

resulting in several model choices, ranging from simple to complex, as shown in Table ??.

```
library(dplyr)
library(kableExtra)
library(ggplot2)
library(mclust)
library(mulgar)
library(patchwork)
library(colorspace)
library(tourr)
```

Note the distribution descriptions “spherical” and “ellipsoidal”. These are descriptions of the shape of the variance-covariance for a multivariate normal distribution. A standard multivariate normal distribution has a variance-covariance matrix with zeros in the off-diagonal elements, which corresponds to spherically shaped data. When the variances (diagonals) are different or the variables are correlated, then the shape of data from a multivariate normal is ellipsoidal.

The models are typically scored using the Bayes Information Criterion (BIC), which is based on the log likelihood, number of variables, and number of mixture

Table 10.1: Parameterizations of the covariance matrix.

Model	Sigma	Family	Volume	Shape	Orientation
EII	λI	Spherical	Equal	Equal	NA
VII	$\lambda_k I$	Spherical	Variable	Equal	NA
EEI	λA	Diagonal	Equal	Equal	Coordinate axes
VEI	$\lambda_k A$	Diagonal	Variable	Equal	Coordinate axes
EVI	λA_k	Diagonal	Equal	Variable	Coordinate axes
VVI	$\lambda_k A_k$	Diagonal	Variable	Variable	Coordinate axes
EEE	λDAD^T	Diagonal	Equal	Equal	Equal
EVE	$\lambda DA_k D^T$	Ellipsoidal	Equal	Variable	Equal
VEE	$\lambda_k DAD^T$	Ellipsoidal	Variable	Equal	Equal
VVE	$\lambda_k DA_k D^T$	Ellipsoidal	Variable	Equal	Equal
EEV	$\lambda D_k A D_k^T$	Ellipsoidal	Equal	Variable	Variable
VEV	$\lambda_k D_k A D_k^T$	Ellipsoidal	Variable	Variable	Variable
EVV	$\lambda D_k A_k D_k^T$	Ellipsoidal	Equal	Variable	Variable
VVV	$\lambda_k D_k A_k D_k^T$	Ellipsoidal	Variable	Variable	Variable

components. They should also be assessed using graphical methods, as we demonstrate using the penguins data.

10.1 Examining the model in 2D

We start with two of the four real-valued variables (`bl`, `fl`) and the three `species`. The goal is to determine whether model-based methods can discover clusters that closely correspond to the three species. Based on the scatterplot in Figure ?? we would expect it to do well, and suggest an elliptical variance-covariance of roughly equal sizes as the model.

```
load("data/penguins_sub.rda")
ggplot(penguins_sub, aes(x=bl,
                        y=fl)) + #,
                        #colour=species)) +

  geom_point() +
  geom_density2d(colour="#3B99B1") +
  theme_minimal() +
  theme(aspect.ratio = 1)
```

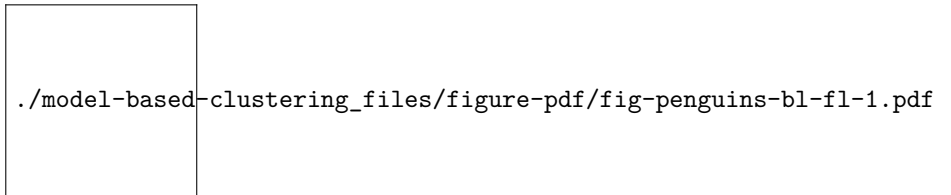



Figure 10.1: Scatterplot of flipper length by bill length of the penguins data.

```

penguins_BIC <- mclustBIC(penguins_sub[,c(1,3)])
ggmc <- ggmcbic(penguins_BIC, cl=2:9, top=4) +
  scale_color_discrete_divergingx(palette = "Roma") +
  ggtitle("(a)") +
  theme_minimal()
penguins_mc <- Mclust(penguins_sub[,c(1,3)],
  G=3,
  modelNames = "EVE")
penguins_mce <- mc_ellipse(penguins_mc)
penguins_cl <- penguins_sub[,c(1,3)]
penguins_cl$cl <- factor(penguins_mc$classification)
ggell <- ggplot() +
  geom_point(data=penguins_cl, aes(x=bl, y=fl,
    colour=cl),
    alpha=0.3) +
  geom_point(data=penguins_mce$ell, aes(x=bl, y=fl,
    colour=cl),
    shape=16) +
  geom_point(data=penguins_mce$mn, aes(x=bl, y=fl,
    colour=cl),
    shape=3, size=2) +
  scale_color_discrete_divergingx(palette = "Zissou 1") +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position="none") +
  ggtitle("(b)")
ggmc + ggell + plot_layout(ncol=2)

```

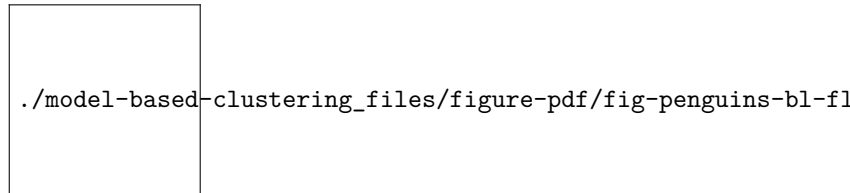


Figure 10.2: Summary plots from model-based clustering: (a) BIC values for clusters 2-9 of top four models, (b) variance-covariance ellipses and cluster means (+) corresponding to the best model. The best model is three-cluster EVE, which has differently shaped variance-covariances albeit the same volume and orientation.

Figure ?? summarises the results. All models agree that three clusters is the best. The different variance-covariance models for three clusters have similar BIC values with EVE (different shape, same volume and orientation) being slightly higher. These plots are made from the `mclust` package output using the `ggmcbic` and `mc_ellipse` functions from the `mulgar` package.

10.2 Extending to higher dimensions

Now we will examine how model-based clustering will group the penguins data using all four variables.

```
penguins_BIC <- mclustBIC(penguins_sub[,1:4])
ggmc <- ggmcbic(penguins_BIC, cl=2:9, top=7) +
  scale_color_discrete_divergingx(palette = "Roma") +
  theme_minimal()
ggmc
```

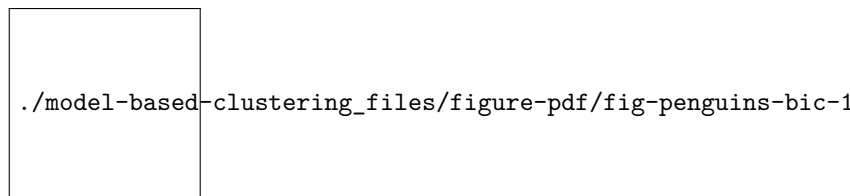


Figure 10.3: BIC values for the top models for 2-9 clusters on the penguins data. The interpretation is mixed: if one were to choose three clusters any of the variance-covariance models would be equally as good, but the very best model is the four-cluster VEE.

```

penguins_mc <- Mclust(penguins_sub[,1:4],
                     G=4,
                     modelNames = "VEE")
penguins_mce <- mc_ellipse(penguins_mc)
penguins_cl <- penguins_sub
penguins_cl$cl <- factor(penguins_mc$classification)

penguins_mc_data <- penguins_cl %>%
  select(bl:bm, cl) %>%
  mutate(type = "data") %>%
  bind_rows(bind_cols(penguins_mce$ell,
                      type=rep("ellipse",
                                nrow(penguins_mce$ell)))) %>%
  mutate(type = factor(type))

animate_xy(penguins_mc_data[,1:4],
           col=penguins_mc_data$cl,
           pch=c(4, 20)[as.numeric(penguins_mc_data$type)],
           axes="off")

#
load("data/penguins_tour_path.rda")
render_gif(penguins_mc_data[,1:4],
           planned_tour(pt1),
           display_xy(col=penguins_mc_data$cl,
                     pch=c(4, 20)[
                       as.numeric(penguins_mc_data$type)],
                     axes="off",
                     half_range = 0.7),
           gif_file="gifs/penguins_best_mc.gif",
           frames=500,
           loop=FALSE)

penguins_mc <- Mclust(penguins_sub[,1:4],
                     G=3,
                     modelNames = "EEE")
penguins_mce <- mc_ellipse(penguins_mc)
penguins_cl <- penguins_sub
penguins_cl$cl <- factor(penguins_mc$classification)

penguins_mc_data <- penguins_cl %>%
  select(bl:bm, cl) %>%
  mutate(type = "data") %>%

```

```

bind_rows(bind_cols(penguins_mce$ell,
                    type=rep("ellipse",
                             nrow(penguins_mce$ell)))) %>%
mutate(type = factor(type))

animate_xy(penguins_mc_data[,1:4],
           col=penguins_mc_data$c1,
           pch=c(4, 20)[as.numeric(penguins_mc_data$type)],
           axes="off")

# Save the animated gif
load("data/penguins_tour_path.rda")
render_gif(penguins_mc_data[,1:4],
           planned_tour(pt1),
           display_xy(col=penguins_mc_data$c1,
                     pch=c(4, 20)[
                       as.numeric(penguins_mc_data$type)],
                     axes="off",
                     half_range = 0.7),
           gif_file="gifs/penguins_simpler_mc.gif",
           frames=500,
           loop=FALSE)

```

Figure 10.4: Examining the model-based clustering results for the penguins data: (a) best model according to BIC value, (b) simpler three-cluster model. Dots are ellipse points, and “x” are data points. It is important to note that the three cluster solution fits the data better, even though it has a lower BIC.

Visualising the final choices of models with similarly high BIC values helps to choose which best fits the data. It may not be the one with the highest value.

Exercises

1. Examine the three cluster EVE, VVE and VEE models with the tour, and explain whether these are distinguishably different from the EEE three cluster model.
2. Fit model-based clustering to the the `clusters`. Does it suggest the data has three clusters? Using the tour examine the best model model. How well does this fit the data?
3. Fit model-based clustering to the the `multiclust`. Does it suggest the data has six clusters? Using the tour examine the best model model. How well does this fit the data?
4. Fit model-based clustering to the `fake_trees` data. Does it suggest that

the data has 10 clusters? If not, why do you think this is? Using the tour examine the best model model. How well does this fit the branching structure?

5. Try fitting model-based clustering to the `aflw` data? What is the best model? Is the solution related to offensive vs defensive vs mid-fielder skills?
6. TODO: Provide some challenge data sets

Chapter 11

Self-organizing maps

A self-organizing map (SOM) Kohonen (2001) is constructed using a constrained k -means algorithm. A 1D or 2D net is stretched through the data. The knots, in the net, form the cluster means, and the points closest to the knot are considered to belong to that cluster. The similarity of nodes (and their corresponding clusters) is defined as proportional to their distance from one another on the net. Unlike k -means one would normally choose a largish net, with more nodes than expected clusters. A well-separated cluster in the data would likely be split across multiple nodes in the net. Examining the net where nodes are empty of points we would interpret this as a gap in the original data.

A self-organising map is like a fisherwoman's net, as the net is pulled in it catches the fish near knots in the net. We would examine the net in

- 2D to extract the fish.
- high-dimensions to see how it was woven through the space to catch fish.

Figure ?? illustrates how the SOM fits the penguins data. SOM is not ideal for clustered data where there are gaps. It is better suited for data that lies on a non-linear low-dimensional manifold. To model data like the penguins the first step is to set up a net that will cover more than the three clusters. Here we have chosen to use a 5×5 rectangular grid. (The option allows for a hexagonal grid, which would make for a better tiled 2D map, but this is not useful for viewing the model in high dimensions.) Like k -means clustering the fitted model can change substantially depending on the initialisation, so setting a seed will ensure a consistent result. We have also initialised the positions of the knots using PCA, which stretches the net in the main two directions of variance of the data, generally giving better results.

```

library(kohonen)
library(aweSOM)
library(mulgar)
library(dplyr)
library(ggplot2)
library(colorspace)
load("data/penguins_sub.rda")

set.seed(947)
p_grid <- kohonen::somgrid(xdim = 5, ydim = 5,
                           topo = 'rectangular')
p_init <- somInit(as.matrix(penguins_sub[,1:4]), 5, 5)
p_som <- som(as.matrix(penguins_sub[,1:4]),
             rlen=2000,
             grid = p_grid,
             init = p_init)

```

The resulting model object is used to construct an object containing the original data, the 2D map, the map in p -D, with edges, and segments to connect points to represent the next using the `som_model()` function from `mulgar`. The 2D map shows a configuration of the data in 2D which best displays the clusters, much like how a PCA or LDA plot would be used.

```

p_som_df_net <- som_model(p_som)
p_som_data <- p_som_df_net$data %>%
  mutate(species = penguins_sub$species)
p_som_map_p <- ggplot() +
  geom_segment(data=p_som_df_net$edges_s,
              aes(x=x, xend=xend, y=y,
                  yend=yend)) +
  geom_point(data=p_som_data,
             aes(x=map1, y=map2,
                 colour=species),
             size=3, alpha=0.5) +
  xlab("map 1") + ylab("map 2") +
  scale_color_discrete_divergingx(
    palette="Zissou 1") +
  theme_minimal() +
  theme(aspect.ratio = 1,
        legend.position = "bottom",
        legend.title = element_blank(),
        axis.text = element_blank())

```

The object can also be modified into the pieces needed to show the net in p -D.

You need the data, points marking the net, and edges indicating which points to connect to draw the net.

```
library(tourr)

# Set up data
p_som_map <- p_som_df_net$net %>%
  mutate(species = "0", type="net")
p_som_data <- p_som_data %>%
  select(bl:bm, species) %>%
  mutate(type="data",
         species = as.character(species))
p_som_map_data <- bind_rows(p_som_map, p_som_data)
p_som_map_data$type <- factor(p_som_map_data$type,
                             levels=c("net", "data"))
p_som_map_data$species <- factor(p_som_map_data$species,
                                levels=c("0", "Adelie", "Chinstrap", "Gentoo"))
p_pch <- c(46, 16)[as.numeric(p_som_map_data$type)]
p_col <- c("black", hcl.colors(3, "Zissou 1"))[as.numeric(p_som_map_data$species)]
animate_xy(p_som_map_data[,1:4],
           col=p_col,
           pch=p_pch,
           edges=as.matrix(p_som_df_net$edges),
           edges.col = "black",
           axes="bottomleft")
```

Figure 11.2: Examining the SOM map views in 2D and with the data in 4D. Points are coloured by species, which was not used for the modeling. The 2D map shows that the **map 2** direction is primarily distinguishing the Gentoo from the others, and **map 1** is imperfectly distinguishing the Chinstrap from Adelie. The map in the data space shows how it is woven into the shape of the data.

The SOM fit, with a 5×5 grid, for the penguins has the data clustered into 25 groups. This doesn't work as a clustering technique on its own, if we remember that the data has three clusters corresponding to three species of penguins. Using species to colour the points helps to see what SOM has done. It has used about seven nodes to capture the separated Gentoo group. These are mostly in the **map 2** direction, which means that this direction (like a direction in PCA) is useful for distinguishing the Gentoo penguins from the others. The other two species are mixed on the map, but roughly spread out on the direction of **map 1**.

Exercises

1. Fit an SOM to the first four PCs of the **aflw** data. Examine the best solution (you choose the size of the net), and describe how the map lays out the data. Does it show offensive vs defensive vs midfield players? Or does it tend to show high skills vs low skills?
2. Fit an SOM to the first 10 PCs of the **fake_trees** data, using your choice of net size. How well does the map show the branching structure?
3. Examine a range of SOM nets fitted to the first 10 PCs of the **fake_trees** data in the 10D space using a tour. Set the values of **r1en** to be 5, 50, 500. How does the net change on this parameter?
4. Plot the distances output for the SOM fit to the penguins data. Mark the observations that have the 5 biggest distances, and show these in a tour. These are the observations where the net has fitted least well, and may be outliers.

Chapter 12

Summarising and comparing results

12.1 Summarising results

The key elements for summarising cluster results are the centres of the clusters and the within-cluster variability of the observations. Adding cluster means to any plot, including tour plots, is easy. You add the additional rows, or a new data set, and set the point shape to be distinct.

Summarising the variability is difficult. For model-based clustering, the shape of the clusters is assumed to be elliptical, so p -dimensional ellipses can be used to show the solution, as done in Chapter ???. Generally, it is common to plot a convex hull of the clusters, as in Figure ???. This can also be done in high-dimensions, using the R package `cxhull` to compute the p -D convex hull.

```
library(mclust)
library(tidyr)
library(dplyr)
library(gt)
library(cxhull)
library(ggplot2)
library(colorspace)
```

```
library(mclust)
library(tidyr)
library(dplyr)
library(gt)
```

```

library(cxhull)
library(ggplot2)
library(colorspace)
load("data/penguins_sub.rda")
p_dist <- dist(penguins_sub[,1:4])
p_hcw <- hclust(p_dist, method="ward.D2")

p_cl <- data.frame(cl_w = cutree(p_hcw, 3))

penguins_mc <- Mclust(penguins_sub[,1:4],
                      G=3,
                      modelNames = "EEE")
p_cl <- p_cl %>%
  mutate(cl_mc = penguins_mc$classification)

p_cl <- p_cl %>%
  mutate(cl_w_j = jitter(cl_w),
         cl_mc_j = jitter(cl_mc))

# Arranging by cluster id is important to define edges
penguins_cl <- penguins_sub %>%
  mutate(cl_w = p_cl$cl_w,
         cl_mc = p_cl$cl_mc) %>%
  arrange(cl_w)

# Penguins in 2D
# Duplicate observations need to be removed for convex hull calculation
psub <- penguins_cl %>%
  select(bl, bd)
dup <- duplicated(psub)
psub <- penguins_cl %>%
  select(bl, bd, cl_w) %>%
  filter(!dup) %>%
  arrange(cl_w)

ncl <- psub %>%
  count(cl_w) %>%
  arrange(cl_w) %>%
  mutate(cum_n = cumsum(n))
phull <- NULL
for (i in unique(psub$cl_w)) {
  x <- psub %>%
    dplyr::filter(cl_w == i) %>%
    select(bl, bd)

```

```

ph <- cxhull(as.matrix(x))$edges
if (i > 1) {
  ph <- ph + ncl$cumn[i-1]
}
ph <- cbind(ph, rep(i, nrow(ph)))
phull <- rbind(phull, ph)
}
phull <- as.data.frame(phull)
colnames(phull) <- c("from", "to", "cl_w")
phull_segs <- data.frame(x = psub$bl[phull$from],
                        y = psub$bd[phull$from],
                        xend = psub$bl[phull$to],
                        yend = psub$bd[phull$to],
                        cl_w = phull$cl_w)
phull_segs$cl_w <- factor(phull$cl_w)
psub$cl_w <- factor(psub$cl_w)
p_chull2D <- ggplot() +
  geom_point(data=psub, aes(x=bl, y=bd,
                           colour=cl_w)) +
  geom_segment(data=phull_segs, aes(x=x, xend=xend,
                                    y=y, yend=yend,
                                    colour=cl_w)) +
  scale_colour_discrete_divergingx(palette = "Zissou 1") +
  theme_minimal() +
  theme(aspect.ratio = 1)

```

```

ncl <- penguins_cl %>%
  count(cl_w) %>%
  arrange(cl_w) %>%
  mutate(cumn = cumsum(n))
phull <- NULL
for (i in unique(penguins_cl$cl_w)) {
  x <- penguins_cl %>%
    dplyr::filter(cl_w == i)
  ph <- cxhull(as.matrix(x[,1:4]))$edges
  if (i > 1) {
    ph <- ph + ncl$cumn[i-1]
  }
  ph <- cbind(ph, rep(i, nrow(ph)))
  phull <- rbind(phull, ph)
}
phull <- as.data.frame(phull)
colnames(phull) <- c("from", "to", "cl_w")

```

```

phull$cl_w <- factor(phull$cl_w)
penguins_cl$cl_w <- factor(penguins_cl$cl_w)

animate_xy(penguins_cl[,1:4], col=penguins_cl$cl_w,
           edges=as.matrix(phull[,1:2]), edges.col=phull$cl_w)
render_gif(penguins_cl[,1:4],
           tour_path = grand_tour(),
           display = display_xy(col=penguins_cl$cl_w,
                               edges=as.matrix(phull[,1:2]),
                               edges.col=phull$cl_w),
           gif_file = "gifs/penguins_chull.gif",
           frames = 500,
           width = 400,
           height = 400)

```

Figure 12.2: Convex hulls summarising the extent of Wards linkage clustering in 2D and 4D.

