

Michael Friendly

Visualizing Multivariate Data and Models in R

A Romance in Many Dimensions

Here is where the dedication goes ...

Table of contents

Preface	ix
ONE, TWO, MANY	ix
Flatland	x
EUREKA!	xii
Multivariate scientific discoveries	xii
What I assume	xv
Conventions used in this book	xv
I Orienting Ideas	1
1 Introduction	3
1.1 Multivariate vs. multivariable methods	3
1.2 Why use a multivariate design	3
1.3 Linear models: Univariate to multivariate	4
1.4 Visualization is harder	4
1.5 Problems in understanding and communicating MLM results	5
2 Getting Started	7
2.1 Why plot your data?	7
2.1.1 Anscombe's Quartet	7
2.1.2 One lousy point can ruin your day	10
2.1.3 Shaken, not stirred: The 1970 Draft Lottery	13
2.2 Plots for data analysis	20
2.2.1 Model plots	20
2.2.2 Diagnostic plots	21
2.2.3 Principles of graphical display	21
2.3 What have we learned?	21
II Exploratory Methods	23
3 Plots of Multivariate Data	25
3.1 Bivariate summaries	26
3.1.1 Smoothers	27
3.1.2 Stratifiers	30
3.1.3 Conditioning	33
3.2 Data Ellipses	34
3.2.1 Ellipse properties	38
3.2.2 R functions for data ellipses	41
3.2.3 Example: Penguins data	47
3.2.4 Visual thinning	50
3.3 Bagplots	51
3.4 Non-parametric bivariate density plots	53
3.5 Simpson's paradox: marginal and conditional relationships	54
3.6 (a) Ignoring species	55
3.7 (b) By species	56

3.8 (c) Within species	56
3.9 Multivariate normality and outliers	57
3.9.1 Galton data	57
3.9.2 Penguin data	60
3.10 Scatterplot matrices	63
3.10.1 Visual thinning	67
3.11 Corrrgrams	69
3.12 Generalized pairs plots	74
3.13 Parallel coordinate plots	79
3.14 Animated tours	84
3.14.1 Projections	84
3.14.2 Touring methods	89
3.15 Network diagrams	94
3.15.1 Crime data	96
3.15.2 Partial correlations	97
3.15.3 Visualizing partial correlations	98
3.16 What have we learned?	99
3.17 Exercises	101
4 Dimension Reduction	103
4.1 <i>Flatland and Spaceland</i>	103
4.1.1 Multivariate juicers	103
4.2 Principal components analysis (PCA)	105
4.2.1 PCA by springs	106
4.2.2 Mathematics and geometry of PCA	108
4.2.3 Finding principal components	113
4.2.4 Visualizing variance proportions: screeplots	115
4.2.5 Visualizing PCA scores and variable vectors	116
4.3 Biplots	121
4.3.1 Constructing a biplot	122
4.3.2 Biplots in R	123
4.3.3 Example: Crime data	123
4.3.4 Biplot contributions and quality	126
4.3.5 Supplementary variables	128
4.3.6 Example: Diabetes data	132
4.4 Nonlinear dimension reduction	134
4.4.1 Multidimensional scaling	135
4.4.2 t-SNE	138
4.5 Application: Variable ordering for data displays	142
4.6 Application: Eigenfaces	144
4.7 Elliptical insights: Outlier detection	151
4.7.1 Example: Penguin data	151
4.8 What have we learned?	153
III Univariate Linear Models	159
5 Overview of Linear models	161
5.1 The General Linear Model	163
5.1.1 Model formulas	164
5.1.2 Model matrices	168
5.1.3 Coding factors and contrasts	169
5.2 What have we learned?	176
6 Plots for univariate response models	177

6.1	The “regression quartet”	178
6.1.1	Example: Duncan’s occupational prestige	178
6.1.2	Diagnostic plots	181
6.1.3	Example: Canadian occupational prestige	182
6.2	Other Model plots	184
6.3	Coefficient displays	185
6.3.1	Displaying coefficients	186
6.3.2	Visualizing coefficients	187
6.3.3	More useful coefficient plots	189
6.4	Added-variable and related plots	192
6.4.1	Properties of AV plots	195
6.4.2	Marginal - conditional plots	196
6.4.3	Prestige data	198
6.4.4	Component + Residual plots	198
6.5	Effect displays	202
6.5.1	Prestige data	204
6.6	Outliers, leverage and influence	208
6.6.1	The leverage-influence quartet	209
6.6.2	Influence plots	216
6.6.3	Duncan data	216
6.6.4	Influence in added-variable plots	217
6.7	What have we learned?	220
7	Topics in Linear Models	221
7.1	Ellipsoids in data space and β space	221
7.1.1	Coffee, stress and heart disease	223
7.2	Measurement error	227
7.2.1	OLS is BLUE	227
7.2.2	Errors in predictors	227
7.2.3	Coffee data: β space	231
7.3	What have we learned?	232
8	Collinearity & Ridge Regression	235
8.1	What is collinearity?	235
8.1.1	Visualizing collinearity	237
8.1.2	Data space and β space	238
8.2	Measuring collinearity	241
8.2.1	Variance inflation factors	241
8.2.2	Collinearity diagnostics	243
8.3	Tableplots	245
8.4	Collinearity biplots	245
8.5	Remedies for collinearity: What can I do?	248
8.6	Ridge regression	252
8.6.1	Properties of ridge regression	252
8.6.2	The <code>genridge</code> package	255
8.7	Univariate ridge trace plots	255
8.8	Bivariate ridge trace plots	258
8.8.1	Visualizing the bias-variance tradeoff	259
8.9	Low-rank views	262
8.9.1	Biplot view	266
8.10	What have we learned?	267
IV	Multivariate Linear Models	269

9 Hotelling's T^2	271
9.1 T^2 as a generalized t -test	271
9.2 T^2 properties	272
Example: Maths score data	273
9.3 HE plot and discriminant axis	277
9.3.1 <code>heplot()</code>	277
9.4 Discriminant analysis	280
9.5 More variables	281
9.5.1 Biplots	284
9.5.2 Testing mean differences	284
9.6 Variance accounted for: Eta square (η^2)	286
9.7 What we've learned	287
9.8 Exercises	287
10 Multivariate Linear Models	289
10.1 Structure of the MLM	290
10.1.1 Assumptions	291
10.2 Fitting the model	292
10.2.1 Example: Dog food data	292
10.2.2 Sums of squares	294
10.2.3 How big is SS_H compared to SS_E ?	296
10.3 Multivariate test statistics	298
10.3.1 Testing contrasts and linear hypotheses	298
10.4 ANOVA → MANOVA	301
10.4.1 Example: Father parenting data	302
10.4.2 Ordered factors	309
10.4.3 Example: Adolescent mental health	309
10.4.4 Factorial MANOVA	314
10.5 MRA → MMRA	314
10.5.1 Example: NLSY data	315
10.5.2 Example: School data	321
10.6 Model diagnostics for MLMs	325
10.6.1 Multivariate normality of residuals	325
10.6.2 Distance plot	327
10.6.3 Multivariate influence	328
10.7 ANCOVA → MANCOVA	330
10.7.1 Example: Paired-associate tasks and academic performance	331
10.8 What we've learned	337
11 Visualizing Multivariate Models	339
11.1 HE plot framework	340
11.2 HE plot construction	341
11.2.1 MANOVA model	345
11.3 HE plots	347
11.4 Significance scaling	348
11.5 Visualizing contrasts and linear hypotheses	349
11.6 HE plot matrices	351
11.7 Low-D views: Canonical analysis	351
11.7.1 Coeficients	353
11.7.2 Canonical scores plot	354
11.7.3 Canonical HE plot	354
11.8 Factorial MANOVA	356
11.9 Quantitative predictors: MMRA	361
11.10 Canonical correlation analysis	364

11.11 MANCOVA models	370
11.12 What have we learned?	374
12 Visualizing Equality of Covariance Matrices	377
12.1 Homogeneity of Variance in Univariate ANOVA	378
12.2 Visualizing Levene's test	380
12.3 Homogeneity of variance in MANOVA	381
12.4 Box's \mathcal{M} test	385
12.5 Visualizing heterogeneity	386
12.6 Visualizing Box's \mathcal{M}	387
12.7 Low-rank views	387
12.7.1 Small dimensions can matter	389
12.8 Other measures of heterogeneity	389
12.9 Multivariate analog of Levine's test	392
12.9.1 Canonical discriminant analysis	392
12.10 What Have We Learned?	394
13 Multivariate Influence and Robust Estimation	397
13.1 Multivariate influence	398
13.1.1 Notation	398
13.1.2 Hat values and residuals	398
13.1.3 Cook's distance	398
13.1.4 Leverage and residual components	399
13.2 The Mysterious Case 9	399
13.2.1 Cook's D	401
13.2.2 DFBETAS	403
13.3 Example: NLSY data	405
13.4 Example: Penguin data	406
13.5 Robust Estimation	409
13.6 Example: Penguin data	409
14 Case studies	413
14.1 Fitting the MLM	415
14.1.1 HE plot	416
14.1.2 Canonical space	417
14.2 Social cognitive measures	420
14.2.1 Model checking	421
14.2.2 Canonical HE plot	423
V End matter	427
Colophon	429
Package versions	429
References	433

Preface

This book is about graphical methods developed recently for multivariate data, and their uses in understanding relationships when there are several aspects to be considered together. Data visualization methods for statistical analysis are well-developed for simple linear models with a single outcome variable. However, with applied research in the sciences, social and behavioral in particular, it is often the case that the phenomena of interest (e.g., depression, job satisfaction, academic achievement, childhood ADHD disorders, etc.) can be measured in several different ways or related aspects. Understanding how these different aspects are *related* contributes to our knowledge of the general phenomenon.

For example, if academic achievement can be measured for adolescents by reading, mathematics, science and history scores, how do predictors such as parental encouragement, school environment and socioeconomic status affect all these outcomes? In a similar way? In different ways? In such cases, much more can be understood from a multivariate approach that considers the correlations among the outcomes. Yet, sadly, researchers typically examine the outcomes one by one which often only tells part of the data story.

However, to do this it is useful to set the stage for multivariate thinking, with a grand scheme for statistics and data visualization, a parable, and an example of multivariate discovery.

ONE, TWO, MANY

There is an old and helpful idea I learned from John Hartigan in my graduate days at Princeton:

In statistics and data visualization *all* methods can be classified by the number of dimensions contemplated, on a scale of **ONE, TWO, MANY**.

By this, he meant that, at a global level, all data, statistical summaries, and graphical displays could be classified as:

- **univariate**: a single variable, considered in isolation (age, COVID cases, pizzas ordered). Univariate numerical summaries are means, medians, measures of variability, and so forth. Univariate displays include dot plots, boxplots, histograms and density estimates.
- **bivariate**: two variables, considered jointly. Numerical summaries include correlations, covariances and two-way tables of frequencies or measures of association for categorical variables. Bivariate displays include scatterplots and mosaic plots.
- **multivariate**: three or more variables, considered jointly. Numerical summaries include correlation and covariance matrices, consisting of all pairwise values, but also derived measures from the analysis of these matrices (eigenvalues, eigenvectors). Graphical displays of multivariate data can sometimes be shown in 3D, but often involve multiple views of the data projected into 2D plots.

As a quasi-numerical scale, I refer to these as **1D**, **2D** and **nD**. This admits the possibility of half-integer cases, such as **1.5D**, where the main focus is on a single variable, but that is classified by a simple factor (e.g., gender), or **2.5D** where a 2D scatterplot can show other variables using color, shape or other visual attributes. His point in this classification was that once you've reached three variables, all higher dimensions involve similar summaries and data displays.

Univariate and bivariate methods and displays are well-known. This book is about how these ideas can be

extended to an n -dimensional world. Three-dimensional data displays are now fairly easy to produce, even if they are sometimes difficult to understand. But how can we even think about four or more dimensions? The difficulty can be appreciated by considering the tale of *Flatland*.

Flatland

To comport oneself with perfect propriety in Polygonal society, one ought to be a Polygon oneself. —
Edwin A. Abbott, *Flatland*

In 1884, an English schoolmaster, Edwin Abbott Abbott, shook the world of Victorian culture with a slim volume, *Flatland: A Romance of Many Dimensions* (Abbott, 1884). He described a two-dimensional world, *Flatland*, inhabited entirely by geometric figures in the plane. His purpose was satirical, to poke fun at the social and gender class system at the time: Women were mere line segments, while men were represented as polygons with varying numbers of sides—a triangle was a working man, but acute isosceles were soldiers or criminals of very small angle; gentlemen and professionals had more sides. Abbott published this under the pseudonym, “A Square”, suggesting his place in the hierarchy.

True, said the Sphere; it appears to you a Plane, because you are not accustomed to light and shade and perspective; just as in Flatland a Hexagon would appear a Straight Line to one who has not the Art of Sight Recognition. But in reality it is a Solid, as you shall learn by the sense of Feeling. —
Edwin A. Abbott, *Flatland*

But how did it *feel* to be a member of a flatland society? How could a point (a newborn child?) understand a line (a woman)? How does a Triangle “see” a Hexagon or even a infinitely-sided Circle? Abbott introduces the very idea of different dimensions of existence through dreams and visions:

- A Square dreams of visiting a one-dimensional *Lineland* where men appear as lines, and women are merely “illustrious points”, but the inhabitants can only see the Square as lines.
- In a vision, the Square is visited by a Sphere, to illustrate what a 2D Flatlander could understand from a 3D sphere (Figure 1) that passes through the plane he inhabits. It is a large circle when seen at the moment of its’ greatest extent. As the Sphere rises, it becomes progressively smaller, until it becomes a point, and then vanishes.

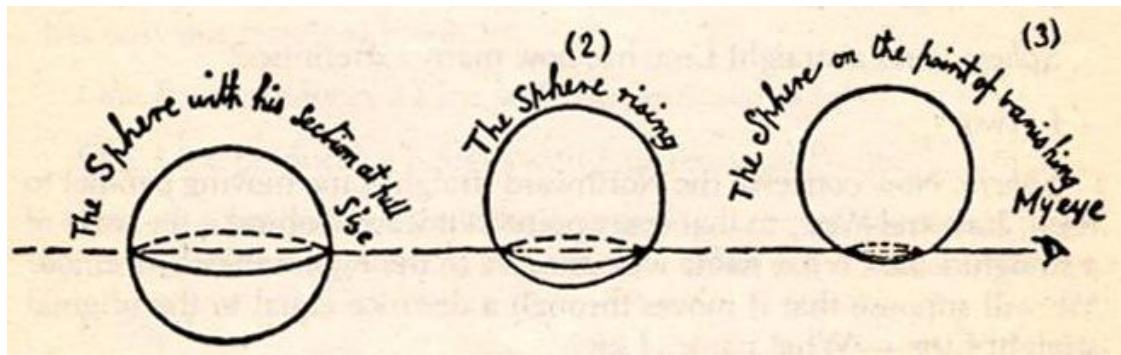


Figure 1: A 2D Flatlander seeing a sphere as it passes through Flatland. The line, labeled ‘My Eye’ indicates what the Flatlander would see. Source: Abbott (1884)

Abbott goes on to state what could be considered as a demonstration (or proof) by induction of the difficulties of seeing in 1, 2, 3 dimensions, and how the idea motion over time (one more dimension) could allow citizens of any 1D, 2D, 3D world to contemplate one more dimension.

In One Dimensions, did not a moving Point produce a Line with two terminal points? In two

Dimensions, did not a moving Line produce a Square with four terminal points? In Three Dimensions, did not a moving Square produce - did not the eyes of mine behold it - that blessed being, a Cube, with eight terminal points? And in Four Dimensions, shall not a moving Cube - alas, for Analogy, and alas for the Progress of Truth if it be not so - shall not, I say the motion of a divine Cube result in a still more divine organization with sixteen terminal points? — Edwin A. Abbott

For Abbot, the way for a citizen of any world to imagine one more dimension was to consider how a higher-dimensional object would change over time.¹ A line moved over time could produce a rectangle as shown in Figure 2; that rectangle moving in another direction over time would produce a 3D figure, and so forth.

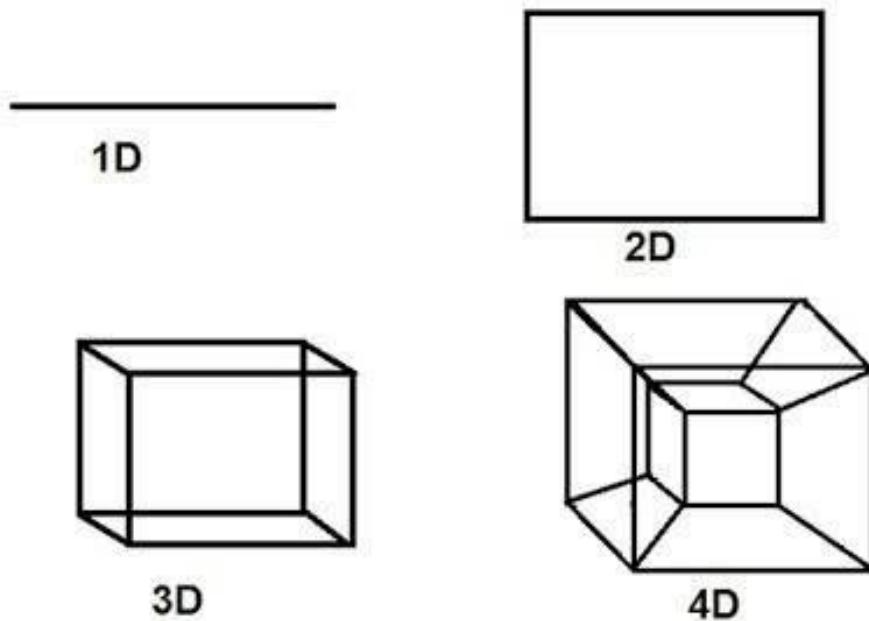


Figure 2: Geometrical objects in 1 to 4 dimensions. One more dimension can be thought of as the trace of movement over time.

But wait! Where does that 4D thing (a *tesseract*) come from? To really see a tesseract it helps to view it in an animation over time ([?@fig-tesseract](#)). But like the Square, contemplating 3D from a 2D world, it takes some imagination.

Yet the deep mathematics of more than three dimensions only emerged in the 19th century. In Newtonian mechanics, space and time were always considered independent of each other. Our familiar three-dimensional space, of length, width, and height had formed the backbone of Euclidean geometry for millenia. However, the idea that space and time are indeed interwoven was first proposed by German mathematician Hermann Minkowski (1864–1909) in 1908. This was a powerful idea. It bore fruit when Albert Einstein revolutionized the Newtonian conceptions of gravity in 1915 when he presented a theory of general relativity which was based primarily on the fact that mass and energy warp the fabric of four-dimensional spacetime.

The parable of *Flatland* can provide inspiration for statistical thinking and data visualization. Once we go beyond bivariate statistics and 2D plots, we are in a multivariate world of possibly MANY dimensions. It takes only some imagination and suitable methods to get there.

¹In his famous TV series, *Cosmos*, Carl Sagan provides [an intriguing video presentation](#) Flatland and the 4th dimension. However, as far back as 1754 ([Cajori, 1926](#)), the idea of adding a fourth dimension appears in Jean le Rond d'Alembert's "Dimensions", and one realization of a four-dimensional object is a *tesseract*, shown in Figure 2.

Like Abbott's *Flatland*, this book is a romance, in many dimensions, of what we can learn from modern methods of data visualization.

EUREKA!

Even modest sized multivariate data can have secrets that can be revealed in the right view. As an example, David Coleman at RCA Laboratories in Princeton, N.J. generated a dataset of five (fictitious) measurements of grains of pollen for the 1986 Data Exposition at the Joint statistical Meetings. The first three variables are the lengths of geometric features 3848 observed sampled pollen grains – in the x, y, and z dimensions: a `ridge` along x, a `nub` in the y direction, and a `crack` in along the z dimension. The fourth variable is pollen grain `weight`, and the fifth is `density`. The challenge was to “find something interesting” in this dataset, now available as `pollen`.

Those who solved the puzzle were able to find an orientation of this 5-dimensional dataset, such that zooming in revealed a magic word, “EUREKA” spelled in points, as in the following figure.

The path to finding the hidden word can be seen better in a 3D animation. The online version of the book uses The `rgl` package ([Adler & Murdoch, 2023](#)) create a 3D scatterplot of the first three variables. Then the `animation` package ([Xie, 2021](#)) is used to record a sequence of images, adjusting the `rgl::par3d(zoom)` value.

Multivariate scientific discoveries

Lest this example seem contrived (which it admittedly is), multivariate visualization has played an important role in quite a few scientific discoveries. Among these, Francis Galton's ([1863](#)) discovery of the anti-cyclonic pattern of wind direction in relation to barometric pressure from many weather measures recorded systematically across all weather stations, lighthouses and observatories in Europe in December 1861 stands out as the best example of a scientific discovery achieved almost entirely through graphical means— something that was totally unexpected, and purely the product of his use of remarkably novel high-dimensional graphs ([Friendly & Wainer, 2021, pp. 170–173](#)).

A more recent example is the discovery of two general classes in the development of Type 2 diabetes by Reaven & Miller ([1979](#)), using PRIM-9 ([Fishkeller et al., 1974](#)), the first computer system for high-dimensional visualization². In an earlier study Reaven & Miller ([1968](#)) examined the relation between blood glucose levels and the production of insulin in normal subjects and in patients with varying degrees of hyperglycemia (elevated blood sugar level). They found a peculiar ‘horse shoe’ shape in this relation (shown in Figure 4), about which they could only speculate: perhaps individuals with the best glucose tolerance also had the lowest levels of insulin as a response to an oral dose of glucose; perhaps those with low glucose response could secrete higher levels of insulin; perhaps those who were low on both glucose and insulin responses followed some other mechanism. In 2D plots, this was a mystery.

```
data(Diabetes, package="heplots")
plot(instest ~ glutest, data=Diabetes,
      pch=16,
      cex.lab=1.25,
      xlab="Glucose response",
      ylab="Insulin response")
```

An answer to their questions came ten years later, when they were able to visualize similar but new data

²PRIM-9 is an acronym for Picturing, Rotation, Isolation and Masking in up to 9 dimensions. These operations are fundamental to interactive and dynamic data visualization.

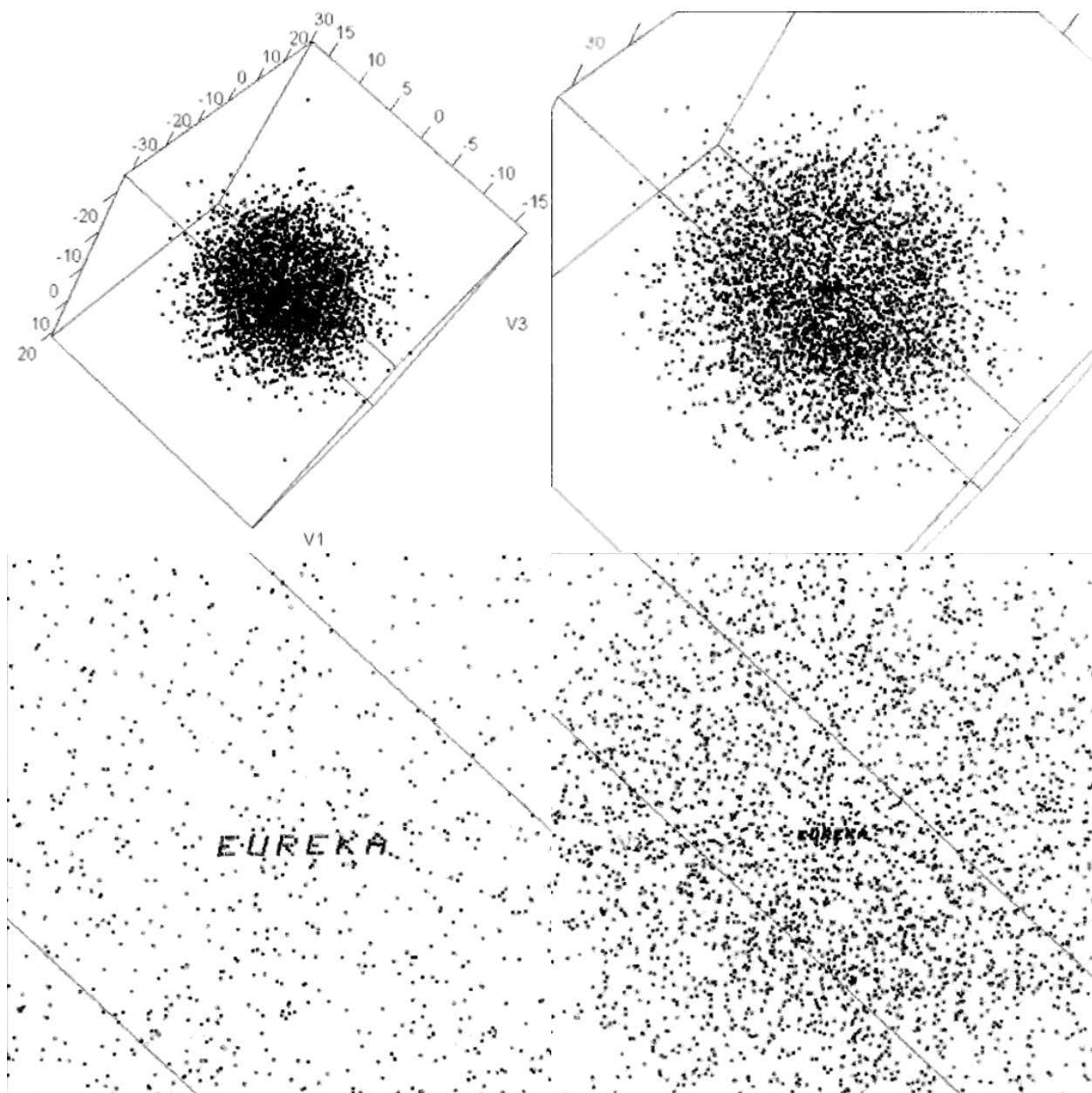


Figure 3: Four views of the pollen data, zooming in, clockwise from the upper left to discover the word “EUREKA”.

in 3D using the PRIM-9 system. In a carefully controlled study, they also measured “steady state plasma glucose” (SSPG), a measure of the efficiency of use of insulin in the body, where large values mean insulin resistance, as well as other variables. PRIM-9 allowed them to explore various sets of three variables, and, more importantly, to rotate a given plot in three dimensions to search for interesting features. One plot that stood out concerned the relation between plasma glucose response, plasma insulin response and SSPG response, shown in Figure 5.

From this graphical insight, they were able to classify the participants into three groups, based on clinical levels of glucose and insulin. The people in the wing on the left in Figure 5 were considered to have overt diabetes, the most advanced form, characterized by elevated fasting blood glucose concentration and classical

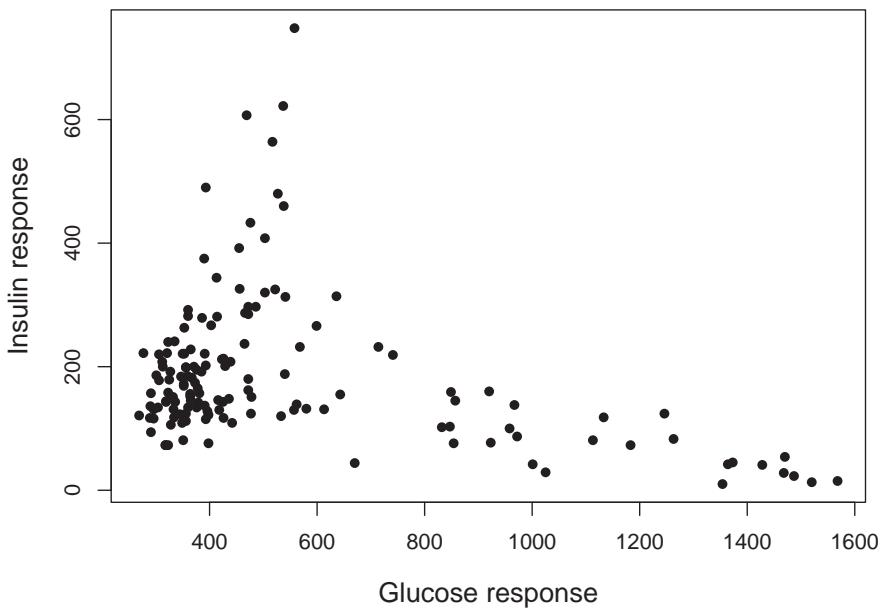


Figure 4: Reproduction of a graph similar to that from Reaven & Miller (1968) on the relationship between glucose and insulin response to being given an oral dose of glucose.

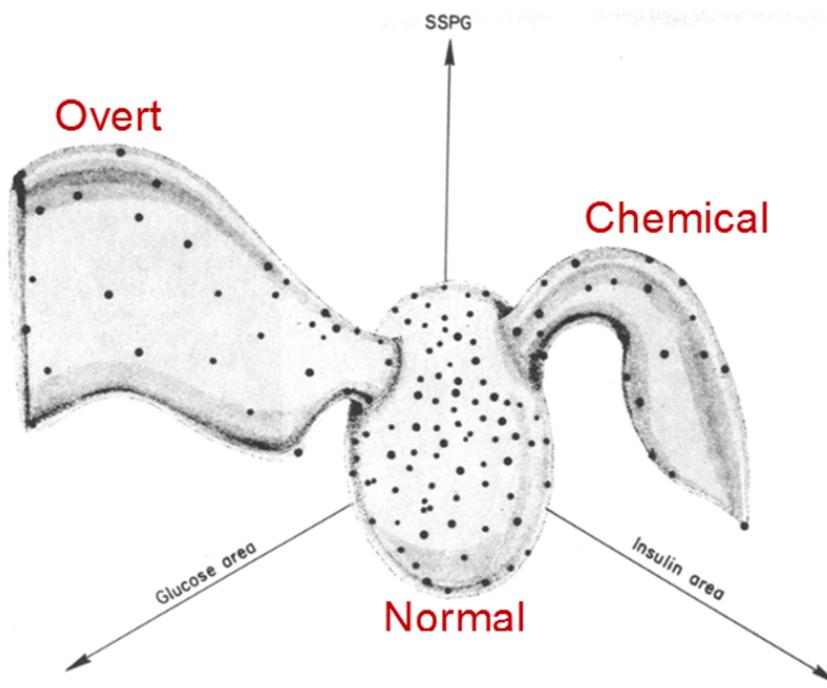


Figure 5: Artist's rendition of data from Reaven & Miller (1979) as seen in three dimensions using the PRIM-9 system. Labels for the clusters have been added, identifying the three groups of patients. *Source:* Reaven & Miller (1979).

diabetic symptoms. Those in the right wing were classified as latent or chemical diabetics, with no symptoms of diabetes but demonstrable abnormality of oral or intravenous glucose tolerance. Those in the central blob were classified as normal.

Previous thinking was that Type 2 diabetes (when the body cannot make *enough* insulin, as opposed to Type I, an autoimmune condition where the pancreatic cells have been destroyed) progressed from the chemical stage to an overt one in a smooth transition. However, it was clear from Figure 5 that the only “path” from one to the other lead through the cluster of normal patients near the origin, so that explanation must be wrong. Instead, this suggested that the chemical and overt diabetics were distinct classes. Indeed, longitudinal studies showed that patients classified as chemical diabetics rarely developed the overt form. The understanding of the etiology of Type 2 diabetes was altered dramatically by the power of high-D interactive graphics.

What I assume

It is assumed that the reader has a background in applied *intermediate* statistics including material on univariate linear models including analysis of variance (ANOVA) and multiple regression. This means you should be familiar with ... **TODO:** Complete this required background

There will also be some mathematics in the book where words and diagrams are not enough. The mathematical level will be intermediate, mostly consisting of simple algebra. No derivations, proofs, theorems here! For multivariate methods, it will be useful to express ideas using matrix notation to simplify presentation. The single symbol I'm using math to express ideas, and all you will need is a reading-level of understanding. For this, the first chapter of Fox (2021), *A mathematical primer for social statistics*, is excellent. If you want to learn something of using matrix algebra for data analysis and statistics, I recommend our package **matlib** (Friendly et al., 2024) .

I also assume the reader to have at least a basic familiarity with R. While R fundamentals are outside the scope of the book, I believe that this language provides a rich set of resources, far beyond that offered by other statistical software packages, and is well worth learning.

For those not familiar with R, I recommend Matloff (2011), Wickham (2014), and Cotton (2013) for introductions to programming in the language. Fox & Weisberg (2018a) and Teator (2011) are great for learning about how to conduct basic statistical analyses in R. **TODO:** Revise this list.

TODO: Add stuff on general books about graphics

Conventions used in this book

The following typographic conventions are used in this book:

- *italic* : indicates terms or phrases to be *emphasized* or defined in the text; **bold** : is used for terms to be given **strong emphasis**, particularly for their first mention.
- Package names are printed in **bold** and colored **brown**, for example **ggplot2** , **lattice** and the **matlib** . These uses generate citations like **ggplot2** (Wickham et al., 2023) . They are automatically indexed, individually and under a “Packages” heading.
- Datasets are rendered as their name in monospaced font, like **Prestige** or indicating the package from which they come, as in **carData::Prestige** . These too are automatically indexed.
- A monospaced **typewriter** font is used in the text to refer to variable and function names, like **education** and **plot()**; also for R statement elements and keywords as well.
- R code in program listings and output is presented in a monospaced (**typewriter**) font, **fira mono**

- For R functions in packages, I use the notation `package::function()`, for example: `car::Anova()`, to identify the package where those functions are defined.

Part I

Orienting Ideas

1

Introduction

This material may or may not survive; it was taken from an earlier article.

1.1 Multivariate vs. multivariable methods

multivariate \neq multivariable

In this era of multivitamins, multitools, multifactor authentication and even the multiverse, it is well to understand the distinction between *multivariate* and *multivariable* methods as these terms are generally used and as I use them here in relation to statistical methods and data visualization. The distinction is simple:

- **Multivariate methods** for linear models such as multivariate regression have more than one dependent, response or outcome variable. Other multivariate methods such as principal components analysis or factor analysis treat all variables on an equal footing.
 - **Multivariable methods** have a single dependent variable and more than one independent variables or covariates.
-

1.2 Why use a multivariate design

A particular research outcome (e.g., depression, neuro-cognitive functioning, academic achievement, self-concept, attention deficit hyperactivity disorders) might take on a multivariate form if it has several observed measurement scales or related aspects by which it is quantified, or if there are multiple theoretically distinct outcomes that should be assessed in conjunction with each other (e.g., using depression, generalized anxiety, and stress inventories to model overall happiness). In this situation, the primary concern of the researcher is to ascertain the impact of potential predictors on two or more response variables simultaneously.

For example, if academic achievement is measured for adolescents by their reading, mathematics, science, and history scores, the following questions are of interest:

- Do predictors such as parent encouragement, socioeconomic status and school environmental variables affect *all* of these outcomes?
- Do they affect them in the *same* or *different* ways?
- How many different aspects of academic achievement can be distinguished in the predictors? Equivalently, is academic achievement *unidimensional* or *multidimensional* in relation to the predictors?

Similarly, if psychiatric patients in various diagnostic categories are measured on a battery of tests related to social skills and cognitive functioning, we might want to know:

- Which measures best discriminate among the diagnostic groups?
- Which measures are most predictive of positive outcomes?

- Further, how are the *relationships* between the outcomes affected by the predictors?

Such questions obviously concern more than just the separate univariate relations of each response to the predictors. Equally, or perhaps more importantly, are questions of how the response variables are predicted *jointly*.

i SEM

Structural equation modeling (SEM) offers another route to explore and analyze the relationships among multiple predictors and multiple responses. They have the advantage of being able to test potentially complex systems of linear equations in very flexible ways; however, these methods are often far removed from data analysis *per se* and except for path diagrams offer little in the way of visualization methods to aid in understanding and communicating the results. The graphical methods we describe here can also be useful in a SEM context.

1.3 Linear models: Univariate to multivariate

For classical linear models for ANOVA and regression, the step from a univariate model for a single response, y , to a multivariate one for a collection of p responses, \mathbf{y} is conceptually very easy. That's because the univariate model,

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q + \epsilon_i,$$

or, in matrix terms,

$$\mathbf{y} = \mathbf{X} \boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \text{with } \mathbf{u} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}),$$

generalizes directly to an analogous multivariate linear model (MLM),

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_p] = \mathbf{X} \mathbf{B} + \boldsymbol{\varepsilon} \quad \text{with } \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

for multiple responses (as will be discussed in detail). The design matrix, \mathbf{X} remains the same, and the vector $\boldsymbol{\beta}$ of coefficients becomes a matrix \mathbf{B} , with one column for each of the p outcome variables.

Happily as well, hypothesis tests for the MLM are also straight-forward generalizations of the familiar F and t -tests for univariate response models. Moreover, there is a rich geometry underlying these generalizations which we can exploit for understanding and visualization.

1.4 Visualization is harder

However, with two or more response variables, visualizations for multivariate models are not as simple as they are for their univariate counterparts for understanding the effects of predictors, model parameters, or model diagnostics. Consequently, the results of such studies are often explored and discussed solely in terms of coefficients and significance, and visualizations of the relationships are only provided for one response variable at a time, if at all. This tradition can mask important nuances, and lead researchers to draw erroneous conclusions.

The aim of this book is to describe and illustrate some central methods that we have developed over the

last ten years that aid in the understanding and communication of the results of multivariate linear models (Friendly, 2007; Friendly & Meyer, 2016). These methods rely on *data ellipsoids* as simple, minimally sufficient visualizations of variance that can be shown in 2D and 3D plots. As will be demonstrated, the *Hypothesis-Error (HE) plot* framework applies this idea to the results of multivariate tests of linear hypotheses.

Further, in the case where there are more than just a few outcome variables, the important nectar of their relationships to predictors can often be distilled in a multivariate juicer—a **projection** of the multivariate relationships to the predictors in the low-D space that captures most of the flavor. This idea can be applied using *canonical correlation plots* and with *canonical discriminant HE plots*.

1.5 Problems in understanding and communicating MLM results

In my consulting practice within the Statistical Consulting Service at York University, I see hundreds of clients each year ranging from advanced undergraduate thesis students, to graduate students and faculty from a variety of fields. Over the last two decades, and across each of these groups, I have noticed an increasing desire to utilize multivariate methods. As researchers are exposed to the utility and power of multivariate tests, they see them as an appealing alternative to running many univariate ANOVAs or multiple regressions for each response variable separately.

However, multivariate analyses are more complicated than such approaches, especially when it comes to understanding and communicating results. Output is typically voluminous, and researchers will often get lost in the numbers. While software (SPSS, SAS and R) make tabular summary displays easy, these often obscure the findings that researchers are most interested in. The most common analytic oversights that we have observed are:

- **Atomistic data screening:** Researchers have mostly learned the assumptions (the Holy Trinity of normality, constant variance and independence) of univariate linear models, but then apply *univariate* tests (e.g., Shapiro-Wilk) and diagnostic plots (normal QQ plots) to every predictor and every response.
- **Bonferroni everywhere:** Faced with the task of reporting the results for multiple response measures and a collection of predictors for each, a common tendency is to run (and sometimes report) each of the separate univariate response models and then apply a correction for multiple testing. Not only is this confusing and awkward to report, but it is largely unnecessary because the multivariate tests provide protection for multiple testing.
- **Reverting to univariate visualizations:** To display results, SPSS and SAS make some visualization methods available through menu choices or syntax, but usually these are the wrong (or at least unhelpful) choices, in that they generate separate univariate graphs for the individual responses.

This book to discusses a few essential procedures for multivariate linear models, how their interpretation can be aided through the use of well-crafted (though novel) visualizations, and provides replicable sample code in R to showcase their use in applied behavioral research. A later section [ref?] provides some practical guidelines for analyzing, visualizing and reporting such models to help avoid these and other problems.

Package summary:

1 packages used here: knitr

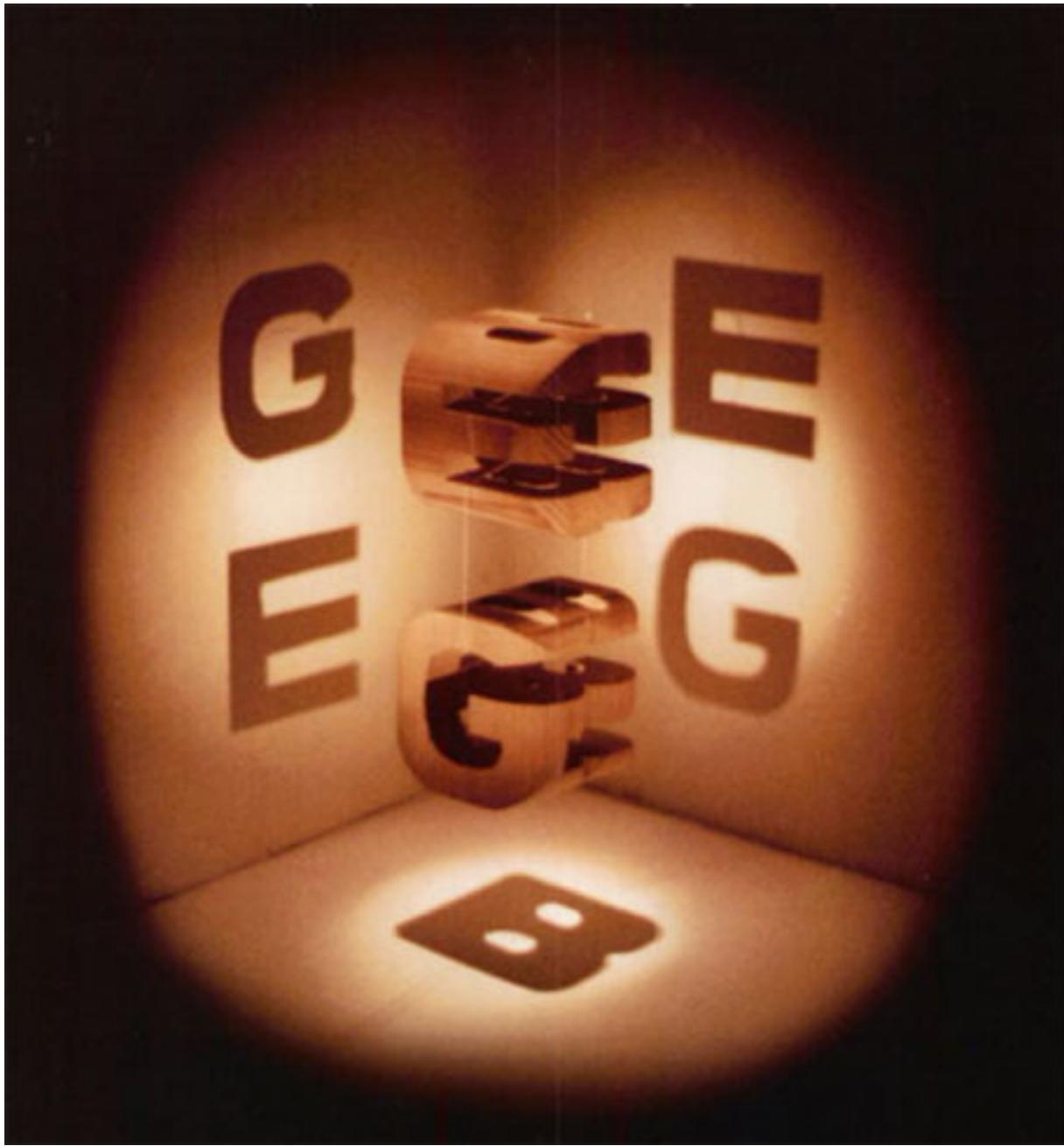


Figure 1.1: Projection: The cover image from Hofstadter's *Gödel, Bach and Escher* illustrates projection of 3D solids onto each 2D plane.

2

Getting Started

2.1 Why plot your data?

Getting information from a table is like extracting sunlight from a cucumber. Farquhar & Farquhar (1891)

At the time the Farquhar brothers wrote this pithy aphorism, graphical methods for understanding data had advanced considerably, but were not universally practiced, prompting their complaint. Most data tables we see are designed for looking something up like the number of measles cases in Ontario compared to other provinces. They are not usually designed to reveal *patterns, trends or anomalies*, although this can be accomplished with modern table generating software such as the `tinytable` or `gt` packages.¹

The main graphic forms we use today—the pie chart, line graphs and bar—were invented by William Playfair around 1800 (Playfair, 1786, 1801). The scatterplot arrived shortly after (Herschel, 1833) and thematic maps showing the spatial distributions of social variables (crime, suicides, literacy) were used for the first time to reason about important societal questions (Guerry, 1833) such as “is increased education associated with lower rates of crime?”

In the last half of the 18th Century, the idea of correlation was developed (Galton, 1886; Pearson, 1896) and the period, roughly 1860–1890, dubbed the “Golden Age of Graphics (Funkhouser, 1937) became the richest period of innovation and beauty in the entire history of data visualization. During this time there was an incredible development of *visual thinking*, represented by the work of Charles Joseph Minard, advances in the role of visualization within scientific discovery, as illustrated through Francis Galton, and graphical excellence, embodied in state statistical atlases produced in France and elsewhere. See Friendly (2008); Friendly & Wainer (2021) for this history. .

This chapter introduces the importance of graphing data through three nearly classic stories with the following themes:

- **summary statistics are not enough:** Anscombe’s Quartet demonstrates datasets that are indistinguishable by numerical summary statistics (mean, standard deviation, correlation), but whose relationships are vastly different.
- **one lousy point can ruin your day:** A researcher is mystified by a difference between a correlation for men and women until she plots the data.
- **finding the signal in noise:** The story of the US 1970 Draft Lottery shows how a weak, but reliable signal, reflecting bias in a process can be revealed by graphical enhancement and summarization.

2.1.1 Anscombe’s Quartet

In 1973, Francis Anscombe (Anscombe, 1973) famously constructed a set of four datasets illustrate the importance of plotting the graphs before analyzing and model building, and the effect of unusual observations

¹For example, a cell in a table can be used to show a “sparklines” (Tufte (1983)), tiny versions of a line graph or bar chart. As well, table rows and/or columns can be sorted to show trends or background colors can be used to show unusual values.

on fitted models. Now known as *Anscombe's Quartet*, these datasets had identical statistical properties: the same means, standard deviations, correlations and regression lines.

His purpose was to debunk three notions that had been prevalent at the time:

- Numerical calculations are exact, but graphs are coarse and limited by perception and resolution;
- For any particular kind of statistical data there is just one set of calculations constituting a correct statistical analysis;
- Performing intricate calculations is virtuous, whereas actually looking at the data is cheating.

The dataset `datasets::anscombe` has 11 observations, recorded in wide format, with variables `x1:x4` and `y1:y4`.

```
data(anscombe)
head(anscombe)
#>   x1 x2 x3 x4   y1   y2   y3   y4
#> 1 10 10 10   8 8.04 9.14 7.46 6.58
#> 2  8   8   8   8 6.95 8.14 6.77 5.76
#> 3 13 13 13   8 7.58 8.74 12.74 7.71
#> 4  9   9   9   8 8.81 8.77 7.11 8.84
#> 5 11 11 11   8 8.33 9.26 7.81 8.47
#> 6 14 14 14   8 9.96 8.10 8.84 7.04
```

The following code transforms this data to long format and calculates some summary statistics for each dataset.

```
anscombe_long <- anscombe |>
  pivot_longer(everything(),
    names_to = c(".value", "dataset"),
    names_pattern = "(.)(.)"
  ) |>
  arrange(dataset)

anscombe_long |>
  group_by(dataset) |>
  summarise(xbar      = mean(x),
            ybar      = mean(y),
            r         = cor(x, y),
            intercept = coef(lm(y ~ x))[1],
            slope     = coef(lm(y ~ x))[2]
  )
#> # A tibble: 4 x 6
#>   dataset xbar  ybar      r intercept slope
#>   <chr>   <dbl> <dbl>  <dbl>     <dbl> <dbl>
#> 1 1       9    7.50  0.816    3.00  0.500
#> 2 2       9    7.50  0.816    3.00  0.5
#> 3 3       9    7.5  0.816    3.00  0.500
#> 4 4       9    7.50  0.817    3.00  0.500
```

As we can see, all four datasets have nearly identical univariate and bivariate statistical measures. You can only see how they differ in graphs, which show their true natures to be vastly different.

Figure 2.1 is an enhanced version of Anscombe's plot of these data, adding helpful annotations to show visually the underlying statistical summaries.

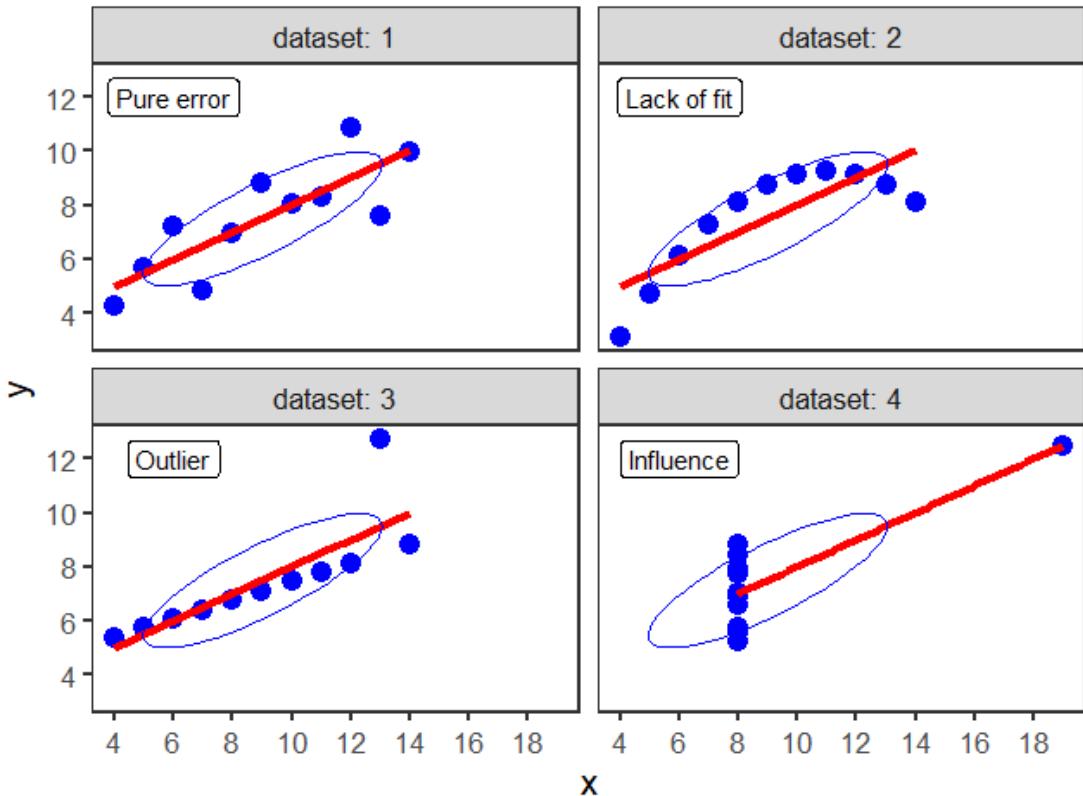


Figure 2.1: Scatterplots of Anscombe's Quartet. Each plot shows the fitted regression line and a 68% data ellipse representing the correlation between x and y .

This figure is produced as follows, using a single call to `ggplot()`, faceted by `dataset`. As we will see later (Section 3.2), the data ellipse (produced by `stat_ellipse()`) reflects the correlation between the variables.

```
desc <- tibble(
  dataset = 1:4,
  label = c("Pure error", "Lack of fit", "Outlier", "Influence")
)

ggplot(anscombe_long, aes(x = x, y = y)) +
  geom_point(color = "blue", size = 4) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE,
             color = "red", linewidth = 1.5) +
  scale_x_continuous(breaks = seq(0,20,2)) +
  scale_y_continuous(breaks = seq(0,12,2)) +
  stat_ellipse(level = 0.5, color=col, type="norm") +
  geom_label(data=desc, aes(label = label), x=6, y=12) +
  facet_wrap(~dataset, labeller = label_both)
```

The subplots are labeled with the statistical idea they reflect:

- dataset 1: **Pure error.** This is the typical case with well-behaved data. Variation of the points around the line reflect only measurement error or unreliability in the response, y .
- dataset 2: **Lack of fit.** The data is clearly curvilinear, and would be very well described by a quadratic,

`y ~ poly(x, 2)`. This violates the assumption of linear regression that the fitted model has the correct form.

- dataset 3: **Outlier**. One point, second from the right, has a very large residual. Because this point is near the extreme of x , it pulls the regression line towards it, as you can see by imagining a line through the remaining points.
- dataset 4: **Influence**. All but one of the points have the same x value. The one unusual point has sufficient influence to force the regression line to fit it **exactly**.

One moral from this example:

Linear regression only “sees” a line. It does its’ best when the data are really linear. Because the line is fit by least squares, it pulls the line toward discrepant points to minimize the sum of squared residuals.

i Datasaurus Dozen

The method Anscombe used to compose his quartet is unknown, but it turns out that there is a method to construct a wider collection of datasets with identical statistical properties. After all, in a bivariate dataset with n observations, the correlation has $(n - 2)$ degrees of freedom, so it is possible to choose $n - 2$ of the (x, y) pairs to yield any given value. As it happens, it is also possible to create any number of datasets with the same means, standard deviations and correlations with nearly any shape you like — even a dinosaur!

The *Datasaurus Dozen* was first publicized by Alberto Cairo in a [blog post](#) and are available in the [datasauRus](#) package ([Davies et al., 2022](#)) . As shown in `?@fig-datasaurus`, the sets include a star, cross, circle, bullseye, horizontal and vertical lines, and, of course the “dino”. The method ([Matejka & Fitzmaurice, 2017](#)) uses *simulated annealing*, an iterative process that perturbs the points in a scatterplot, moving them towards a given shape while keeping the statistical summaries close to the fixed target value.

The [datasauRus](#) package just contains the datasets, but a general method, called *statistical metamers*, for producing such datasets has been described by [Elio Campitelli](#) and implemented in the [metamer](#) package.

i Quartets

The essential idea of a statistical “quartet” is to illustrate four quite different datasets or circumstances that seem superficially the same, but yet are paradoxically very different when you look behind the scenes. For example, in the context of causal analysis Gelman et al. ([2023](#)), illustrated sets of four graphs, within each of which all four represent the same average (latent) causal effect but with much different patterns of individual effects; McGowan et al. ([2023](#)) provide another illustration with four seemingly identical data sets each generated by a different causal mechanism. As an example of machine learning models, Biecek et al. ([2023](#)), introduced the “Rashamon Quartet”, a synthetic dataset for which four models from different classes (linear model, regression tree, random forest, neural network) have practically identical predictive performance. In all cases, the paradox is solved when their visualization reveals the distinct ways of understanding structure in the data. The [quartets](#) package contains these and other variations on this theme.

2.1.2 One lousy point can ruin your day

In the mid 1980s, a consulting client had a strange problem.² She was conducting a study of the relation between body image and weight preoccupation in exercising and non-exercising people ([Davis, 1990](#)). As part of the design, the researcher wanted to know if self-reported weight could be taken as a reliable indicator

²This story is told apocryphally. The consulting client actually did plot the data, but needed help making better graphs.

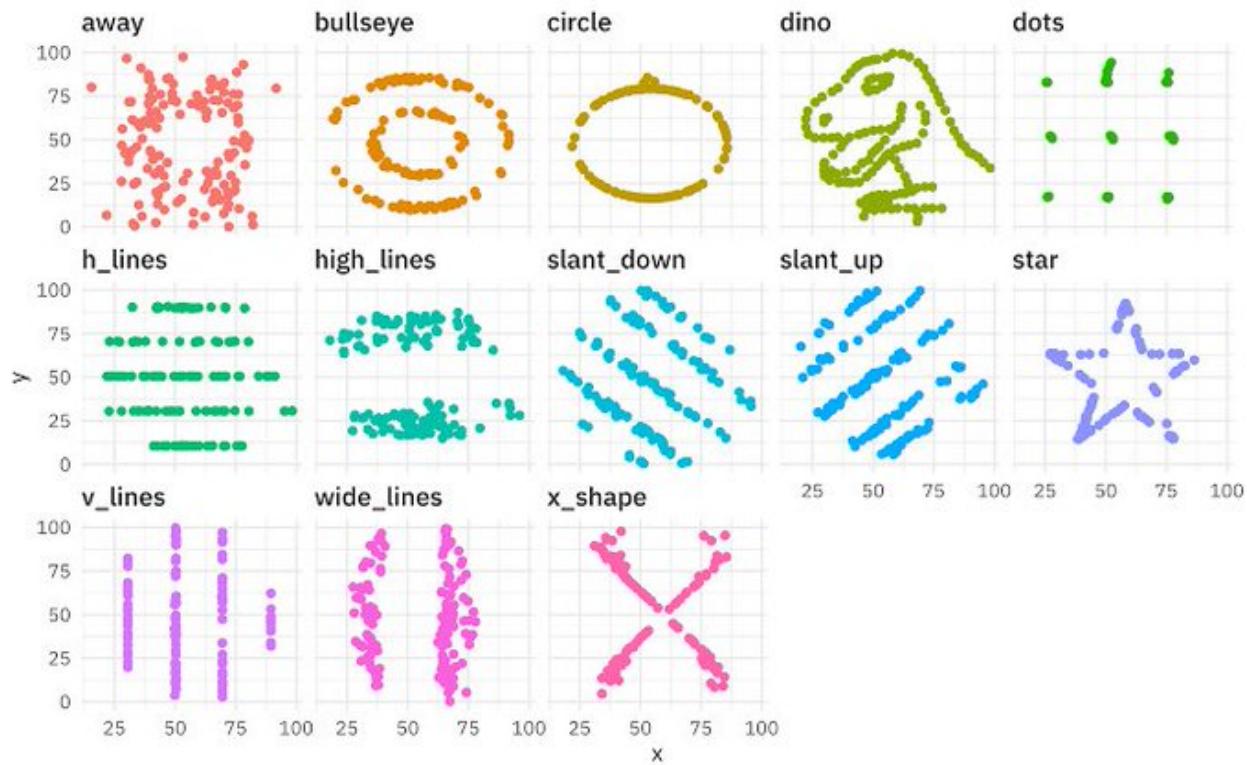


Figure 2.2: Plots of the Dinosaur Dozen datasets. Source: [Selçuk Korkmaz on X](#)

of true weight measured on a scale. It was expected that the correlations between reported and measured weight should be close to 1.0, and the slope of the regression lines for men and women should also be close to 1.0. The dataset is `carData::Davis`.

She was therefore very surprised to see the following numerical results: For men, the correlation was nearly perfect, but not so for women.

```
data(Davis, package="carData")
Davis <- Davis |>
  drop_na()          # drop missing cases
Davis |>
  group_by(sex) |>
  select(sex, weight, repwt) |>
  summarise(r = cor(weight, repwt))
#> # A tibble: 2 x 2
#>   sex      r
#>   <fct> <dbl>
#> 1 F      0.501
#> 2 M      0.979
```

Similarly, the regression lines showed the expected slope for men, but that for women was only 0.26.

```
Davis |>
  nest(data = -sex) |>
  mutate(model = map(data, ~ lm(repwt ~ weight, data = .)),
```

```

tidied = map(model, tidy)) |>
unnest(tidied) |>
filter(term == "weight") |>
select(sex, term, estimate, std.error)
#> # A tibble: 2 x 4
#>   sex    term   estimate std.error
#>   <fct> <chr>     <dbl>     <dbl>
#> 1 M      weight     0.990    0.0229
#> 2 F      weight     0.262    0.0459

```

“What could be wrong here?”, the client asked. The consultant replied with the obvious question:

Did you plot your data?

The answer turned out to be one discrepant point, a female, whose measured weight was 166 kg (366 lbs!). This single point exerted so much influence that it pulled the fitted regression line down to a slope of only 0.26.

```

# shorthand to position legend inside the figure
legend_inside <- function(position) {
  theme(legend.position = "inside",
        legend.position.inside = position)
}

Davis |>
  ggplot(aes(x = weight, y = repwt,
             color = sex, shape = sex, linetype = sex)) +
  geom_point(size = ifelse(Davis$weight==166, 6, 2)) +
  geom_smooth(method = "lm", formula = y~x, se = FALSE) +
  labs(x = "Measured weight (kg)",
       y = "Reported weight (kg)") +
  scale_linetype_manual(values = c(F = "longdash",
                                    M = "solid")) +
  legend_inside(c(.8, .8))

```

In this example, it was arguable that x and y axes should be reversed, to determine how well measured weight can be predicted from reported weight. In `ggplot` this can easily be done by reversing the `x` and `y` aesthetics.

```

Davis |>
  ggplot(aes(y = weight, x = repwt, color = sex, shape=sex)) +
  geom_point(size = ifelse(Davis$weight==166, 6, 2)) +
  labs(y = "Measured weight (kg)",
       x = "Reported weight (kg)") +
  geom_smooth(method = "lm", formula = y~x, se = FALSE) +
  legend_inside(c(.8, .8))

```

In Figure 2.4, this discrepant observation again stands out like a sore thumb, but it makes very little difference in the fitted line for females. The reason is that this point is well within the range of the x variable (`repwt`). To impact the slope of the regression line, an observation must be unusual in *both* x and y . We take up the topic of how to detect influential observations and what to do about them in Chapter 6.