12.2 Comparing two clusterings

Each cluster analysis will result in a vector of class labels for the data. To compare two results we would tabulate and plot the pair of integer variables. The labels given to each cluster will likely differ. If the two methods agree, there will be just a few cells with large counts among mostly empty cells.

Below is a comparison between the three cluster results of Wards linkage hierarchical clustering (rows) and model-based clustering (columns). The two methods mostly agree, as seen from the three cells with large counts, and most cells with zeros. They disagree only on eight penguins. These eight penguins would be considered to be part of cluster 1 by Wards, but model-based considers them to be members of cluster 2.

The two methods label them clusters differently: what Wards labels as cluster 3, model-based labels as cluster 2. The labels given by any algorithm are arbitrary, and can easily be changed to coordinate between methods.

```

p_cl %>%
  count(cl_w, cl_mc) %>%
  pivot_wider(names_from = cl_mc,
              values_from = n,
              values_fill = 0) %>%
  gt() %>%
  tab_spanner(label = "cl_mc", columns=c(`2`, `3`, `1`)) %>%

```

```
cols_width(everything() ~ px(60))
```

cl_w	cl_mc		
	2	3	1
1	8	0	149
2	0	119	0
3	57	0	0

We can examine the disagreement by linking a plot of the table, with a tour plot. Here is how to do this with `liminal`. Figure ?? and Figure ?? show screenshots of the exploration of the eight penguins on which the methods disagree. It makes sense that there is some confusion. These penguins are part of the large clump of observations that don't separate cleanly into two clusters. The eight penguins are in the middle of this clump. Realistically, both methods result in a plausible clustering, and it is not clear how these penguins should be grouped.

```
library(liminal)
limn_tour_link(
  p_cl[,3:4],
  penguins_cl,
  cols = bl:bm,
  color = cl_w
)
```

Exercises

1. Compare the results of the four cluster model-based clustering with that of the four cluster Wards linkage clustering of the penguins data.

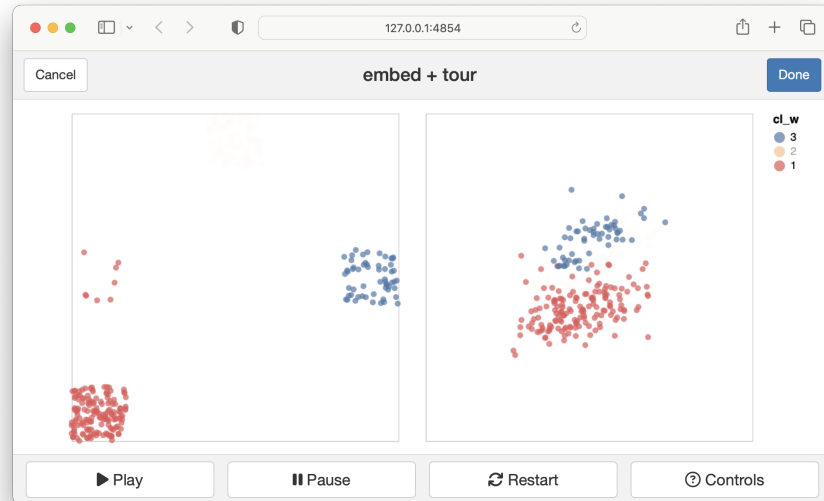


Figure 12.3: Linking the confusion table with a tour using liminal. Points are coloured according to Wards linkage. The disagreement on eight penguins is with cluster 1 from Wards and cluster 2 from model-based.



Figure 12.4: Highlighting the penguins where the methods disagree so we can see where these observations are located relative to the two clusters.

Part IV

Supervised classification

Chapter 13

Linear discriminant analysis

Linear discriminant analysis (LDA) dates to the early 1900s. It's one of the most elegant and simple techniques for both modeling separation between groups, and as an added bonus, producing a low-dimensional representation of the differences between groups. LDA has two strong assumptions: the groups are samples from multivariate normal distributions, and each have the same variance-covariance. If the latter assumption is relaxed, a slightly less elegant solution results from quadratic discriminant analysis.

Useful explanations can be found in Venables & Ripley (2002a) and Ripley (1996). A good general treatment of parametric methods for supervised classification can be found in R. A. Johnson & Wichern (2002) or another similar multivariate analysis textbook. It's also useful to know that hypothesis testing for the difference in multivariate means using multivariate analysis of variance (MANOVA) has similar assumptions to LDA. Also model-based clustering assumes that each cluster arises from a multivariate normal distribution, and is related to LDA. The methods described here can be used to check these assumptions when applying these methods, too..

Because LDA is a parametric model it is important to check that the assumptions are reasonable:

- shape of clusters are elliptical
- cluster sizes are the same.

13.1 Extracting the key elements of the model

LDA builds the model on the between-group sum-of-square matrix

$$B = \sum_{k=1}^g n_k (\bar{X}_k - \bar{X})(\bar{X}_k - \bar{X})^\top$$

which measures the differences between the class means, compared with the overall data mean \bar{X} and the within-group sum-of-squares matrix,

$$W = \sum_{k=1}^g \sum_{i=1}^{n_k} (X_{ki} - \bar{X}_k)(X_{ki} - \bar{X}_k)^\top$$

which measures the variation of values around each class mean. The linear discriminant space is generated by computing the eigenvectors (canonical coordinates) of $W^{-1}B$, and this is the $(g - 1)$ -D space where the group means are most separated with respect to the pooled variance-covariance.

$$\delta_k(x) = (x - \mu_k)^\top W^{-1} \mu_k + \log \pi_k$$

where π_k is a prior probability for class k that might be based on unequal sample sizes, or cost of misclassification. The LDA classifier rule is to *assign a new observation to the class with the largest value*.

We can fit an LDA model using the `lda()` function from the **MASS** package. Here we have used the **penguins** data, assuming equal prior probability, to illustrate.

```
library(dplyr)
library(mulgar)
library(MASS)
load("data/penguins_sub.rda")

p_lda <- lda(species~bl+bd+fl+bm, data=penguins_sub, prior=c(1/3, 1/3, 1/3))
options(digits=2)
p_lda
```

Call:

```
lda(species ~ bl + bd + fl + bm, data = penguins_sub, prior = c(1/3,
  1/3, 1/3))
```

Prior probabilities of groups:

Adelie	Chinstrap	Gentoo
0.33	0.33	0.33

Group means:

	bl	bd	fl	bm
Adelie	-0.95	0.60	-0.78	-0.62
Chinstrap	0.89	0.64	-0.37	-0.59

```
Gentoo      0.65 -1.10  1.16  1.10
```

Coefficients of linear discriminants:

```
      LD1   LD2
b1  0.24 -2.31
bd -2.04  0.19
fl  1.20  0.08
bm  1.22  1.24
```

Proportion of trace:

```
      LD1   LD2
0.83 0.17
```

Because there are three classes the dimension of the discriminant space is 2D. We can easily extract the group means from the model.

```
p_lda$means
```

```
      b1   bd   fl   bm
Adelie -0.95 0.60 -0.78 -0.62
Chinstrap 0.89 0.64 -0.37 -0.59
Gentoo  0.65 -1.10  1.16  1.10
```

The coefficients to project the data into the discriminant space, that is the eigenvectors of $W^{-1}B$ are:

```
p_lda$scaling
```

```
      LD1   LD2
b1  0.24 -2.31
bd -2.04  0.19
fl  1.20  0.08
bm  1.22  1.24
```

and the predicted values, which include class predictions, and coordinates in the discriminant space are generated as:

```
p_lda_pred <- predict(p_lda, penguins_sub)
```

The best separation between classes can be viewed from this object, which can be shown to match the original data projected using the `scaling` component of the model object.

```
library(colorspace)
library(ggplot2)
```

```

library(ggpubr)
p_lda_pred_x1 <- data.frame(p_lda_pred$x)
p_lda_pred_x1$species <- penguins_sub$species
p_lda1 <- ggplot(p_lda_pred_x1,
                 aes(x=LD1, y=LD2,
                     colour=species)) +
  geom_point() +
  xlim(-6, 8) + ylim(-6.5, 5.5) +
  scale_color_discrete_divergingx("Zissou 1") +
  ggtitle("(a)") +
  theme_minimal() +
  theme(aspect.ratio = 1, legend.title = element_blank())

p_lda_pred_x2 <- data.frame(as.matrix(penguins_sub[,1:4]) %*%
                           p_lda$scaling)
p_lda_pred_x2$species <- penguins_sub$species
p_lda2 <- ggplot(p_lda_pred_x2,
                 aes(x=LD1, y=LD2,
                     colour=species)) +
  geom_point() +
  xlim(-6, 8) + ylim(-7, 5.5) +
  scale_color_discrete_divergingx("Zissou 1") +
  ggtitle("(b)") +
  theme_minimal() +
  theme(aspect.ratio = 1, legend.title = element_blank())
ggarrange(p_lda1, p_lda2, ncol=2,
          common.legend = TRUE, legend = "bottom")

```

./LDA_files/figure-pdf/fig-p-lda-1.pdf

Figure 13.1: Penguins projected into the 2D discriminant space, done two ways: (a) using the predicted values, (b) directly projecting using the model component. The scale is not quite the same but the projected data is identical in shape.

The W and B matrices cannot be extracted from the model object, so we need to compute these separately. We only need W actually. It is useful to think of this as the pooled variance-covariance matrix. Because the assumption for LDA is that the population group variance-covariances are identical, we estimate this by computing them for each class and then averaging them to get the pooled

variance-covariance matrix. It's laborious, but easy.

```
p_vc_pool <- mulgar::pooled_vc(penguins_sub[,1:4],
                               penguins_sub$species)

p_vc_pool
```

	b1	bd	f1	bm
b1	0.31	0.18	0.13	0.18
bd	0.18	0.32	0.14	0.20
f1	0.13	0.14	0.23	0.16
bm	0.18	0.20	0.16	0.31

This can be used to draw an ellipse corresponding to the pooled variance-covariance that is used by the LDA model.

13.2 Checking assumptions

This LDA approach is widely applicable, but it is useful to check the underlying assumptions on which it depends: (1) that the cluster structure corresponding to each class forms an ellipse, showing that the class is consistent with a sample from a multivariate normal distribution, and (2) that the variance of values around each mean is nearly the same. Figure ?? and Figure ?? illustrates two datasets, of which only one is consistent with these assumptions. Other parametric models, such as quadratic discriminant analysis or logistic regression, also depend on assumptions about the data which should be validated.

```
lda1 <- ggplot(penguins_sub, aes(x=b1,
                                y=bd,
                                colour=species)) +

  geom_point() +
  scale_color_discrete_diverging("Zissou 1") +
  xlim(-2.5, 3) + ylim(-2.5, 2.5) +
  ggtitle("(a)") +
  theme_minimal() +
  theme(aspect.ratio = 1)

p_ell <- NULL
for (i in unique(penguins_sub$species)) {
  x <- penguins_sub %>% dplyr::filter(species == i)
  e <- gen_xvar_ellipse(x[,1:2], n=150, nstd=1.5)
  e$species <- i
  p_ell <- bind_rows(p_ell, e)
}

lda2 <- ggplot(p_ell, aes(x=b1,
                          y=bd,
```

```

                                colour=species)) +
  geom_point() +
  scale_color_discrete_divergingx("Zissou 1") +
  xlim(-2.5, 3) + ylim(-2.5, 2.5) +
  ggtitle("(b)") +
  theme_minimal() +
  theme(aspect.ratio = 1)
ggarrange(lda1, lda2, ncol=2,
          common.legend = TRUE, legend = "bottom")

```

./LDA_files/figure-pdf/fig-lda-assumptions1-1.pdf

Figure 13.2: Scatterplot of flipper length by bill length of the penguins data, and corresponding variance-covariance ellipses. There is a small amount of difference between the ellipses, but they are similar enough to be confident in assuming the population variance-covariances are equal.

```

# Now repeat for a data set that violates assumptions
data(bushfires)
lda3 <- ggplot(bushfires, aes(x=log_dist_cfa,
                             y=log_dist_road,
                             colour=cause)) +

  geom_point() +
  scale_color_discrete_divergingx("Zissou 1") +
  xlim(6, 11) + ylim(-1, 10.5) +
  ggtitle("(a)") +
  theme_minimal() +
  theme(aspect.ratio = 1)
b_ell <- NULL
for (i in unique(bushfires$cause)) {
  x <- bushfires %>% dplyr::filter(cause == i)
  e <- gen_xvar_ellipse(x[,c(57, 59)], n=150, nstd=2)
  e$cause <- i
  b_ell <- bind_rows(b_ell, e)
}
lda4 <- ggplot(b_ell, aes(x=log_dist_cfa,
                          y=log_dist_road,
                          colour=cause)) +

```



```

geom_point() +
scale_color_discrete_divergingx("Zissou 1") +
xlim(6, 11) + ylim(-1, 10.5) +
ggtitle("(b)") +
theme_minimal() +
theme(aspect.ratio = 1)
ggarrange(lda3, lda4, ncol=2,
          common.legend = TRUE, legend = "bottom")

```

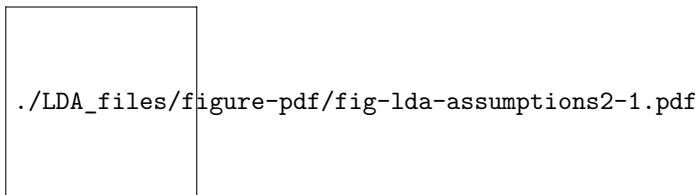


Figure 13.3: Scatterplot of distance to cfa and road for the bushfires data, and corresponding variance-covariance ellipses. There is a lot of difference between the ellipses, so it cannot be assumed that the population variance-covariances are equal.

This approach extends to any dimension. We would use the same projection sequence to view both the data and the variance-covariance ellipses, as in Figure ???. It can be seen that there is some difference in the shape and size of the ellipses in some projections, as there is with the spread of points in the projected data. However,

```

library(tourr)
p_ell <- NULL
for (i in unique(penguins_sub$species)) {
  x <- penguins_sub %>% dplyr::filter(species == i)
  e <- gen_xvar_ellipse(x[,1:4], n=150, nstd=1.5)
  e$species <- i
  p_ell <- bind_rows(p_ell, e)
}
p_ell$species <- factor(p_ell$species)
load("data/penguins_tour_path.rda")
animate_xy(p_ell[,1:4], col=factor(p_ell$species))
render_gif(penguins_sub[,1:4],
           planned_tour(pt1),
           display_xy(half_range=0.9, axes="off", col=penguins_sub$species),
           gif_file="gifs/penguins_lda1.gif",
           frames=500,

```

```

        loop=FALSE)
render_gif(p_ell[,1:4],
          planned_tour(pt1),
          display_xy(half_range=0.9, axes="off", col=p_ell$species),
          gif_file="gifs/penguins_lda2.gif",
          frames=500,
          loop=FALSE)

```

Figure 13.4: Checking the assumption of equal variance-covariance matrices for the 4D penguins data.

As a further check, we could generate three ellipses corresponding to the pooled variance-covariance matrix, as would be used in the model, centered at each of the means. Overlay this with the data. Now you will compare the spread of the observations in the data, with the elliptical shape of the pooled variance-covariance. If it matches reasonably we can safely use LDA. This can also be done group by group when multiple groups make it difficult to view all together.

```

library(tourr)
# Create an ellipse corresponding to pooled vc
pool_ell <- gen_vc_ellipse(p_vc_pool, xm=rep(0, ncol(p_vc_pool)))

# Add means to produce ellipses for each species
p_lda_pool <- data.frame(rbind(pool_ell +
                                matrix(rep(p_lda$means[1,],
                                             each=nrow(pool_ell)),
                                         ncol=4),
                                pool_ell +
                                matrix(rep(p_lda$means[2,],
                                             each=nrow(pool_ell)),
                                         ncol=4),
                                pool_ell +
                                matrix(rep(p_lda$means[3,],
                                             each=nrow(pool_ell)),
                                         ncol=4)))
# Create one data set with means, data, ellipses
p_lda_pool$species <- factor(rep(levels(penguins_sub$species),
                                rep(nrow(pool_ell), 3)))
p_lda_pool$type <- "ellipse"
p_lda_means <- data.frame(p_lda$means,
                          species=factor(rownames(p_lda$means)),
                          type="mean")
p_data <- data.frame(penguins_sub[,1:5], type="data")

```

```

p_lda_all <- bind_rows(p_lda_means,
                      p_data,
                      p_lda_pool)
p_lda_all$type <- factor(p_lda_all$type,
                        levels=c("mean", "data", "ellipse"))
shapes <- c(3, 4, 20)
p_pch <- shapes[p_lda_all$type]

# Code to run the tour
animate_xy(p_lda_all[,1:4], col=p_lda_all$species, pch=p_pch)
load("data/penguins_tour_path.rda")
render_gif(p_lda_all[,1:4],
           planned_tour(pt1),
           display_xy(col=p_lda_all$species, pch=p_pch,
                     axes="off", half_range = 0.7),
           gif_file="gifs/penguins_lda_pooled1.gif",
           frames=500,
           loop=FALSE)

# Focus on one species
render_gif(p_lda_all[p_lda_all$species == "Gentoo",1:4],
           planned_tour(pt1),
           display_xy(col="#F5191C",
                     pch=p_pch[p_lda_all$species == "Gentoo"],
                     axes="off", half_range = 0.7),
           gif_file="gifs/penguins_lda_pooled2.gif",
           frames=500,
           loop=FALSE)

```

Figure 13.5: Tour of penguins data, pooled variance-covariance ellipse overlaid. We can see that the pooled variance-covariance is a reasonable estimate of the spread of the data.