The value of such plots is not only that they can reveal possible problems with an analysis, but also help identify their reasons and suggest corrective action. What went wrong here? Examination of the original

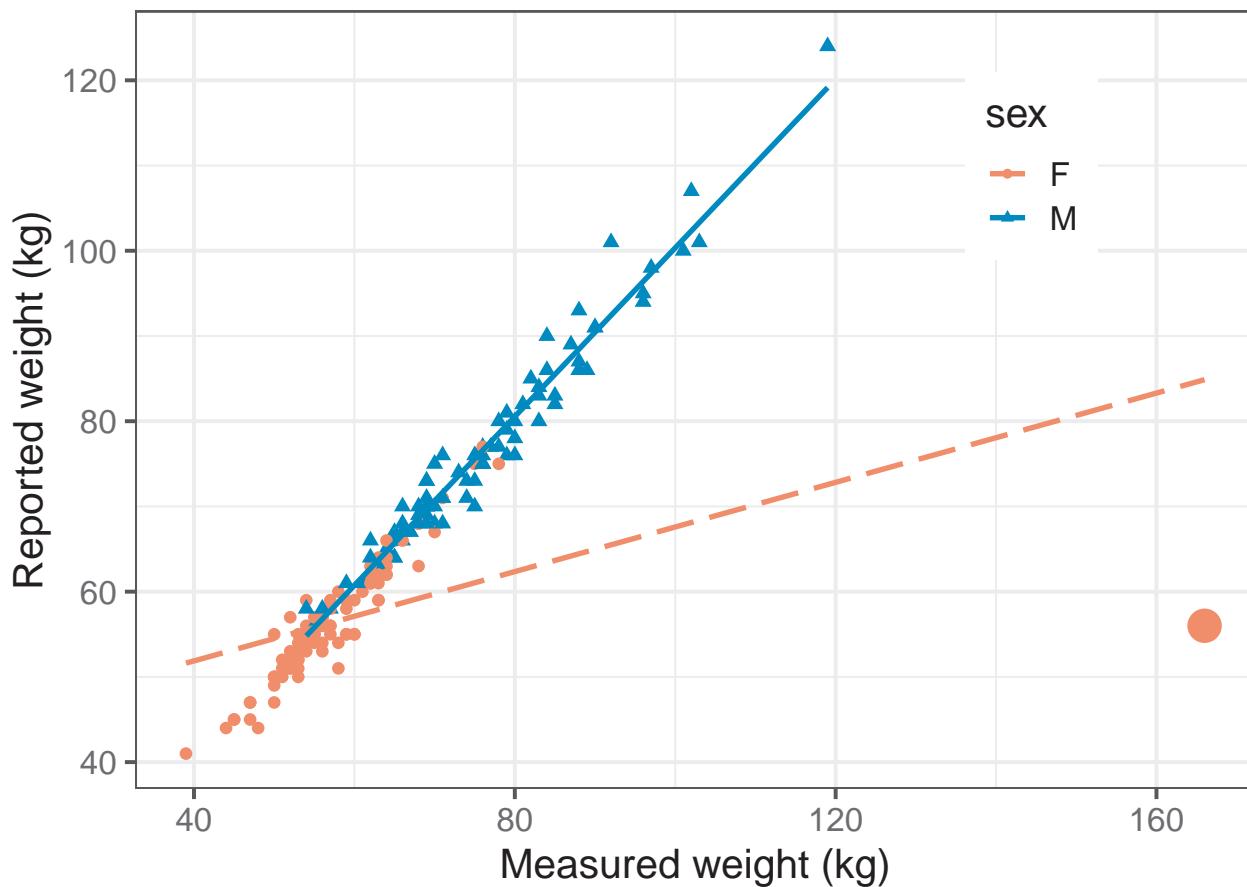


Figure 2.3: Regression for Davis' data on reported weight and measures weight for men and women. Separate regression lines, predicting reported weight from measured weight are shown for males and females. One highly unusual point is highlighted.

data showed that this person switched the values, recording her reported weight in the box for measured weight and vice versa.

2.1.3 Shaken, not stirred: The 1970 Draft Lottery

Although we often hear that data speak for themselves, their voices can be soft and sly.—Frederick Mosteller (1983), *Beginning Statistics with Data Analysis*, p. 234.

The power of graphics is particularly evident when data contains a weak signal embedded in a field of noise. To the casual glance, there may seem to be nothing going on, but the signal can be made apparent in an incisive graph.

A dramatic example of this occurred in 1969 when the U.S. military conducted a lottery, the first since World War II, to determine which young men would be called up to serve in the Vietnam War for 1970. The U.S. Selective Service had devised a system to rank eligible men according to a random drawing of their birthdays. There were 366 blue plastic capsules containing birth dates placed in a transparent glass container and drawn by hand to assign ranked order-of-call numbers to all men within the 18-26 age range.

In an attempt to make the selection process also transparent, the proceeding was covered on radio, TV and film and the dates posted in order on a large display board. The first capsule—drawn by Congressman Alexander Pirnie (R-NY) of the House Armed Services Committee—contained the date September 14, so all men born on September 14 in any year between 1944 and 1950 were assigned lottery number 1, and would be

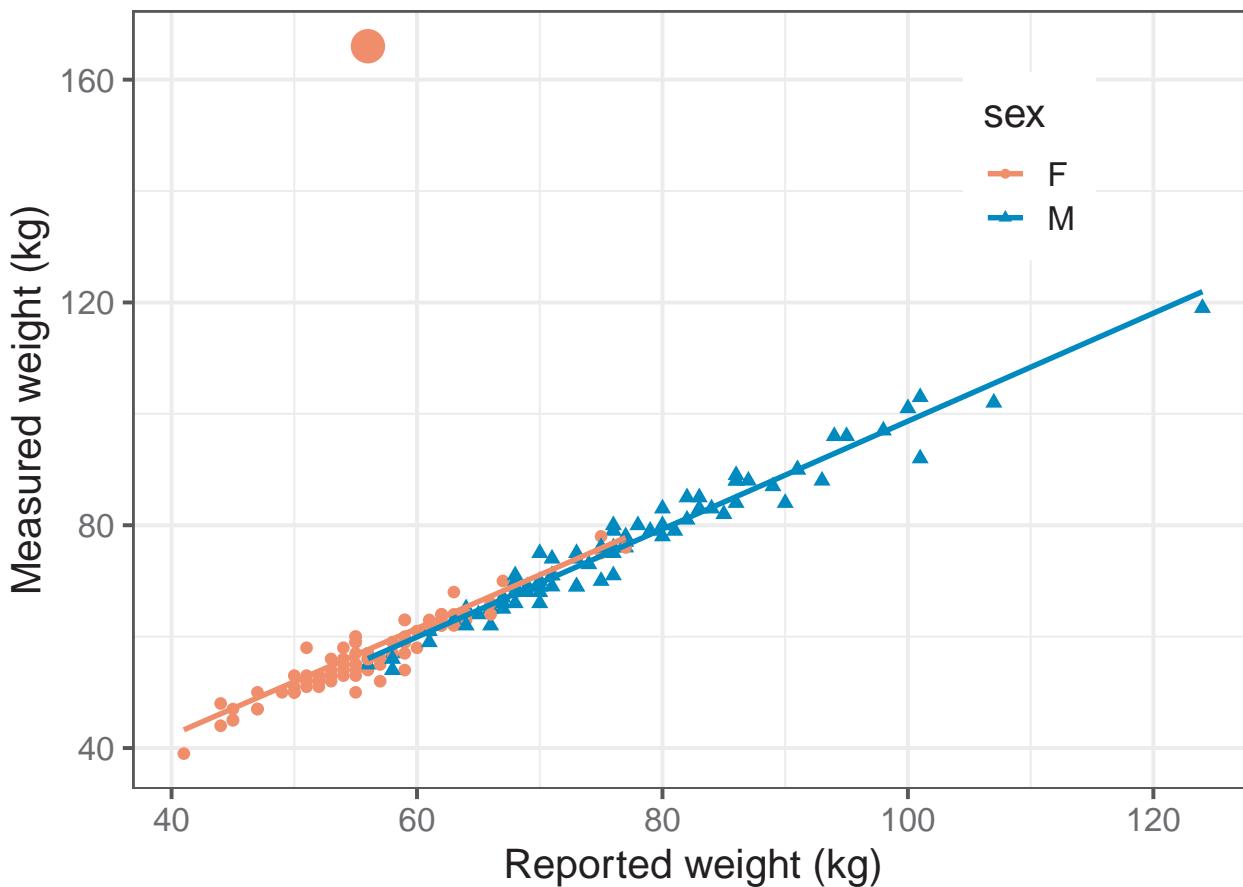


Figure 2.4: Regression for Davis' data on reported weight and measures weight for men and women. Separate regression lines, predicting measured weight from reported weight are shown for males and females. The highly unusual point no longer has an effect on the fitted lines.

drafted first. April 24 was drawn next, then December 30, February 14, and so on until June 8, selected last. At the time of the drawing, US officials stated that those with birthdays drawn in the first third would almost certainly be drafted, while those in the last third would probably avoid the draft (Fienberg, 1971).

I watched this unfold with considerable interest because I was eligible for the Draft that year. I was dismayed when my birthday, May 7, came up ranked 35. **Ugh!** Could some data analysis and graphics get me out of my funk?

The data, from the official [Selective Service listing](#) are contained in the dataset `vcdExtra::Draft1970`, ordered by `Month` and `birthdate` (`Day`), with `Rank` as the order in which the birthdates were drawn.

```
library(ggplot2)
library(dplyr)
data(Draft1970, package = "vcdExtra")
dplyr::glimpse(Draft1970)
#> Rows: 366
#> Columns: 3
#> $ Day    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
#> $ Rank   <int> 305, 159, 251, 215, 101, 224, 306, 199, 194, 325, 32~
#> $ Month  <ord> Jan, Jan, Jan, Jan, Jan, Jan, Jan, Jan, Ja~
```



Figure 2.5: Congressman Alexander Pirnie (R-NY) drawing the first capsule for the Selective Service draft, Dec 1, 1969. Source: <https://bit.ly/45c23sB>

A basic scatterplot, slightly prettified, is shown in Figure 2.6. The points are colored by month, and month labels are shown at the bottom.

```
# make markers for months at their mid points
months <- data.frame(
  month =unique(Draft1970$Month),
  mid = seq(15, 365-15, by = 30))

ggplot2:: theme_set(theme_bw(base_size = 16))
gg <- ggplot(Draft1970, aes(x = Day, y = Rank)) +
  geom_point(size = 2.5, shape = 21,
             alpha = 0.3,
             color = "black",
             aes(fill=Month)
  ) +
  scale_fill_manual(values = rainbow(12)) +
  geom_text(data=months, aes(x=mid, y=0, label=month), nudge_x = 5) +
  geom_smooth(method = "lm", formula = y ~ 1,
              col = "black", fill="grey", linetype = "dashed", alpha=0.6) +
  labs(x = "Day of the year",
       y = "Lottery rank") +
  theme(legend.position = "none")
gg
```

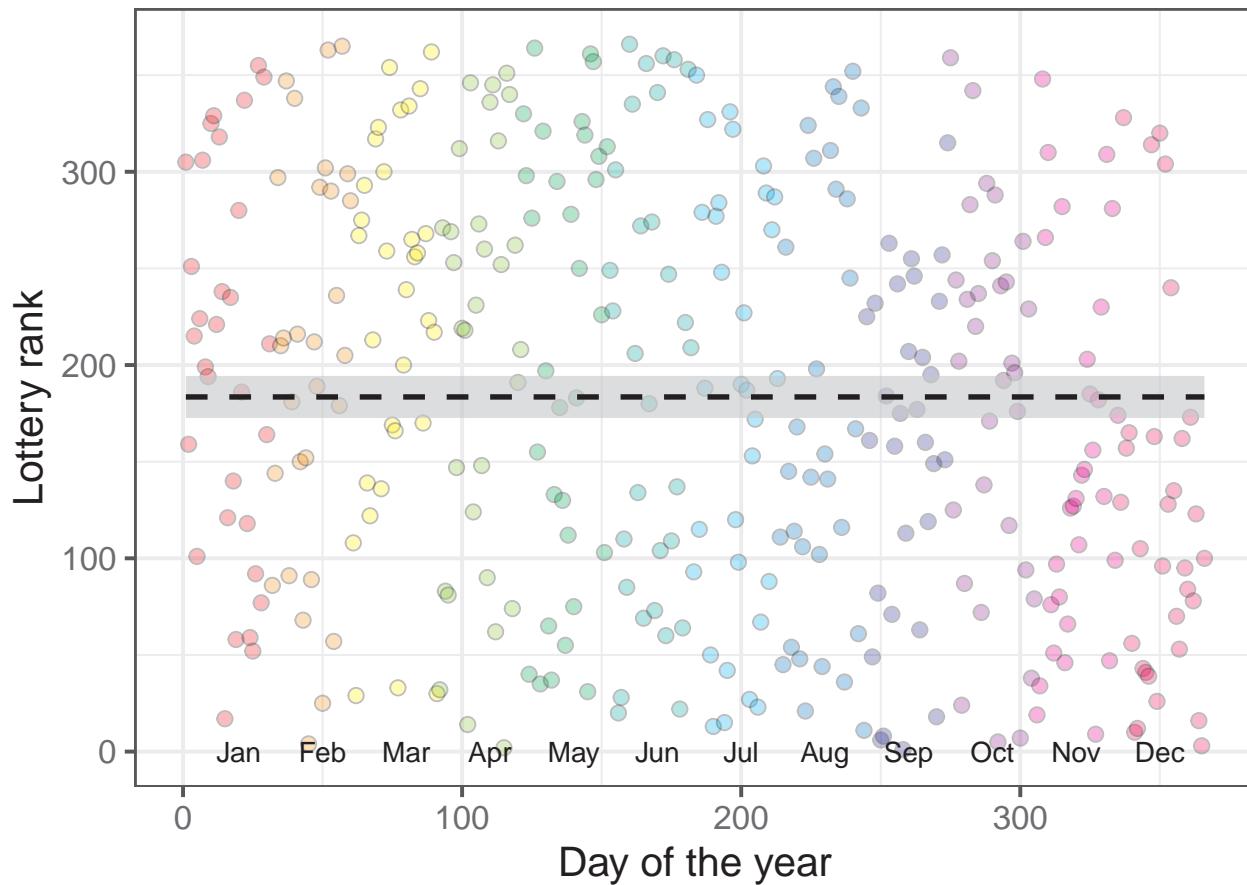


Figure 2.6: Basic scatterplot of 1970 Draft Lottery data plotting rank order of selection against birthdates in the year. Points are colored by month. The horizontal line is at the average rank.

The ranks do seem to be essentially random. Is there any reason to suspect a flaw in the selection process, as I firmly hoped at the time?

If you stare at the graph in Figure 2.6 long enough, you just can make out a sparsity of points in the upper right corner and also in the lower left corner compared to the opposite corners. But probably not until I told you.

Visual smoothers

Fitting a linear regression line or a smoothed (loess) curve can bring out the signal lurking in the background of a field of nearly random points. Figure 2.7 shows a definite trend to lower ranks for birthdays toward the end of the year. Those born earlier in the year were more likely to be given lower ranks, calling them up sooner for the draft.

```
ggplot(Draft1970, aes(x = Day, y = Rank)) +
  geom_point(size = 2.5, shape = 21,
             alpha = 0.3,
             color = "black",
             aes(fill=Month)) +
  scale_fill_manual(values = rainbow(12)) +
  geom_smooth(method = "lm", formula = y~1,
```

```

    se = FALSE,
    col = "black", fill="grey", linetype = "dashed", alpha=0.6) +
geom_smooth(method = "loess", formula = y~x,
            color = "blue", se = FALSE,
            alpha=0.25) +
geom_smooth(method = "lm", formula = y~x,
            color = "darkgreen",
            fill = "darkgreen",
            alpha=0.25) +
geom_text(data=months, aes(x=mid, y=0, label=month), nudge_x = 5) +
labs(x = "Day of the year",
     y = "Lottery rank") +
theme(legend.position = "none")

```

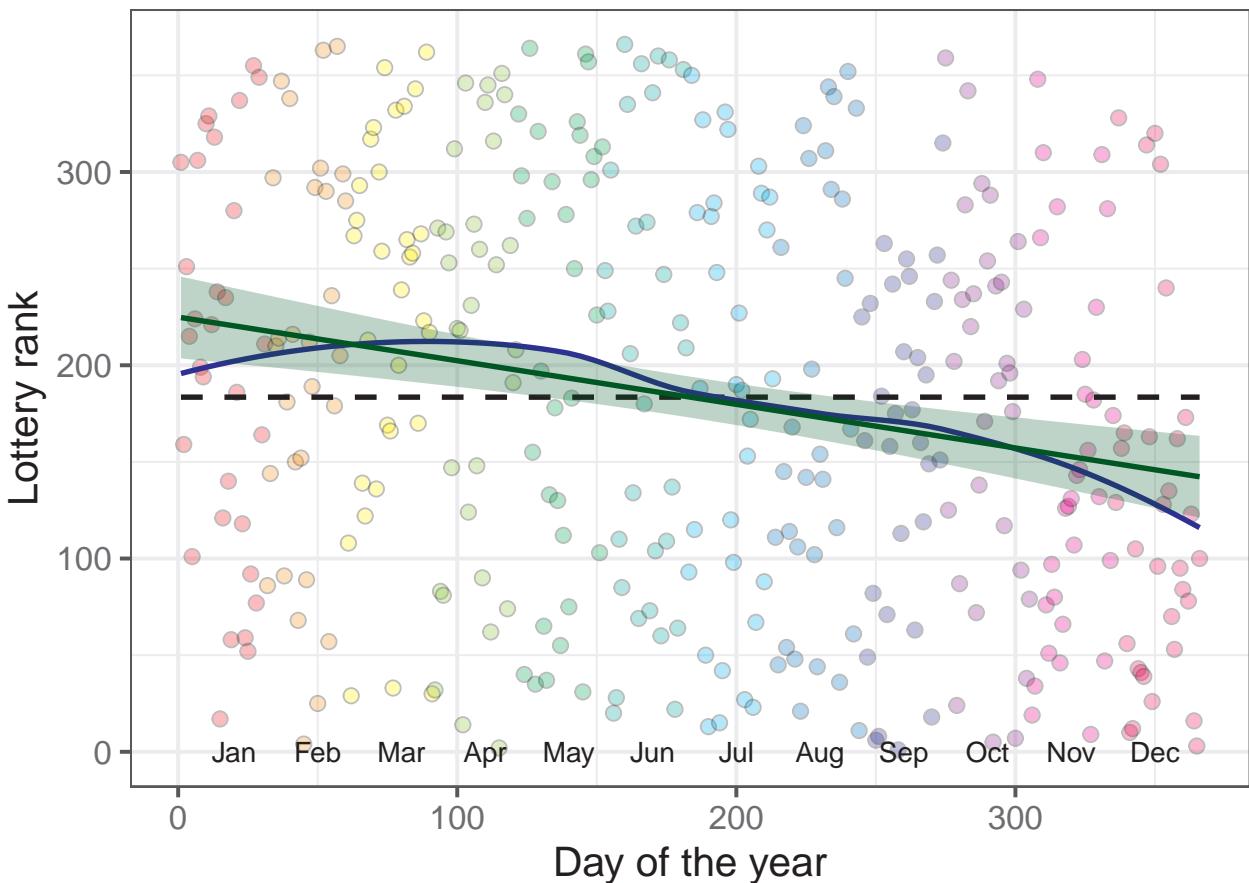


Figure 2.7: Enhanced scatterplot of 1970 Draft Lottery data adding a linear regression line and loess smooth.

Is this a real effect? Even though the points seem to be random over the year, linear regression of Rank on Day shows a highly significant negative effect even though the correlation³ is small ($r = -0.226$). The slope, -0.226 , means that for each additional day in the year the lottery rank decreases about $1/4$ toward the front of the draft line; that's nearly 7 ranks per month.

³Because both days of the year and rank in the lottery are the integers, 1 to 366, the Pearson correlation and Spearman rank order correlation are identical.

```

draft.mod <- lm(Rank ~ Day, data=Draft1970)
with(Draft1970, cor(Day, Rank))
#> [1] -0.226
coef(draft.mod)
#> (Intercept)      Day
#>     224.913    -0.226

```

So, smoothing the data, using either the linear regression line or a nonparametric smoother is one important technique for seeing a weak signal in a noisy background.

Statistical summaries

Another way to enhance the signal-to-noise ratio of a graph is to plot summaries of the messy data points. For example, you might make boxplots of the ranks by month, or calculate and plot the mean or median rank by month and plot those together with some indication of variability within month.

Figure 2.8 plots the average Rank for each month with error bars showing the mean ± 1 standard errors against the average Day. The message of rank decreasing nearly linearly with month is now more dramatic. The correlation between the means is $r = -0.867$; the slope is -0.231 , similar to what we found for the raw data.

```

means <- Draft1970 |>
  group_by(Month) |>
  summarize(Day = mean(Day),
            se = sd(Rank / sqrt(n())),
            Rank = mean(Rank))

ggplot(aes(x = Day, y = Rank), data=means) +
  geom_point(size = 4) +
  geom_smooth(method = "lm", formula = y~x,
              color = "blue", fill = "blue", alpha = 0.1) +
  geom_errorbar(aes(ymin = Rank-se, ymax = Rank+se),
                width = 8, linewidth = 1.3) +
  geom_text(data=months, aes(x=mid, y=0, label=month), nudge_x = 5) +
  ylim(100, 250) +
  labs(x = "Average day of the year",
       y = "Average lottery rank")

```

The visual impression of a linearly decreasing trend in lottery rank is much stronger in Figure 2.8 than in Figure 2.7 for two reasons:

- Replacing the data points with their means strengthens the signal in relation to noise, which is essentially eliminated by plotting means and error bars rather than the raw data.
- The narrower vertical range (100–250) in the plot of means makes the slope of the line appear steeper. (However, the slope of the means, $b = -0.231$ is nearly the same as that for the data points.) The narrower range also makes deviations from the regression line more noticeable.

What happened here?

Previous lotteries carried out by drawing capsules from a container had occasionally suffered the embarrassment that an empty capsule was selected because of vigorous mixing (Fienberg, 1971). So for the 1970 lottery, the birthdate capsules were put in cardboard boxes, one for each month and these were carefully emptied into the glass container in order of month: Jan., Feb., ... Dec., gently shaken in atop the pile already there. All might have been well had the persons drawing the capsules put their hand in truly randomly, but generally

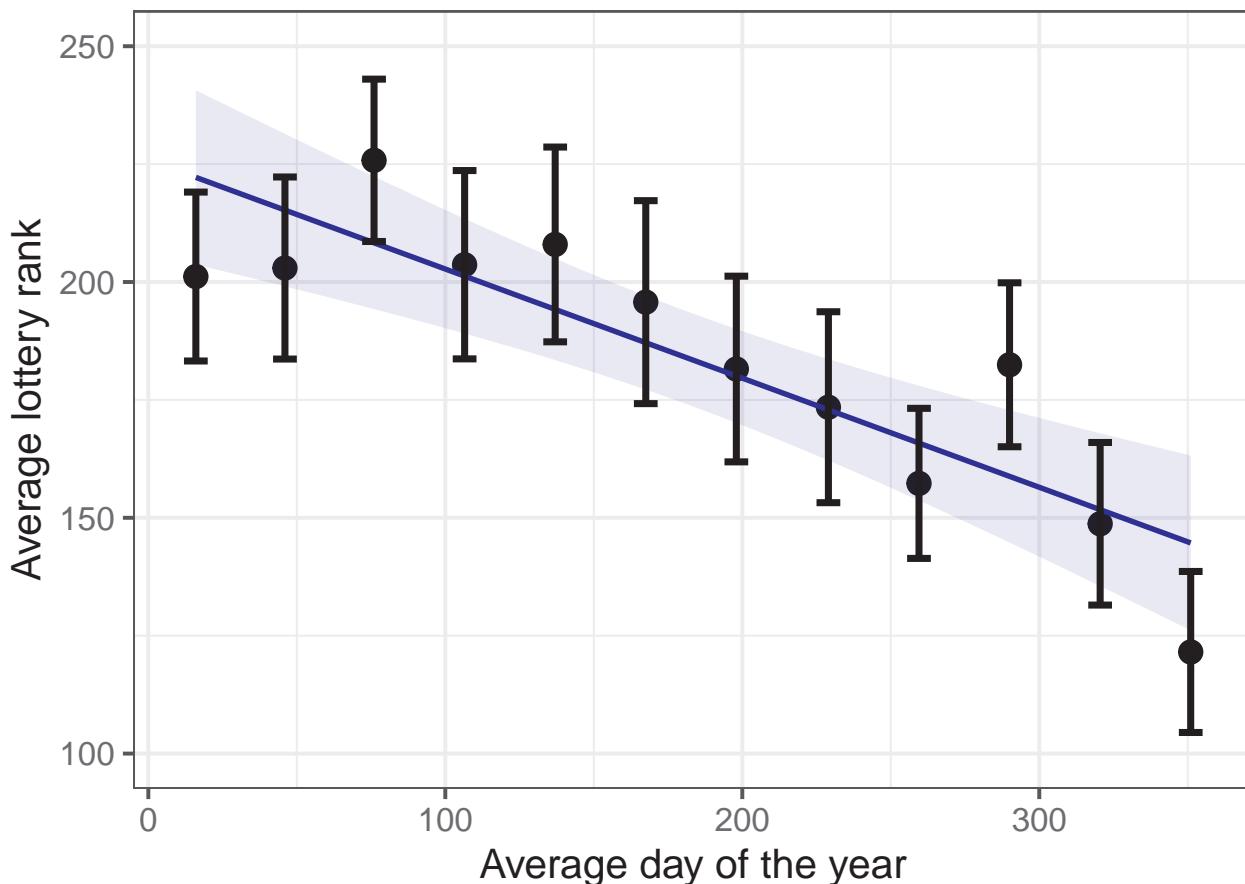


Figure 2.8: Plot of the average rank per month with ± 1 standard error bars. The line shows the least squares regression line, treating months as equally spaced.

they picked from toward the top of the container. Consequently, those born later in the year had a greater chance of being picked earlier.

There was considerable criticism of this procedure once the flaw had been revealed by analyses such as described here. In the following year, the Selective Service called upon the National Bureau of Standards to devise a better procedure. In 1971 they used two drums, one with the dates of the year and another with the rank numbers 1–366. As a date capsule was drawn randomly from the first drum, another from the numbers drum was picked simultaneously, giving a doubly-randomized sequence.

Of course, if they had R, the entire process could have been done using `sample()`:

```
set.seed(42)
date = seq(as.Date("1971-01-01"), as.Date("1971-12-31"), by="+1 day")
rank = sample(seq_along(date))
draft1971 <- data.frame(date, rank)

head(draft1971, 4)
#>      date rank
#> 1 1971-01-01   49
#> 2 1971-01-02  321
#> 3 1971-01-03  153
```

```
#> 4 1971-01-04 74
tail(draft1971, 4)
#>           date rank
#> 362 1971-12-28 247
#> 363 1971-12-29   8
#> 364 1971-12-30 333
#> 365 1971-12-31 132
```

And, what would have happened to me and all others born on a May 7th, if they did it this way? My lottery rank would have 274!⁴

```
me <- as.Date("1971-05-07")
draft1971[draft1971$date == me,]
#>           date rank
#> 127 1971-05-07 274
```

2.2 Plots for data analysis

Visualization methods take an enormous variety of forms, so it is useful to distinguish several broad categories according to their *use in data analysis*:

- **data plots** : primarily plot the raw data, often with annotations to aid interpretation (regression lines and smooths, data ellipses, marginal distributions). A survey of these methods is presented in Section 3.1.
- **reconnaissance plots** : with more than a few variables, reconnaissance plots provide a high-level, bird's-eye overview of the data, allowing you to see patterns that might not be visible in a set of separate plots. Some examples are scatterplot matrices (Section 3.10) showing all bivariate plots of variables in a dataset; correlation diagrams (Section 3.11), using visual glyphs to represent the correlations between all pairs of variables and “trellis” or faceted plots that show how a focal relation of one or more variables differs across values of other variables.
- **model plots** : plot the results of a fitted model, such as a regression line or curve to show uncertainty, or a regression surface in 3D, or a plot of coefficients in model together with confidence intervals. Other model plots try to take into account that a fitted model may involve more variables than can be shown in a static 2D plot. Some examples of these are added variable plots (Section 6.4), and marginal effect plots (Section 6.5), both of which attempt to show the *net* relation of two focal variables, controlling or adjusting for other variables in a model.
- **diagnostic plots** : indicating potential problems with the fitted model. These include residual plots, influence plots, plots for testing homogeneity of variance and so forth, illustrated in Section 6.1.2.
- **dimension reduction plots** : plot representations of the data into a space of fewer dimensions than the number of variables in the dataset. Simple examples include principal components analysis (PCA) and the related biplots, and multidimensional scaling (MDS) methods. This is the topic of Chapter 4, but this powerful idea runs through the rest of the book.

2.2.1 Model plots

Model plots show the fitted or predicted values from a statistical model and provide visual summaries...

⁴A personal note: I escaped being drafted, but I moved to Canada in 1971. Looking back today, it's one of the best decisions I ever made.