13.3 Examining results

The boundaries for a classification model can be examined by:

1. generating a large number of observations in the domain of the data
2. predicting the class for each

We'll look at this for 2D using the LDA model fitted to `b1`, and `bd` of the penguins data.

```
library(classifly)

load("data/penguins_sub.rda")
p_bl_bd_lda <- lda(species~bl+bd, data=penguins_sub,
                  prior = c(1/3, 1/3, 1/3))
```

The fitted model parameters are the means: $\bar{x}_{Adelie} = (-0.95, 0.6)^\top$, $\bar{x}_{Chinstrap} = (0.89, 0.64)^\top$, and $\bar{x}_{Gentoo} = (0.65, -1.1)^\top$.

The boundaries can be examined using the `explore()` function from the `classifly` package, which generates observations in the range of all values of `bl` and `bd` and predicts their class. We can overlay the sample means and an ellipse corresponding to the variance-covariance also, by extracting these from the model object.

```
p_bl_bd_lda_boundaries <- explore(p_bl_bd_lda, penguins_sub)
p_bl_bd_lda_m1 <- ggplot(p_bl_bd_lda_boundaries) +
  geom_point(aes(x=bl, y=bd, colour=species, shape=.TYPE)) +
  scale_color_discrete_divergingx("Zissou 1") +
  scale_shape_manual(values=c(46, 16)) +
  theme_minimal() +
  theme(aspect.ratio = 1, legend.position = "none")

p_bl_bd_lda_means <- data.frame(p_bl_bd_lda$means, species=rownames(p_bl_bd_lda$means))
```

```
p_lda <- lda(species ~ ., penguins_sub[,1:5], prior = c(1/3, 1/3, 1/3))
p_lda_boundaries <- explore(p_lda, penguins_sub)
```

```
# Code to run the tour
p_lda_boundaries$species
animate_slice(p_lda_boundaries[p_lda_boundaries$.TYPE == "simulated",1:4], col=p_lda_boundaries$species,
render_gif(p_lda_boundaries[p_lda_boundaries$.TYPE == "simulated",1:4],
  planned_tour(pt1),
  display_slice(v_rel=0.02,
    col=p_lda_boundaries$species[p_lda_boundaries$.TYPE == "simulated"],
    axes="bottomleft"), gif_file="gifs/penguins_lda_boundaries.gif",
  frames=500,
  loop=FALSE
)
```

```
# Project the boundaries into the 2D discriminant space
p_lda_b_sub <- p_lda_boundaries[
```

```

p_lda_boundaries$.TYPE == "simulated",
c(1:4, 6)]
p_lda_b_sub_ds <- data.frame(as.matrix(p_lda_b_sub[,1:4]) %*%
  p_lda$scaling)
p_lda_b_sub_ds$species <- p_lda_b_sub$species
p_lda_b_sub_ds_p <- ggplot(p_lda_b_sub_ds,
  aes(x=LD1, y=LD2,
      colour=species)) +
  geom_point(alpha=0.2) +
  scale_color_discrete_divergingx("Zissou 1") +
  theme_minimal() +
  theme(aspect.ratio = 1,
        legend.position = "bottom",
        legend.title = element_blank())

```

Figure 13.7: Examining the boundaries produced by the LDA model in the full 4D with a slice tour and in the discriminant space.

The LDA boundaries divide the classes in the discriminant space, and the remaining directions are irrelevant.

Exercises

1. For the `simple_clusters` compute the lda model, and make a plot of the data, with points coloured by the class. Overlay variance-covariance ellipses, and a + indicating the sample mean for each class. Is it reasonable to assume that the two classes are sampled from populations with the same variance-covariance?
2. Examine the clusters corresponding to the classes in the `clusters` data set, using a tour. Based on the shape of the data is the assumption of equal variance-covariance reasonable?
3. Examine the pooled variance-covariance for the `clusters` data, overlaid on the data in a tour on the 5D. Does it fit the variance of each cluster nicely?
4. Fit an LDA model to the `simple_clusters` data. Examine the boundaries produced by the model, in 2D.
5. Fit an LDA model to the `clusters` data. Examine the boundaries produced by the model in 5D.
6. Assess the LDA assumptions for the `multicluster` data. Is LDA an appropriate model?

Chapter 14

Trees and forests

14.1 Trees

The tree algorithm Breiman et al. (1984) is a simple and versatile algorithmic method for supervised classification. The basic tree algorithm generates a classification rule by sequentially splitting the data into two buckets. Splits are made between sorted data values of individual variables, with the goal of obtaining pure classes on each side of the split. The inputs for a simple tree classifier commonly include (1) an impurity measure, an indication of the relative diversity among the cases in the terminal nodes; (2) a parameter that sets the minimum number of cases in a node, or the minimum number of observations in a terminal node of the tree; and (3) a complexity measure that controls the growth of a tree, balancing the use of a simple generalizable tree against a more accurate tree tailored to the sample. When applying tree methods, exploring the effects of the input parameters on the tree is instructive; for example, it helps us to assess the stability of the tree model.

Although algorithmic models do not depend on distributional assumptions, that does not mean that every algorithm is suitable for all data. For example, the tree model works best when all variables are independent within each class, because it does not take such dependencies into account. Visualization can help us to determine whether a particular model should be applied. In classification problems, it is useful to explore the cluster structure, comparing the clusters with the classes and looking for evidence of correlation within each class. The plots in Figure ?? and Figure ?? shows a strong correlation between the variables within each species, which indicates that the tree model may not give good results for the penguins data. We'll show how this is the case with two variables initially, and then extend to the four variables.

The plots in Figure ?? show the inadequacies of the tree fit. The background color indicates the class predictions, and thus boundaries produced by the tree

```

library(mulgar)

library(rpart)

library(rpart.plot)

library(colorspace)

library(classify)

library(ggplot2)

load("data/penguins_sub.rda")

p_bl_bd_tree <- rpart(species~bl+bd, data=penguins_sub)

rpart.plot(p_bl_bd_tree, box.palette="Grays")

p_bl_bd_tree_boundaries <- explore(p_bl_bd_tree, pe

ggplot(p_bl_bd_tree_boundaries) +

  geom_point(aes(x=bl, y=bd, colour=species, shape=

  scale_color_discrete_divergingx(palette="Zissou 1

  scale_shape_manual(values=c(46, 16)) +

  theme_minimal() +

  theme(aspect.ratio = 1, legend.position = "none")

./forests_files/figure-pdf/fig-p-bl-bd-tree1-1.pdf

```

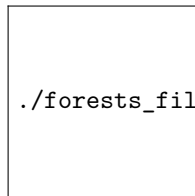


Figure 14.1: Default tree fit

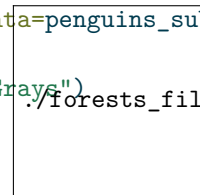


Figure 14.2: Boundaries of tree fit

./forests_files/figure-pdf/fig-p-bl-bd-tree1-1.pdf

Figure 14.3: The correlation between variables causes problems for using a tree model on the penguins data.

fit. They can be seen to be boxy, and missing the elliptical nature of the penguin clusters. This produces errors in the classification of observations which are indefensible. One could always force the tree to fit the data more closely by adjusting the parameters, but the main problem persists: that one is trying to fit elliptical data using boxes.

The boundaries for the tree model on all four variables of the penguins data can be viewed similarly using the tour. The default fitted tree is delightfully simple, with just six splits of the data.

```
p_tree <- rpart(species~., data=penguins_sub[,1:5])
rpart.plot(p_tree, box.palette="Grays")

p_tree_boundaries <- explore(p_tree, penguins_sub)
animate_slice(p_tree_boundaries[p_tree_boundaries$.TYPE == "simulated",1:4], col=p_tree_boundaries$.TYPE)
load("data/penguins_tour_path.rda")
render_gif(p_tree_boundaries[p_tree_boundaries$.TYPE == "simulated",1:4],
           planned_tour(pt1),
           display_slice(v_rel=0.02,
                        col=p_tree_boundaries[p_tree_boundaries$.TYPE == "simulated",6],
                        axes="bottomleft"),
           gif_file="gifs/penguins_tree_boundaries.gif",
           frames=500,
           loop=FALSE
           )
```

Figure 14.4: Comparison of the boundaries produced by the LDA model and the tree models.

14.2 Random forests

A random forest Breiman & Cutler (2004) is a classifier that is built from multiple trees generated by randomly sampling the cases and the variables. The random sampling (with replacement) of cases has the fortunate effect of creating a training (“in-bag”) and a test (“out-of-bag”) sample for each tree computed. The class of each case in the out-of-bag sample for each tree is predicted, and the predictions for all trees are combined into a vote for the class identity.

A random forest is a computationally intensive method, a “black box” classifier, but it produces several diagnostics that make the outcome less mysterious. Some diagnostics that help us to assess the model are the votes, the measure of variable importance, and the proximity matrix.

14.2.1 Examining the votes matrix

Here we show how to use the `randomForest` (Liaw & Wiener, 2002) votes matrix for the penguins data to investigate confusion between classes, and observations which are problematic to classify. With only three classes the votes matrix is only a 2D object, and thus easy to examine. With four or more classes the votes matrix needs to be examined in a tour.

```
library(randomForest)
library(dplyr)
penguins_rf <- randomForest(species~.,
                             data=penguins_sub[,1:5],
                             importance=TRUE)

penguins_rf
```

Call:

```
randomForest(formula = species ~ ., data = penguins_sub[, 1:5],
              Type of random forest: classification
              Number of trees: 500
              No. of variables tried at each split: 2
```

importance = TRUE)

OOB estimate of error rate: 2.4%

Confusion matrix:

	Adelie	Chinstrap	Gentoo	class.error
Adelie	143	2	1	0.020547945
Chinstrap	4	64	0	0.058823529
Gentoo	0	1	118	0.008403361

To examine the votes matrix, we extract the `votes` element from the random forest model object. This will have three columns corresponding to the three species, but because each row is a set of proportions it is only a 2D object, as seen in [?@fig-p-votes-tour](#). To reduce the dimension from 3D to the 2D we use a Helmert matrix (Lancaster, 1965). Helmert matrices are orthogonal matrices, where the first row is all 1's, and then subsequent rows sequentially replace one element with a 0. The rows are usually normalised to have length 1. They are used to create contrasts to test combinations of factor levels for post-testing after Analysis of Variance (ANOVA). For compositional data, like the votes matrix, when the first row is removed a Helmert matrix can be used to reduce the dimension appropriately. For three classes, this will generate the common 2D ternary diagram, but for higher dimensions it will reduce to a $(g - 1)$ -dimensional simplex. For the penguins data, the Helmert matrix for 3D is

```
geozoo::f_helmert(3)
```

	[,1]	[,2]	[,3]
helmert	0.5773503	0.5773503	0.5773503

```
x      0.7071068 -0.7071068  0.0000000
x      0.4082483  0.4082483 -0.8164966
```

We drop the first row, transpose it, and use matrix multiplication with the votes matrix to get the ternary diagram.

```
# Project 4D into 3D
library(geozoo)
proj <- t(geozoo::f_helmert(3)[-1,])
p_rf_v_p <- as.matrix(penguins_rf$votes) %*% proj
colnames(p_rf_v_p) <- c("x1", "x2")
p_rf_v_p <- p_rf_v_p %>%
  as.data.frame() %>%
  mutate(species = penguins_sub$species)
```

We can use the `geozoo` package to generate the surrounding simplex, which is a triangle for 2D.

```
# Add simplex
simp <- simplex(p=2)
sp <- data.frame(cbind(simp$points), simp$points[c(2,3,1),])
colnames(sp) <- c("x1", "x2", "x3", "x4")
sp$species = sort(unique(penguins_sub$species))
library(ggthemes)
p_ternary <- ggplot() +
  geom_segment(data=sp, aes(x=x1, y=x2, xend=x3, yend=x4)) +
  geom_text(data=sp, aes(x=x1, y=x2, label=species),
            nudge_x=c(-0.06, 0.07, 0),
            nudge_y=c(0.05, 0.05, -0.05)) +
  geom_point(data=p_rf_v_p, aes(x=x1, y=x2, colour=species), size=2, alpha=0.5) +
  scale_color_discrete_divergingx(palette="Zissou 1") +
  theme_map() +
  theme(aspect.ratio=1, legend.position="none")

# Look at the votes matrix, in its 3D space
animate_xy(penguins_rf$votes, col=penguins_sub$species)

# Save an animated gif
render_gif(penguins_rf$votes,
  grand_tour(),
  display_xy(v_rel=0.02,
    col=penguins_sub$species,
    axes="bottomleft"),
  gif_file="gifs/penguins_rf_votes.gif",
```



```
frames=500,
loop=FALSE
)
```

Figure 14.6: Examining the votes matrix from a random forest fit to the penguins.

The votes matrix, reports the proportion of trees each observation is classified as each class. From the tour of the votes matrix, it can be seen to be 2D in 3D space. This is due to the constraint that the three proportions for each observation sum to 1. Using a Helmert matrix, this data can be projected into the 2D space, or more generally the $(g - 1)$ -dimensional space where it resides. In 2D this is called a ternary diagram, and in higher dimensions the bounding shapes might be considered to be a simplex. The vertices of this shape correspond to $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$ (and analogously for higher dimensions), which represent perfect confidence, that an observation is classified into that group all the time.

What we can see here is a concentration of points in the corners of the triangle indicates that most of the penguins are confidently classified into their correct class. Then there is more separation between the Gentoo and the others, than between Chinstrap and Adelie. That means that as a group Gentoo are more distinguishable. Only one of the Gentoo penguins has substantial confusion, mostly confused as a Chinstrap, but occasionally confused as an Adelie – if it was only ever confused as a Chinstrap it would fall on the edge between Gentoo and Chinstrap. There are quite a few Chinstrap and Adelie penguins confused as each other, with a couple of each more confidently predicted to be the other class. This can be seen because there are points of the wrong colour close to those vertices.

The votes matrix is useful for investigating the fit, but one should remember that there are some structural elements of this data that don't lend themselves to tree models. Although a forest has the capacity to generate non-linear boundaries by combining predictions from multiple trees, it is still based on the boxy boundaries of trees. This makes it less suitable for the penguins data with elliptical classes. You could use the techniques from the previous section to explore the boundaries produced by the forest, and you will find that they are more boxy than the LDA models.

To examine a vote matrix for a problem with more classes, we will examine the 10 class `fake_trees` data example. The full data has 100 variables, and we have seen from Chapter ?? that reducing to 10 principal components allows the linear branching structure in the data to be seen. Given that the branches correspond to the classes, it will be interesting to see how well the random forest model performs.

```

library(mulgar)
library(dplyr)
library(liminal)
ft_pca <- prcomp(fake_trees[,1:100],
                 scale=TRUE, retx=TRUE)
ft_pc <- as.data.frame(ft_pca$x[,1:10])
ft_pc$branches <- fake_trees$branches
library(randomForest)
ft_rf <- randomForest(branches~., data=ft_pc,
                     importance=TRUE)

ft_rf

```

Call:

```

randomForest(formula = branches ~ ., data = ft_pc, importance = TRUE)
      Type of random forest: classification
      Number of trees: 500

```

No. of variables tried at each split: 3

OOB estimate of error rate: 4.2%

Confusion matrix:

	0	1	2	3	4	5	6	7	8	9	class.error
0	265	5	3	3	3	3	6	3	6	3	0.11666667
1	13	287	0	0	0	0	0	0	0	0	0.04333333
2	8	0	289	0	3	0	0	0	0	0	0.03666667
3	5	0	0	290	0	0	0	3	0	2	0.03333333
4	12	0	0	0	288	0	0	0	0	0	0.04000000
5	11	0	0	0	0	289	0	0	0	0	0.03666667
6	9	0	0	0	0	0	290	0	1	0	0.03333333
7	5	0	0	5	0	0	0	290	0	0	0.03333333
8	8	0	0	0	0	0	0	0	292	0	0.02666667
9	6	0	0	0	0	0	0	0	0	294	0.02000000

```

ft_rf_votes <- ft_rf$votes %>%
  as_tibble() %>%
  mutate(branches = fake_trees$branches)

proj <- t(geozoo::f_helmert(10)[-1,])
f_rf_v_p <- as.matrix(ft_rf_votes[,1:10]) %*% proj
colnames(f_rf_v_p) <- c("x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9")
f_rf_v_p <- f_rf_v_p %>%
  as.data.frame() %>%
  mutate(branches = fake_trees$branches)

simp <- geozoo::simplex(p=9)

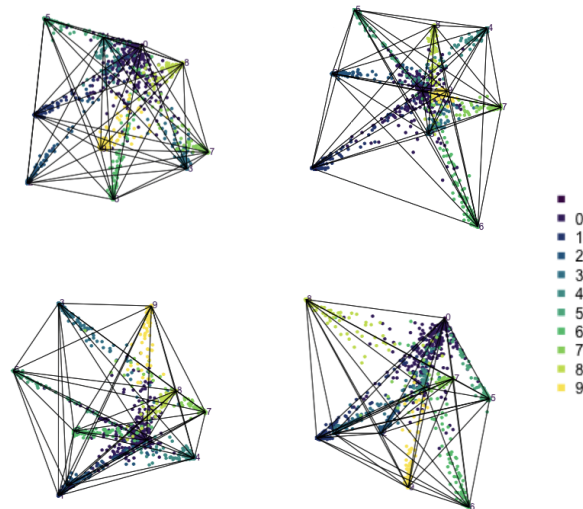
```

```

sp <- data.frame(simp$points)
colnames(sp) <- c("x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9")
sp$branches = ""
f_rf_v_p_s <- bind_rows(sp, f_rf_v_p) %>%
  mutate(branches = factor(branches))
labels <- c("0" , "1", "2", "3", "4", "5", "6", "7", "8", "9",
           rep("", 3000))
animate_xy(f_rf_v_p_s[,1:9], col = f_rf_v_p_s$branches,
           axes = "off", half_range = 0.8,
           edges = as.matrix(simp$edges),
           obs_labels = labels, palette = "Viridis")

render_gif(f_rf_v_p_s[,1:9],
           grand_tour(),
           display_xy(col = f_rf_v_p_s$branches,
                     axes = "off", half_range = 0.8,
                     edges = as.matrix(simp$edges),
                     obs_labels = labels, palette="Viridis"),
           gif_file="gifs/ft_votes.gif",
           frames=500)

```



(a) Several static views from the tour revealing how clusters connect.

Figure 14.7: The votes matrix for the `fake_trees` data has a very striking geometric shape. The branching nature of the clusters is very clear. Most classes are distinct except for a connection with class 0.

The votes matrix is 9D, but the structure of it is easy to read, and very interesting. The observations are coloured by class. There is one vertex (0) which has connections to all other vertexes. That is, there are points stretching without big breaks from this vertex to every other. It means that some observations in every other class can be confused with class 0, and class 0 observations can be confused with every other class. All of the other vertexes have a string of points almost entirely along one edge, the edge leading to vertex 0. This shows the lack of confusion with any other class, except 0. Cluster 0 could be considered the trunk of the tree, from which the other clusters grow.

This pattern is what can be inferred from the confusion matrix, but we can't determine whether the clusters are mostly separated except for a few observations, or some other clustering shape. The visual pattern in the votes matrix is so striking, and gives additional information about the clustering distribution, and shapes of clusters. It reinforces the clusters are linear extending into different dimensions in the 100D space, but really only into about 8D (as we'll see from the variable importance explanation below). We also see that 9 of the clusters are all connected to one cluster.

14.2.2 Using variable importance

The variable importance score across all classes, and for each class is useful for choosing variables to enter into a tour, to explore class differences. This is particularly so when there are many variables, as in the `fake_trees` data. We would also expect that this data will have a difference between importance for some classes.

```
library(gt)
ft_rf$importance %>%
  as_tibble(rownames="Variable") %>%
  rename(Accuracy=MeanDecreaseAccuracy,
         Gini=MeanDecreaseGini) %>%
  #arrange(desc(Gini)) %>%
  gt() %>%
  fmt_number(columns = c(`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, Accuracy),
             decimals = 2) %>%
  fmt_number(columns = Gini,
             decimals = 0)
```

Table 14.1: Variable importance from the random forest fit to the `fake_trees` data.

Variable	0	1	2	3	4	5	6	7	8	9	Accuracy	Gini
PC1	0.10	0.36	0.44	0.26	0.19	0.46	0.40	0.17	0.30	0.30	0.30	472
PC2	0.13	0.24	0.22	0.52	0.31	0.31	0.18	0.40	0.24	0.29	0.28	379

PC3	0.09	0.05	0.11	0.14	0.54	0.15	0.09	0.14	0.19	0.17	0.17	318
PC4	0.09	0.45	0.06	0.04	0.10	0.04	0.36	0.14	0.09	0.09	0.14	344
PC5	0.14	0.12	0.36	0.07	0.16	0.24	0.10	0.11	0.29	0.22	0.18	333
PC6	0.10	0.25	0.24	0.16	0.04	0.12	0.04	0.26	0.14	0.16	0.15	276
PC7	0.07	0.03	0.15	0.05	0.04	0.07	0.11	0.36	0.12	0.15	0.11	261
PC8	0.05	0.07	0.01	0.25	0.06	0.08	0.02	0.04	0.08	0.27	0.09	217
PC9	0.07	0.01	0.01	0.01	0.01	0.01	0.01	0.04	0.05	0.02	0.02	56
PC10	0.03	0.01	0.01	0.00	0.01	0.02	0.01	0.01	0.01	0.02	0.01	44

From the variable importance, we can see that PC9 and PC10 do not substantially contribute. That means the 100D data can be reduced to 8 PCs while maintaining the information about the clustering. PC1 is most important overall, but each cluster has a different set of variables that are important. For example, the variables important for distinguishing cluster 1 are PC1, PC2, PC4 and PC6, and for cluster 7 they are PC2, PC6, PC7. We can use this information to choose variables to provide to the tour. It can be helpful to reduce the class variable to focus on a particular class, by creating a new class variable, as follows.

```
ft_pc <- ft_pc %>%
  mutate(cl1 = factor(case_when(
    branches == "0" ~ "0",
    branches == "1" ~ "1",
    .default = "other"
  )))

animate_xy(ft_pc[,c("PC1", "PC2", "PC4", "PC6")], col=ft_pc$cl1, palette="Viridis")
render_gif(ft_pc[,c("PC1", "PC2", "PC4", "PC6")],
  grand_tour(),
  display_xy(col=ft_pc$cl1, palette="Viridis"),
  gif_file="gifs/ft_cl1.gif",
  frames=500)

ft_pc_cl1 <- ggplot(ft_pc, aes(x=PC4, y=PC2, col=cl1)) +
  geom_point(alpha=0.7, size=1) +
  scale_color_discrete_sequential(palette="Viridis", rev=FALSE) +
  theme_minimal() +
  theme(aspect.ratio = 1)
```

From Figure ?? we can see how cluster 1 is distinct from all of the other observations, albeit with a close connection to the trunk of the tree (cluster 0). The distinction is visible in PC1, PC2, PC4, PC6, but can be seen clearly with just two of these.

Figure 14.9: Focusing on class 1 in the `fake_trees` data. The most important variables were PC1, PC2, PC4, PC6. A combination of PC2 and PC4 reveals the difference between cluster 1 and all the other clusters.

For a problem with this many classes it can be useful to focus on several groups together. We've chosen cluster 8, because it appears to have less of a connection with cluster 0. See the light green cluster in bottom right static plot of Figure ?? is connected more to a different vertex, which is cluster 6 when carefully viewed. This is also suggested by the confusion matrix, where there is one observation from cluster 8 confused with cluster 6, although somewhat contradictory information, most are confused with cluster 0. We have also added cluster 1 to the investigation because it is closely connected to 6 and 8.

```
ft_pc <- ft_pc %>%
  mutate(cl8 = factor(case_when(
    branches == "0" ~ "0",
    branches == "6" ~ "6",
    branches == "1" ~ "1",
    branches == "8" ~ "8",
    .default = "other"
  )))

animate_xy(ft_pc[,c("PC1", "PC2", "PC4", "PC5", "PC6")], col=ft_pc$cl8, palette="Viridis")
render_gif(ft_pc[,c("PC1", "PC2", "PC4", "PC5", "PC6")],
  grand_tour(),
  display_xy(col=ft_pc$cl8, palette="Viridis"),
  gif_file="gifs/ft_cl8.gif",
  frames=500)

ft_pc_cl8 <- ggplot(ft_pc, aes(x=PC1, y=PC5, col=cl8)) +
  geom_point(alpha=0.7, size=1) +
  scale_color_discrete_sequential(palette="Viridis", rev=FALSE) +
  theme_minimal() +
  theme(aspect.ratio = 1)
```

From Figure ?? we can see that clusters 1, 6, and 8 share one end of the trunk (cluster 0). Cluster 8 is almost more closely connected with cluster 6, though, than cluster 0. PC1 and PC5 mostly show the distinction between cluster 8 and the rest of the points, but it is clearer if more variables are used.

Figure 14.11: Focusing on class 8 in the `fake_trees` data, relative to nearby clusters 1 and 6. The most important variables for cluster 8 are PC1, PC2, PC5, but to explore in association with clusters 1 and 6, we include PC4 and PC6. A combination of PC1 and PC5 reveals the difference between cluster 8, 6, 1 and 0.

Exercises

1. Using a grand tour compare the boundaries from the random forest model on the penguins data to that of (a) a default tree model, (b) an LDA model. Is it less boxy than the tree model, but still more boxy than that of the LDA model?
2. Tinker with the parameters of the tree model to force it to fit a tree more closely to the data. Compare the boundaries from this with the default tree, and with the forest model. Is it less boxy than the default tree, but more boxy than the forest model?
3. Fit a random forest model to the `bushfires` data using the `cause` variable as the class. It is a highly imbalanced classification problem. What is the out-of-bag error rate for the forest? Are there some classes that have lower error rate than others? Examine the 4D votes matrix with a tour, and describe the confusion between classes. This is interesting because it is difficult to accurately classify the fire ignition cause, and only some groups are often confused with each other. You should be able to see this from the 3D votes matrix.
4. Explore the 5D votes matrix for a random forest on the sketches data. Why does it look star-shaped?
5. Choose a cluster (or group of clusters) from the `fake_trees` data (2, 3, 4, 5, 7, 9) to explore in detail like done in Section ???. Be sure to choose which PCs are the most useful using a tour, and follow-up by making a scatterplot showing the best distinction between your chosen cluster and the other observations.

Chapter 15

Support vector machines

A support vector machine (SVM) (Vapnik, 1999) looks for gaps between clusters in the data, based on the extreme observations in each class. In this sense it mirrors the graphical approach described Chapter ??, in which we searched for gaps between groups. It can be viewed as similar to LDA, in that the boundary between classes is a hyperplane. The difference between LDA and SVM is the placement of the boundary. LDA uses the means and covariance matrices of the classes to place the boundary, but SVM uses extreme observations.

The key elements of the SVM model to examine are:

- support vectors
- separating hyperplane.

15.1 Components of the SVM model

To illustrate the approach, we use two simple simulated data examples. Both have only two variables, and two classes. Explaining SVM is easier when there are just two groups. In the first data set the two classes have different covariances matrices, which will cause trouble for LDA, but SVM should see the gap between the two clusters and place the separating hyperplane in the middle of the gap. In the second data set the two groups are concentric circles, with the inner one solid. A non-linear SVM should be fitted to this data, which should see circular gap between the two classes.

Note that the `svm` function in the `e1071` package will automatically scale observations into the range $[0, 1]$. To make it easier to examine the fitted model, it is best to scale your data first, and then fit the model.

```

# Toy examples
library(mulgar)
library(ggplot2)
library(geozoo)
library(tourr)

set.seed(1071)
n1 <- 162
vc1 <- matrix(c(1, -0.7, -0.7, 1), ncol=2, byrow=TRUE)
c1 <- rmvn(n=n1, p=2, mn=c(-2, -2), vc=vc1)
vc2 <- matrix(c(1, -0.4, -0.4, 1)*2, ncol=2, byrow=TRUE)
n2 <- 138
c2 <- rmvn(n=n2, p=2, mn=c(2, 2), vc=vc2)
df1 <- data.frame(x1=mulgar::scale2(c(c1[,1], c2[,1])),
                  x2=mulgar::scale2(c(c1[,2], c2[,2])),
                  cl = factor(c(rep("A", n1),
                                rep("B", n2))))

c1 <- sphere.hollow(p=2, n=n1)$points*3 +
  c(rnorm(n1, sd=0.3), rnorm(n1, sd=0.3))
c2 <- sphere.solid.random(p=2, n=n2)$points
df2 <- data.frame(x1=mulgar::scale2(c(c1[,1], c2[,1])),
                  x2=mulgar::scale2(c(c1[,2], c2[,2])),
                  cl = factor(c(rep("A", n1),
                                rep("B", n2))))

library(classify)
library(e1071)
df1_svm <- svm(cl~., data=df1,
               probability=TRUE,
               kernel="linear",
               scale=FALSE)
df1_svm_e <- explore(df1_svm, df1)

df2_svm <- svm(cl~., data=df2,
               probability=TRUE,
               kernel="radial")
df2_svm_e <- explore(df2_svm, df2)

library(patchwork)
library(colorspace)
s1 <- ggplot() +
  geom_point(data=df1, aes(x=x1, y=x2, colour=cl),
             shape=20) +

```

```

scale_colour_discrete_divergingx(palette="Zissou 1") +
geom_point(data=df1_svm_e[(!df1_svm_e$.BOUNDARY)&(df1_svm_e$.TYPE=="simulated"),],
  aes(x=x1, y=x2, colour=c1), shape=3) +
geom_point(data=df1[df1_svm$index,],
  aes(x=x1, y=x2, colour=c1),
  shape=4, size=4) +
theme_minimal() +
theme(aspect.ratio=1, legend.position = "none") +
ggtitle("(a)")

s2 <- ggplot() +
  geom_point(data=df2, aes(x=x1, y=x2, colour=c1), shape=20) +
  scale_colour_discrete_divergingx(palette="Zissou 1") +
  geom_point(data=df2_svm_e[(!df2_svm_e$.BOUNDARY)&(df2_svm_e$.TYPE=="simulated"),],
    aes(x=x1, y=x2, colour=c1),
    shape=3) +
  geom_point(data=df2[df2_svm$index,],
    aes(x=x1, y=x2, colour=c1),
    shape=4, size=4) +
  theme_minimal() +
  theme(aspect.ratio=1, legend.position = "none") +
  ggtitle("(b)")

s1+s2

```

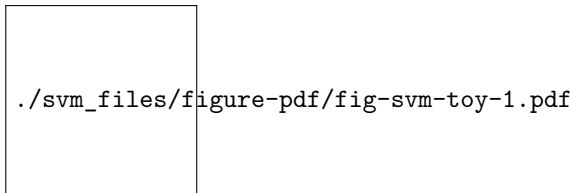


Figure 15.1: SVM classifier fit overlaid on two simulated data examples: (a) groups with different variance-covariance, fitted using a linear kernel, (b) groups with non-linear separation, fitted using a radial kernel. The band of points shown as '+' mark the SVM boundary, and points marked by 'x' are the support vectors used to define the boundary.

Figure ?? shows the two data sets and the important aspects of the fitted SVM model for each. The observations are represented by dots, the separating hyperplane (just a line for 2D) is represented by '+'. Where the two colours merge is the actual location of the boundary between classes. It can be seen that this is located right down the middle of the gap, for both data sets. Even though the boundary is circular for the second data set, in a transformed high-dimensional

space it would be linear.

SVMs use a subset of the observations to define the boundary, and these are called the support vectors. For each of the data sets these are marked with 'x'. For the linear boundary, there are nine support vectors, five in one group and four in the other. There is one interesting observation in the red group, which falls on the other side of the boundary. It is marked as a support vector, but its contribution to the fitted hyperplane is limited by a control parameter in the model fitting process.

Linear SVMs can be assessed similarly to regression models. The components of the model are:

1. The points that are the support vectors:

```
df1_svm$index
```

```
[1] 15 45 123 135 155 180 202 239 292
```

2. Their coefficients:

```
df1_svm$coefs
```

```

          [,1]
[1,]  0.3771240
[2,]  0.1487726
[3,]  1.0000000
[4,]  1.0000000
[5,]  1.0000000
[6,] -0.5258966
[7,] -1.0000000
[8,] -1.0000000
[9,] -1.0000000

```

which indicate that all but 15, 45 and 180 are actually bounded support vectors (their coefficients are bounded to magnitude 1).

3. that when used with the intercept:

```
df1_svm$rho
```

```
[1] 0.3520001
```

can be used to compute the equation of the fitted hyperplane.

```

w = t(df1_svm$SV) %*% df1_svm$coefs
w

```

```

      [,1]
x1 -1.501086
x2 -1.356237

```

Giving the equation to be $-1.5 x_1 + -1.36 x_2 + -0.35 = 0$, or alternatively, $x_2 = -1.11 x_1 + -0.26$.

which can be used to generate a line to show the boundary with the data.

```

s1 + geom_abline(intercept=df1_svm$rho/w[2],
                  slope=-w[1]/w[2])

```

Note that care in scaling of data is important to get the intercept calculated exactly. We have standardised the data, and set the `scale=FALSE` parameter in the `svm` function. The slope calculation is quite robust to the data scaling.

15.2 Examining the model components in high-dimensions

For higher dimensions, the procedures are similar, with the hyperplane and support vectors being examined using a tour. Here we examine the model for differentiating male and female Chinstrap penguins. The Chinstrap penguins have a noticeable difference in size of the sexes, unlike the other two species. Working with a two-class problem is easier for explaining SVM, but multi-class calculations can also follow this approach.

```

library(dplyr)
load("data/penguins_sub.rda")
chinstrap <- penguins_sub %>%
  filter(species == "Chinstrap") %>%
  select(-species) %>%
  mutate_if(is.numeric, mulgar::scale2)
chinstrap_svm <- svm(sex~., data=chinstrap,
                    kernel="linear",
                    probability=TRUE,
                    scale=FALSE)
chinstrap_svm_e <- explore(chinstrap_svm, chinstrap)

# Tour raw data
animate_xy(chinstrap[,1:4], col=chinstrap$sex)
# Add all SVs, including bounded
c_pch <- rep(20, nrow(chinstrap))
c_pch[chinstrap_svm$index] <- 4

```

```

animate_xy(chinstrap[,1:4], col=chinstrap$sex, pch=c_pch)
# Only show the SVs with |coefs| < 1
c_pch <- rep(20, nrow(chinstrap))
c_pch[chinstrap_svm$index[abs(chinstrap_svm$coefs)<1]] <- 4
c_cex <- rep(1, nrow(chinstrap))
c_cex[chinstrap_svm$index[abs(chinstrap_svm$coefs)<1]] <- 2
animate_xy(chinstrap[,1:4], col=chinstrap$sex,
           pch=c_pch, cex=c_cex)
render_gif(chinstrap[,1:4],
           grand_tour(),
           display_xy(col=chinstrap$sex, pch=c_pch, cex=c_cex),
           gif_file="gifs/chinstrap_svs.gif",
           width=400,
           height=400,
           frames=500)

# Tour the separating hyperplane also
symbols <- c(3, 20)
c_pch <- symbols[as.numeric(chinstrap_svm_e$.TYPE[!chinstrap_svm_e$.BOUNDARY])]
animate_xy(chinstrap_svm_e[!chinstrap_svm_e$.BOUNDARY,1:4],
           col=chinstrap_svm_e$sex[!chinstrap_svm_e$.BOUNDARY],
           pch=c_pch)
render_gif(chinstrap_svm_e[!chinstrap_svm_e$.BOUNDARY,1:4],
           grand_tour(),
           display_xy(col=chinstrap_svm_e$sex[!chinstrap_svm_e$.BOUNDARY], pch=c_pch),
           gif_file="gifs/chinstrap_svm.gif",
           width=400,
           height=400,
           frames=500)

```

Figure 15.2: SVM model for distinguishing the sexes of the Chinstrap penguins. The separating hyperplane is 3D, and separates primarily on variables `b1` and `bd`, as seen because these two axes extend out from the plane when it is seen on its side, separating the two groups.