2.2.2 Diagnostic plots

2.2.3 Principles of graphical display

[This could be a separate chapter]

- Criteria for assessing graphs: communication goals
- Effective data display:
 - Make the data stand out
 - Make graphical comparison easy
 - Effect ordering: For variables and unordered factors, arrange them according to the effects to be seen
- Visual thinning: As the data becomes more complex, focus more on impactful summaries

->

2.3 What have we learned?

This chapter demonstrates why visualization isn't just a "nice-to-have" feature in data analysis—it's absolutely essential. Through compelling historical examples and modern techniques, we've discovered some fundamental idea that every data analyst should embrace:

- **Summary statistics can lie (beautifully):** Anscombe's Quartet reveals that datasets can have identical means, correlations, and regression coefficients yet tell completely different stories. The quartet's four plots—pure error, lack of fit, outliers, and influence—show that numerical summaries without visualization can lead us wildly astray. Modern extensions like the Datasaurus Dozen prove this isn't just a quirky historical example—you can literally hide a dinosaur in your data while maintaining identical statistical properties! Talk about the dinosaur in the room!
- **One rogue data point can hijack your entire analysis:** Plotting the raw data facilitates critical engagement with our statistical models. The Davis weight study demonstrates how a single influential observation (one participant who accidentally switched their reported and measured weights) can completely distort relationships between variables. What appeared to be a puzzling gender difference in the reliability of self-reported weight vanished once the data was plotted and the outlier revealed itself as a simple recording error.
- **Meaning becomes more apparent through thoughtful visualizations of our well-considered models:** Statistical modeling helps guide our attention in what might otherwise be a chaotic plot of raw data. The 1970 Draft Lottery story shows how graphics can reveal systematic bias hiding in apparently random data. While individual lottery numbers seemed random, smoothing techniques and summary plots exposed a clear pattern—later birthdays were systematically favored due to poor mixing of the lottery capsules. Sometimes the most important patterns are the ones that whisper rather than shout.
- **Different plot types serve different purposes:** The chapter introduces a taxonomy of visualization goals that helps us choose the right tool for each analytical task. Data plots show raw observations, reconnaissance plots provide bird's-eye overviews of complex datasets, model plots reveal fitted relationships, diagnostic plots expose model problems, and dimension reduction plots tame high-dimensional complexity. Each serves a distinct role in the analytical process.
- **Visual enhancement amplifies signal over noise:** Whether through smoothing lines, statistical summaries, or careful use of color and annotation, the chapter shows how thoughtful visual design can make weak patterns stand out dramatically. The Draft Lottery analysis becomes far more convincing when we plot monthly averages rather than individual data points, transforming a subtle correlation into an unmistakable trend.

The overarching message is clear: in an era where we can compute any statistic imaginable, the humble graph

remains our most powerful tool for understanding what our data are really trying to tell us. As the Farquhar brothers noted over a century ago, getting information from tables alone is like extracting sunlight from a cucumber—possible in theory, but why make it so hard on yourself?

Part II

Exploratory Methods

3

Plots of Multivariate Data

There is no excuse for failing to plot and look.

The greatest value of a picture is when it forces us to notice what we never expected to see. — John W. Tukey, *Exploratory Data Analysis*, 1977

These quotes from John Tukey remind us that data analysis should nearly always start with graphs to help us understand the main features of our data. It is important to understand the general *patterns* and *trends*: Are relationships increasing or decreasing? Are they approximately linear or non-linear? But it is also important to spot *anomalies*: “unusual” observations, groups of points that seem to differ from the rest, and so forth. As we saw with Anscombe’s quartet (Section 2.1.1) numerical summaries hide features that are immediately apparent in a plot.

This chapter introduces a toolbox of basic graphical methods for visualizing multivariate datasets. It starts with some simple techniques to enhance the basic scatterplot with graphical *annotations* such as fitted lines, curves and data ellipses to *summarize* the relation between two variables.

To visualize more than two variables, we can view all pairs of variables in a scatterplot matrix or shift gears entirely to show multiple variables along a set of parallel axes. As the number of variables increases, we may need to suppress details with stronger summaries for a high-level reconnaissance of our data terrain, as we do by zooming out on a map. For example, we can simply remove the data points or make them nearly transparent to focus on the visual summaries provided by fitted lines or other graphical summaries.

Packages

In this chapter I use the following packages. Load them now:

```
library(car)
library(ggplot2)
library(dplyr)
library(tidyr)
library(corrplot)
library(corrgram)
library(GGally)
library(ggdensity)
library(patchwork)
library(ggpcp)
library(tourr)
library(hplots)
library(gggda)
```

3.1 Bivariate summaries

The basic scatterplot is the workhorse of multivariate data visualization, showing how one variable, y , often an outcome to be explained by or varies with another, x . It is a building block for many useful techniques, so it is helpful to understand how it can be used as a tool for thinking in a wider, multivariate context.

The essential idea is that we can start with a simple version of the scatterplot and add annotations to show interesting features more clearly. We consider the following here:

- **Smoothers:** Showing overall trends, perhaps in several forms, as visual summaries such as fitted regression lines or curves and nonparametric smoothers.
- **Stratifiers:** Using color, shape or other features to identify subgroups; more generally, *conditioning* on other variables in multi-panel displays;
- **Data ellipses:** A compact 2D visual summary of bivariate linear relations and uncertainty assuming normality; more generally, contour plots of bivariate density.

Example 3.1. Academic salaries

Let's start with data on the academic salaries of faculty members collected at a U.S. college for the purpose of assessing salary differences between male and female faculty members, and perhaps address anomalies in compensation. The dataset `carData::Salaries` gives data on nine-month salaries and other variables for 397 faculty members in the 2008-2009 academic year.

```
data(Salaries, package = "carData")
str(Salaries)
#> 'data.frame': 397 obs. of 6 variables:
#> $ rank      : Factor w/ 3 levels "AsstProf","AssocProf",...: 3 3 1 3 3 2 3 3 3 ...
#> $ discipline : Factor w/ 2 levels "A","B": 2 2 2 2 2 2 2 2 2 ...
#> $ yrs.since.phd: int 19 20 4 45 40 6 30 45 21 18 ...
#> $ yrs.service : int 18 16 3 39 41 6 23 45 20 18 ...
#> $ sex        : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 1 ...
#> $ salary     : int 139750 173200 79750 115000 141500 97000 175000 147765 119250 129000 ...
```

The most obvious, but perhaps naive, predictor of `salary` is `years.since.phd`. For simplicity, I'll refer to this as years of "experience." Before looking at differences between males and females, we would want consider faculty `rank` (related also to `yrs.service`) and `discipline`, recorded here as "A" ("theoretical" departments) or "B" ("applied" departments). But, for a basic plot, we will ignore these for now to focus on what can be learned from plot annotations.

```
library(ggplot2)
gg1 <- ggplot(Salaries,
  aes(x = yrs.since.phd, y = salary)) +
  geom_jitter(size = 2) +
  scale_y_continuous(labels = scales::dollar_format(
    prefix = "$", scale = 0.001, suffix = "K")) +
  labs(x = "Years since PhD",
       y = "Salary")

gg1 + geom_rug(position = "jitter", alpha = 1/4)
```

There is quite a lot we can see "just by looking" at this simple plot, but the main things are:

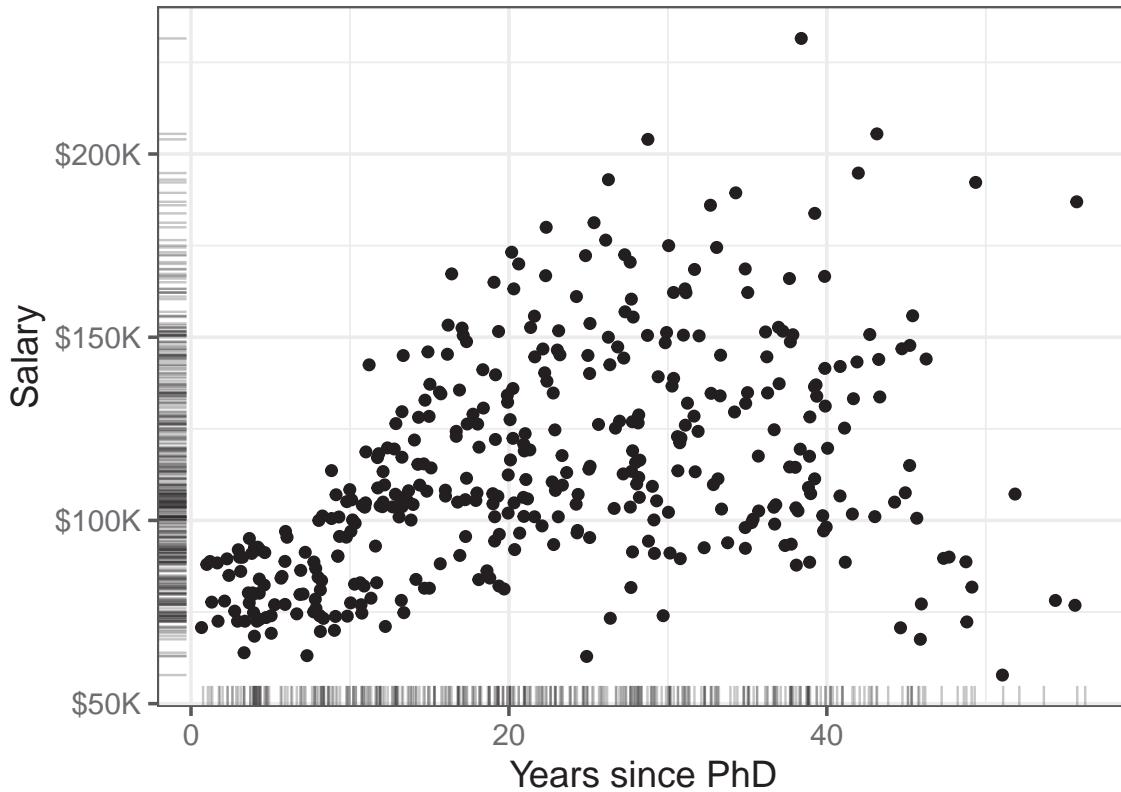


Figure 3.1: Naive scatterplot of Salary vs. years since PhD, ignoring other variables, and without graphical annotations.

- Salary increases generally from 0 - 40 years since the PhD, but then maybe begins to drop off (partial retirement?);
- Variability in salary increases among those with the same experience, a “fan-shaped” pattern that signals a violation of homogeneity of variance in simple regression;
- Data beyond 50 years is thin, but there are some quite low salaries there. Adding rug plots to the X and Y axes is a simple but effective way to show the marginal distributions of the observations. Jitter and transparency helps to avoid overplotting due to discrete values.

3.1.1 Smoothers

Smoothers are among the most useful graphical annotations you can add to such plots, giving a visual summary of how y changes with x . The most common smoother is a line showing the linear regression for y given x , expressed in math notation as $\mathbb{E}(y|x) = b_0 + b_1x$. If there is doubt that a linear relation is an adequate summary, you can try a quadratic or other polynomial smoothers.

In `ggplot2`, these are easily added to a plot using `geom_smooth()` with `method = "lm"`, and a model `formula`, which (by default) is `y ~ x` for a linear relation or `y ~ poly(x, k)` for a polynomial of degree k .

```
gg1 +
  geom_smooth(method = "lm", formula = "y ~ x",
              color = "red", fill= "pink",
              linewidth = 2) +
  geom_smooth(method = "lm", formula = "y ~ poly(x,2)",
```

```
color = "darkgreen", fill = "lightgreen",
linewidth = 2)
```

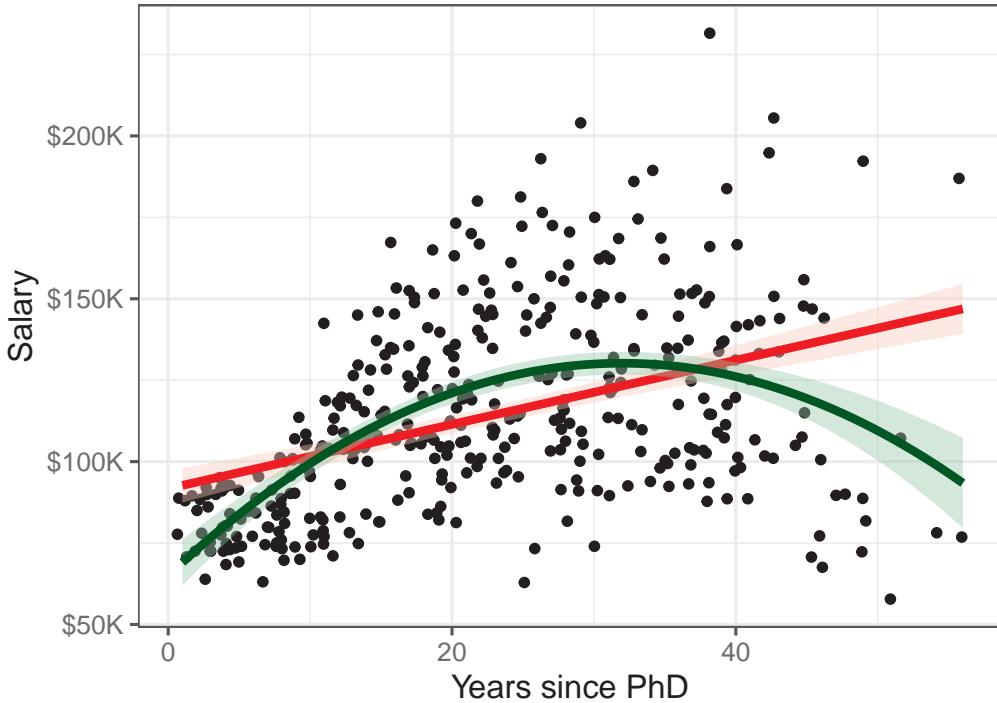


Figure 3.2: Scatterplot of Salary vs. years since PhD, showing linear and quadratic smooths with 95% confidence bands.

This serves to highlight some of our impressions from the basic scatterplot shown in Figure 3.1, making them more apparent. And that's precisely the point: the regression smoother draws attention to a possible pattern that we can consider as a visual summary of the data. You can think of this as showing what a linear (or quadratic) regression “sees” in the data. Statistical tests can help you decide if there is more evidence for a quadratic fit compared to the simpler linear relation.

It is useful to also show some indication of *uncertainty* (or inversely, *precision*) associated with the predicted values. Both the linear and quadratic trends are shown in Figure 3.2 with 95% pointwise confidence bands.¹ These are necessarily narrower in the center of the range of x where there is typically more data; they get wider toward the highest values of experience where the data are thinner.

Non-parametric smoothers

The most generally useful idea is a smoother that tracks an average value, $\mathbb{E}(y|x)$, of y as x varies across its range *without* assuming any particular functional form, and so avoiding the necessity to choose among $y \sim \text{poly}(x, 1)$, or $y \sim \text{poly}(x, 2)$, or $y \sim \text{poly}(x, 3)$, etc.

¹Confidence bands allow us to visualize the uncertainty around a fitted regression curve, which can be of two types: *pointwise intervals* or *simultaneous intervals*. The default setting in ‘`ggplot2::geom_smooth()`’ calculates pointwise intervals (using `stats::predict.lm(..., interval="confidence")`) at a confidence level $1 - \alpha$ for the predicted response at *each value* x_i of a predictor, and have the frequentist interpretation that over repeated sampling only 100α of the predictions at x_i will be outside that interval. In contrast, simultaneous intervals are calculated so that $1 - \alpha$ is the probability that *all of them* cover their corresponding true values simultaneously. These are necessarily wider than pointwise intervals. Commonly used methods for constructing simultaneous confidence bands in regression are the Bonferroni and Scheffé methods, which control the family-wise error rate over all values of x_i . See for precise definitions of these terms. These are different from a *prediction band*, which is used to represent the uncertainty about the value of a **new** data-point on the curve, but subject to the additional variance reflected in one observation.

Non-parametric smoothers attempt to estimate $\mathbb{E}(y|x) = f(x)$ where $f(x)$ is some smooth function. These typically use a collection of weighted *local regressions* for each x_i within a window centered at that value. In the method called *lowess* or *loess* (Cleveland, 1979; Cleveland & Devlin, 1988), a weight function is applied, giving greatest weight to x_i and a weight of 0 outside a window containing a certain fraction, s , called *span*, of the nearest neighbors of x_i . The fraction, s , is usually within the range $1/3 \leq s \leq 2/3$, and it determines the smoothness of the resulting curve; smaller values produce a wigglier curve and larger values giving a smoother fit (an optimal span can be determined by k -fold cross-validation to minimize a measure of overall error of approximation).

Non-parametric regression is a broad topic; see Fox (2016), Ch. 18 for a more general treatment including smoothing splines, and Wood (2006) for generalized additive models, fit using `method = "gam"` in `ggplot2`, which is the default when the largest group has more than 1,000 observations.

Figure 3.3 shows the addition of a loess smooth to the plot in Figure 3.2, suppressing the confidence band for the linear regression. The loess fit is nearly coincident with the quadratic fit, but has a slightly wider confidence band.

```
gg1 +
  geom_smooth(method = "loess", formula = "y ~ x",
              color = "blue", fill = scales::muted("blue"),
              linewidth = 2) +
  geom_smooth(method = "lm", formula = "y ~ x", se = FALSE,
              color = "red",
              linewidth = 2) +
  geom_smooth(method = "lm", formula = "y ~ poly(x,2)",
              color = "darkgreen", fill = "lightgreen",
              linewidth = 2)
```

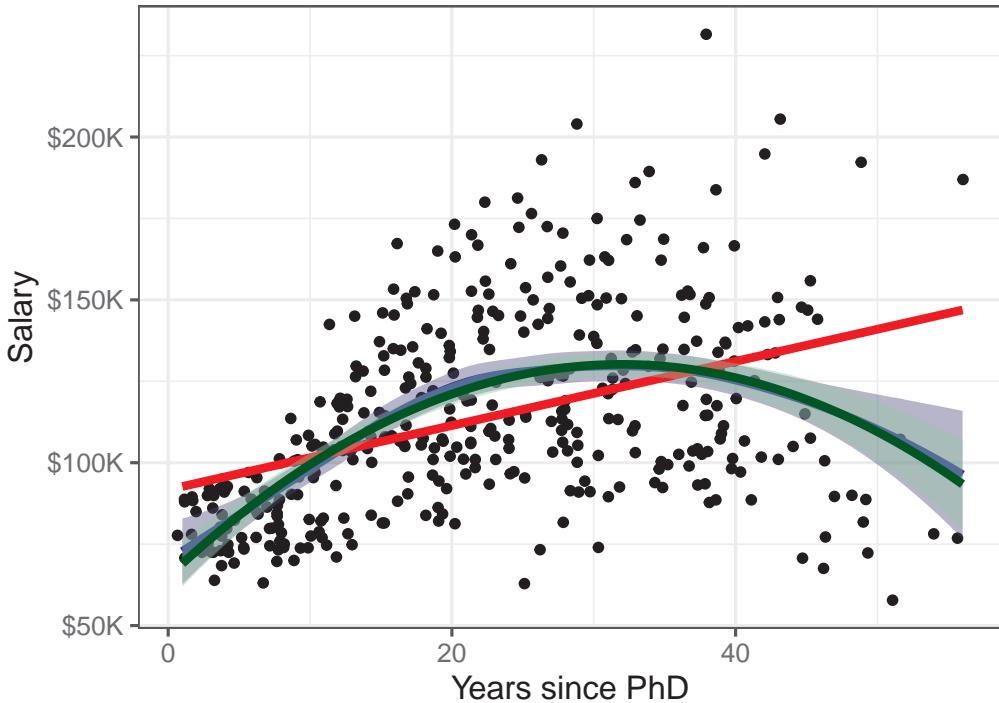


Figure 3.3: Scatterplot of Salary vs. years since PhD, adding the loess smooth. The loess smooth curve and confidence band in green is nearly indistinguishable from a quadratic fit in blue.

But now comes an important question: is it reasonable that academic salary should increase up to about 40 years since the PhD degree and then decline? The predicted salary for someone still working 50 years after earning their degree is about the same as a person at 15 years. What else is going on here?

3.1.2 Stratifiers

Very often, we have a main relationship of interest, but various groups in the data are identified by discrete factors (like faculty `rank` and `sex`, their type of `discipline` here), or there are quantitative predictors for which the main relation might vary. In the language of statistical models such effects are *interaction* terms, as in $y \sim \text{group} + x + \text{group}:x$, where the term `group:x` fits a different slope for each group and the grouping variable is often called a *moderator* variable. Common moderator variables are ethnicity, health status, social class and level of education. Moderators can also be continuous variables as in $y \sim x_1 + x_2 + x_1:x_2$.

I call these *stratifiers*, recognizing that we should consider breaking down the overall relation to see whether and how it changes over such “other” variables. Such variables are most often factors, but we can cut a continuous variable into ranges (*shingles*) and do the same graphically. There are two general stratifying graphical techniques:

- **Grouping:** Identify subgroups in the data by assigning different visual attributes, such as color, shape, line style, etc. within a single plot. This is quite natural for factors; quantitative predictors can be accommodated by cutting their range into ordered intervals. Grouping has the advantage that the levels of a grouping variable can be shown within the same plot, facilitating direct comparison.
- **Conditioning:** Showing subgroups in different plot panels. This has the advantages that relations for the individual groups more easily discerned and one can easily stratify by two (or more) other variables jointly, but visual comparison is more difficult because the eye must scan from one panel to another.

i History Corner

Recognition of the roles of visual grouping by factors within a panel and conditioning in multi-panel displays was an important advance in the development of modern statistical graphics. It began at A.T.&T. Bell Labs in Murray Hill, NJ in conjunction with the **S** language, the mother of R.

Conditioning displays (originally called *coplots* (Chambers & Hastie, 1991)) are simply a collection of 1D, 2D or 3D plots separate panels for subsets of the data broken down by one or more factors, or, for quantitative variables, subdivided into a factor with several overlapping intervals (*shingles*). The first implementation was in *Trellis* plots (Becker et al., 1996; Cleveland, 1985).

Trellis displays were extended in the **lattice** package (Sarkar, 2025), which offered:

- A **graphing syntax** similar to that used in statistical model formulas: $y \sim x | g$ conditions the data by the levels of `g`, with `|` read as “given”; two or more conditioning are specified as $y \sim x | g_1 + g_2 + \dots$, with `+` read as “and”.
- **Panel functions** define what is plotted in a given panel. `panel.xyplot()` is the default for scatterplots, plotting points, but you can add `panel.lmline()` for regression lines, `latticeExtra::panel.smoother()` for loess smooths and a wide variety of others.

The **car** package (Fox et al., 2023) supports this graphing syntax in many of its functions. **ggplot2** does not; it uses aesthetics (`aes()`), which map variables in the data to visual characteristics in displays.

The most obvious variable that affects academic salary is `rank`, because faculty typically get an increase in salary with a promotion that carries through in their future salary. What can we see if we group by `rank` and fit a separate smoothed curve for each?

In **ggplot2** thinking, grouping is accomplished simply by adding an aesthetic, such as `color = rank`. What happens then is that points, lines, smooths and other `geom_*`() inherit the feature that they are differentiated by color. In the case of `geom_smooth()`, we get a separate fit for each subset of the data, according to `rank`.

```
# make some re-useable pieces to avoid repetitions
scale_salary <- scale_y_continuous(
  labels = scales::dollar_format(prefix="$",
                                 scale = 0.001,
                                 suffix = "K"))

# position the legend inside the plot
legend_pos <- theme(legend.position = "inside",
                     legend.position.inside = c(.1, 0.95),
                     legend.justification = c(0, 1))

ggplot(Salaries,
       aes(x = yrs.since.phd, y = salary,
           color = rank, shape = rank)) +
  geom_point() +
  scale_salary +
  labs(x = "Years since PhD",
       y = "Salary") +
  geom_smooth(aes(fill = rank),
              method = "loess", formula = "y ~ x",
              linewidth = 2) +
  legend_pos
```

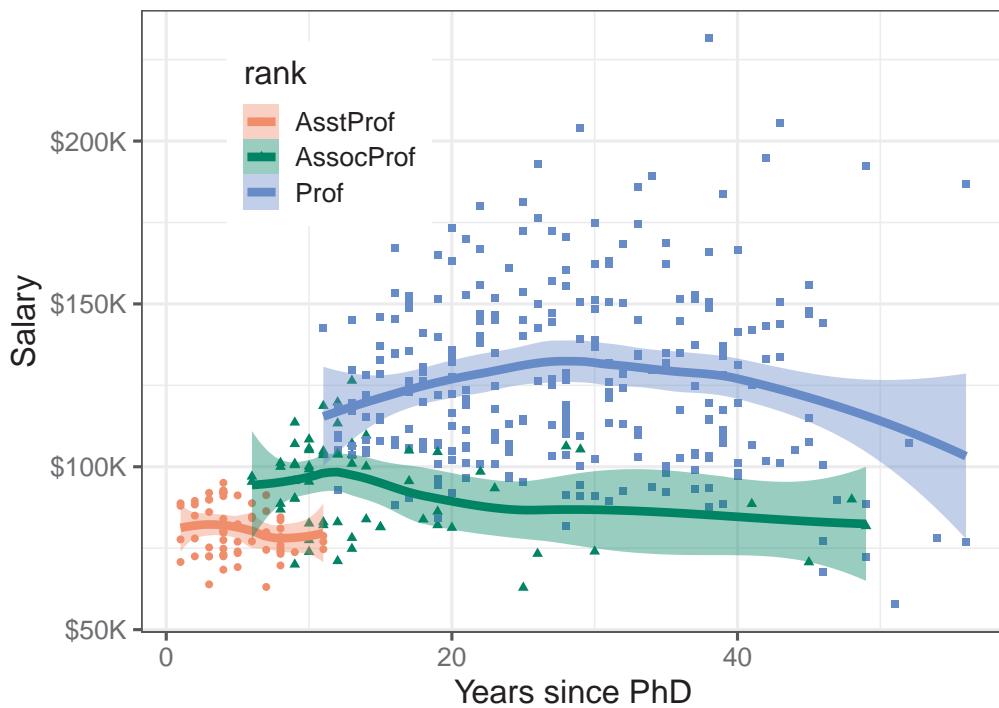


Figure 3.4: Scatterplot of Salary vs. years since PhD, grouped by rank.

Well, there is a different story here. Salaries generally occupy separate vertical levels, increasing with academic rank. The horizontal extents of the smoothed curves show their ranges. Within each rank there is some initial increase after promotion, and then some tendency to decline with increasing years. But, by and large, years since the PhD doesn't make as much difference once we've taken academic rank into account.

What about the `discipline` which is classified, perhaps peculiarly, as “theoretical” vs. “applied”? The values are just “A” and “B”, so I map these to more meaningful labels before making the plot.

```
Salaries <- Salaries |>
  mutate(discipline =
    factor(discipline,
           labels = c("A: Theoretical", "B: Applied")))

Salaries |>
  ggplot(aes(x = yrs.since.phd, y = salary, color = discipline)) +
  geom_point() +
  scale_salary +
  geom_smooth(aes(fill = discipline),
              method = "loess", formula = "y ~ x",
              linewidth = 2) +
  labs(x = "Years since PhD",
       y = "Salary") +
  legend_pos
```

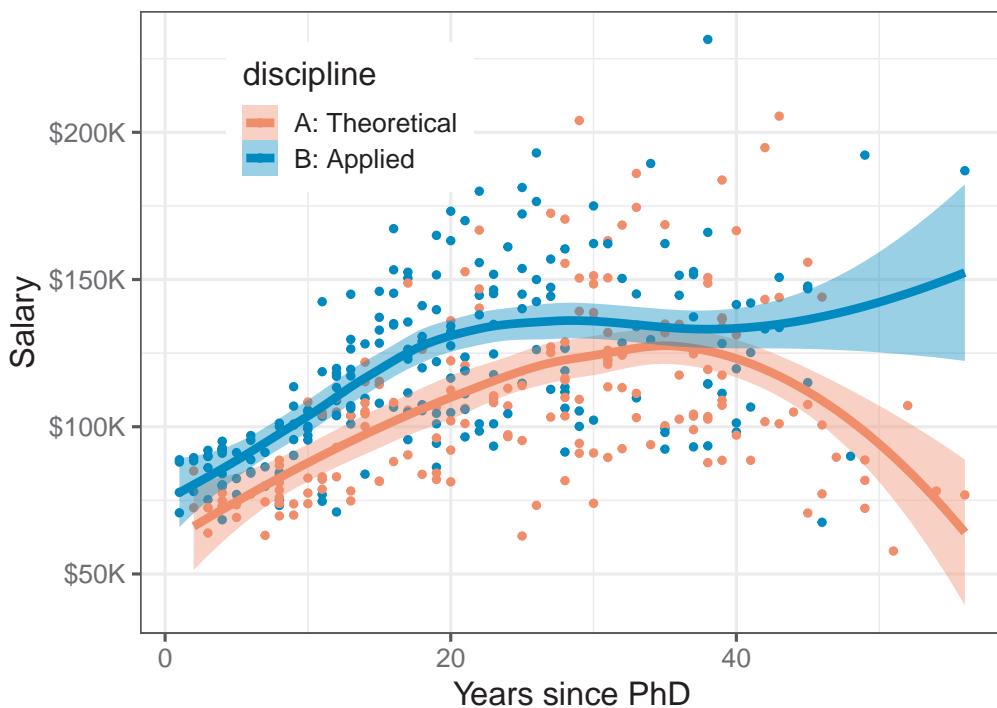


Figure 3.5: Scatterplot of Salary vs. years since PhD, grouped by discipline.

The story in Figure 3.5 is again different. Faculty in applied disciplines on average earn about 10,000\$ more per year on average than their theoretical colleagues.

```
Salaries |>
  group_by(discipline) |>
  summarize(mean = mean(salary))
#> # A tibble: 2 x 2
#>   discipline      mean
```

```
#> <fct>      <dbl>
#> 1 A: Theoretical 108548.
#> 2 B: Applied     118029.
```

For both groups, there is an approximately linear relation up to about 30–40 years, but the smoothed curves then diverge into the region where the data is thinner.

This result is more surprising than differences among faculty ranks. The effect of annotation with smoothed curves as visual summaries is apparent, and provides a stimulus to think about *why* these differences (if they are real) exist between theoretical and applied professors, and maybe *should* theoreticians be paid more!

3.1.3 Conditioning

The previous plots use grouping by color to plot the data for different subsets inside the same plot window, making comparison among groups easier, because they can be directly compared along a common vertical scale ². This gets messy, however, when there are more than just a few levels, or worse—when there are two (or more) variables for which we want to show separate effects. In such cases, we can plot separate panels using the `ggplot2` concept of *faceting*. There are two options: `facet_wrap()` takes one or more conditioning variables and produces a ribbon of plots for each combination of levels; `facet_grid(row ~ col)` takes two or more conditioning variables and arranges the plots in a 2D array identified by the `row` and `col` variables.