We can check this interpretation using the radial tour. Using the components from the model, the coefficients of the hyperplane are:

```

t(chinstrap_svm$SV) %*% chinstrap_svm$coefs

      [,1]
b1 -0.9102439
bd -1.1073475

```



```
fl -0.5223364
bm -0.2846370
```

This supports the observation that **bl** and **bd** are most important, because they have the largest magnitudes. We can use this vector to set the starting point for radial tour. It needs to be normalised. A randomly generated second vector orthonormal to this one can be added to make a 2D projection from which to see the boundary.

```
set.seed(1022)
prj1 <- mulgar::norm_vec(t(chinstrap_svm$SV) %*% chinstrap_svm$coefs)
prj2 <- basis_random(4, 1)
prj <- orthonormalise(cbind(prj1, prj2))
prj

      [,1]      [,2]
bl -0.5865081 -0.06412875
bd -0.7135101  0.51192498
fl -0.3365631 -0.77713899
bm -0.1834035 -0.36038216

animate_xy(chinstrap_svm_e[!chinstrap_svm_e$.BOUNDARY,1:4],
  tour_path = radial_tour(start=prj, mvar = 2),
  col=chinstrap_svm_e$sex[!chinstrap_svm_e$.BOUNDARY],
  pch=c_pch)
render_gif(chinstrap_svm_e[!chinstrap_svm_e$.BOUNDARY,1:4],
  radial_tour(start=prj, mvar = 2),
  display_xy(col=chinstrap_svm_e$sex[!chinstrap_svm_e$.BOUNDARY], pch=c_pch),
  gif_file="gifs/chinstrap_rad_bd.gif",
  apf = 1/30,
  width=400,
  height=400,
  frames=500)
render_gif(chinstrap_svm_e[!chinstrap_svm_e$.BOUNDARY,1:4],
  radial_tour(start=prj, mvar = 1),
  display_xy(col=chinstrap_svm_e$sex[!chinstrap_svm_e$.BOUNDARY], pch=c_pch),
  gif_file="gifs/chinstrap_rad_bl.gif",
  apf = 1/30,
  width=400,
  height=400,
  frames=500)
render_gif(chinstrap_svm_e[!chinstrap_svm_e$.BOUNDARY,1:4],
  radial_tour(start=prj, mvar = 3),
  display_xy(col=chinstrap_svm_e$sex[!chinstrap_svm_e$.BOUNDARY], pch=c_pch),
  gif_file="gifs/chinstrap_rad_fl.gif",
```

```

    apf = 1/30,
    width=400,
    height=400,
    frames=500)
render_gif(chinstrap_svm_e[!chinstrap_svm_e$.BOUNDARY,1:4],
  radial_tour(start=prj, mvar = 4),
  display_xy(col=chinstrap_svm_e$sex[!chinstrap_svm_e$.BOUNDARY], pch=c_pch,
  gif_file="gifs/chinstrap_rad_bm.gif",
  apf = 1/30,
  width=400,
  height=400,
  frames=500)

```

Figure 15.3: Exploring the importance of the four variables to the separating hyperplane using a radial tour where the contribution of each variable is reduced to 0, and then increased to it's original value. You can see that **b1** and **bd** contribute most to the plane, because when they are removed the plane is no longer on it side marking the boundary. Variables **f1** and **bm** contribute a small amount to the separating hyperplane, but it is possible that these two could be removed without affecting the strength of the separation between the sexes.

Chapter 16

Neural networks and deep learning

Chapter 17

Exploring misclassifications

To examine misclassifications, we can create a separate variable that identifies the errors or not. Constructing this for each class, and exploring in small steps is helpful. Let's do this using the random forest model for the penguins fit. The random forest fit has only a few misclassifications. There are four Adelie penguins confused with Chinstrap, and similarly four Chinstrap confused with Adelie. There is one Gentoo penguin confused with a Chinstrap. This is interesting, because the Gentoo cluster is well separated from the clusters of the other two penguin species.

```
library(randomForest)
library(dplyr)
load("data/penguins_sub.rda")

penguins_rf <- randomForest(species~.,
                             data=penguins_sub[,1:5],
                             importance=TRUE)

penguins_rf
```

Call:

```
randomForest(formula = species ~ ., data = penguins_sub[, 1:5], importance = TRUE)
```

```
      Type of random forest: classification
```

```
      Number of trees: 500
```

```
No. of variables tried at each split: 2
```

```
      OOB estimate of  error rate: 2.4%
```

```
Confusion matrix:
```

```
      Adelie Chinstrap Gentoo class.error
```

Adelie	143	3	0	0.020547945
Chinstrap	4	64	0	0.058823529
Gentoo	0	1	118	0.008403361

```
penguins_errors <- penguins_sub %>%
  mutate(err = ifelse(penguins_rf$predicted != penguins_rf$y, 1, 0))
```

```
library(tourr)
symbols <- c(1, 16)
p_pch <- symbols[penguins_errors$err+1]
p_cex <- rep(1, length(p_pch))
p_cex[penguins_errors$err==1] <- 2
animate_xy(penguins_errors[,1:4],
           col=penguins_errors$species,
           pch=p_pch, cex=p_cex)
render_gif(penguins_errors[,1:4],
           grand_tour(),
           display_xy(col=penguins_errors$species,
                     pch=p_pch, cex=p_cex),
           gif_file="gifs/p_rf_errors.gif",
           frames=500,
           width=400,
           height=400)

animate_xy(penguins_errors[,1:4],
           guided_tour(lda_pp(penguins_errors$species)),
           col=penguins_errors$species,
           pch=pch)

render_gif(penguins_errors[,1:4],
           guided_tour(lda_pp(penguins_errors$species)),
           display_xy(col=penguins_errors$species,
                     pch=p_pch, cex=p_cex),
           gif_file="gifs/p_rf_errors_guided.gif",
           frames=500,
           width=400,
           height=400,
           loop=FALSE)
```

Figure ?? shows a grand tour, and a guided tour, of the penguins data, where the misclassifications are marked by an asterisk. (If these gifs are too small to see the different glyphs, zoom in to make the figures larger.) It can be seen that the one Gentoo penguin that is mistaken for a Chinstrap by the forest model is always moving with its other Gentoo (yellow) family. It can occasionally be seen

to be on the edge of the group, closer to the Chinstraps, in some projections in the grand tour. But in the final projection from the guided tour it is hiding well among the other Gentoos. This is an observation where a mistake has been made because of the inadequacies of the forest algorithm. Forests are only as good as the trees they are constructed from, and we have seen from Section ?? that the splits only on single variables done by trees does not adequately utilise the covariance structure in each class. They make mistakes based on the boxy nature of the boundaries. This can carry through to the forests model. Even though many trees are combined to generate smoother boundaries, forests do not effectively utilise covariance in clusters either. The other mistakes, where Chinstrap are predicted to be Adelie, and vice versa, are more sensible. These mistaken observations can be seen to lie in the border region between the two clusters, and reflect genuine uncertainty about the classification of penguins in these two species.

Figure 17.1: Examining the misclassified cases (marked as asterisks) from a random forest fit to the penguins data. The one Gentoo penguin mistaken for a Chinstrap is a mistake made because the forest method suffers from the same problems as trees - cutting on single variables rather than effectively using covariance structure. The mistakes between the Adelie and Chinstrap penguins are more sensible because all of these observations lie in the bordering regions between the two clusters.

Exercises

1. Examine misclassifications for the `fake_trees` data between cluster 1 and 0, using the votes matrix instead of the principal components. Describe where these errors fall in the simplex.
2. Examine the misclassifications for the `sketches` data, focusing on cactus sketches that were mistaken for bananas. Follow up by plotting the images of these errors, and describe whether the classifier is correct that these sketches are so poor their true cactus identity cannot be determined.

Part V

Miscellaneous

Chapter 18

Multiple time series

Potential topics:

- computing features, and exploring feature space linked to individual time series plots
- projection pursuit of multiple time series, 1D indexes against time
- longitudinal data

References

- Abbott, E. (1884). *Flatland: A romance of many dimensions*. Dover Publications.
- Ahlberg, C., Williamson, C., & Shneiderman, B. (1991). Dynamic Queries for Information Exploration: An Implementation and Evaluation. *ACM CHI '92 Conference Proceedings*, 619–626.
- Anderson, E. (1957). A Semigraphical Method for the Analysis of Complex Problems. *Proceedings of the National Academy of Science*, 13, 923–927.
- Andrews, D. F. (1972). Plots of High-dimensional Data. *Biometrics*, 28, 125–136.
- Andrews, D. F., Gnanadesikan, R., & Warner, J. L. (1971). Transformations of Multivariate Data. *Biometrics*, 27, 825–840.
- Anselin, L., & Bao, S. (1997). Exploratory Spatial Data Analysis Linking Space-Stat and ArcView. In M. M. Fischer & A. Getis (Eds.), *Recent Developments in Spatial Analysis* (pp. 35–59). Springer.
- Arnold, J. B. (2021). *Ggthemes: Extra themes, scales and geoms for ggplot2*. <https://github.com/jrnold/ggthemes>
- ASA Statistical Graphics Section. (2023). *Video Library*. <https://community.amstat.org/jointscsg-section/media/videos>.
- Asimov, D. (1985). The Grand Tour: A Tool for Viewing Multidimensional Data. *SIAM Journal of Scientific and Statistical Computing*, 6(1), 128–143.
- Auguie, B. (2017). *gridExtra: Miscellaneous functions for "grid" graphics*. <https://CRAN.R-project.org/package=gridExtra>
- Australian Bureau of Agricultural and Resource Economics and Sciences. (2018). *Forests of Australia*. <https://www.agriculture.gov.au/abares/forestsaustralia/forest-data-maps-and-tools/spatial-data/forest-cover>
- Becker, R. A., & Chambers, J. M. (1984). *S: An environment for data analysis and graphics*. Wadsworth.
- Becker, R. A., & Cleveland, W. S. (1988). Brushing Scatterplots. In W. S. Cleveland & M. E. McGill (Eds.), *Dynamic graphics for statistics* (pp. 201–224). Wadsworth.
- Becker, R., Cleveland, W. S., & Shyu, M.-J. (1996). The Visual Design and Control of Trellis Displays. *Journal of Computational and Graphical Statistics*, 6(1), 123–155.
- Bederson, B. B., & Shneiderman, B. (2003). *The craft of information visualization: Readings and reflections*. Morgan Kaufmann.

- Bellman, R. (1961). *Adaptive control processes : A guided tour*.
- Bickel, P. J., Kur, G., & Nadler, B. (2018). Projection pursuit in high dimensions. *Proceedings of the National Academy of Sciences*, 115, 9151–9156. <https://doi.org/10.1073/pnas.1801177115>
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Boehmke, B., & Greenwell, B. M. (2019). *Hands-on machine learning with r (1st ed.)*. Chapman; Hall/CRC. <https://doi.org/10.1201/9780367816377>
- Boelaert, J., Ollion, E., & Sodge, J. (2022). *aweSOM: Interactive self-organizing maps*. <https://CRAN.R-project.org/package=aweSOM>
- Bonneau, G.-P., Ertl, T., & Nielson, G. M. (Eds.). (2006). *Scientific visualization: The visual extraction of knowledge from data*. Springer.
- Borg, I., & Groenen, P. J. F. (2005). *Modern Multidimensional Scaling*. Springer.
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
- Breiman, L., & Cutler, A. (2004). *Random Forests*. http://www.math.usu.edu/~adele/forests/cc_ho
- Breiman, L., Cutler, A., Liaw, A., & Wiener, M. (2022). *randomForest: Breiman and cutler's random forests for classification and regression*. <https://www.stat.berkeley.edu/~breiman/RandomForests/>
- Breiman, L., Friedman, J., Olshen, C., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth; Brooks/Cole.
- Buja, A. (1996). Interactive Graphical Methods in the Analysis of Customer Panel Data: Comment. *Journal of Business & Economic Statistics*, 14(1), 128–129.
- Buja, A., & Asimov, D. (1986). Grand Tour Methods: An Outline. *Computing Science and Statistics*, 17, 63–67.
- Buja, A., Asimov, D., Hurley, C., & McDonald, J. A. (1988). Elements of a Viewing Pipeline for Data Analysis. In W. S. Cleveland & M. E. McGill (Eds.), *Dynamic graphics for statistics* (pp. 277–308). Wadsworth.
- Buja, A., Cook, D., Asimov, D., & Hurley, C. (1997). *Dynamic Projections in High-Dimensional Visualization: Theory and Computational Methods*. AT&T Labs.
- Buja, A., Cook, D., Asimov, D., & Hurley, C. (2005). Computational Methods for High-Dimensional Rotations in Data Visualization. In C. R. Rao, E. J. Wegman, & J. L. Solka (Eds.), *Handbook of statistics: Data mining and visualization* (pp. 391–414). Elsevier/North-Holland.
- Buja, A., Cook, D., & Swayne, D. (1996). Interactive High-Dimensional Data Visualization. *Journal of Computational and Graphical Statistics*, 5(1), 78–99.
- Buja, A., Hurley, C., & McDonald, J. A. (1986). A Data Viewer for Multivariate Data. *Computing Science and Statistics*, 17(1), 171–174.
- Buja, A., & Swayne, D. F. (2002). Visualization Methodology for Multidimensional Scaling. *Journal of Classification*, 19(1), 7–43.
- Buja, A., Swayne, D. F., Littman, M. L., Dean, N., Hofmann, H., & Chen, L. (2008). Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 17(2), 444–472. <https://doi.org/10.1198/106186008X318440>
- Buja, A., & Tukey, P. (Eds.). (1991). *Computing and Graphics in Statistics*.

- Springer-Verlag.
- Card, S. K., Mackinlay, J. D., & Schneiderman, B. (1999). *Readings in information visualization*. Morgan Kaufmann Publishers.
- Carr, D. B., Wegman, E. J., & Luo, Q. (1996). *ExplorN: Design Considerations Past and Present* (Technical Report No. 129). Center for Computational Statistics, George Mason University.
- Chatfield, C. (1995). *Problem solving: A statistician's guide*. Chapman; Hall/CRC Press.
- Chen, C., Härdle, W., & Unwin, A. (Eds.). (2006). *Handbook of computational statistics (volume III) data visualization*. Springer.
- Chen, C.-H., Härdle, W., & Unwin, A. (Eds.). (2007). *Handbook of Data Visualization*. Springer.
- Cheng, B., & Titterton, M. (1994). Neural Networks: A Review from a Statistical Perspective. *Statistical Science*, 9(1), 2–30.
- Cheng, J., & Sievert, C. (2021). *Crosstalk: Inter-widget interactivity for HTML widgets*. <https://rstudio.github.io/crosstalk/>
- Chernoff, H. (1973). The Use of Faces to Represent Points in k -dimensional Space Graphically. *Journal of the American Statistical Association*, 68, 361–368.
- Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of American Statistics Association*, 74, 829–836.
- Cleveland, W. S. (1993). *Visualizing Data*. Hobart Press.
- Cleveland, W. S., & McGill, M. E. (Eds.). (1988). *Dynamic graphics for statistics*. Wadsworth.
- Cook, D. (2023). *Mulgar: Functions for pre-processing data for multivariate data visualisation using tours*.
- Cook, D., & Buja, A. (1997). Manual Controls For High-Dimensional Data Projections. *Journal of Computational and Graphical Statistics*, 6(4), 464–480.
- Cook, D., Buja, A., & Cabrera, J. (1993). Projection Pursuit Indexes Based on Orthonormal Function Expansions. *Journal of Computational and Graphical Statistics*, 2(3), 225–250.
- Cook, D., Buja, A., Cabrera, J., & Hurley, C. (1995a). Grand Tour and Projection Pursuit. *Journal of Computational and Graphical Statistics*, 4(3), 155–172.
- Cook, D., Buja, A., Cabrera, J., & Hurley, C. (1995b). Grand Tour and Projection Pursuit. *Journal of Computational and Graphical Statistics*, 4(3), 155–172.
- Cook, D., Hofmann, H., Lee, E.-K., Yang, H., Nikolau, B., & Wurtele, E. (2007). Exploring Gene Expression Data, Using Plots. *Journal of Data Science*, 5(2), 151–182.
- Cook, D., Lee, E.-K., Buja, A., & Wickham, H. (2006). Grand Tours, Projection Pursuit Guided Tours and Manual Controls. In C.-H. Chen, W. Härdle, & A. Unwin (Eds.), *Handbook of Data Visualization*. Springer.
- Cook, D., Majure, J. J., Symanzik, J., & Cressie, N. (1996). Dynamic Graphics in a GIS: Exploring and Analyzing Multivariate Spatial Data using Linked Software. *Computational Statistics: Special Issue on Computer Aided Analyses*

- of *Spatial Data*, 11(4), 467–480.
- Cook, D., & Swayne, D. F. (2007). *Interactive and dynamic graphics for data analysis: With R and GGobi*. Springer-Verlag. <https://doi.org/10.1007/978-0-387-71762-3>
- Cortes, C., Pregibon, D., & Volinsky, C. (2003). Computational Methods for Dynamic Graphs. *Journal of Computational & Graphical Statistics*, 12(4), 950–970.
- Cortes, C., & Vapnik, V. N. (1995). Support-Vector Networks. *Machine Learning*, 20(3), 273–297.
- d’Ocagne, M. (1885). *Coordonnées Parallèles et Axiales: Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèles*. Gauthier-Villars.
- Dalgaard, P. (2002). *Introductory statistics with R*. Springer.
- Dasu, T., Swayne, D. F., & Poole, D. (2005). Grouping Multivariate Time Series: A Case Study. *Proceedings of the IEEE Workshop on Temporal Data Mining: Algorithms, Theory and Applications, in Conjunction with the Conference on Data Mining, Houston, November 27, 2005*, 25–32.
- de Vries, A., & Ripley, B. D. (2022). *Ggdendro: Create dendrograms and tree diagrams using ggplot2*. <https://github.com/andrie/ggdendro>
- Department of Environment, Land, Water & Planning. (2019). *Fire Origins - Current and Historical*. <https://discover.data.vic.gov.au/dataset/fire-origins-current-and-historical>
- Department of Environment, Land, Water & Planning. (2020a). *CFA - Fire Station*. https://discover.data.vic.gov.au/dataset/cfa-fire-station-vmfeat-geomark_point
- Department of Environment, Land, Water & Planning. (2020b). *Recreation Sites*. <https://discover.data.vic.gov.au/dataset/recreation-sites>
- Diaconis, P., & Freedman, D. (1984b). Asymptotics of Graphical Projection Pursuit. *Annals of Statistics*, 12, 793–815.
- Diaconis, P., & Freedman, D. (1984a). Asymptotics of graphical projection pursuit. *Annals of Statistics*, 12(3), 793–815. <https://doi.org/10.1214/aos/1176346703>
- Dykes, J., MacEachren, A. M., & Kraak, M.-J. (2005). *Exploring geovisualization*. Elsevier.
- Everitt, B. S., Landau, S., & Leese, M. (2001). *Cluster Analysis (4th ed)*. Edward Arnold.
- Fienberg, S. E. (1979). Graphical Methods in Statistics. *Journal of American Statistical Association*, 33(4), 165–178.
- Fisher, R. A. (1936b). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7, 179–188.
- Fisher, R. A. (1936a). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2), 179–188. <https://doi.org/10.1111/j.1469-1809.1936.tb02137.x>
- Fisher, R. A. (1938). The Statistical Utilization of Multiple Measurements. *Annals of Eugenics*, 8, 376–386.
- Fisher, M. A., Friedman, J. H., & Tukey, J. W. (1973). *PRIM-9, an*

- interactive multidimensional data display and analysis system*. <https://www.youtube.com/watch?v=B7XoW2qiFUA>
- Fisher-Keller, M. A., Friedman, J. H., & Tukey, J. W. (1974). PRIM-9, an interactive multidimensional data display and analysis system. In W. S. Cleveland (Ed.), *The collected works of John W. Tukey: Graphics 1965-1985, volume v* (pp. 340–346).
- Forbes, J., Cook, D., & Hyndman, R. J. (2020). Spatial modelling of the two-party preferred vote in Australian federal elections: 2001–2016. *Australian & New Zealand Journal of Statistics*, 62(2), 168–185. <https://doi.org/https://doi.org/10.1111/anzs.12292>
- Ford, B. J. (1992). *Images of science: A history of scientific illustration*. The British Library.
- Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3), 768–769.
- Fraley, C., & Raftery, A. E. (2002). Model-based Clustering, Discriminant Analysis, Density Estimation. *Journal of the American Statistical Association*, 97, 611–631.
- Fraley, C., Raftery, A. E., & Scrucca, L. (2022). *Mclust: Gaussian mixture modelling for model-based clustering, classification, and density estimation*. <https://mclust-org.github.io/mclust/>
- Friedman, J. H. (1987). Exploratory Projection Pursuit. *Journal of American Statistical Association*, 82, 249–266.
- Friedman, J. H., & Tukey, J. W. (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computing C*, 23, 881–889.
- Friendly, M., & Denis, D. J. (2004). *Milestones in the history of thematic cartography, statistical graphics, and data visualization*. <http://www.math.yorku.ca/SCS/Gallery/milestone/>.
- Friendly, M., & Denis, D. J. (2006). *Graphical Milestones*. <http://www.math.yorku.ca/SCS/Gallery/milestone>.
- Furnas, G. W., & Buja, A. (1994). Prosection Views: Dimensional Inference Through Sections and Projections. *Journal of Computational and Graphical Statistics*, 3(4), 323–385.
- Gabriel, K. R. (1971). The Biplot Graphical Display of Matrices with Applications to Principal Component Analysis. *Biometrika*, 58, 453–467.
- Gentle, J. E., Härdle, W., & Mori, Y. (Eds.). (2004). *Handbook of computational statistics: Concepts and methods*. Springer.
- Giordani, P., Ferraro, M. B., & Martella, F. (2020). *An introduction to clustering with R*. Springer Singapore. <https://doi.org/10.1007/978-981-13-0553-5>
- Glover, D. M., & Hopke, P. K. (1992). Exploration of Multivariate Chemical Data by Projection Pursuit. *Chemometrics and Intelligent Laboratory Systems*, 16, 45–59.
- Good, P. (2005). *Permutation, Parametric, and Bootstrap Tests of Hypotheses*. Springer.
- Gower, J. C., & Hand, D. J. (1996). *Biplots*. Chapman; Hall.
- Hansen, C., & Johnson, C. R. (2004). *Visualization handbook*. Academic Press.
- Harrison, P. (2023). *Langevitour: Langevin tour*. <https://logarithmic.net/lang>

- evitour/
Hart, C., & Wang, E. (2023). *Detourr: Portable and performant tour animations*. <https://casperhart.github.io/detourr/>
- Hartigan, J. A., & Kleiner, B. (1981). Mosaics for Contingency Tables. *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface*, 268–273.
- Hartigan, J., & Kleiner, B. (1984). A Mosaic of Television Ratings. *The American Statistician*, 38, 32–35.
- Haslett, J., Bradley, R., Craig, P., Unwin, A., & Wills, G. (1991). Dynamic Graphics for Exploring Spatial Data with Application to Locating Global and Local Anomalies. *The American Statistician*, 45(3), 234–242.
- Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.
- Hennig, C., Meila, M., Murtagh, F., & Rocci, R. (Eds.). (2015). *Handbook of cluster analysis*. Chapman; Hall/CRC. <https://doi.org/10.1201/b19706>
- Hofmann, H. (2001). *Graphical Tools for the Exploration of Multivariate Categorical Data*. Books on Demand.
- Hofmann, H. (2003). Constructing and Reading Mosaicplots. *Computational Statistics and Data Analysis*, 43(4), 565–580.
- Hofmann, H., & Theus, M. (1998). Selection Sequences in MANET. *Computational Statistics*, 13(1), 77–87.
- Horst, A., Hill, A., & Gorman, K. (2022). *Palmerpenguins: Palmer archipelago (antarctica) penguin data*. <https://CRAN.R-project.org/package=palmerpenguins>
- Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6), 417–441. <https://doi.org/10.1037/h0071325>
- Huber, P. J. (1985). Projection Pursuit (with discussion). *Annals of Statistics*, 13, 435–525.
- Hurley, C. (1987). *The data viewer: An interactive program for data analysis* [PhD thesis]. University of Washington.
- Iannone, R., Cheng, J., Schloerke, B., Hughes, E., & Seo, J. (2022). *Gt: Easily create presentation-ready display tables*. <https://CRAN.R-project.org/package=gt>
- Ihaka, R., & Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, 5, 299–314.
- Ihaka, R., Murrell, P., Hornik, K., Fisher, J. C., Stauffer, R., Wilke, C. O., McWhite, C. D., & Zeileis, A. (2023). *Colorspace: A toolbox for manipulating and assessing colors and palettes*. <https://CRAN.R-project.org/package=colorspace>
- Inselberg, A. (1985). The Plane with Parallel Coordinates. *The Visual Computer*, 1, 69–91.
- Iowa State University. (2020). *ASOS-AWOS-METAR data download*. https://mesonet.agron.iastate.edu/request/download.phtml?network=AU__ASOS
- Johnson, D., & Travis, J. (2007). *Flatland: The movie*. <https://round-drum-w7xh.squarespace.com/our-story>.

- Johnson, R. A., & Wichern, D. W. (2002). *Applied multivariate statistical analysis (5th ed)*. Prentice-Hall.
- Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Phil. Trans. R. Soc. A.*, *374*, 20150202. <https://doi.org/10.1098/rsta.2015.0202>
- Jones, M. C., & Sibson, R. (1987). What is Projection Pursuit? (With discussion). *Journal of the Royal Statistical Society, Series A*, *150*, 1–36.
- Kassambara, A. (2017). *Practical guide to cluster analysis in r: Unsupervised machine learning*. STHDA.
- Kassambara, A. (2023). *Ggpubr: ggplot2 based publication ready plots*. <https://rpkgs.datanovia.com/ggpubr/>
- Kohonen, T. (2001). *Self-Organizing Maps (3rd ed)*. Springer.
- Koschat, M. A., & Swayne, D. F. (1996). Interactive Graphical Methods in the Analysis of Customer Panel Data (with discussion). *Journal of Business and Economic Statistics*, *14*(1), 113–132.
- Krijthe, J. (2022). *Rtsne: T-distributed stochastic neighbor embedding using a barnes-hut implementation*. <https://github.com/jkrijthe/Rtsne>
- Kruskal, J. B. (1964a). Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, *29*, 1–27.
- Kruskal, J. B. (1964b). Nonmetric Multidimensional Scaling: A Numerical Method. *Psychometrika*, *29*, 115–129.
- Kruskal, J. B., & Wish, M. (1978). *Multidimensional Scaling*. Sage Publications.
- Laa, U., Cook, D., & Valencia, G. (2020a). A slice tour for finding hollowness in high-dimensional data. *Journal of Computational and Graphical Statistics*, *29*(3), 681–687. <https://doi.org/10.1080/10618600.2020.1777140>
- Laa, U., Cook, D., & Valencia, G. (2020b). A slice tour for finding hollowness in high-dimensional data. *Journal of Computational and Graphical Statistics*, *29*(3), 681–687. <https://doi.org/10.1080/10618600.2020.1777140>
- Lancaster, H. O. (1965). The helmert matrices. *The American Mathematical Monthly*, *72*(1), 4–12.
- Lee, E. K., Cook, D., Klinke, S., & Lumley, T. (2005a). Projection Pursuit for Exploratory Supervised Classification. *Journal of Computational and Graphical Statistics*, *14*(4), 831–846.
- Lee, E.-K. (2018). PPtreeViz: An r package for visualizing projection pursuit classification trees. *Journal of Statistical Software*, *83*(8), 1–30. <https://doi.org/10.18637/jss.v083.i08>
- Lee, E.-K., & Cook, D. (2009). A projection pursuit index for large p small n data. *Statistics and Computing*, *20*, 381–392. <https://doi.org/10.1007/s11222-009-9131-1>
- Lee, E.-K., Cook, D., Klinke, S., & Lumley, T. (2005b). Projection Pursuit for Exploratory Supervised Classification. *Journal of Computational and Graphical Statistics*, *14*(4), 831–846.
- Lee, S. (2021). *Liminal: Multivariate data visualization with tours and embeddings*. <https://CRAN.R-project.org/package=liminal>
- Lee, S., Cook, D., Silva, N. da, Laa, U., Spyrisson, N., Wang, E., & Zhang, H. S. (2022). The state-of-the-art on tours for dynamic visualization of

- high-dimensional data. *WIREs Computational Statistics*, 14(4), e1573. <https://doi.org/10.1002/wics.1573>
- Lee, Y. D., Cook, D., Park, J., & Lee, E.-K. (2013). PPtree: Projection pursuit classification tree. *Electronic Journal of Statistics*, 7(none), 1369–1386. <https://doi.org/10.1214/13-EJS810>
- Leisch, F., & Gruen, B. (2023). *CRAN task view: Cluster analysis & finite mixture models*. <https://cran.r-project.org/web/views/Cluster.html>.
- Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News*, 2(3), 18–22. <https://CRAN.R-project.org/doc/Rnews/>
- Littman, M. L., Swayne, D. F., Dean, N., & Buja, A. (1992). Visualizing the Embedding of Objects in Euclidean Space. *Computing Science and Statistics: Proceedings of the 24th Symposium on the Interface*, 208–217.
- Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129–137. <https://doi.org/10.1109/TIT.1982.1056489>
- Longley, P. A., Maguire, D. J., Goodchild, M. F., & Rhind, D. W. (2005). *Geographic information systems and science*. John Wiley & Sons.
- Loperfido, N. (2018). Skewness-based projection pursuit: A computational approach. *Computational Statistics & Data Analysis*, 120, 42–57. <https://doi.org/https://doi.org/10.1016/j.csda.2017.11.001>
- Maaten, L. van der, & Hinton, G. (2008). Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9(Nov), 2579–2605. <http://www.jmlr.org/papers/v9/vandermaten08a.html>
- MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. L. Cam & J. Neyman (Eds.), *Proc. Of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). University of California Press.
- Maindonald, J., & Braun, J. (2003). *Data analysis and graphics using r - an example-based approach*. Cambridge University Press.
- Martin, E. (1965). *Flatland*. <http://www.der.org/films/flatland.html>.
- McFarlane, M., & Young, F. W. (1994). Graphical Sensitivity Analysis for Multidimensional Scaling. *Journal of Computational and Graphical Statistics*, 3, 23–33.
- McInnes, L., Healy, J., & Melville, J. (2018). *UMAP: Uniform manifold approximation and projection for dimension reduction*. <http://arxiv.org/abs/1802.03426>
- McNeil, D. (1977). *Interactive Data Analysis*. John Wiley & Sons.
- McVicar, T. (2011). *Near-surface wind speed. v10. CSIRO. Data collection*. <https://doi.org/10.25919/5c5106acbc02>
- Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2023). *e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), TU wien*. <https://CRAN.R-project.org/package=e1071>
- Milborrow, S. (2022). *Rpart.plot: Plot rpart models: An enhanced version of plot.rpart*. <http://www.milbo.org/rpart-plot/index.html>
- Mock, T. (2022). *gtExtras: Extending gt for beautiful HTML tables*. <https://CRAN.R-project.org/package=gtExtras>

- Moon, K. R., Dijk, D. van, Wang, Z., Gigante, S., Burkhardt, D. B., Chen, W. S., Yim, K., Elzen, A. van den, Hirn, M. J., Coifman, R. R., Ivanova, N. B., Wolf, G., & Krishnaswamy, S. (2019). Visualizing structure and transitions for biological data exploration. *Nature Biotechnology*, *37*, 1482–1492. <https://doi.org/10.1038/s41587-019-0336-3>
- Murrell, P. (2005). *R graphics*. Chapman & Hall/CRC.
- OpenStreetMap contributors. (2020). *Planet dump retrieved from https://planet.osm.org*. <https://www.openstreetmap.org>.
- Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. <https://doi.org/10.1080/14786440109462720>
- Pedersen, T. L. (2022). *Patchwork: The composer of plots*. <https://CRAN.R-project.org/package=patchwork>
- Perisic, I., & Posse, C. (2005). Projection pursuit indices based on the empirical distribution function. *Journal of Computational and Graphical Statistics*, *14*(3), 700–715. <https://doi.org/10.1198/106186005X69440>
- Polzehl, J. (1995). Projection Pursuit Discriminant Analysis. *Computational Statistics and Data Analysis*, *20*, 141–157.
- Posse, C. (1992). Projection Pursuit Discriminant Analysis for Two Groups. *Communications in Statistics, Part A – Theory and Methods*, *21*, 1–19.
- Posse, C. (1995). Tools for Two-dimensional Projection Pursuit. *Journal of Computational and Graphical Statistics*, *4*(2), 83–100.
- P-Tree System. (2020). *JAXA Himawari Monitor - User's Guide*. <https://www.eorc.jaxa.jp/ptree/userguide.html>
- R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. <https://www.R-project.org/>
- Rao, C. R. (1948). The Utilization of Multiple Measurements in Problems of Biological Classification (with discussion). *Journal of the Royal Statistical Society, Series B*, *10*, 159–203.
- Rao, C. R. (Ed.). (1993). *Handbook of Statistics, Vol. 9*. Elsevier Science Publishers.
- Rao, C. R., Wegman, E. J., & Solka, J. L. (Eds.). (2006). *Handbook of Statistics: Data Mining and Visualization*. Elsevier/North-Holland.
- Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.
- Ripley, B. (2023). *MASS: Support functions and datasets for venables and ripley's MASS*. <http://www.stats.ox.ac.uk/pub/MASS4/>
- Rothkopf, E. Z. (1957). A Measure of Stimulus Similarity and Errors in Some Paired-associate Learning Tasks. *Journal of Experimental Psychology*, *53*, 94–101.
- Schloerke, B. (2016). *Geozoo: Zoo of geometric objects*. <https://CRAN.R-project.org/package=geozoo>
- Schloerke, B., Cook, D., Larmanange, J., Briatte, F., Marbach, M., Thoen, E., Elberg, A., & Crowley, J. (2021). *GGally: Extension to ggplot2*. <https://CRAN.R-project.org/package=GGally>

- Scrucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). mclust 5: Clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1), 289–317. <https://doi.org/10.32614/RJ-2016-021>
- Shepard, R. N. (1962). The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function, I and II. *Psychometrika*, 27, 125-139 and 219-246.
- Sievert, C. (2020). *Interactive web-based data visualization with r, plotly, and shiny*. Chapman; Hall/CRC. <https://plotly-r.com>
- Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., & Despouy, P. (2023). *Plotly: Create interactive web graphics via plotly.js*.
- Sjoberg, D. D., Larmarange, J., Curry, M., Lavery, J., Whiting, K., & Zabor, E. C. (2023). *Gtsummary: Presentation-ready data summary and analytic result tables*. <https://CRAN.R-project.org/package=gtsummary>
- Sjoberg, D. D., Whiting, K., Curry, M., Lavery, J. A., & Larmarange, J. (2021). Reproducible summary tables with the gtsummary package. *The R Journal*, 13, 570–580. <https://doi.org/10.32614/RJ-2021-053>
- Slowikowski, K. (2023). *Ggrepel: Automatically position non-overlapping text labels with ggplot2*. <https://github.com/slowkow/ggrepel>
- Sparks, A. H., Carroll, J., Goldie, J., Marchiori, D., Melloy, P., Padgham, M., Parsonage, H., & Pembleton, K. (2020). *bomrang: Australian government bureau of meteorology (BOM) data client*. <https://CRAN.R-project.org/package=bomrang>
- Spence, R. (2007). *Information visualization: Design for interaction*. Prentice Hall.
- Stauffer, R., Mayr, G. J., Dabernig, M., & Zeileis, A. (2009). Somewhere over the rainbow: How to make effective use of colors in meteorological visualizations. *Bulletin of the American Meteorological Society*, 96(2), 203–216. <https://doi.org/10.1175/BAMS-D-13-00155.1>
- Sutherland, P., Rossini, A., Lumley, T., Lewin-Koh, N., Dickerson, J., Cox, Z., & Cook, D. (2000b). Orca: A Visualization Toolkit for High-Dimensional Data. *Journal of Computational and Graphical Statistics*, 9(3), 509–529.
- Sutherland, P., Rossini, A., Lumley, T., Lewin-Koh, N., Dickerson, J., Cox, Z., & Cook, D. (2000a). Orca: A visualization toolkit for high-dimensional data. *Journal of Computational and Graphical Statistics*, 9(3), 509–529. <https://doi.org/10.1080/10618600.2000.10474896>
- Swayne, D. F., Buja, A., & Temple Lang, D. (2004). Exploratory visual analysis of graphs in GGobi. In J. Antoch (Ed.), *CompStat: Proceedings in computational statistics, 16th symposium*. Physica-Verlag.
- Swayne, D. F., Cook, D., & Buja, A. (1992). XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S. *American Statistical Association 1991 Proceedings of the Section on Statistical Graphics*, 1–8.
- Swayne, D. F., Cook, D., & Buja, A. (1998). XGobi: Interactive dynamic data visualization in the x window system. *Journal of Computational and Graphical Statistics*, 7(1), 113–130. <https://doi.org/10.1080/10618600.1998.10474764>