Let's look at salary broken down by the combinations of discipline and rank. Here, I chose to stratify using color by rank within each of panels faceting by discipline. Because there is more going on in this plot, a linear smooth is used to represent the trend.

```
Salaries |>
  ggplot(aes(x = yrs.since.phd, y = salary,
             color = rank, shape = rank)) +
  geom_point() +
  scale_salary +
  labs(x = "Years since PhD",
       y = "Salary") +
  geom_smooth(aes(fill = rank),
              method = "lm", formula = "y ~ x",
              linewidth = 2, alpha = 1/4) +
  facet_wrap(~ discipline) +
  legend_pos
```

Once both of these factors are taken into account, there does not seem to be much impact of years of service. Salaries in theoretical disciplines are noticeably greater than those in applied disciplines at all ranks, and there are even greater differences among ranks.

Finally, to shed light on the question that motivated this example— are there anomalous differences in salary for men and women— we can look at differences in salary according to sex, when discipline and rank are taken into account. To do this graphically, condition by both variables, but use `facet_grid(discipline ~ rank)` to arrange their combinations in a grid whose rows are the levels of `discipline` and columns are those of `rank`. I want to make the comparison of males and females most direct, so I use `color = sex` to stratify the panels. The smoothed regression lines and error bands are calculated separately for each combination of discipline, rank and sex.

²The classic study by Cleveland & McGill (1984);Cleveland & McGill (1985) shows that judgements of magnitude along a common scale are more accurate than those along separate, aligned scales.

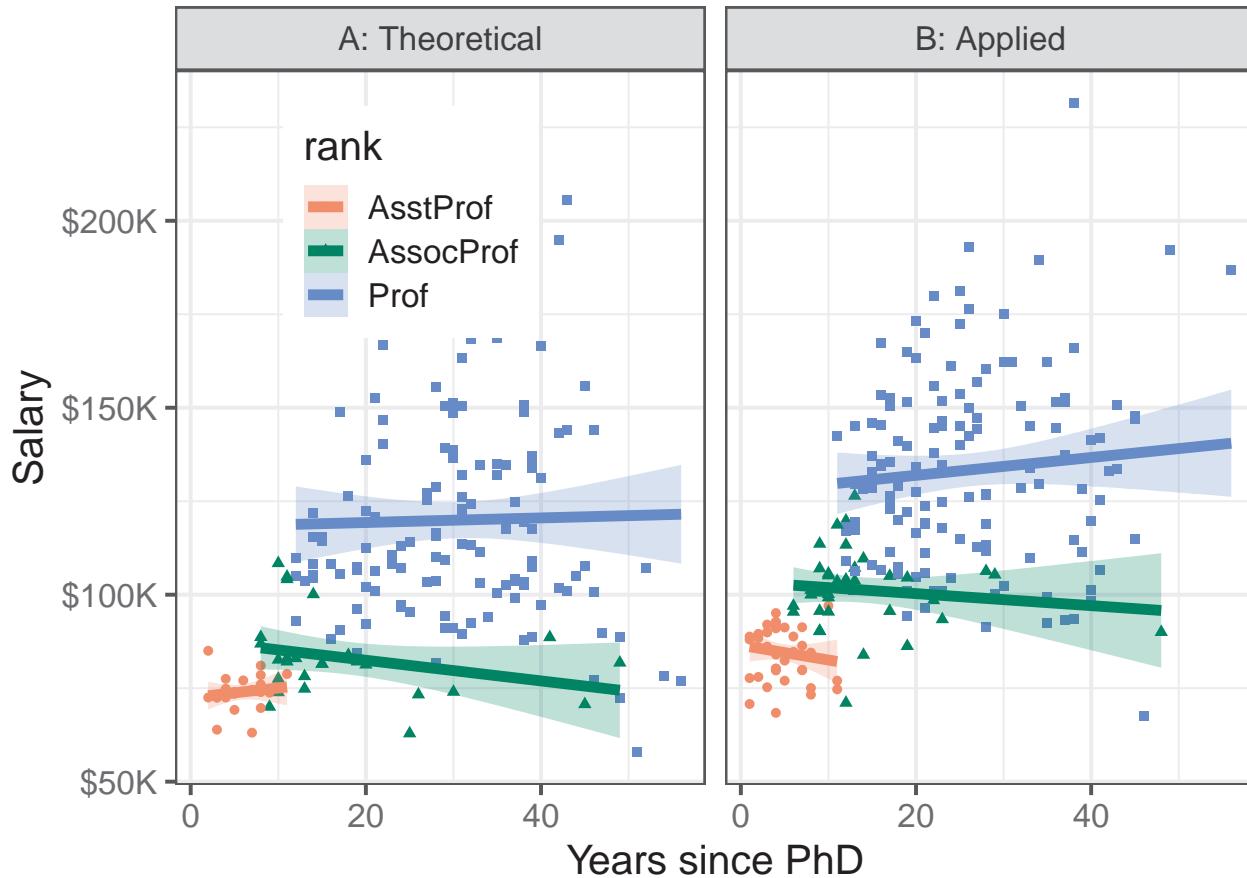


Figure 3.6: Scatterplot of Salary vs. years since PhD, grouped by rank, with separate panels for discipline.

```
Salaries |>
  ggplot(aes(x = yrs.since.phd, y = salary, color = sex)) +
  geom_point() +
  scale_salary +
  labs(x = "Years since PhD",
       y = "Salary") +
  geom_smooth(aes(fill = sex),
              method = "lm", formula = "y ~ x",
              linewidth = 2, alpha = 1/4) +
  facet_grid(discipline ~ rank) +
  theme_bw(base_size = 14) +
  legend_pos
```

3.2 Data Ellipses

The *data ellipse* (Monette, 1990), or *concentration ellipse* (Dempster, 1969) is a remarkably simple and effective display for viewing and understanding bivariate relationships in multivariate data. The data ellipse

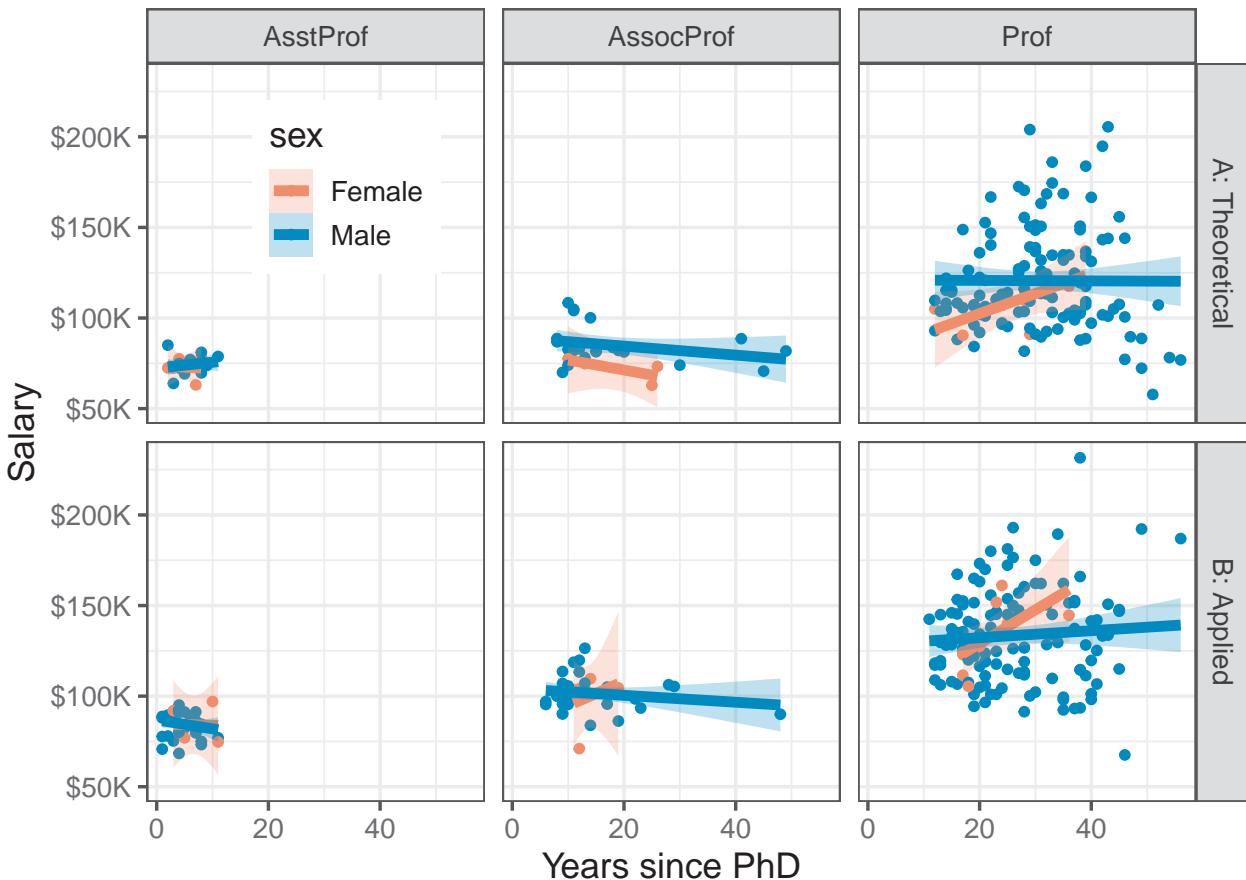


Figure 3.7: Scatterplot of Salary vs. years since PhD, grouped by sex, faceted by discipline and rank.

is typically used to add a visual summary to a scatterplot, that shows all together the means, standard deviations, correlation, and slope of the regression line for two variables, perhaps stratified by another variable.

Under the classical assumption that the data are bivariate normally distributed, the data ellipse is also a **sufficient** visual summary, in the sense that it captures **all** relevant features of the data. See Friendly et al. (2013) for a complete discussion of the role of ellipsoids in statistical data visualization.

The data ellipse is based on the idea that in a bivariate normal distribution, the contours of equal probability form a series of concentric ellipses. If the variables were uncorrelated and had the same variances, these would be circles, and Euclidean distance would measure the distance of each observation from the mean. When the variables are correlated, a different measure, *Mahalanobis distance* is the proper measure of how far a point is from the mean, taking the correlation into account.

To illustrate, Figure 3.8 shows a scatterplot with labels for two points, “A” and “B”. Which is further from the mean, “X”? A contour of constant Euclidean distance, shown by the red dashed circle, ignores the apparent negative correlation, so point “A” is further. The blue ellipse for Mahalanobis distance takes the correlation into account, so point “B” has a greater distance from the mean.

Mathematically, Euclidean (squared) distance for p variables, $j = 1, 2, \dots, p$, is just a generalization of the square of a univariate standardized (z) score, $z^2 = [(y - \bar{y})/s]^2$,

$$D_E^2(\mathbf{y}) = \sum_j^p z_j^2 = \mathbf{z}^\top \mathbf{z} = (\mathbf{y} - \bar{\mathbf{y}})^\top \text{diag}(\mathbf{S})^{-1}(\mathbf{y} - \bar{\mathbf{y}}),$$

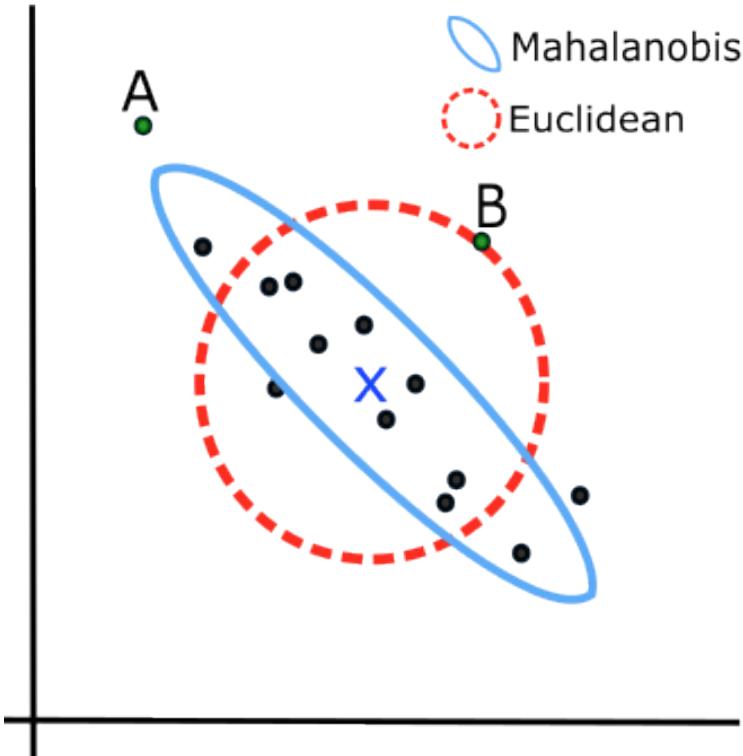


Figure 3.8: 2D data with curves of constant distance from the centroid. The blue solid ellipse shows a contour of constant Mahalanobis distance, taking the correlation into account; the dashed red circle is a contour of equal Euclidean distance. Given the data points, Which of the points **A** and **B** is further from the mean ($\bar{\mathbf{y}}$)? *Source:* Re-drawn from Ou Zhang

where \mathbf{S} is the sample variance-covariance matrix, $\mathbf{S} = (n - 1)^{-1} \sum_{i=1}^n (\mathbf{y}_i - \bar{\mathbf{y}})^\top (\mathbf{y}_i - \bar{\mathbf{y}})$.

Mahalanobis distance takes the correlations into account simply by using the covariances as well as the variances,

$$D_M^2(\mathbf{y}) = (\mathbf{y} - \bar{\mathbf{y}})^\top \mathbf{S}^{-1} (\mathbf{y} - \bar{\mathbf{y}}) . \quad (3.1)$$

In Equation 3.1, the inverse \mathbf{S}^{-1} serves to “divide” the matrix $(\mathbf{y} - \bar{\mathbf{y}})^\top (\mathbf{y} - \bar{\mathbf{y}})$ of squared distances by the variances (and covariances) of the variables, as in the univariate case.

For p variables, the data *ellipsoid* \mathcal{E}_c of size c is a p -dimensional ellipse, defined as the set of points $\mathbf{y} = (y_1, y_2, \dots, y_p)$ whose squared Mahalanobis distance, $D_M^2(\mathbf{y})$ is less than or equal to c^2 ,

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{S}) := \{D_M^2(\mathbf{y}) \leq c^2\} .$$

A computational definition recognizes that the boundary of the ellipsoid can be found by transforming a unit sphere \mathcal{P} centered at the origin, $\mathcal{P} := \{\mathbf{x}^\top \mathbf{x} = 1\}$, by $\mathbf{S}^{1/2}$ and then shifting that to centroid of the data,

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{S}) = \bar{\mathbf{y}} \oplus \mathbf{S}^{1/2} \mathcal{P} ,$$

where $\mathbf{S}^{1/2}$ represents a rotation and scaling and the notation \oplus represents translation to a new centroid, $\bar{\mathbf{y}}$ here. The matrix $\mathbf{S}^{1/2}$ is commonly computed as the Cholesky factor of \mathbf{S} . Slightly abusing notation and taking the unit sphere \mathcal{P} as given (like an identity matrix \mathbf{I}), we can write the data ellipsoid as simply:

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{S}) = \bar{\mathbf{y}} \oplus c\sqrt{\mathbf{S}} . \quad (3.2)$$

When \mathbf{y} is (at least approximately) bivariate normal, $D_M^2(\mathbf{y})$ has a large-sample χ^2 distribution (χ^2 with 2 df), so

- $c^2 = \chi_2^2(0.5) = 1.39$ gives a data ellipse covering 50% of the data points, a bivariate analog of the central box of a boxplot.
- $c^2 = \chi_2^2(0.68) = 2.28$ gives a “1 standard deviation bivariate ellipse,” an analog of the standard interval $\bar{y} \pm 1s$,
- $c^2 = \chi_2^2(0.95) = 5.99 \approx 6$ gives a data ellipse of 95% coverage.

In not-large samples, the radius c of the ellipsoid is better approximated by a multiple of a $F_{p,n-p}$ distribution, becoming $c = \sqrt{2F_{2,n-2}^{1-\alpha}}$ in the bivariate case ($p = 2$) for coverage $1 - \alpha$.

These are illustrated in Figure 3.9 with ellipses of 50%, 68% and 95% coverage for a the matrix \mathbf{S} defined below and $\bar{\mathbf{y}} = \mathbf{0}$. Here, the variance of \mathbf{y}_1 is twice that of \mathbf{y}_2 and the correlation works out to $r(y_1, y_2) = 0.35$.

```
ybar <- c(0, 0)
S <- matrix(c(1, .5, .5, 2), 2, 2)
rownames(S) <- colnames(S) <- c("y1", "y2")
S
#>      y1   y2
#> y1 1.0 0.5
#> y2 0.5 2.0
```

Statistical ellipses are conveniently drawn using `car::ellipse()`. `heplots::ellipse.label()` provides flexible ways to add labels to ellipses at various locations around the ellipse shown in Figure 3.9. These are called repeatedly to overlay the three ellipses.

```
levels <- c(0.50, 0.68, 0.95)
c <- qchisq(levels, df = 2) |> round(2) |> print()
#> [1] 1.39 2.28 5.99

# labels for ellipses, using plotmath
lab1 <- bquote(paste("c =", chi[2]^2, "(", .(levels[1]), ") =", .(c[1])))
lab2 <- bquote(paste("c =", chi[2]^2, "(", .(levels[2]), ") =", .(c[2])))
lab3 <- bquote(paste("c =", chi[2]^2, "(", .(levels[3]), ") =", .(c[3])))

e1 <- ellipse(ybar, S, radius=qchisq(levels[1], 2),
               col = "blue", fill=TRUE, fill.alpha = 0.5,
               add=FALSE,
               xlim=c(-8, 8), ylim=c(-9.5, 9.5),
               asp=1, grid = FALSE,
               xlab = expression(y[1]),
               ylab = expression(y[2]),
               cex.lab = 1.5)
label.ellipse(e1, label = lab1, label.pos = "S",
              cex = 1.2)

e2 <- ellipse(ybar, S, radius=qchisq(levels[2], 2),
               col="blue", fill=TRUE, fill.alpha=0.3)
label.ellipse(e2, label = lab2, label.pos = "N",
              cex = 1.2)
```

```
e3 <- ellipse(ybar, S, radius=qchisq(levels[3], 2),
               col="blue", fill=TRUE, fill.alpha=0.1)
label.ellipse(e3, label = lab3, label.pos = "N",
              cex = 1.2)
```

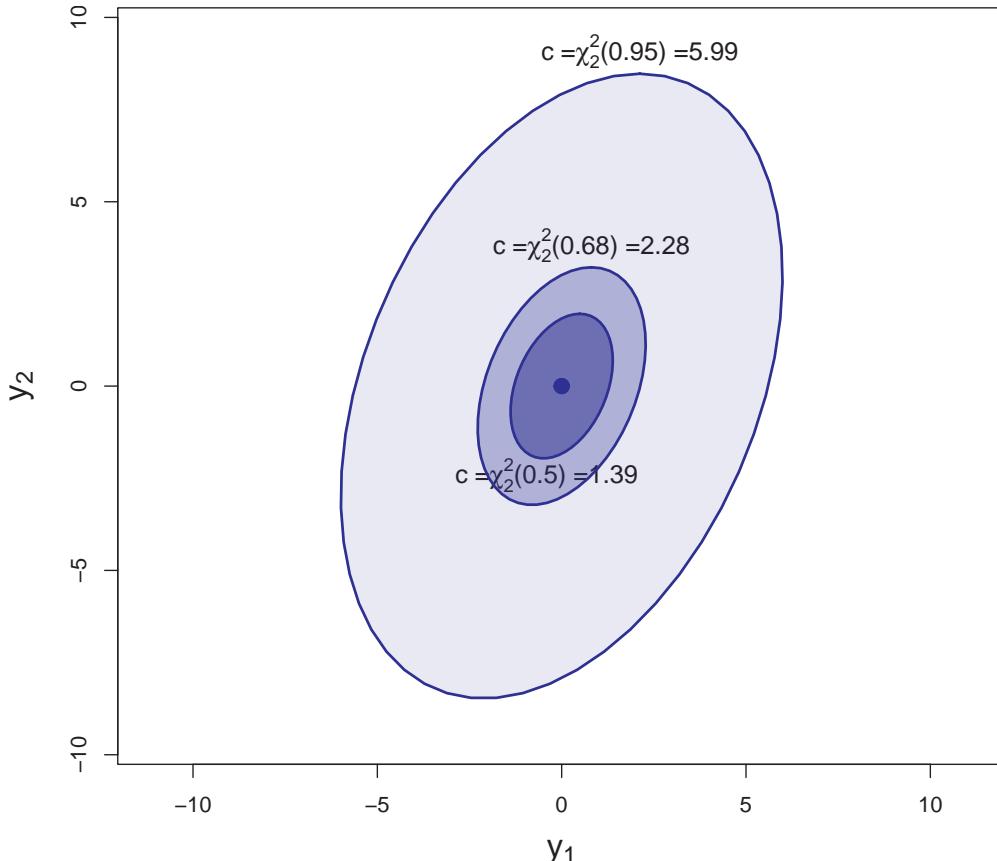


Figure 3.9: Data ellipses of 50%, 68% and 95% coverage when the means are $\bar{\mathbf{y}} = \mathbf{0}$ and the variance-covariance matrix is \mathbf{S} .

As always, graphic details matter. Figure 3.9 uses `asp = 1` so that units in the plot are the same for y_1 and y_2 , and we can see the greater variance for y_2 as well as the correlation. Facilities of `grDevices::plotmath()` are used to provide mathematical annotations in the plot.

3.2.1 Ellipse properties

The essential ideas of correlation and regression and their relation to ellipses go back to Galton (1886). Galton's goal was to predict (or explain) how a heritable trait, Y , (e.g., height) of children was related to that of their parents, X . He made a semi-graphic table of the frequencies of 928 observations of the average height of father and mother versus the height of their child, shown in Figure 3.10. (Today, we would put child height on the y axis, but Galton was working from the table, so he organized it with parent height as the rows.) He then drew smoothed contour lines of equal frequencies and had the wonderful visual insight that these formed concentric shapes that were tolerably close to ellipses.

He then calculated summaries, $\text{Ave}(Y|X)$, and, for symmetry, $\text{Ave}(X|Y)$, and plotted these as lines of means on his diagram. Lo and behold, he had a second visual insight: the lines of means of $(Y|X)$ and $(X|Y)$ corresponded approximately to the loci of horizontal and vertical tangents to the concentric ellipses. To

complete the picture, he added lines showing the geometric major and minor axes of the family of ellipses (which turned out to be the principal components) with the result shown in Figure 3.10.

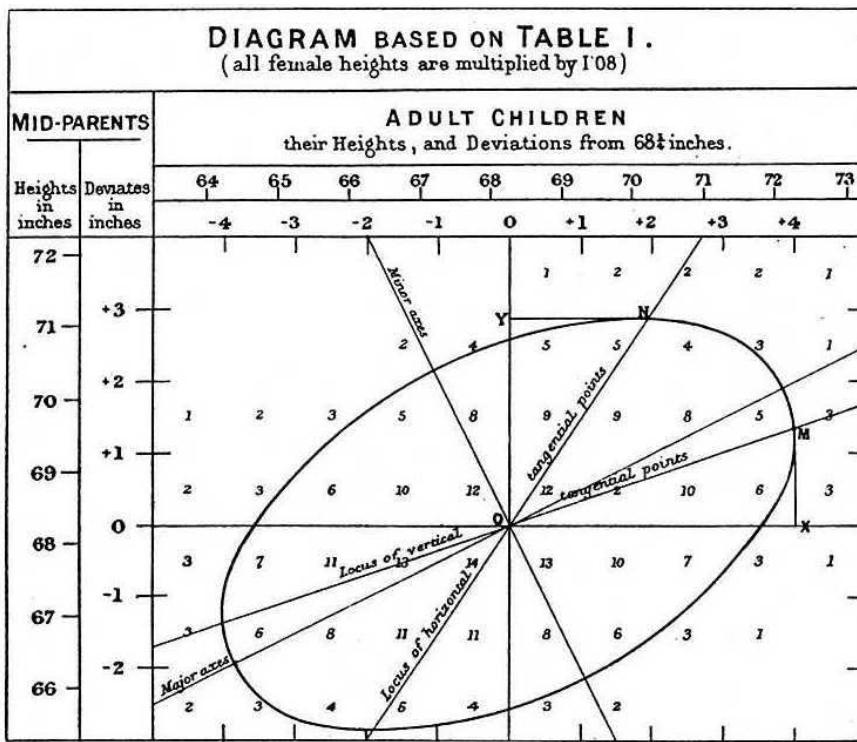


Figure 3.10: Galton's 1886 diagram, showing the relationship of height of children to the average of their parents' height. The diagram is essentially an overlay of a geometrical interpretation on a bivariate grouped frequency distribution, shown as numbers.

For two variables, x and y , the remarkable properties of the data ellipse are illustrated in Figure 3.11, a modern reconstruction of Galton's diagram.

```

data(Galton, package = "HistData")
sunflowerplot(parent ~ child, data=Galton,
  xlim=c(61,75),
  ylim=c(61,75),
  seg.col="black",
  xlab="Child height",
  ylab="Mid Parent height")

y.x <- lm(parent ~ child, data=Galton)      # regression of y on x
abline(y.x, lwd=2)
x.y <- lm(child ~ parent, data=Galton)      # regression of x on y
cc <- coef(x.y)
abline(-cc[1]/cc[2], 1/cc[2], lwd=2, col="gray")

with(Galton,
  car::dataEllipse(child, parent,
    plot.points=FALSE,
    levels=c(0.40, 0.68, 0.95),
    )
  )

```

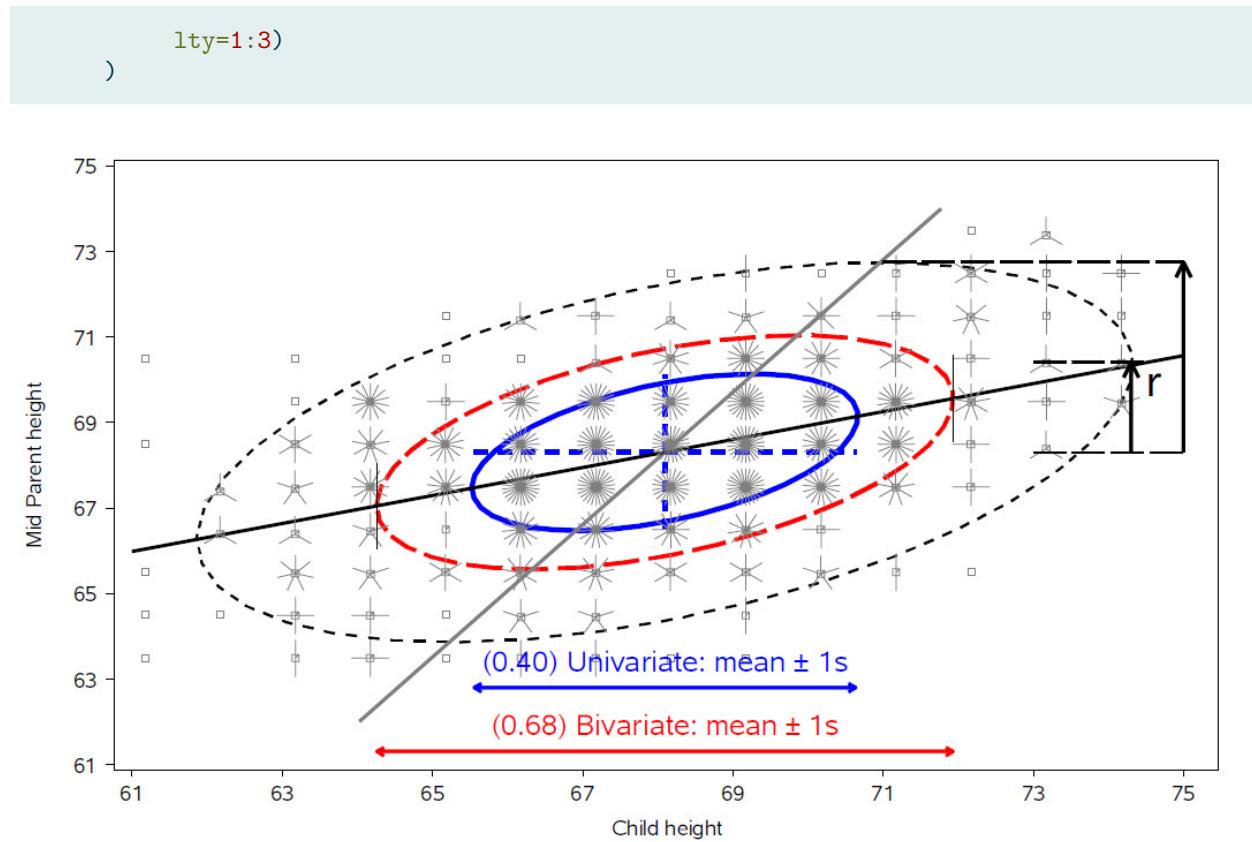


Figure 3.11: Sunflower plot of Galton’s data on heights of parents and their children (in.), with 40%, 68% and 95% data ellipses and the regression lines of y on x (black) and x on y (grey).

- The ellipses have the mean vector (\bar{x}, \bar{y}) as their center.
- The lengths of arms of the blue dashed central cross show the standard deviations of the variables, which correspond to the shadows of the ellipse covering 40% of the data. These are the bivariate analogs of the standard intervals $\bar{x} \pm 1s_x$ and $\bar{y} \pm 1s_y$.
- More generally, shadows (projections) on the coordinate axes, or any linear combination of them, give any standard interval, $\bar{x} \pm ks_x$ and $\bar{y} \pm ks_y$. Those with $k = 1, 1.5, 2.45$, have bivariate coverage 40%, 68% and 95% respectively, corresponding to these quantiles of the χ^2 distribution with 2 degrees of freedom, i.e., $\chi^2_{(0.40)} \approx 1^2$, $\chi^2_{(0.68)} \approx 1.5^2$, and $\chi^2_{(0.95)} \approx 2.45$. The shadows of the 68% ellipse are the bivariate analog of a univariate $\bar{x} \pm 1s_x$ interval.
- The regression line predicting y from x goes through the points where the ellipses have vertical tangents. The other regression line, predicting x from y goes through the points of horizontal tangency.
- The correlation $r(x, y)$ is the ratio of the vertical segment from the mean of y to the regression line to the vertical segment going to the top of the ellipse as shown at the right of the figure. It is $r = 0.46$ in this example.
- The residual standard deviation, $s_e = \sqrt{MSE} = \sqrt{\sum(y - \bar{y})^2/n - 2}$, is the half-length of the ellipse at the mean \bar{x} .