- Swayne, D. F., & Klinke, S. (1998). Editorial commentary. *Computational Statistics: Special Issue on The Use of Interactive Graphics*, 14(1).
- Swayne, D. F., Temple Lang, D., Buja, A., & Cook, D. (2003). GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization. *Computational Statistics & Data Analysis*, 43, 423–444.
- Swayne, D., & Buja, A. (1998). Missing Data in Interactive High-Dimensional Data Visualization. *Computational Statistics*, 13(1), 15–26.
- Symanzik, J. (2002). New applications of the image grand tour. *Computing Science and Statistics*, 34, 500–512. https://math.usu.edu/symanzik/papers/2002_interface.pdf
- Symanzik, J. (2004). Interactive and Dynamic Graphics. In J. E. Gentle, W. Härdle, & Y. Mori (Eds.), *Handbook of computational statistics: Concepts and methods* (pp. 293–336). Springer.
- Takatsuka, M., & Gahegan, M. (2002). GeoVISTA Studio: A Codeless Visual Programming Environment for Geoscientific Data Analysis and Visualization. *The Journal of Computers and Geosciences*, 28(10), 1131–1144.
- Tarpey, T., Li, L., & Flury, B. (1995). Principal points and self-consistent points of elliptical distributions. *The Annals of Statistics*, 23, 103–112.
- Temple Lang, D., Swayne, D., Wickham, H., & Lawrence, M. (2006). *rggobi: An Interface between R and GGobi*. <http://www.R-project.org>.
- Therneau, T., & Atkinson, B. (2022). *Rpart: Recursive partitioning and regression trees*. <https://CRAN.R-project.org/package=rpart>
- Theus, M. (2002). Interactive Data Visualization Using Mondrian. *Journal of Statistical Software*, 7(11), <http://www.jstatsoft.org>.
- Theus, M., Hofmann, H., & Wilhelm, A. F. X. (1998). Selection Sequences – Interactive Analysis of Massive Data Sets. *Computing Science and Statistics*, 29(1), 439–444.
- Thompson, G. L. (1993). Generalized Permutation Polytopes and Exploratory Graphical Methods for Ranked Data. *The Annals of Statistics*, 21, 1401–1430.
- Tierney, L. (1991). *LispStat: An Object-Orientated Environment for Statistical Computing and Dynamic Graphics*. John Wiley & Sons.
- Torgerson, W. S. (1952). Multidimensional Scaling. 1. Theory and Method. *Psychometrika*, 17, 401–419.
- Tufte, E. (1983). *The visual display of quantitative information*. Graphics Press.
- Tufte, E. (1990). *Envisioning information*. Graphics Press.
- Tukey, J. W. (1965). The Technical Tools of Statistics. *The American Statistician*, 19, 23–28.
- Unwin, A. R., Hawkins, G., Hofmann, H., & Siegl, B. (1996). Interactive Graphics for Data Sets with Missing Values - MANET. *Journal of Computational and Graphical Statistics*, 5(2), 113–122.
- Unwin, A., Hofmann, H., & Wilhelm, A. (2002). Direct Manipulation Graphics for Data Mining. *Journal of Image and Graphics*, 2(1), 49–65.
- Unwin, A., Theus, M., & Hofmann, H. (2006). *Graphics of Large Datasets: Visualizing a Million*. Springer.
- Unwin, A., Volinsky, C., & Winkler, S. (2003). Parallel Coordinates for Exploratory Modelling Analysis. *Comput. Stat. Data Anal.*, 43(4), 553–564.

- [https://doi.org/10.1016/S0167-9473\(02\)00292-X](https://doi.org/10.1016/S0167-9473(02)00292-X)
- Urbanek, S., & Theus, M. (2003). iPlots: High Interaction Graphics for R. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd international workshop on distributed statistical computing (DSC 2003)*.
- Vaidyanathan, R., Xie, Y., Allaire, J., Cheng, J., Sievert, C., & Russell, K. (2023). *Htmlwidgets: HTML widgets for r*. <https://github.com/ramnathv/htmlwidgets>
- van der Maaten, L. J. P. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15, 3221–3245.
- van der Maaten, L. J. P., & Hinton, G. E. (2008). Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.
- Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory*. Springer.
- Velleman, P. F., & Velleman, A. Y. (1985). *Data desk handbook*. Data Description, Inc.
- Venables, W. N., & Ripley, B. (2002a). *Modern Applied Statistics with S*. Springer-Verlag.
- Venables, W. N., & Ripley, B. D. (2002b). *Modern applied statistics with s* (Fourth). Springer. <https://www.stats.ox.ac.uk/pub/MASS4/>
- Wainer, H. (2000). *Visual Revelations (2nd ed)*. LEA, Inc.
- Wainer, H., & Spence, I. (eds). (2005a). *The Commercial and Political Atlas, Representing, by means of Stained Copper-Plate Charts, The Progress of the Commerce, Revenues, Expenditure, and Debts of England, during the whole of the Eighteenth Century, by William Playfair*. Cambridge University Press.
- Wainer, H., & Spence, I. (eds). (2005b). *The Statistical Breviary; Shewing on a Principle entirely new, the resources of every state and kingdom in Europe; illustrated with Stained Copper-Plate Charts, representing the physical powers of each distinct nation with ease and perspicuity by William Playfair*. Cambridge University Press.
- Wang, P. C. C. (Ed.). (1978). *Graphical Representation of Multivariate Data*. Academic Press.
- Wegman, E. (1990). Hyperdimensional Data Analysis Using Parallel Coordinates. *Journal of American Statistics Association*, 85, 664–675.
- Wegman, E. J. (1991). *The Grand Tour in k-Dimensions* (Technical Report No. 68). Center for Computational Statistics, George Mason University.
- Wegman, E. J., & Carr, D. B. (1993). *Statistical Graphics and Visualization* (C. R. Rao, Ed.; pp. 857–958). Elsevier Science Publishers.
- Wegman, E. J., Poston, W. L., & Solka, J. L. (1998). Image Grand Tour. *Automatic Target Recognition VIII - Proceedings of SPIE*, 3371, 286–294.
- Wehrens, R., & Buydens, L. M. C. (2007). Self- and super-organizing maps in R: The kohonen package. *Journal of Statistical Software*, 21(5), 1–19. <https://doi.org/10.18637/jss.v021.i05>
- Wehrens, R., & Kruisselbrink, J. (2018). Flexible self-organizing maps in kohonen 3.0. *Journal of Statistical Software*, 87(7), 1–18. <https://doi.org/10.18637/jss.v087.i07>
- Wehrens, R., & Kruisselbrink, J. (2022). *Kohonen: Supervised and unsupervised self-organising maps*. <https://CRAN.R-project.org/package=kohonen>

- Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>
- Wickham, H. (2022). *Classify: Explore classification models in high dimensions*. <http://had.co.nz/classify>
- Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., Woo, K., Yutani, H., & Dunnington, D. (2023). *ggplot2: Create elegant data visualisations using the grammar of graphics*. <https://CRAN.R-project.org/package=ggplot2>
- Wickham, H., & Cook, D. (2023). *Tourr: Tour methods for multivariate data visualisation*. <https://github.com/ggobi/tourr>
- Wickham, H., Cook, D., & Hofmann, H. (2015). Visualizing statistical models: Removing the blindfold. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(4), 203–225. <https://doi.org/10.1002/sam.11271>
- Wickham, H., Cook, D., Hofmann, H., & Buja, A. (2011a). Tourr: An R Package for Exploring Multivariate Data with Projections. *Journal of Statistical Software*, 40(2). <https://doi.org/10.18637/jss.v040.i02>
- Wickham, H., Cook, D., Hofmann, H., & Buja, A. (2011b). tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2), 1–18. <https://doi.org/10.18637/jss.v040.i02>
- Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). *Dplyr: A grammar of data manipulation*. <https://CRAN.R-project.org/package=dplyr>
- Wickham, H., Hester, J., & Bryan, J. (2023). *Readr: Read rectangular text data*. <https://CRAN.R-project.org/package=readr>
- Wilhelm, A. F. X., Wegman, E. J., & Symanzik, J. (1999). Visual Clustering and Classification: The Oronsay Particle Size Data Set Revisited. *Computational Statistics: Special Issue on Interactive Graphical Data Analysis*, 14(1), 109–146.
- Wilkinson, L. (2005). *The grammar of graphics*. Springer.
- Wills, G. (1999). NicheWorks – Interactive Visualization of Very Large Graphs. *Journal of Computational and Graphical Statistics*, 8(2), 190–212.
- Xie, Y., Hofmann, H., & Cheng, X. (2014). Reactive Programming for Interactive Graphics. *Statistical Science*, 29(2), 201–213. <https://doi.org/10.1214/14-STS477>
- Young, F. W., Valero-Mora, P. M., & Friendly, M. (2006). *Visual Statistics: Seeing Data with Dynamic Interactive Graphics*. John Wiley & Sons.
- Zeileis, A., Fisher, J. C., Hornik, K., Ihaka, R., McWhite, C. D., Murrell, P., Stauffer, R., & Wilke, C. O. (2020). colorspace: A toolbox for manipulating and assessing colors and palettes. *Journal of Statistical Software*, 96(1), 1–49. <https://doi.org/10.18637/jss.v096.i01>
- Zeileis, A., Hornik, K., & Murrell, P. (2009). Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53(9), 3259–3270. <https://doi.org/10.1016/j.csda.2008.11.033>
- Zhang, C., Ye, J., & Wang, X. (2023). A computational perspective on projection pursuit in high dimensions: Feasible or infeasible feature extraction. *International Statistical Review*, 91(1), 140–161. <https://doi.org/10.1111/insr.12517>

- Zhu, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*.
<https://CRAN.R-project.org/package=kableExtra>

Appendix A

Toolbox

A.1 Using tours in the `tourr` package

A.1.1 Installation

You can install the released version of `tourr` from CRAN with:

```
install.packages("tourr")
```

and the development version from github with:

```
# install.packages("remotes")
remotes::install_github("ggobi/tourr")
```

A.1.2 Getting started

To run a tour in R, use one of the `animate` functions. This code will show a 2D tour displayed as a scatterplot on a 6D data set with three labelled classes.

```
animate_xy(flea[, -7], col=flea$species)
```

Wickham et al. (2011a) remains a good reference for learning more about this package.

A.1.3 Different tours

There are two main components of the tour algorithm:

1. The dimension of the projection, which will impact the type of display to use.
2. The algorithm that delivers the projections to show.

A.1.3.1 Dimension of projection and display

- `display_dist()`: choice of density, histogram or average shifted histogram (ash) display of the 1D projections.
- `display_xy()`, `display_density2d()`, `display_groupxy()`, `display_pca()`, `display_sage()`, `display_slice()`, `display_trails()`: choices in display of 2D projections.
- `display_depth()`, `display_stereo()`: choices to display 3D projections.
- `display_pcp()`, `display_scattermat()`, `display_stars()`, `display_faces()`: choices for displaying three or more variables.
- `display_image()`: to use with multispectral images, where different combinations of spectral bands are displayed. See E. J. Wegman et al. (1998) and Symanzik (2002) for applications.
- `dependence_tour()`: displaying two groups of variables as in multiple regression, or multivariate regression or canonical correlation analysis, as two independent 1D projections.
- `display_andrews()`: 1D projections as Andrews curves.

A.1.3.2 Algorithms for projection delivery

- `grand_tour()`: Smooth sequence of random projections to view all possible projections as quickly as possible. Good for getting an overview of the high-dimensional data, especially when you don't know what you are looking for.
- `guided_tour()`: Follow a projection pursuit optimisation to find projections that have particular patterns. This is used when you want to learn if the data has particular patterns, such as clustering or outliers. Use the `holes()` index to find projections with gaps that allow one to see clusters, or `lda_pp()` or `pda_pp()` when class labels are known and you want to find the projections where the clusters are separated.
- `little_tour()`: Smoothly interpolate between pairs of variables, to show all the marginal views of the data.
- `local_tour()`: Makes small movements around a chosen projections to explore a small neighbourhood. Very useful to learn if small distances away from a projection change the pattern substantially or not.
- `radial_tour()`: Interpolates a chosen variable out of the projection, and then back into the projection. This is useful for assessing importance of variables to pattern in a projection. If the pattern changes a lot when the variable is rotated out, then the variable is important for producing it.
- `dependendence_tour()`: Delivers two sequences of 1D grand tours, to examine associations between two sets of variables.
- `frozen_tour()`: This is an interesting one! it allows the coefficient for

some variables to be fixed, and others to vary.

A.1.4 The importance of scale

Scaling of multivariate data is really important in many ways. It affects most model fitting, and can affect the perception of patterns when data is visualised. Here we describe a few scaling issues to take control of when using tours.

A.1.4.1 Pre-processing data

It is generally useful to standardise your data to have mean 0 and variance-covariance equal to the identity matrix before using the tour. We use the tour to discover associations between variables. Characteristics of single variables should be examined and understood before embarking on looking for high-dimensional structure.

The `rescale` parameter in the `animate()` function will scale all variables to range between 0 and 1, prior to starting the tour. This will force all to have the same range. It is the default, and without this data with different ranges across variable may have some strange patterns. If you have already scaled the data yourself, even if using a different scaling such as using standardised variables you should set `rescale=FALSE`.

A more severe transformation that can be useful prior to starting a tour is to **sphere** the data. This is also an option in the `animate()` function, but is `FALSE` by default. Sphering is the same as conducting a principal component analysis, and using the principal components as the variables. It removes all linear association between variables! This can be especially useful if you want to focus on finding non-linear associations, including clusters, and outliers.

A.1.4.2 Scaling to fit into plot region

The `half_range` parameter in most of the display types sets the range used to scale the data into the plot. It is estimated when a tour is started, but you may need to change it if you find that the data keeps escaping the plot window or is not fully using the space. Space expands exponentially as dimension increases, and the estimation takes this into account. However, different distributions of data points lead to different variance of observations in high-dimensional space. A skewed distribution will be more varied than a normal distribution. It is hard to estimate precisely how the data should be scaled so that it fits nicely into the plot space for all projections viewed.

The `center` parameter is used to centre each projection by setting the mean to be at the middle of the plot space. With different distributions the mean of the data can vary around the plot region, and this can be distracting. Fixing the mean of each projection to always be at the center of the plot space makes it easier to focus on other patterns.

A.1.5 Saving your tour

The functions `save_history()` and `planned_tour()` allow the tour path to be pre-computed, and re-played in your chosen way. The tour path is saved as a list of projection vectors, which can also be passed to external software for displaying tours easily. Only a minimal set of projections is saved, by default, and a full interpolation path of projections can always be generated from it using the `interpolate()` function.

Versions and elements of tours can be saved for publication using a variety of functions:

- `render_gif()`: Save a tour as an animated gif, using the `gifski` package.
- `render_proj()`: Save an object that can be used to produce a polished rendering of a single projection, possibly with `ggplot`.
- `render_anim()`: Creates an object containing a sequence of projections that can be used with `plotly()` to produce an HTML animation, with interactive control.

A.1.6 Understanding your tour path

Grand tour, and show paths on the space, starting with 1D. How more time gives more coverage of the sphere. Then the torus and paths on torus.

A.2 What not to do

- Categorical data, or discrete data
- When time or space is a variable

A.3 Tours in other software

Also include other software now available

- `detourr`
- `langevitour`
- `woylier`
- `spinifex`
- `ferrn`

Appendix B

Data

This chapter describes the datasets used throughout the book, one possibility is that this could be moved to an appendix and in the online version of the book any mention of the dataset in the main text could be cross-linked to its description. One thing that I really liked in the first edition is the emphasis on analysis question of interest and the little notes about the data. I believe the original layout was to put the data in order of appearance of the book, which naturally clusters them by technique.

Name	Description
aflw	Player statistics from the AFLW
bushfires	Multivariate spatio-temporal data for locations of bushfires
oz_election_2001	Socioeconomic characteristics of Australian electorates in 2001
oz_election_2016	Socioeconomic characteristics of Australian electorates in 2016
penguins	Measure four physical characteristics of three species of penguins
pisa	OECD programme for international student assessment data
sketches	Google's Quickdraw data
clusterchallenge	Simulated data used to show various cluster examples
duolingo	Learning traces for duolingo users
harp	Chemical abundances from HARPS-GTO stellar survey
plastics	Plastic pollution from different companies and countries 2019-2020
pdfsense	Physics model fits analysed in Cook, Laa, Valencia (2018)
pedestrian	Melbourne pedestrian counts
mouseretina	Single cell RNA-seq data from Macosko et al.
mosquitos	An audio "civic science" dataset containing wingbeat frequencies for classifying mosquito species
melbhouseprices	Melbourne housing prices
books	Features of common books
wages	NA

B.1 Australian Football League Women

Description

The `aflw` data is from the 2021 Women's Australian Football League. These are average player statistics across the season, with game statistics provided by the `fitzRoy` package. If you are new to the game of AFL, there is a nice explanation on Wikipedia.