Because Galton’s values of `parent` and `child` height were recorded in class intervals of 1 in., they are shown as sunflower symbols in Figure 3.11, with multiple ‘petals’ reflecting the number of observations at each location. This plot (except for annotations) is constructed using `sunflowerplot()` and `car::dataEllipse()` for the ellipses.

```

data(Galton, package = "HistData")

sunflowerplot(parent ~ child, data=Galton,
  xlim=c(61,75),
  ylim=c(61,75),
  seg.col="black",
  xlab="Child height",
  ylab="Mid Parent height")

y.x <- lm(parent ~ child, data=Galton)      # regression of y on x
abline(y.x, lwd=2)
x.y <- lm(child ~ parent, data=Galton)      # regression of x on y
cc <- coef(x.y)
abline(-cc[1]/cc[2], 1/cc[2], lwd=2, col="gray")

with(Galton,
  car::dataEllipse(child, parent,
    plot.points=FALSE,
    levels=c(0.40, 0.68, 0.95),
    lty=1:3)
)

```

Finally, as Galton noted in his diagram, the principal major and minor axes of the ellipse have important statistical properties. Pearson (1901) would later show that their directions are determined by the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots$ of the covariance matrix \mathbf{S} and their radii by the square roots, $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots$ of the corresponding eigenvalues.

3.2.2 R functions for data ellipses

A number of packages provide functions for drawing data ellipses in a scatterplot, with various features.

- `car::scatterplot()`: uses base R graphics to draw 2D scatterplots, with a wide variety of plot enhancements including linear and non-parametric smoothers (loess, gam), a formula method, e.g., `y ~ x | group`, and marking points and lines using symbol shape, color, etc. Importantly, the `car` package generally allows automatic identification of “noteworthy” points by their labels in the plot using a variety of methods. For example, `method = "mahal"` labels cases with the most extreme Mahalanobis distances; `method = "r"` selects points according to their value of `abs(y)`, which is appropriate in residual plots.
- `car::dataEllipse()`: plots classical or robust data ellipses for one or more groups, with the same facilities for point identification. The robust version (`robust=TRUE`) uses the multivariate t distribution (using `MASS::cov.trob()`) rather than the Gaussian,
- `heplots::covEllipses()`: draws classical or robust data ellipses for one or more groups in a one-way design and optionally for the pooled total sample, where the focus is on homogeneity of within-group covariance matrices.
- `ggplot2::stat_ellipse()`: uses the calculation methods of `car::dataEllipse()` to add unfilled (`geom = "path"`) or filled (`geom = polygon`) data ellipses in a `ggplot` scatterplot, using inherited aesthetics.

Example 3.2. Canadian occupational prestige

These examples use the data on the prestige of 102 occupational categories and other measures from the 1971 Canadian Census, recorded in `Prestige`.³ Our interest is in understanding how `prestige` (the Pineo & Porter (2008) prestige score for an occupational category, derived from a social survey) is related to census

³The dataset was collected by Bernard Blishen, William Carroll and Catherine Moore, but apparently unpublished. A version updated to the 1981 census is described in Blishen et al. (1987).

measures of the average education, income, percent women of incumbents in those occupations. Occupation type is a factor with levels "bc" (blue collar), "wc" (white collar) and "prof" (professional).

```
data(Prestige, package="carData")
# `type` is really an ordered factor. Make it so.
Prestige$type <- ordered(Prestige$type,
                           levels=c("bc", "wc", "prof"))
str(Prestige)
#> 'data.frame': 102 obs. of 6 variables:
#>   $ education: num 13.1 12.3 12.8 11.4 14.6 ...
#>   $ income    : int 12351 25879 9271 8865 8403 11030 8258 14163 11377 11023 ...
#>   $ women     : num 11.16 4.02 15.7 9.11 11.68 ...
#>   $ prestige   : num 68.8 69.1 63.4 56.8 73.5 77.6 72.6 78.1 73.1 68.8 ...
#>   $ census    : int 1113 1130 1171 1175 2111 2113 2133 2141 2143 2153 ...
#>   $ type      : Ord.factor w/ 3 levels "bc" <"wc" <"prof": 3 3 3 3 3 3 3 3 3 ...
```

I first illustrate the relation between `income` and `prestige` in Figure 3.12 using `car::scatterplot()` with many of its bells and whistles, including marginal boxplots for the variables, the linear regression line, loess smooth and the 68% data ellipse.

```
scatterplot(prestige ~ income, data=Prestige,
            pch = 16, cex.lab = 1.25,
            regLine = list(col = "red", lwd=3),
            smooth = list(smoother=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "darkgreen",
                          col.var = "darkgreen"),
            ellipse = list(levels = 0.68),
            id = list(n=4, method = "mahal", col="black", cex=1.2))
#> general.managers      lawyers      ministers      physicians
#>                 2             17            20            24
```

There is a lot that can be seen here:

- `income` is positively skewed, as is often the case.
- The loess smooth, on the scale of income, shows `prestige` increasing up to \$15,000 (these are 1971 incomes), and then leveling off.
- The data ellipse, centered at the means encloses approximately 68% of the data points. It adds visual information about the correlation and precision of the linear regression; but here, the non-linear trend for higher incomes strongly suggests a different approach.
- The four points identified by their labels are those with the largest Mahalanobis distances. `scatterplot()` prints their labels to the console.

Figure 3.13 shows a similar plot for education, which from the boxplot appears to be reasonably symmetric. The smoothed curve is quite close to the linear regression, according to which `prestige ~ education`, `data=Prestige`)`["education"] = 5.361` with each year of education.

```
scatterplot(prestige ~ education, data=Prestige,
            pch = 16, cex.lab = 1.25,
            regLine = list(col = "red", lwd=3),
            smooth = list(smoother=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "darkgreen",
```

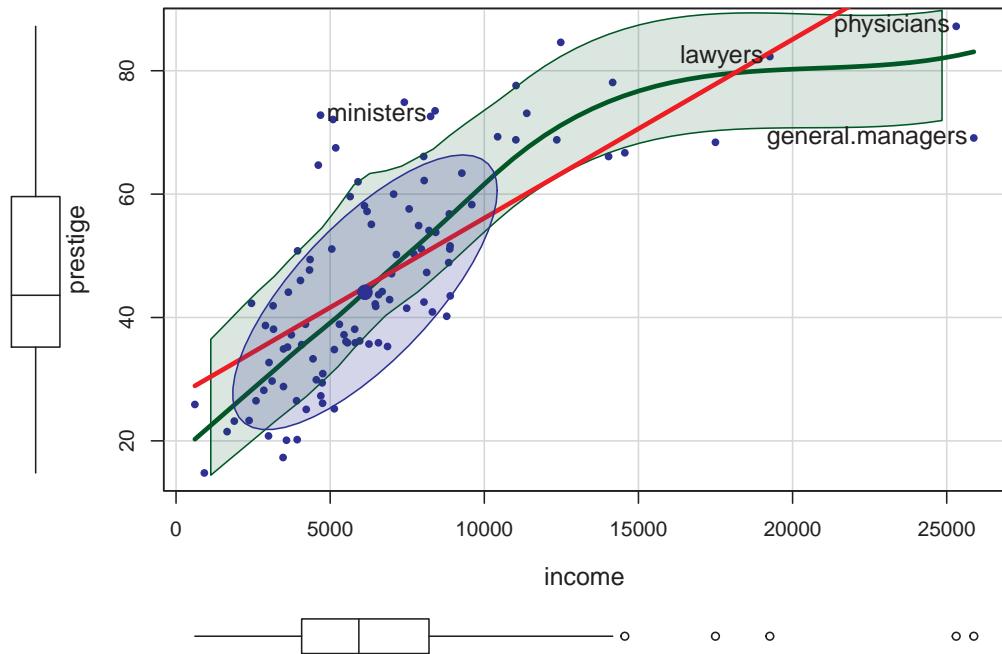


Figure 3.12: Scatterplot of prestige vs. income, showing the linear regression line (red), the loess smooth with a confidence envelope (darkgreen) and a 68% data ellipse. Points with the 4 largest D^2 values are labeled.

```

col.var = "darkgreen",
ellipse = list(levels = 0.68),
id = list(n=4, method = "mahal", col="black", cex=1.2))
#> physicians file.clerks    newsboys    farmers
#>        24          41          53          67

```

In this plot, farmers, newsboys, file.clerks and physicians are identified as noteworthy, for being furthest from the mean by Mahalanobis distance. In relation to their typical level of education, these are mostly understandable, but it is nice that farmers are rated of higher prestige than their level of education would predict.

Note that the `method` argument for point identification can take a vector of case numbers indicating the points to be labeled. So, to label the observations with large absolute standardized residuals in the linear model `m`, you can use `method = which(abs(rstandard(m)) > 2)`.

```

m <- lm(prestige ~ education, data=Prestige)
scatterplot(prestige ~ education, data=Prestige,
            pch = 16, cex.lab = 1.25,
            boxplots = FALSE,
            regLine = list(col = "red", lwd=3),
            smooth = list(smoother=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "black",
                          col.var = "darkgreen"),
            ellipse = list(levels = 0.68),
            id = list(n=4, method = which(abs(rstandard(m))>2),
                      col="black", cex=1.2)) |> invisible()

```

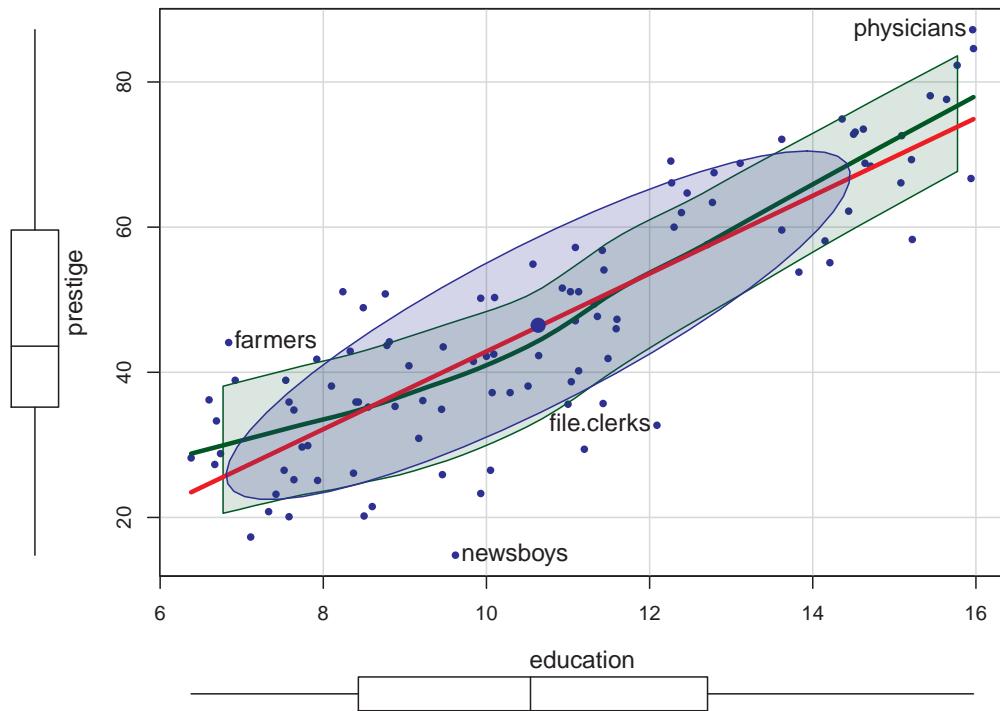


Figure 3.13: Scatterplot of prestige vs. education, showing the linear regression line (red), the loess smooth with a confidence envelope (darkgreen) and a 68% data ellipse.

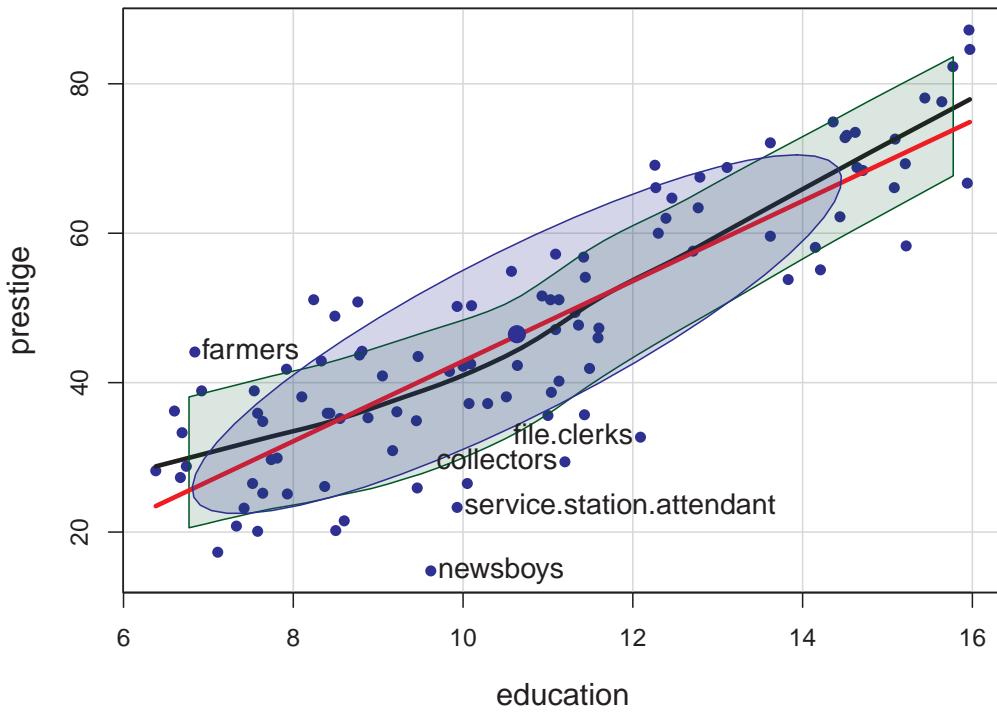


Figure 3.14: Scatterplot of prestige vs. education, labeling points whose absolute standardized residual is > 2 .

3.2.2.1 Plotting on a log scale

A typical remedy for the non-linear relationship of income to prestige is to plot income on a log scale. This usually makes sense, and expresses a belief that a **multiple** of or **percentage increase** in income has a constant impact on prestige, as opposed to the **additive** interpretation for income itself.

For example, the slope of the linear regression line in Figure 3.12 is given by `coef(lm(prestige ~ income, data=Prestige))["income"] = 0.003`. Multiplying this by 1000 says that a \$1000 increase in `income` is associated with an average increase of `prestige` of 2.9.

In the plot below, `scatterplot(..., log = "x")` re-scales the x-axis to the $\log_e()$ scale. The slope, `coef(lm(prestige ~ log(income), data=Prestige))["log(income)"] = 21.556` says that a 1% increase in salary is associated with an average change of 21.55 / 100 in prestige.

```
scatterplot(prestige ~ income, data=Prestige,
            log = "x",
            pch = 16, cex.lab = 1.25,
            regLine = list(col = "red", lwd=3),
            smooth = list(smooth=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "darkgreen", col.var = "darkgreen"),
            ellipse = list(levels = 0.68),
            id = list(n=4, method = "mahal", col="black", cex=1.2))
#> general.managers      ministers      newsboys      babysitters
#>                      2             20            53            63
```

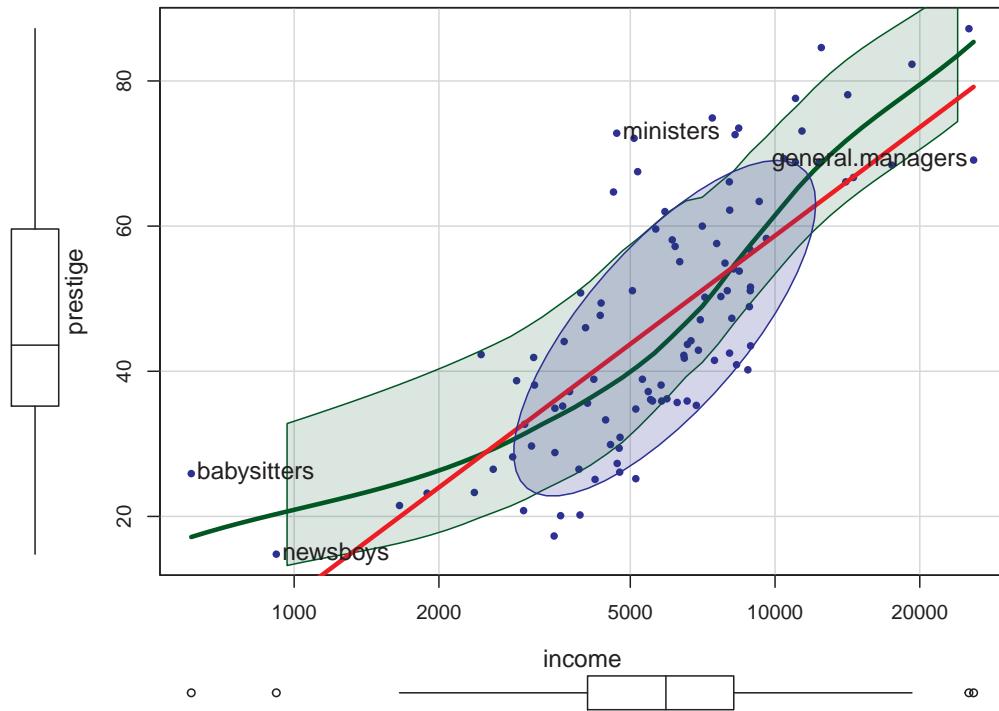


Figure 3.15: Scatterplot of prestige vs. $\log(\text{income})$.

The smoothed curve in Figure 3.15 exhibits a slight tendency to bend upwards, but a linear relation is a reasonable approximation.

3.2.2.2 Stratifying

Before going further, it is instructive to ask what we could see in the relationship between income and prestige if we stratified by type of occupation, fitting separate regressions and smooths for blue collar, white collar and professional incumbents in these occupations.

The formula `prestige ~ income | type` (read: income *given* type) is a natural way to specify grouping by `type`; separate linear regressions and smooths are calculated for each group, applying the color and point shapes specified by the `col` and `pch` arguments.

```
scatterplot(prestige ~ income | type, data=Prestige,
            col = c("blue", "red", "darkgreen"),
            pch = 15:17,
            grid = FALSE,
            legend = list(coords="bottomright"),
            regLine = list(lwd=3),
            smooth=list(smoothers=loessLine,
                        var=FALSE, lwd.smooth=2, lty.smooth=1))
```

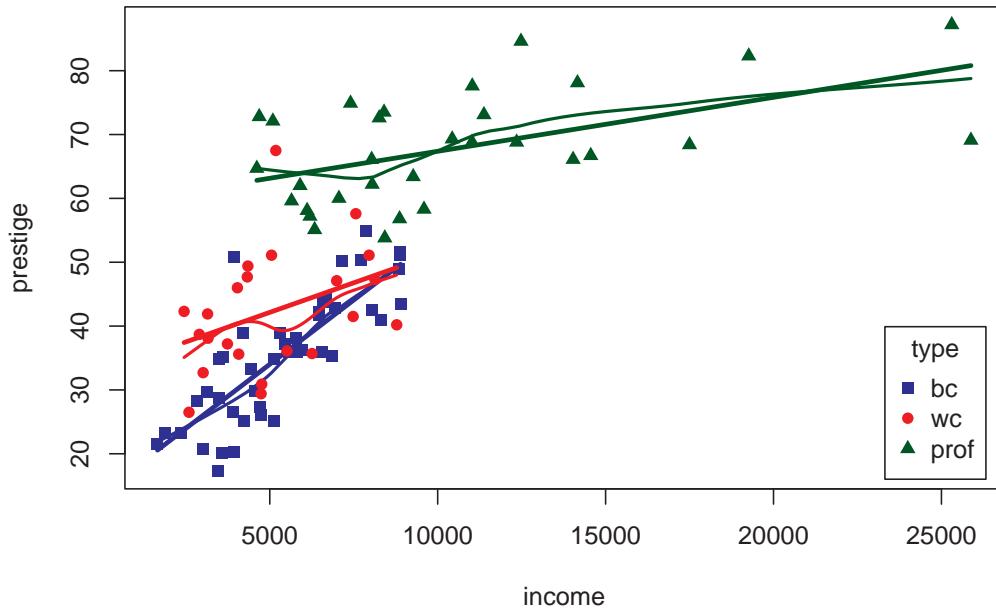


Figure 3.16: Scatterplot of prestige vs. income, stratified by occupational type. This implies a different interpretation, where occupation type is a moderator variable.

This visual analysis offers a different interpretation of the dependence of prestige on income, which appeared to be non-linear when occupation type was ignored. Instead, Figure 3.16 suggests an *interaction* of income by type. In a model formula this would be expressed as one of:

```
lm(prestige ~ income + type + income:type, data = Prestige)
lm(prestige ~ income * type, data = Prestige)
```

These models signify that there are different slopes (and intercepts) for the three occupational types. In this interpretation, `type` is a moderator variable, with a different story. The slopes of the fitted lines suggest that among blue collar workers, prestige increases sharply with their income. For white collar and professional workers, there is still an increasing relation of prestige with income, but the effect of income (slope) diminishes with higher occupational category. A different plot entails a different story.

3.2.3 Example: Penguins data

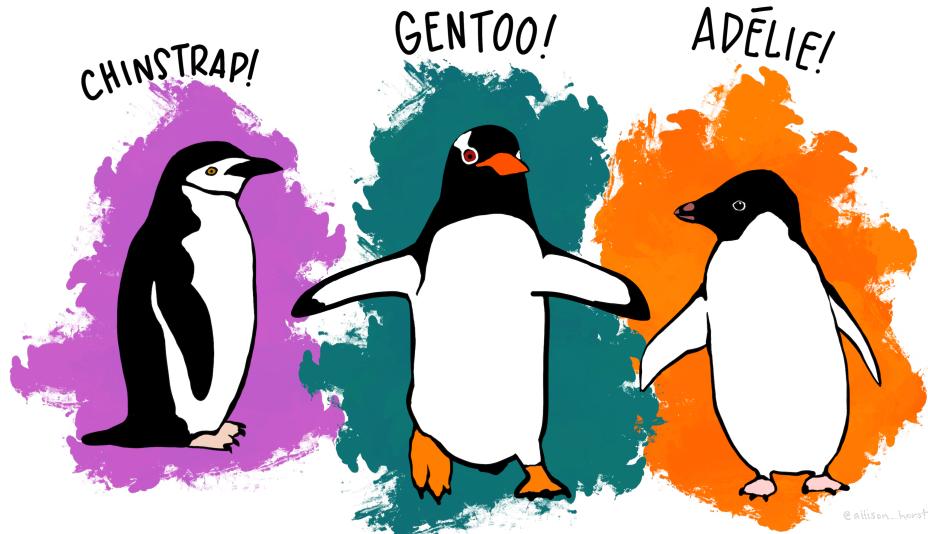


Figure 3.17: Penguin species observed in the Palmer Archipelago. This is a cartoon, but it illustrates some features of penguin body size measurements, and the colors typically used for species. Image: Allison Horst

The `penguins` dataset from the [palmerpenguins](#) package (Horst et al., 2022) provides further instructive examples of plots and analyses of multivariate data. The data consists of measurements of body size (flipper length, body mass, bill length and depth) of 344 penguins collected at the [Palmer Research Station](#) in Antarctica.

There were three different species of penguins (Adélie, Chinstrap & Gentoo) collected from 3 islands in the Palmer Archipelago between 2007–2009 (Gorman et al., 2014). The purpose was to examine differences in size or appearance of these species, particularly differences among the sexes (sexual dimorphism) in relation to foraging and habitat.

Here, I use a slightly altered version of the dataset, `heplots::peng`, constructed by renaming variables to remove the units, making factors of character variables and deleting a few cases with missing data.⁴

```
data(penguins, package = "palmerpenguins")
peng <- penguins |>
  rename(
    bill_length = bill_length_mm,
    bill_depth = bill_depth_mm,
    flipper_length = flipper_length_mm,
    body_mass = body_mass_g
  ) |>
  mutate(species = as.factor(species),
         island = as.factor(island),
         sex = as.factor(substr(sex,1,1))) |>
  tidyr::drop_na()

str(peng)
```

⁴In R 4.5.0, a revised version of the `palmerpenguins` dataset, named `penguins` was added to the base R `datasets`. This corrected a few of the problems that led me to create `heplots::peng`, but still retains the cases with missing data.

```
#> tibble [333 x 8] (S3: tbl_df/tbl/data.frame)
#> $ species      : Factor w/ 3 levels "Adelie","Chinstrap",...: 1 1 1 1 1 1 1 1 1 ...
#> $ island       : Factor w/ 3 levels "Biscoe","Dream",...: 3 3 3 3 3 3 3 3 3 ...
#> $ bill_length  : num [1:333] 39.1 39.5 40.3 36.7 39.3 38.9 39.2 41.1 38.6 34.6 ...
#> $ bill_depth   : num [1:333] 18.7 17.4 18 19.3 20.6 17.8 19.6 17.6 21.2 21.1 ...
#> $ flipper_length: int [1:333] 181 186 195 193 190 181 195 182 191 198 ...
#> $ body_mass    : int [1:333] 3750 3800 3250 3450 3650 3625 4675 3200 3800 4400 ...
#> $ sex          : Factor w/ 2 levels "f","m": 2 1 1 1 2 1 2 1 2 2 ...
#> $ year         : int [1:333] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

There are quite a few variables to choose for illustrating data ellipses in scatterplots. Here I focus on the measures of their bills, `bill_length` and `bill_depth` (indicating curvature) and show how to use `ggplot2` for these plots.

I'll be using the penguins data quite a lot, so it is useful to set up custom colors like those used in Figure 3.17, and shown in Figure 3.18 with their color codes. These are shades of:

- Adelie: orange,
- Chinstrap: purple, and
- Gentoo: green.

Adelie	Chinstrap	Gentoo
#FDBF6F	#CAB2D6	#B2DF8A
#F89D38	#9A78B8	#73C05B
#F37A00	#6A3D9A	#33a02c

Figure 3.18: Color palettes used for penguin species.

To use these in `ggplot2` I define a function `peng.colors()` that allows shades of light, medium and dark and then functions `scale_*_penguins()` for color and fill.

```
peng.colors <- function(shade=c("medium", "light", "dark")) {
  shade = match.arg(shade)
  #           light     medium     dark
  oranges <- c("#FDBF6F", "#F89D38", "#F37A00") # Adelie
  purples <- c("#CAB2D6", "#9A78B8", "#6A3D9A") # Chinstrap
  greens <- c("#B2DF8A", "#73C05B", "#33a02c") # Gentoo
```

```

cols.vec <- c(oranges, purples, greens)
cols.mat <-
  matrix(cols.vec, 3, 3,
         byrow = TRUE,
         dimnames = list(species = c("Adelie", "Chinstrap", "Gentoo"),
                         shade = c("light", "medium", "dark")))
# get shaded colors
cols.mat[, shade ]
}

# define color and fill scales
scale_fill_penguins <- function(shade=c("medium", "light", "dark"), ...){
  shade = match.arg(shade)
  ggplot2::discrete_scale(
    "fill", "penguins",
    scales::manual_pal(values = peng.colors(shade)), ...)
}

scale_colour_penguins <- function(shade=c("medium", "light", "dark"), ...){
  shade = match.arg(shade)
  ggplot2::discrete_scale(
    "colour", "penguins",
    scales::manual_pal(values = peng.colors(shade)), ...)
}
scale_color_penguins <- scale_colour_penguins

```

This is used to define a `theme_penguins()` function that I use to simply change the color and fill scales for plots below.

```

theme_penguins <- function(shade=c("medium", "light", "dark"), ...) {
  shade = match.arg(shade)
  list(scale_color_penguins(shade=shade),
       scale_fill_penguins(shade=shade)
     )
}

```

An initial plot using `ggplot2` shown in Figure 3.19 uses color and point shape to distinguish the three penguin species. I annotate the plot of points using the linear regression lines, loess smooths to check for non-linearity and 95% data ellipses to show precision of the linear relation.

```

ggplot(peng,
       aes(x = bill_length, y = bill_depth,
           color = species, shape = species, fill=species)) +
  geom_point(size=2) +
  geom_smooth(method = "lm", formula = y ~ x,
              se=FALSE, linewidth=2) +
  geom_smooth(method = "loess", formula = y ~ x,
              linewidth = 1.5, se = FALSE, alpha=0.1) +
  stat_ellipse(geom = "polygon", level = 0.95, alpha = 0.2) +
  theme_penguins("dark") +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.15))

```

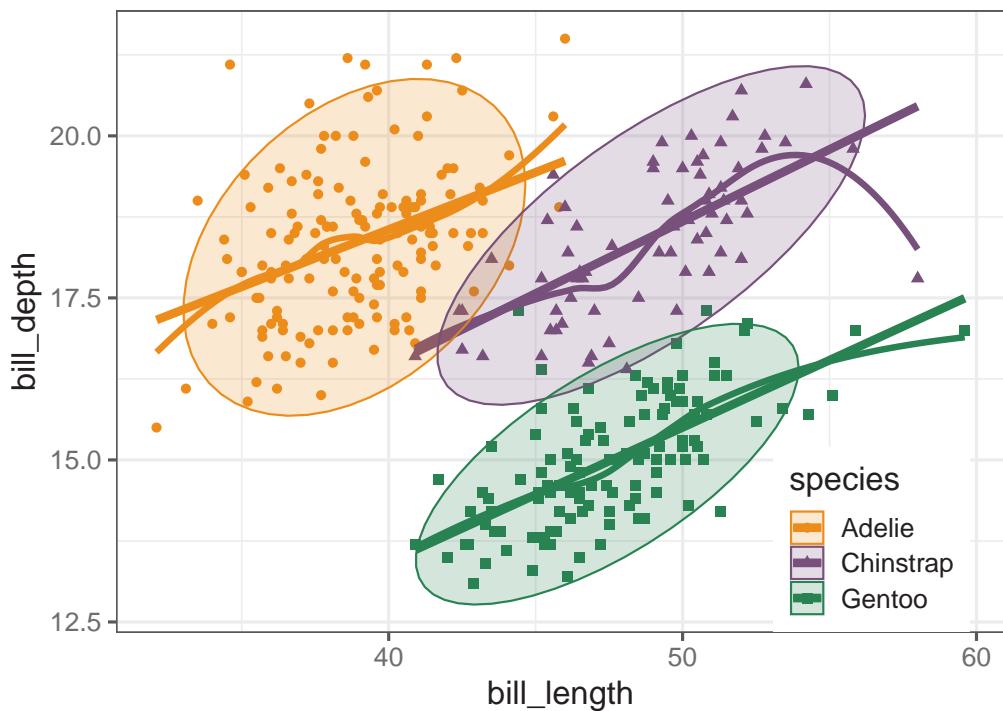


Figure 3.19: Penguin bill length and bill depth according to species.

3.2.4 Visual thinning

Overall, the three species occupy different regions of this 2D space and for each species the relation between bill length and depth appears reasonably linear. Given this, we can suppress plotting the data points to get a visual summary of the data using the fitted regression lines and data ellipses, as shown in Figure 3.20.

This idea, of **visual thinning** a graph to focus on what should be seen, becomes increasingly useful as the data becomes more complex. The `ggplot2` framework encourages this, because we can think of various components as layers, to be included or not. In Figure 3.20 I chose to include only the regression line and add data ellipses of 40%, 68% and 95% coverage to highlight the increasing bivariate density around the group means.

```
ggplot(peng,
       aes(x = bill_length, y = bill_depth,
           color = species, shape = species, fill=species)) +
  geom_smooth(method = "lm", se=FALSE, linewidth=2) +
  stat_ellipse(geom = "polygon", level = 0.95, alpha = 0.2) +
  stat_ellipse(geom = "polygon", level = 0.68, alpha = 0.2) +
  stat_ellipse(geom = "polygon", level = 0.40, alpha = 0.2) +
  theme_penguins("dark") +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.15))
```

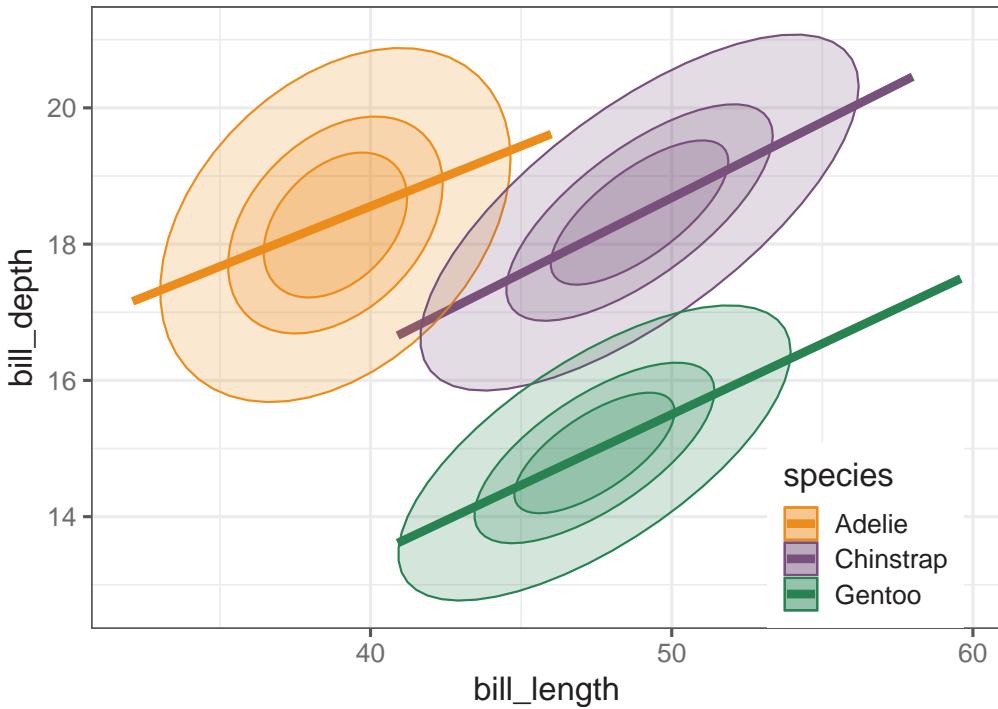


Figure 3.20: Visual thinning: Suppressing the data points gives a visual summary of the relation between bill length and bill depth using the regression line and data ellipses.

3.3 Bagplots

If you are concerned about the assumption of bivariate normality entailed by the data ellipse, a very nice non-parametric and robust alternative is a 2D generalization of a boxplot called a **bagplot**, introduced by Peter J. Rousseeuw et al. (1999). The idea is very simple. The bagplot consists of three nested polygons, called the “bag”, the “fence”, and the “loop”:

- **bag:** The central 50% box of the boxplot is replaced by a polygon called the “bag”, constructed on the basis of *depth* of points from the bivariate depth median point. Depth generalizes the univariate concept of rank, but counted from the medians point outward in any direction. The bag contains at most 50% of the most central data points.
- **fence:** The univariate fences are replaced by a “fence” polygon, by expanding the bag outward by a factor (`coef`), usually 3, from the depth median.
- **loop:** Points beyond the fence (the “loop”) are potential outliers, sometimes plotted as a convex hull surrounding *all* of the observations, but better rendered simply as a scatterplot of just those points.

In this way, the bagplot visualizes the bivariate location (median), spread, correlation, skewness, and tails of the data. Compared with a standard data ellipse, it serves as a visual test of normality and provides a simple way to identify outliers.

Bagplots are implemented in `gggda` (Brunson & Gracey, 2025) in the `ggplot2` framework as `geom_bagplot()` with computations done via `stat_bagplot()`. For the Penguin data, the bagplot version of Figure 3.20 is shown in Figure 3.21.

```
ggplot(peng,
       aes(x = bill_length, y = bill_depth,
           color = species, shape = species, fill=species)) +
  geom_smooth(method = "lm", formula = y ~ x,
              se=FALSE, linewidth=2) +
  geom_bagplot(bag.alpha = 0.5,
               outlier.size = 5,
               fraction = 0.5,      # bag fraction
               coef = 2.5,          # fence factor
               show.legend = FALSE) +
  theme_penguins("dark") +
  theme(legend.position = "inside",
        legend.position.inside = c(0.87, 0.15))
```

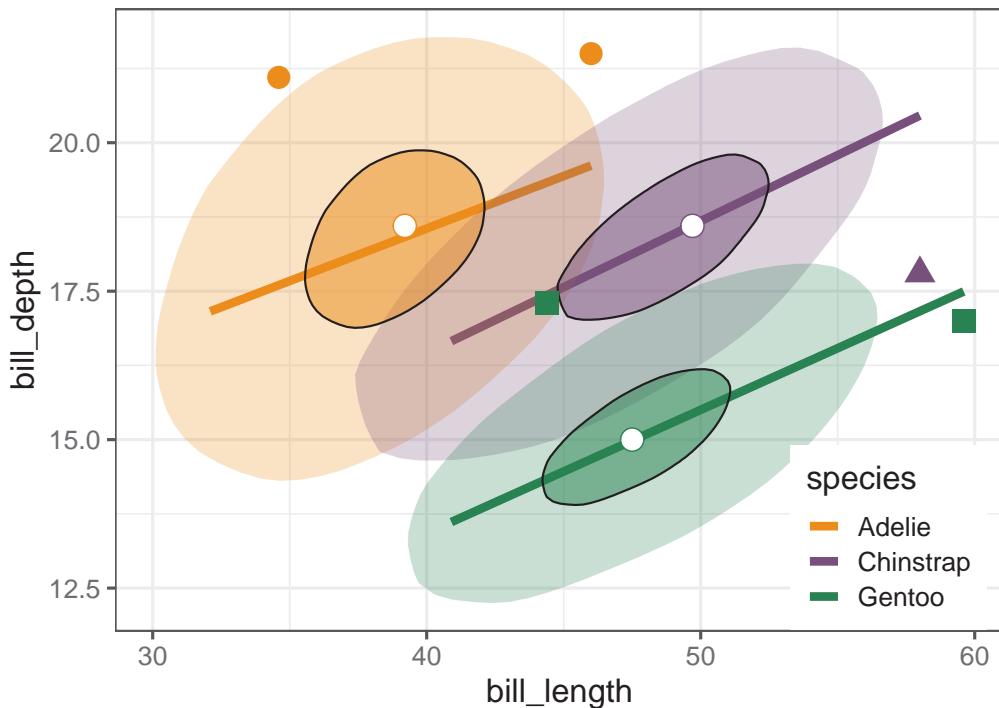


Figure 3.21: Bagplot: For each Penguin species the darker inner (bag) polygon reflects the innermost 50% of the data points. The outer (fence) polygon corresponds to points enclosed by a multiple of the bag. Points outside the fence for each species are plotted individually.

Compared with the data ellipses shown in Figure 3.20, the bag polygons have shapes very close to ellipses, giving credence to the assumption of (approximate) normality. Using the factor `coef = 2.5` causes five points to be flagged as potential outliers. Bivariate skewness is indicated when the bag and fence is not symmetric around the central median in some direction, as is slightly true for all three species.

I discuss other visual tests of multivariate normality in Section 3.9 and consider outlier identification for the Penguin data in Section 3.9.2 below.

3.4 Non-parametric bivariate density plots

While I emphasize data ellipses (because I like their beautiful geometry), other visual summaries of the bivariate density are possible and often useful. To see more detail about the “shape” of bivariate data in a non-parametric way you can use one of the methods described below to see a model-free representation of your data.

For a single variable, `stats::density()` and `ggplot2::geom_density()` calculate a smoothed estimate of the density using nonparametric kernel methods (Silverman, 1986) whose smoothness is controlled by a bandwidth parameter, analogous to the span in a loess smoother. This idea extends to two (and more) variables (Scott, 1992). For bivariate data, `MASS::kde2d()` estimates the density on a square $n \times n$ grid over the ranges of the variables.

`ggplot2` provides `geom_density_2d()` which uses `MASS::kde2d()` and displays these as contours—horizontal slices of the 3D surface at equally-spaced heights and projects these onto the 2D plane. The `ggdensity` package (Otto & Kahle, 2023) extends this with `geom_hdr()`, computing the high density regions that bound given levels of probability and maps these to the `alpha` transparency aesthetic. A `method` argument allows you to specify various nonparametric (`method = "kde"` is the default) and parametric (`method = "mvnorm"` gives normal data ellipses) ways to estimate the underlying bivariate distribution.

Figure 3.22 shows these side-by-side for comparison. With `geom_density_2d()` you can specify either the number of contour `bins` or the width of these bins (`binwidth`). For `geom_hdr()`, the `probs` argument gives a result that is easier to understand.

```
library(ggdensity)
library(patchwork)
p1 <- ggplot(peng,
  aes(x = bill_length, y = bill_depth,
      color = species)) +
  geom_smooth(method = "lm", se=FALSE, linewidth=2) +
  geom_density_2d(binwidth = 1.1, bins = 8) +
  ggtitle("geom_density_2d") +
  theme_bw(base_size = 14) +
  theme_penguins() +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.15))

p2 <- ggplot(peng,
  aes(x = bill_length, y = bill_depth,
      color = species, fill = species)) +
  geom_smooth(method = "lm", se=FALSE, linewidth=2) +
  geom_hdr(probs = c(0.95, 0.68, 0.4), show.legend = FALSE) +
  ggtitle("ggdensity::geom_hdr") +
  theme_bw(base_size = 14) +
  theme_penguins() +
  theme(legend.position = "none")

p1 + p2
```

Compared with the data ellipse, which is highly smoothed by the Gaussian assumption and the bagplot, which is smoothed by the concept of contours of data depth, the plots in Figure 3.22 show considerably more detail, and an intriguing suggestion of two peaks for our Chinstrap penguins.

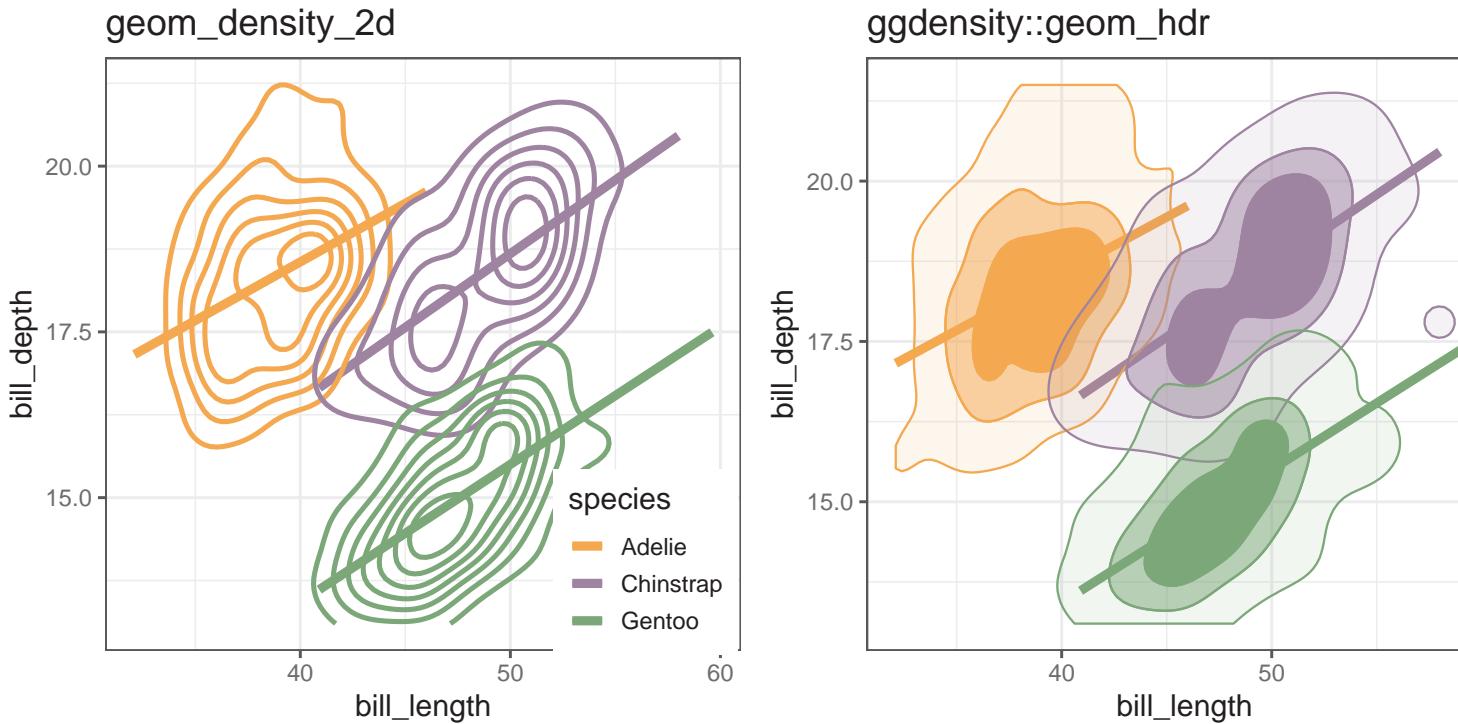


Figure 3.22: Bivariate densities show the contours of the 3D surface representing the frequency in the joint distribution of bill length and bill depth.

3.5 Simpson's paradox: marginal and conditional relationships

Because it provides a visual representation of means, variances, and correlations, the data ellipse is ideally suited as a tool for illustrating and explicating various phenomena that occur in the analysis of linear models. One class of simple, but important, examples concerns the difference between the *marginal relationship* between variables, ignoring some important factor or covariate, and the *conditional* relationship, adjusting (controlling) for that variable.

An important example is **Simpson's paradox** (Simpson, 1951) which occurs when the marginal and conditional relationships differ in direction. That is, the overall correlation in a model $y \sim x$ might be negative, while the within-group correlations in separate models for each group $y[g] \sim x[g]$ might be positive, or vice versa. For Flatlanders in 2D space, this is a puzzlement.

We can see this paradox in the plots of bill length against bill depth for the penguin data shown in Figure 3.23. Ignoring penguin species, the marginal, total-sample correlation is slightly negative as seen in panel (a). The individual-sample ellipses in panel (b) show that the conditional, within-species correlations are all positive, with approximately equal regression slopes. However the group means have a negative relationship, accounting for the negative marginal correlation when species is ignored.

The regression line in panel (a) is that for the linear model `lm(bill_depth ~ bill_length)`, while the separate lines in panel (b) are those for the model `lm(bill_depth ~ bill_length * species)` which allows a different slope and intercept for each species.

A correct analysis of the (conditional) relationship between these variables, controlling or adjusting for mean differences among species, is based on the pooled within-sample covariance matrix, a weighted average of the individual within-group \mathbf{S}_i ,

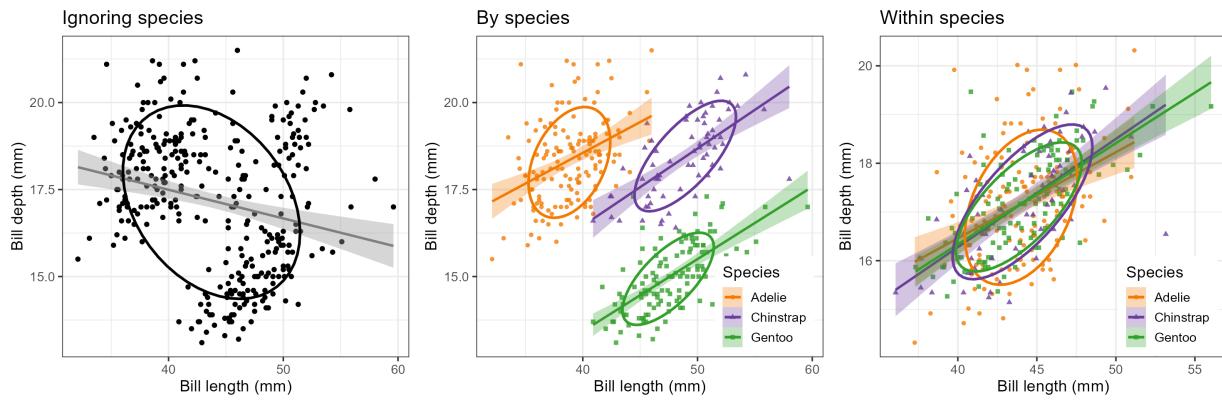


Figure 3.23: Marginal (a), conditional (b), and pooled within-sample (c) relationships of bill length and depth in the Penguins data. Each plot shows the 68% data ellipse and regression line(s) with 95% confidence bands.

$$\mathbf{S}_{\text{within}} = \sum_{i=1}^g (n_i - 1) \mathbf{S}_i / (N - g) ,$$

where $N = \sum n_i$. The result is shown in panel (c) of Figure 3.23.

In this graph, the data for each species were first transformed to deviations from the species means on both variables and then translated back to the grand means. You can also see here that the shapes and sizes of the individual data ellipses are roughly comparable, but perhaps not identical. This visual idea of centering groups to a common mean will become important in Chapter 12 when we want to test the assumption of equality of error covariances in multivariate models.

The `ggplot2` code for the panels in this figure are shown below. Note that for components that will be the same across panels, you can define elements (e.g., `labels`, `theme_penguins()`, `legend_position`) once, and then re-use these across several graphs.

TODO This panel tabset looks fine in HTML but is awkward in PDF

3.6 (a) Ignoring species

```
labels <- labs(
  x = "Bill length (mm)",
  y = "Bill depth (mm)",
  color = "Species",
  shape = "Species",
  fill = "Species")

plt1 <- ggplot(data = peng,
                 aes(x = bill_length,
                     y = bill_depth)) +
  geom_point(size = 1.5) +
  geom_smooth(method = "lm", formula = y ~ x,
```

```

            se = TRUE, color = "gray50") +
stat_ellipse(level = 0.68, linewidth = 1.1) +
ggttitle("Ignoring species") +
labels

plt1

```

3.7 (b) By species

```

legend_position <-
theme(legend.position = "inside",
      legend.position.inside = c(0.83, 0.16))

plt2 <- ggplot(data = peng,
                 aes(x = bill_length,
                     y = bill_depth,
                     color = species,
                     shape = species,
                     fill = species)) +
geom_point(size = 1.5,
           alpha = 0.8) +
geom_smooth(method = "lm", formula = y ~ x,
            se = TRUE, alpha = 0.3) +
stat_ellipse(level = 0.68, linewidth = 1.1) +
ggttitle("By species") +
labels +
theme_penguins("dark") +
legend_position

plt2

```

3.8 (c) Within species

```

# center within groups, translate to grand means
means <- colMeans(peng[, 3:4])
peng.centered <- peng |>
  group_by(species) |>
  mutate(bill_length = means[1] + scale(bill_length, scale = FALSE),
         bill_depth = means[2] + scale(bill_depth, scale = FALSE))

plt3 <- ggplot(data = peng.centered,
                 aes(x = bill_length,
                     y = bill_depth,
                     color = species,

```

```

      shape = species,
      fill = species)) +
geom_point(size = 1.5,
           alpha = 0.8) +
geom_smooth(method = "lm", formula = y ~ x,
            se = TRUE, alpha = 0.3) +
stat_ellipse(level = 0.68, linewidth = 1.1) +
labels +
ggtitle("Within species") +
theme_penguins("dark") +
legend_position

plt3

```

3.9 Multivariate normality and outliers

The relation of the data ellipsoid for p variables to the χ_p^2 distribution with p degrees of freedom described in Section 3.2 is based on the assumption that the data in \mathbf{y} is a sample from a multivariate normal distribution (MVN), with a mean vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$, so each one implies the other:

$$\mathbf{y}_{p \times 1} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \iff D_M^2(\mathbf{y}) \sim \chi_p^2.$$

This fact can be used to assess whether sample data in \mathbf{y} does indeed follow a MVN distribution by plotting the *quantiles*⁵ of the *sorted* sample D^2 values (denoted $D_{(i)}^2$) calculated from Equation 3.1 against the corresponding χ_p^2 quantiles found from `qchisq(df = p)`. This is called a χ^2 QQ plot.

The essential idea is that the plotted points should then approximately fall along a 45° line of slope = 1 when the data are MVN (and axes are scaled equally). This also provides a simple method to identify potential outliers as those points which are furthest from the centroid; that is those for which $D_{(i)}^2$ is greater than the $1 - \alpha$ quantile of χ_p^2 .

The topics of assessing multivariate normality and detecting multivariate outliers are larger than I consider here. These topics are also intertwined, because outliers inflate the variance (\mathbf{S}) of the data, making extreme observations appear less extreme. A variety of *robust* methods described later (sec ??) work by down-weighting outliers, which is particularly important in assessing whether the residuals from a multivariate linear model are multivariate normal.

In this section, I focus on the χ^2 QQ plot and graphical methods to relate the the points identified as potential outliers to plots in data space.

3.9.1 Galton data

Mahalanobis D^2 values are calculated by `heplots::Mahalanobis()`. I illustrate finding the largest $D_{(i)}^2$ for the `Galton` data as shown below. I've used $\alpha = 0.01$, giving $\chi_p^2(0.99) = 9.21$ as an outlier cutoff, just for the sake of this example. With 928 cases, we would expect about 1% or 9 to have larger squared distances than this cutoff. (Normally, allowing for the fact that we are looking at the largest values in a sample of size n , we would use a much smaller individual significance level, say $\alpha = 0.001$ or smaller still.) Three cases are identified here.

⁵The p^{th} -quantile of a random variable Y is a value, denoted by $Q_Y(p)$, such that $X \leq Q_Y(p)$ with probability p , i.e., $\Pr(Y \leq Q_Y(p)) = p$. For continuous variables, the quantile is the inverse of the cumulative distribution function.

```

data(Galton, package = "HistData")
DSQ <- Mahalanobis(Galton)
alpha <- 0.01
cutoff <- (qchisq(p = 1 - alpha, df = ncol(Galton))) |>
  print()
#> [1] 9.21
outliers <- which(DSQ > cutoff) |>
  print()
#> [1] 1 13 897
GaltonD <- cbind(Galton, DSQ = DSQ)
GaltonD[outliers,]
#>   parent child   DSQ
#> 1     70.5 61.7 13.67
#> 13    70.5 63.2  9.45
#> 897   65.5 72.2  9.49

```

χ^2 QQ plots are constructed by `heplots::cpplot()`. For the Galton data, the result is shown in Figure 3.24, where I've asked for the `id.n = 3` with the greatest $D_{(i)}^2$ values to be identified with their row numbers in the plot. The function returns (invisibly) the $D_{(i)}^2$ and corresponding χ^2 quantile along with the upper-tail p -value.

```

out <- cpplot(Galton, id.n = 3)
out
#>   DSQ quantile      p
#> 1  13.67    15.1 0.000539
#> 897 9.49    12.9 0.001616
#> 13  9.45    11.8 0.002694

```

In the typical use of QQ plots, it is essential to have something in the nature of a confidence band around the points to be able to appreciate whether, and to what degree the observed data points differ from the reference distribution. For `cpplot()`, this helps to assess whether the data are reasonably distributed as multivariate normal and also to flag potential outliers.

The pointwise 95% confidence envelope is calculated as $D_{(i)}^2 \pm 1.96 \times \text{se}(D_{(i)}^2)$ where the standard error is calculated (Chambers et al., 1983, sec. 8.6) as

$$\text{se}(D_{(i)}^2) = \frac{\hat{b}}{d(q_i)} \times \sqrt{p_i(1-p_i)/n} .$$

Here, \hat{b} is an estimate of the *actual* slope of the reference line obtained from the ratio of the interquartile range of the D^2 values to that of the corresponding χ_p^2 distribution. $d(q_i)$ is the density of the chi square distribution at the quantile q_i and p_i is the corresponding percentile.

So, in Figure 3.24, you can see the same three points, with largest D^2 identified earlier. The confidence band shows that uncertainty increases as points get further from the mean, but by and large, nearly all the points lie within it. The fact that larger values fall *beneath* the reference line indicates that the sample D^2 are somewhat *less concentrated* (has a shorter upper tail) than values in the χ^2 distribution.

To help understand what we are seeing in this example, it is helpful to view the data in a scatterplot equivalent of Figure 3.11, with the same three observations labeled and made distinctive in the plot. For this plot, because the heights are recorded in whole inches, I draw the data ellipses first without plotting the points and then draw the points using `jitter()` on top of the data ellipses.

Chi-Square Q-Q Plot of Galton

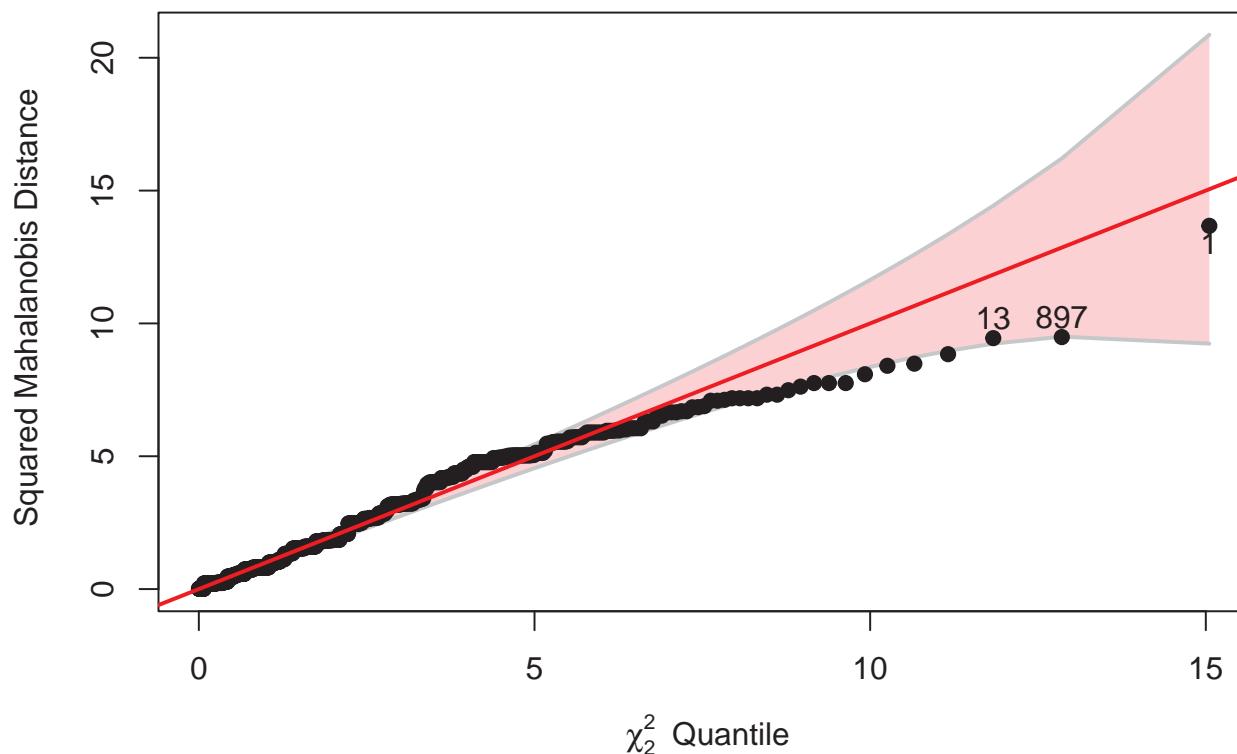


Figure 3.24: Chi-square QQ plot of Galton's data on heights of parents and their offspring, with a 95% pointwise confidence envelope

```

set.seed(47)
dataEllipse(parent ~ child, data = GaltonD,
            levels = c(0.68, 0.95),
            add = FALSE, plot.points = FALSE,
            center.pch = "+", center.cex = 3,
            cex.lab = 1.5)
with(GaltonD,{
  points(jitter(child), jitter(parent),
         col = ifelse(DSQ > cutoff, "red", "black"),
         pch = ifelse(DSQ > cutoff, 16, 1),
         cex = ifelse(DSQ > cutoff, 2, 0.8))
  text(child[outliers], parent[outliers], labels = outliers, pos = 3)
})

```

It is clear from this plot that case 897 is for an exceptionally tall child of quite short parents, while case 1 and 13 are from very short children of tall parents; you could call them poster children for regression toward the mean.

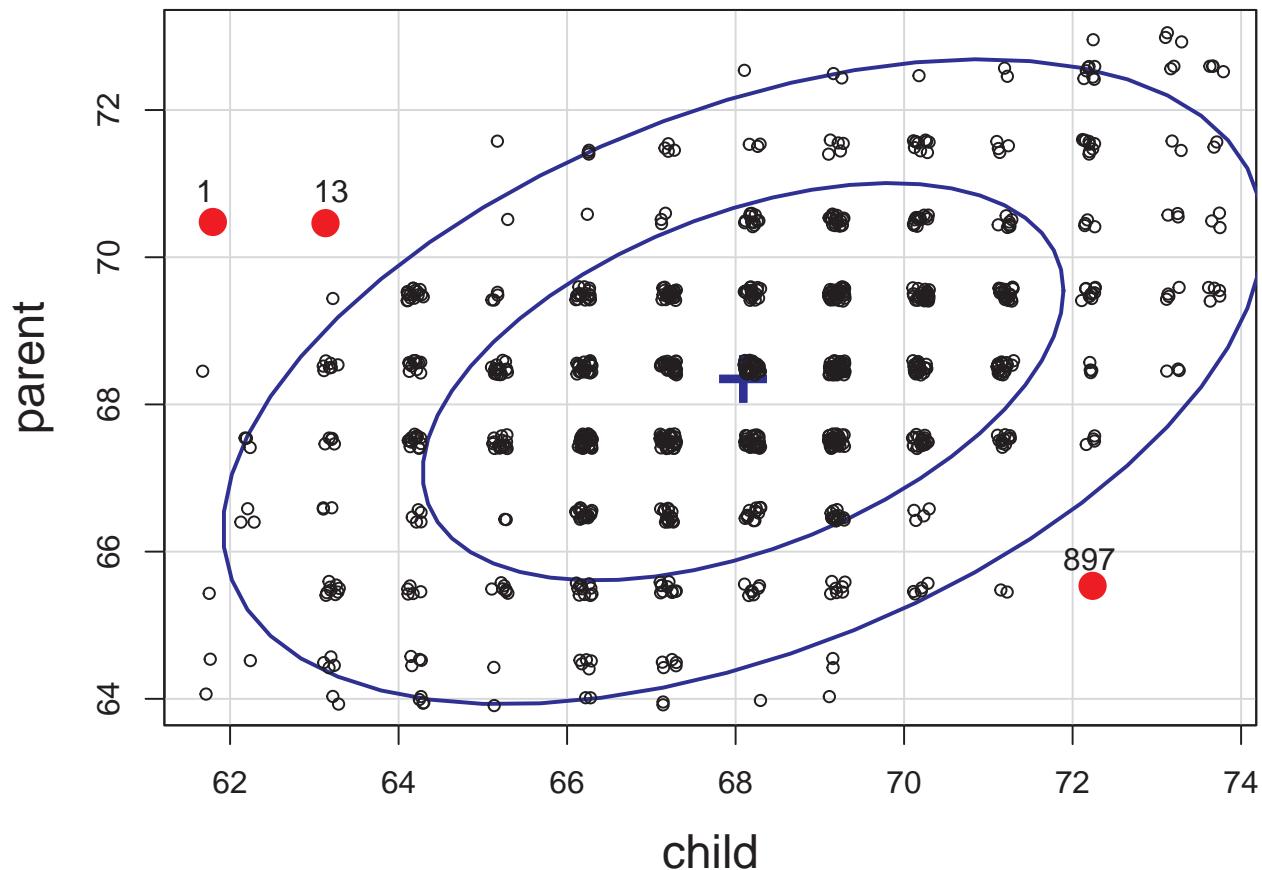


Figure 3.25: Scatterplot of Galton’s data showing 68% and 95% data ellipses for the observations on parent and child height. The discrete points have been jittered to avoid overplotting. The three observations identified as having large D^2 values are labeled with their observation number and given distinctive size, color and shape.

3.9.2 Penguin data

Let’s do a similar analysis to assess multivariate normality and identify possible outliers for the Penguin data. In the `cqplot()` shown in Figure 3.26, I use the same point symbols and colors as in Figure 3.19. For illustration, I again label the three most extreme points.