Variables

Rows: 381

Columns: 35

\$ id	<chr> "CD_I1001678", "CD_I1001679", "CD_I1001681", "CD_I1001~
\$ given_name	<chr> "Jordan", "Brianna", "Jodie", "Ebony", "Emma", "Pepa",~
\$ surname	<chr> "Zanchetta", "Green", "Hicks", "Antonio", "King", "Ran~
\$ number	<int> 2, 3, 5, 12, 60, 21, 22, 23, 35, 14, 3, 8, 16, 12, 19,~
\$ team	<chr> "Brisbane Lions", "West Coast Eagles", "GWS Giants", "~
\$ position	<chr> "INT", "INT", "HFFR", "WL", "RK", "BPL", "INT", "INT",~
\$ time_pct	<dbl> 63.00000, 61.25000, 76.50000, 74.90000, 85.10000, 77.4~
\$ goals	<dbl> 0.0000000, 0.0000000, 0.0000000, 0.1000000, 0.6000000,~
\$ behinds	<dbl> 0.0000000, 0.0000000, 0.5000000, 0.4000000, 0.4000000,~
\$ kicks	<dbl> 5.000000, 2.500000, 3.750000, 8.800000, 4.100000, 3.22~
\$ handballs	<dbl> 2.500000, 3.750000, 3.000000, 3.600000, 2.700000, 2.22~
\$ disposals	<dbl> 7.500000, 6.250000, 6.750000, 12.400000, 6.800000, 5.4~
\$ marks	<dbl> 1.5000000, 0.2500000, 1.0000000, 3.7000000, 2.2000000,~
\$ bounces	<dbl> 0.0000000, 0.0000000, 0.0000000, 0.6000000, 0.1000000,~
\$ tackles	<dbl> 3.000000, 2.250000, 2.250000, 3.900000, 2.000000, 1.77~
\$ contested	<dbl> 3.500000, 2.250000, 3.500000, 5.700000, 4.400000, 2.66~
\$ uncontested	<dbl> 3.500000, 4.500000, 3.000000, 7.000000, 2.800000, 1.77~
\$ possessions	<dbl> 7.000000, 6.750000, 6.500000, 12.700000, 7.200000, 4.4~
\$ marks_in50	<dbl> 1.0000000, 0.0000000, 0.2500000, 0.5000000, 0.9000000,~
\$ contested_marks	<dbl> 1.0000000, 0.0000000, 0.0000000, 0.4000000, 1.2000000,~
\$ hitouts	<dbl> 0.0000000, 0.0000000, 0.0000000, 0.0000000, 19.4000000~
\$ one_pct	<dbl> 0.0000000, 1.5000000, 0.5000000, 1.2000000, 2.6000000,~
\$ disposal	<dbl> 60.25000, 67.15000, 37.20000, 65.96000, 61.72000, 66.8~
\$ clangers	<dbl> 2.000000, 0.500000, 2.500000, 3.100000, 2.400000, 1.33~
\$ frees_for	<dbl> 1.0000000, 0.5000000, 0.2500000, 2.5000000, 0.5000000,~
\$ frees_against	<dbl> 1.0000000, 0.5000000, 1.2500000, 1.3000000, 1.1000000,~
\$ rebounds_in50	<dbl> 0.0000000, 0.5000000, 0.2500000, 1.1000000, 0.0000000,~
\$ assists	<dbl> 0.00000000, 0.00000000, 0.00000000, 0.20000000, 0.2000~
\$ accuracy	<dbl> 0.00000, 0.00000, 0.00000, 5.00000, 30.00000, 0.00000,~
\$ turnovers	<dbl> 1.500000, 1.000000, 2.500000, 4.000000, 1.700000, 1.22~
\$ intercepts	<dbl> 2.0000000, 2.0000000, 0.5000000, 5.3000000, 1.3000000,~
\$ tackles_in50	<dbl> 0.5000000, 0.0000000, 0.7500000, 0.5000000, 0.5000000,~
\$ shots	<dbl> 0.5000000, 0.0000000, 0.7500000, 1.0000000, 1.2000000,~


```
$ metres      <dbl> 72.50000, 58.50000, 76.00000, 225.90000, 89.80000, 76.~
$ clearances  <dbl> 0.5000000, 0.2500000, 1.2500000, 0.4000000, 0.9000000,~
```

Purpose

The primary analysis is to summarise the variation using principal component analysis, which gives information about relationships between the statistics or skills sets common in players. One also might be tempted to cluster the players, but there are no obvious clusters so it could be frustrating. At best one could partition the players into groups, while recognising there are no absolutely distinct and separated groups.

Source

See the information provided with the `fitzRoy` package.

Pre-processing

The code for downloading and pre-processing the data is available at the mulgar website in the `data-raw` folder. The data provided by the `fitzRoy` package was pre-processed to reduce the variables to only those that relate to player skills and performance. It is possible that using some transformations on the variables would be useful to make them less skewed.

B.2 Bushfires

Description

This data was collated by Weihao (Patrick) Li as part of his Honours research at Monash University. It contains fire ignitions as detected from satellite hotspots, and processed using the `spotaroo` package, augmented with measurements on weather, vegetation, proximity to human activity. The cause variable is predicted based on historical fire ignition data collected by County Fire Authority personnel.

Variables

Rows: 1,021

Columns: 60

```
$ id          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1~
$ lon         <dbl> 141.1300, 141.3000, 141.4800, 147.1600, 148.1050, 144.18~
$ lat         <dbl> -37.13000, -37.65000, -37.35000, -37.85000, -37.57999, --
$ time        <date> 2019-10-01, 2019-10-01, 2019-10-02, 2019-10-02, 2019-10~
$ FOR_CODE    <dbl> 41, 41, 91, 44, 0, 44, 0, 102, 0, 91, 45, 41, 45, 45, 45~
$ FOR_TYPE    <chr> "Eucalypt Medium Woodland", "Eucalypt Medium Woodland", ~
$ FOR_CAT     <chr> "Native forest", "Native forest", "Commercial plantation~
$ COVER       <dbl> 1, 1, 4, 2, 6, 2, 6, 5, 6, 4, 2, 1, 2, 2, 2, 2, 6, 6, 6,~
```

```

$ HEIGHT <dbl> 2, 2, 4, 2, 6, 2, 6, 5, 6, 4, 3, 2, 3, 3, 3, 2, 6, 6, 6,~
$ FOREST <dbl> 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,~
$ rf <dbl> 0.0, 0.0, 15.4, 4.8, 6.0, 11.6, 11.6, 0.6, 0.2, 0.6, 0.0~
$ arf7 <dbl> 5.0857143, 2.4000000, 2.4000000, 0.7142857, 0.8571429, 1~
$ arf14 <dbl> 2.8142857, 1.7428571, 1.8000000, 1.6714286, 1.5714286, 1~
$ arf28 <dbl> 1.9785714, 1.5357143, 1.5357143, 3.7857143, 1.9000000, 1~
$ arf60 <dbl> 2.3033333, 1.7966667, 1.7966667, 4.0000000, 2.5333333, 1~
$ arf90 <dbl> 1.2566667, 1.0150000, 1.0150000, 2.9600000, 2.1783333, 1~
$ arf180 <dbl> 0.9355556, 0.8444444, 0.8444444, 2.3588889, 1.7866667, 1~
$ arf360 <dbl> 1.3644444, 1.5255556, 1.5255556, 1.7272222, 1.4716667, 1~
$ arf720 <dbl> 1.3011111, 1.5213889, 1.5213889, 1.7111111, 1.5394444, 1~
$ se <dbl> 3.8, 4.6, 14.2, 23.7, 23.8, 16.8, 18.0, 12.9, 14.7, 12.9~
$ ase7 <dbl> 18.02857, 18.50000, 21.41429, 23.08571, 23.11429, 22.014~
$ ase14 <dbl> 17.03571, 17.44286, 18.03571, 19.17143, 18.45714, 18.628~
$ ase28 <dbl> 19.32857, 18.47500, 19.33929, 18.23571, 16.86071, 19.375~
$ ase60 <dbl> 20.38644, 19.99153, 20.39492, 19.90847, 19.26780, 20.449~
$ ase90 <dbl> 22.54118, 21.93193, 22.04370, 20.59328, 20.04538, 21.809~
$ ase180 <dbl> 20.79106, 19.93966, 19.99385, 19.11006, 18.66760, 19.810~
$ ase360 <dbl> 15.55153, 14.83259, 14.87883, 14.69276, 14.44318, 14.755~
$ ase720 <dbl> 15.52350, 14.75049, 14.77427, 14.53463, 14.32656, 14.540~
$ maxt <dbl> 21.3, 17.8, 15.4, 20.8, 19.8, 15.8, 19.5, 12.6, 18.8, 12~
$ amaxt7 <dbl> 22.38571, 20.44286, 22.21429, 24.21429, 23.14286, 21.671~
$ amaxt14 <dbl> 21.42857, 19.72857, 19.86429, 21.80000, 20.89286, 19.578~
$ amaxt28 <dbl> 20.71071, 19.10000, 19.18929, 19.75000, 19.05714, 18.885~
$ amaxt60 <dbl> 24.02667, 22.28000, 22.38667, 22.93167, 22.12000, 21.031~
$ amaxt90 <dbl> 27.07750, 25.77667, 25.89833, 24.93667, 23.93750, 23.164~
$ amaxt180 <dbl> 26.92000, 25.92722, 25.98500, 24.84056, 23.95389, 23.343~
$ amaxt360 <dbl> 21.55389, 20.79778, 20.81333, 20.21972, 19.99389, 19.505~
$ amaxt720 <dbl> 21.47750, 20.57222, 20.57694, 20.13153, 20.03875, 19.650~
$ mint <dbl> 9.6, 9.0, 7.3, 7.7, 8.3, 8.3, 6.1, 5.9, 7.4, 5.9, 6.9, 7~
$ amint7 <dbl> 9.042857, 7.971429, 9.171429, 10.328571, 11.200000, 10.6~
$ amint14 <dbl> 9.928571, 9.235714, 9.421429, 10.007143, 10.900000, 10.7~
$ amint28 <dbl> 8.417857, 7.560714, 7.353571, 8.671429, 9.575000, 10.060~
$ amint60 <dbl> 11.156667, 9.903333, 9.971667, 10.971667, 11.975000, 12.~
$ amint90 <dbl> 11.96667, 10.81250, 10.87833, 12.49000, 13.46167, 13.638~
$ amint180 <dbl> 11.96778, 11.01056, 11.02000, 12.41944, 13.42500, 13.695~
$ amint360 <dbl> 9.130556, 8.459722, 8.448333, 9.588611, 10.456389, 11.03~
$ amint720 <dbl> 8.854861, 8.266250, 8.254028, 9.674861, 10.517083, 10.96~
$ dist_cfa <dbl> 9442.206, 6322.438, 7957.374, 7790.785, 10692.055, 6054.~
$ dist_camp <dbl> 50966.485, 6592.893, 31767.235, 8816.272, 15339.702, 941~
$ ws <dbl> 1.263783, 1.263783, 1.456564, 5.424445, 4.219751, 4.1769~
$ aws_m0 <dbl> 2.644795, 2.644795, 2.644795, 5.008369, 3.947659, 5.2316~
$ aws_m1 <dbl> 2.559202, 2.559202, 2.559202, 5.229680, 4.027398, 4.9704~
$ aws_m3 <dbl> 2.446211, 2.446211, 2.446211, 5.386005, 3.708622, 5.3045~
$ aws_m6 <dbl> 2.144843, 2.144843, 2.144843, 5.132617, 3.389890, 5.0355~
$ aws_m12 <dbl> 2.545008, 2.545008, 2.548953, 5.045297, 3.698736, 5.2341~

```

```

$ aws_m24      <dbl> 2.580671, 2.580671, 2.584047, 5.081100, 3.745286, 5.2522~
$ dist_road    <dbl> 498.75145, 102.22032, 1217.22446, 281.69151, 215.56176, ~
$ log_dist_cfa <dbl> 9.152945, 8.751860, 8.981854, 8.960697, 9.277256, 8.7084~
$ log_dist_camp <dbl> 10.838924, 8.793748, 10.366191, 9.084354, 9.638200, 9.15~
$ log_dist_road <dbl> 6.212108, 4.627130, 7.104329, 5.640813, 5.373247, 5.0047~
$ cause        <chr> "lightning", "lightning", "lightning", "lightning", "lig~

```

Purpose

The primary goal is to predict the cause of the bushfire using the weather and distance from human activity variables provided.

Source

Collated data was part of Weihao Li's Honours thesis, which is not publicly available. The hotspots data was collected from P-Tree System (2020), climate data was taken from the Australian Bureau of Meteorology using the **bomrang** package (Sparks et al., 2020), wind data from McVicar (2011) and Iowa State University (2020), vegetation data from Australian Bureau of Agricultural and Resource Economics and Sciences (2018), distance from roads calculated using OpenStreetMap contributors (2020), CFA stations from Department of Environment, Land, Water & Planning (2020a), and campsites from Department of Environment, Land, Water & Planning (2020b). The cause was predicted from training data provided by Department of Environment, Land, Water & Planning (2019).

Pre-processing

The 60 variables are too many to view with a tour, so it should be pre-processed using principal component analysis. The categorical variables of **FOR_TYPE** and **FOR_CAT** are removed. It would be possible to keep these if they are converted to dummy (binary variables).

B.3 Australian election data

Description

This is data from a study on the relationship between voting patterns and socio-demographic characteristics of Australian electorates reported in Forbes et al. (2020). These are the predictor variables upon which voting percentages are modelled. There are two years of data in **oz_election_2001** and **oz_election_2016**.

Variables

```
load("data/oz_election_2001.rda")
load("data/oz_election_2016.rda")
glimpse(oz_election_2001)
```

Purpose

The tour is used to check for multicollinearity between predictors, that might adversely affect the linear model fit.

Source

The data was compiled from Australian Electoral Commission (AEC) and the Australian 38 Bureau of Statistics (ABS). Code to construct the data, and the original data are available at <https://github.com/jforbes14/eechidna-paper>.

Pre-processing

Considerable pre-processing was done to produce these data sets. The original data was wrangled into tidy form, some variables were log transformed to reduce skewness, and a subset of variables was chosen.

B.4 Palmer penguins

```
library(palmerpenguins)
penguins <- penguins %>%
  na.omit() # 11 observations out of 344 removed
# use only vars of interest, and standardise
# them for easier interpretation
penguins_sub <- penguins %>%
  select(bill_length_mm,
         bill_depth_mm,
         flipper_length_mm,
         body_mass_g,
         species,
         sex) %>%
  mutate(across(where(is.numeric), ~ scale(.)[,1])) %>%
  rename(bl = bill_length_mm,
         bd = bill_depth_mm,
         fl = flipper_length_mm,
         bm = body_mass_g)
save(penguins_sub, file="data/penguins_sub.rda")
```

Description

This data measure four physical characteristics of three species of penguins.

Variables

Name	Description
bl	a number denoting bill length (millimeters)
bd	a number denoting bill depth (millimeters)
fl	an integer denoting flipper length (millimeters)
bm	an integer denoting body mass (grams)
species	a factor denoting penguin species (Adélie, Chinstrap and Gentoo)

Purpose

The primary goal is to find a combination of the four variables where the three species are distinct. This is also a useful data set to illustrate cluster analysis.

Source

Details of the penguins data can be found at <https://allisonhorst.github.io/palmerpenguins/>, and Horst et al. (2022) is the package source.

Pre-processing

The data is loaded from the `palmerpenguins` package. The four physical measurement variables and the species are selected, and the penguins with missing values are removed. Variables are standardised, and their names are shortened.

```
library(palmerpenguins)
penguins <- penguins %>%
  na.omit() # 11 observations out of 344 removed
# use only vars of interest, and standardise
# them for easier interpretation
penguins_sub <- penguins[,c(3:6, 1)] %>%
  mutate(across(where(is.numeric), ~ scale(.)[,1])) %>%
  rename(bl = bill_length_mm,
         bd = bill_depth_mm,
         fl = flipper_length_mm,
         bm = body_mass_g) %>%
  as.data.frame()
save(penguins_sub, file="data/penguins_sub.rda")
```

B.5 Program for International Student Assessment

Description

The `pisa` data contains plausible scores for math, reading and science of Australian and Indonesian students from the 2018 testing cycle. The plausible scores are simulated from a model fitted to the original data, to preserve privacy of the students.

Variables

	Name	Description
	CNT	country, either AUS for Australia or IDN for Indonesia
	PV1MATH-PV10MATH	plausible scores for math
	PV1READ-PV10READ	plausible scores for reading
	PV1SCIE-PV10SCIE	plausible scores for science

Purpose

Primarily this data is useful as an example for dimension reduction.

Source

The full data is available from <https://www.oecd.org/pisa/>. There are records of the student test scores, along with survey data from the students, their households and their schools.

Pre-processing

The data was reduced to country and the plausible scores, and filtered to the two countries. It may be helpful to know that the SPSS format data was used, and was read into R using the `read_sav()` function in the `haven` package.

B.6 Sketches

Description

This data is a subset of images from <https://quickdraw.withgoogle.com>. The subset was created using the quickdraw R package at <https://huizezhangsherry.github.io/quickdraw/>. It has 6 different groups: banana, boomerang, cactus, flip flops, kangaroo. Each image is 28x28 pixels. The `sketches_train` data would be used to train a classification model, and the unlabelled `sketches_test` can be used for prediction.

Variables

	Name	Description
V1-V784	grey scale 0-255	
word	what the person was asked to draw, NA in the test data	
id	unique id for each sketch	

Purpose

Primarily this data is useful as an example for supervised classification, and also dimension reduction.

Source

The full data is available from <https://quickdraw.withgoogle.com>.

Pre-processing

It is typically useful to pre-process this data into principal components. This code can also be useful for plotting one of the sketches in a recognisable form:

```
library(mulgar)
library(ggplot2)
data("sketches_train")
set.seed(77)
x <- sketches_train[sample(1:nrow(sketches_train), 1), ]
xm <- data.frame(gry=t(as.matrix(x[,1:784])),
                 x=rep(1:28, 28),
                 y=rep(28:1, rep(28, 28)))
ggplot(xm, aes(x=x, y=y, fill=gry)) +
  geom_tile() +
  scale_fill_gradientn(colors = gray.colors(256, start = 0, end = 1, rev = TRUE)) +
  ggtitle(x$word) +
  theme_void() +
  theme(legend.position="none")
```

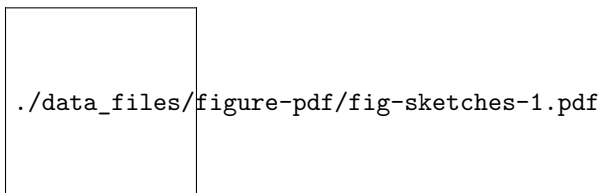


Figure B.1: One of the sketches in the subset of training data.

Appendix C

Glossary

Term	Synonyms
variable	feature, attribute
observations	cases, items, experimental units, observational units, records, statistical units, instances, examples
data set	-
response	target
predictor	independent variable, feature
similarity	correlation
dissimilarity	distance
PCA	-
LDA	-
SOM	-

Index

- Bayes Information Criterion (BIC), 99
- brushing
 - persistent, 63, 69
- classification methods
 - logistic regression, 123
 - multivariate analysis of variance (MANOVA), 119
 - neural network, 151
 - quadratic discriminant analysis (QDA), 123
 - random forest, 133
 - support vector machines (SVM), 143
- cluster analysis
 - algorithms, 77
 - confusion table, 111
 - dendrogram, 77
 - hierarchical, 77
 - intercluster distance (linkage), 77
 - interpoint distance, 63, 64, 66
 - model-based, 99, 119
 - nuisance variable, 63
 - self-organizing maps (SOM), 107
 - spin and brush, 63, 69
- parallel coordinate plot, 64
- R package
 - randomForest, 139
- tour, 69