```

clr <- peng.colors("dark")
pch <- c(19, 17, 15) # ggplot symbol defaults for a factor

out <- cqplot(peng[, 3:6],
  id.n = 3,
  col = clr[peng$species],
  pch = pch[peng$species],
  ref.col = "grey",
  what = "Penguin numeric variables",
  cex.lab = 1.25)
out
#>   DSQ quantile      p
#> 283 27.8      17.6 0.00150

```

```
#> 10 13.3      15.1 0.00450
#> 35 12.4      13.9 0.00751
```

Chi-Square Q-Q Plot of Penguin numeric variables

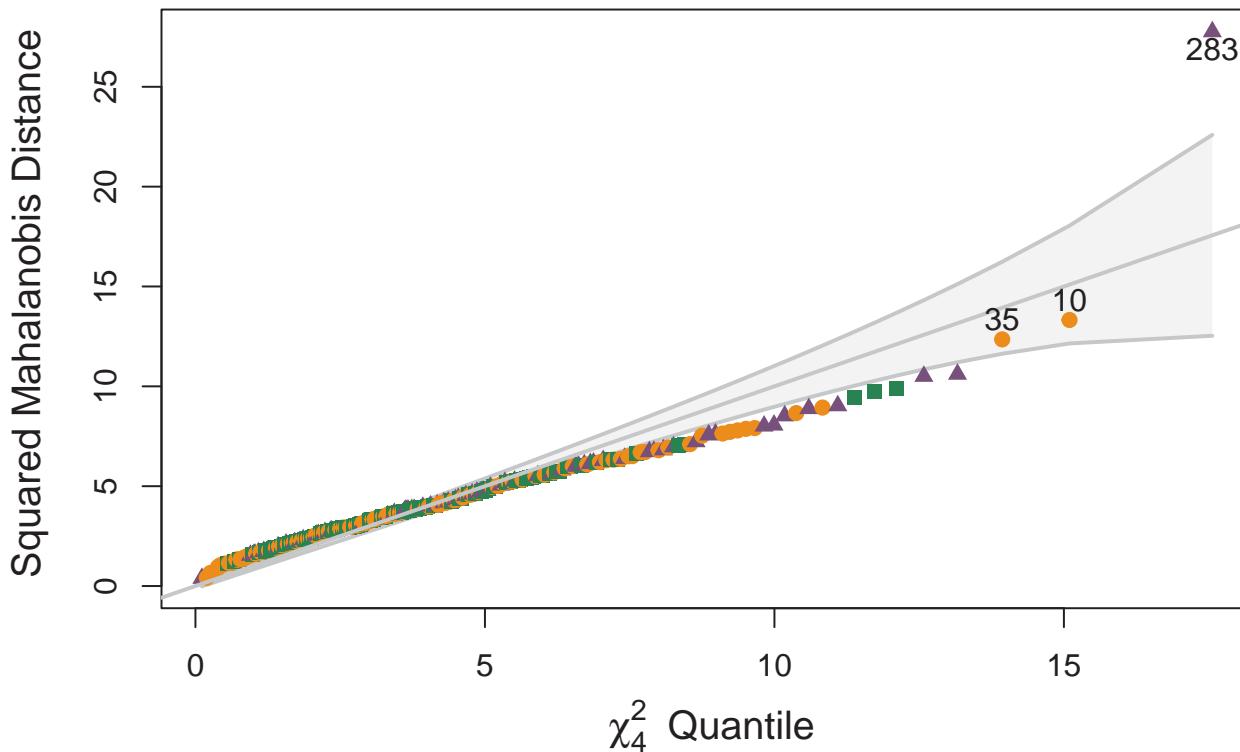


Figure 3.26: Chi-square QQ plot of the Penguin data. The three cases with the largest D^2 values are identified with their case numbers.

One point (case 283) stands out as an extreme multivariate outlier. The other two (10, 35) are well within the confidence envelope.

To relate this to the data, we can plot the data, as was done in Figure 3.19, and label the points identified as noteworthy. It is a bit tricky to label points selectively in `ggplot2` when the criterion for which points to label is complex, or involves variables outside the data frame. Here, I create a logical variable `note`, which is `TRUE` for the noteworthy ones. This is then used to subset the data for `geom_text()` that writes the case id numbers. Figure 3.27 shows the resulting plot.

```
DSQ <- Mahalanobis(peng[, 3:6])
noteworthy <- order(DSQ, decreasing = TRUE)[1:3] |> print()
#> [1] 283 10 35

peng_plot <- peng |>
  tibble::rownames_to_column(var = "id") |>
  mutate(note = id %in% noteworthy)

ggplot(peng_plot,
  aes(x = bill_length, y = bill_depth,
```

```

color = species, shape = species, fill=species)) +
geom_point(aes(size=note), show.legend = FALSE) +
scale_size_manual(values = c(1.5, 4)) +
geom_text(data = subset(peng_plot, note==TRUE),
aes(label = id),
nudge_y = .4, color = "black", size = 5) +
geom_smooth(method = "lm", formula = y ~ x,
se=FALSE, linewidth=2) +
stat_ellipse(geom = "polygon", level = 0.95, alpha = 0.1) +
theme_penguins() +
theme_bw(base_size = 14) +
theme(legend.position = "inside",
legend.position.inside=c(0.85, 0.15))

```

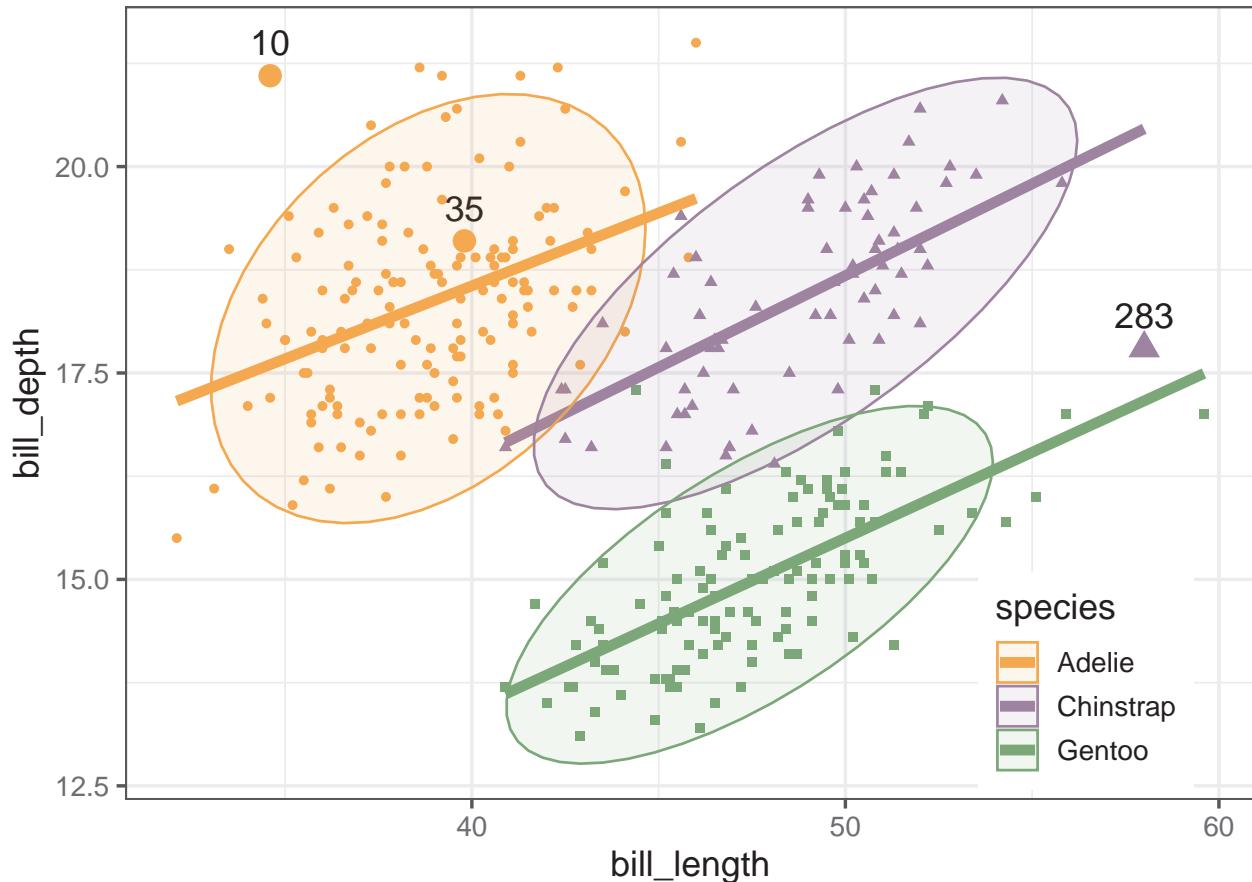


Figure 3.27: Plot of bill length and bill depth, with the noteworthy points labeled. Only one (case 283) is a true multivariate outlier.

Two cases (10, 283) appear to be very unusual in this plot, in relation to other members of their species. But only case 283 is a true multivariate outlier, as shown in the cqplot Figure 3.26. We could call this long-billed penguin “Cyrano”⁶. I’ll call case 10, with a very short and curved bill, “Hook Nose”.

Of course, we are only looking at the data in the 2D space of the bill variables, but possible outliers exist

⁶After the long-nosed Cyrano de Bergerac from Edmund Rostand’s play. Actually, bird 283 is a female. But, it would be a mistake to call her “Rozanne” instead.

in four dimensional Penguin space. Case 35 is well inside the Adelie ellipse, so perhaps it is unusual on one of the other variables. It turns out that multivariate outliers can most often be easily seen as unusual observations in a projection of the data into the space of the *smallest* principal components. I return to this example in Section 4.7.

It bears noting that for linear models, multivariate normality is **not required** for the response variables or predictors, but rather is an assumption for the *residuals* from a model. As well, multivariate outliers in the responses may not turn out to be unusual when the predictor variables are taken into account. In this example, a multivariate model would include the effect of `species`,

```
peng.mod <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~ species,
                 data = peng)
```

and there would be cause for concern if the residuals from this model, `residuals(peng.mod)` were highly non-normal or showed outliers.

3.10 Scatterplot matrices

Going beyond bivariate scatterplots, a *pairs* plot (or *scatterplot matrix*) displays all possible $p \times p$ pairs of p variables in a matrix-like display where variables (x_i, x_j) are shown in a plot for row i , column j . This idea, due to Hartigan (1975b), uses small multiple plots, so that the eye can easily scan across a row or down a column to see how a given variable is related to all the others.

The most basic version is provided by `pairs()` in base R. When one variable is considered as an outcome or response, it is usually helpful to put this in the first row and column. For the `Prestige` data, in addition to income and education, we also have a measure of % women in each occupational category.

Plotting these together gives Figure 3.28. In such plots, the diagonal cells give labels for the variables, but they are also a guide to interpreting what is shown. In each row, say row 2 for `income`, income is the vertical y variable in plots against other variables. In each column, say column 3 for `education`, education is the horizontal x variable.

```
pairs(~ prestige + income + education + women,
      data=Prestige)
```

The plots in the first row show what we have seen before for the relations between prestige and income and education, adding to those the plot of prestige vs. % women. Plots in the first column show the same data, but with x and y interchanged.

But this basic `pairs()` plot is very limited. A more feature-rich version is provided by `car:::scatterplotMatrix()` which can add the regression lines, loess smooths and data ellipses for each pair, as shown in Figure 3.29.

The diagonal panels show density curves for the distribution of each variable; for example, the distribution of `education` appears to be multi-modal and that of `women` shows that most of the occupations have a low percentage of women.

The combination of the regression line with the loess smoothed curve, but without their confidence envelopes, provides about the right amount of detail to take in at a glance where the relations are non-linear. We've already seen (Figure 3.12) the non-linear relation between prestige and income (row 1, column 2) when occupational type is ignored. But all relations with income in column 2 are non-linear, reinforcing our idea (Section 3.2.2.1) that effects of income should be assessed on a log scale.

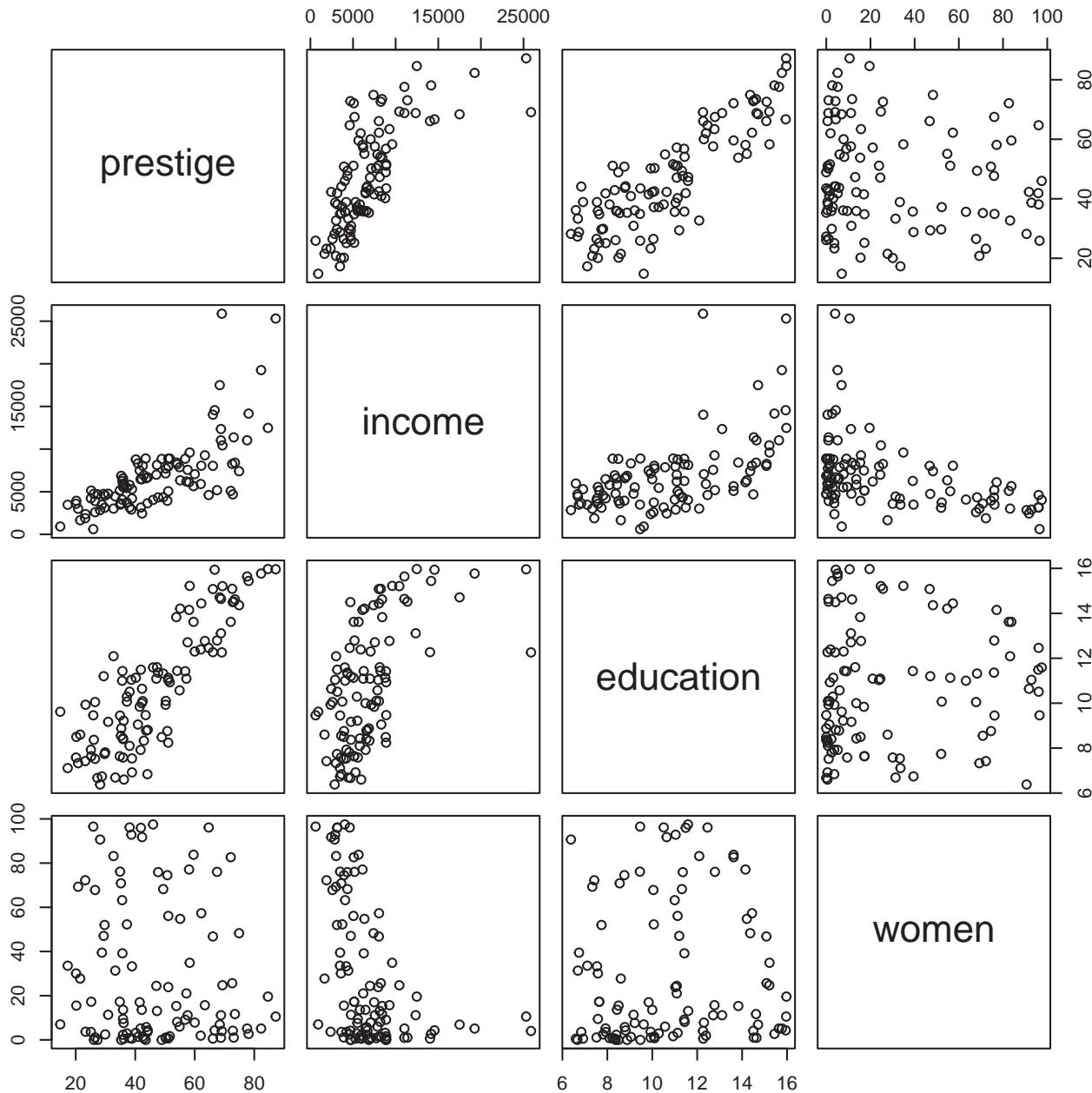


Figure 3.28: Scatterplot matrix of the variables in the Prestige dataset produced by `pairs()`

```
scatterplotMatrix(~ prestige + income + education + women,
  data=Prestige,
  regLine = list(method=lm, lty=1, lwd=2, col="black"),
  smooth=list(smooth=loessLine, spread=FALSE,
              lty.smooth=1, lwd.smooth=3, col.smooth="red"),
  ellipse=list(levels=0.68, fill.alpha=0.1))
```

`scatterplotMatrix()` can also label points using the `id =` argument (though this can get messy) and can stratify the observations by a grouping variable with different symbols and colors. For example, Figure 3.30 uses the syntax `~ prestige + education + income + women | type` to provide separate regression lines,

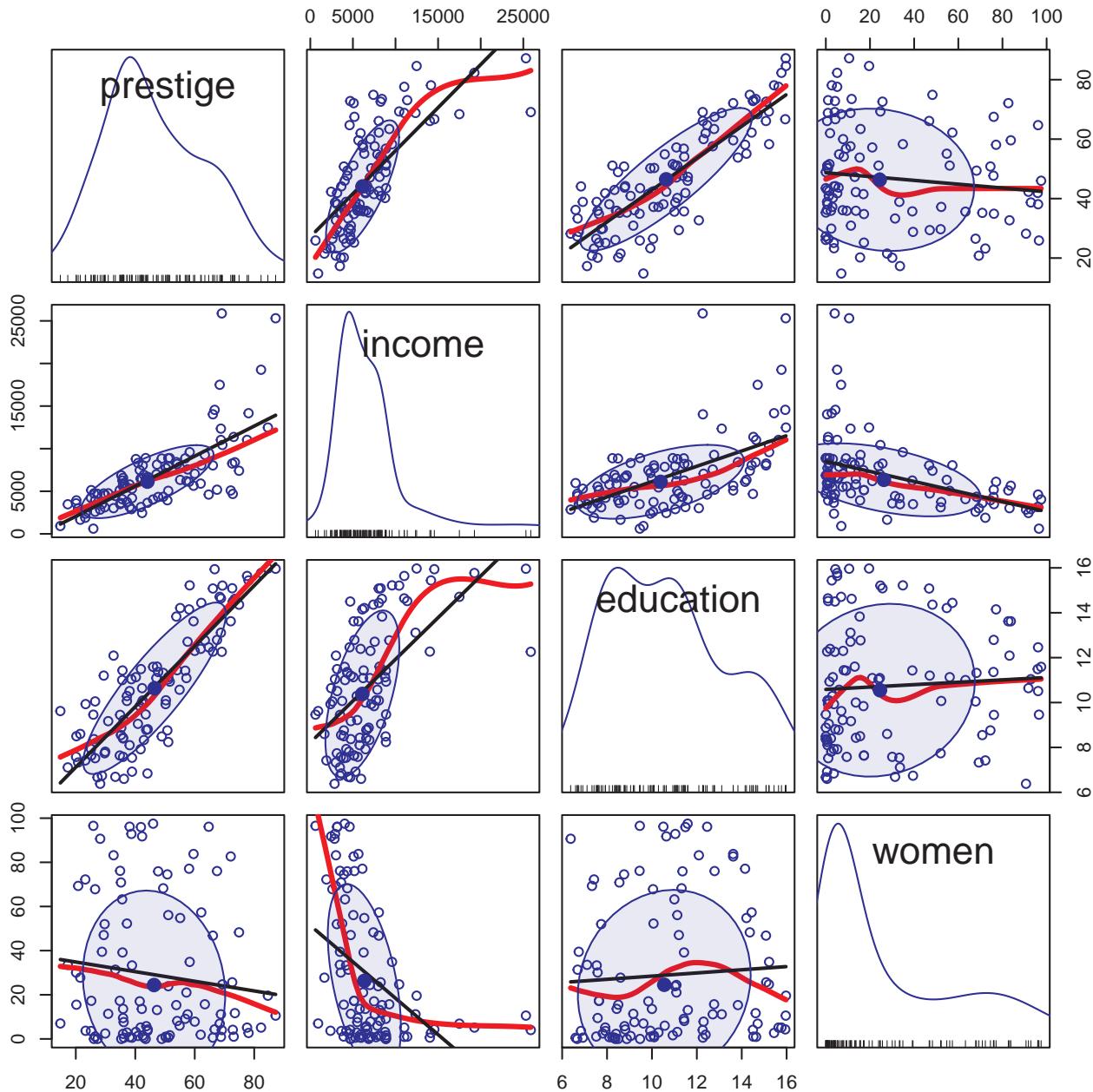


Figure 3.29: Scatterplot matrix of the variables in the Prestige dataset from `car::scatterplotMatrix()`.

smoothed curves and data ellipses for the three types of occupations. (The default colors are somewhat garish, so I use `scales::hue_pal()` to mimic the discrete color scale used in `ggplot2`).

```
scatterplotMatrix(~ prestige + income + education + women | type,
  data = Prestige,
  col = scales::hue_pal()(3),
  pch = 15:17,
  smooth=list(smoker=loessLine, spread=FALSE,
             lty.smooth=1, lwd.smooth=3, col.smooth="black"),
  ellipse=list(levels=0.68, fill.alpha=0.1))
```

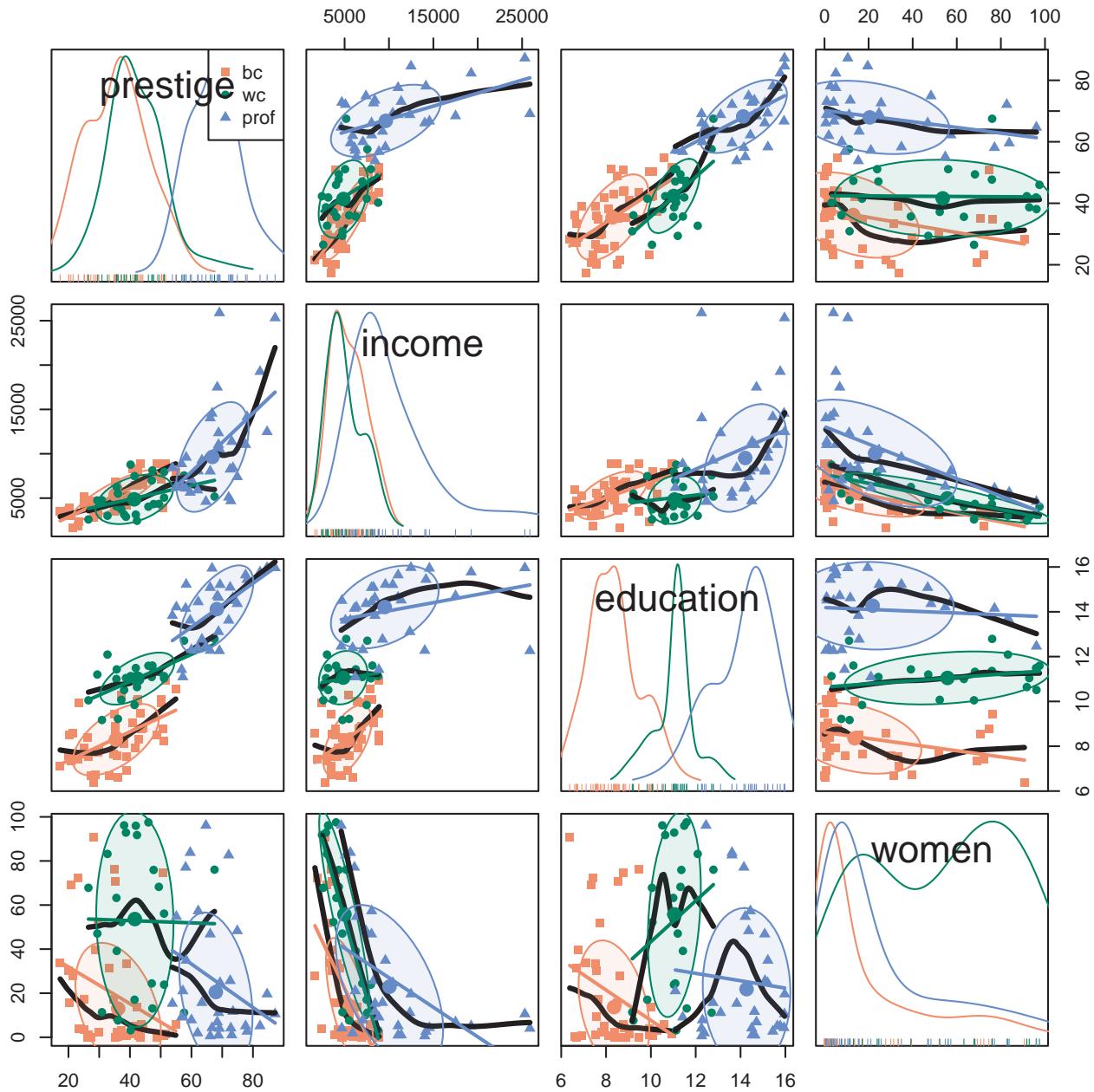


Figure 3.30: Scatterplot matrix of the variables in the Prestige dataset from `car::scatterplotMatrix()`, stratified by type of occupation.

It is now easy to see why education is multi-modal: blue collar, white collar and professional occupations have largely non-overlapping years of education. As well, the distribution of % women is much higher in the white collar category.

For the `penguins` data, given what we've seen before in Figure 3.19 and Figure 3.20, we may wish to suppress details of the points (`plot.points = FALSE`) and loess smooths (`smooth = FALSE`) to focus attention on the similarity of regression lines and data ellipses for the three penguin species. In Figure 3.31, I've chosen to show boxplots rather than density curves in the diagonal panels in order to highlight differences in the means and interquartile ranges of the species, and to show 68% and 95% data ellipses in the off-diagonal panels.

```
scatterplotMatrix(~ bill_length + bill_depth + flipper_length + body_mass | species,
  data = peng,
  col = peng.colors("medium"),
  legend=FALSE,
  ellipse = list(levels = c(0.68, 0.95),
                 fill.alpha = 0.1),
  regLine = list(lwd=3),
  diagonal = list(method = "boxplot"),
  smooth = FALSE,
  plot.points = FALSE,
  cex.labels=1)
```

It can be seen that the species are widely separated in most of the bivariate plots. As well, the regression lines for species have similar slopes and the data ellipses have similar size and shape in most of the plots. From the boxplots, we can also see that **Adelie** penguins have shorter bill lengths than the others, while **Gentoo** penguins have smaller bill depth, but longer flippers and are heavier than **Chinstrap** and **Adelie** penguins.

Looking ahead

Figure 3.31 provides a reasonably complete visual summary of the data in relation to multivariate models that ask “do the species differ in their means on these body size measures?” This corresponds to the MANOVA model,

```
peng.mod <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~ species,
  data=peng)
```

Hypothesis-error (HE) plots, described in Chapter 11 provide a better summary of the evidence for the MANOVA test of differences among means on all variables together. These give an **H** ellipse reflecting the differences among means, to be compared with an **E** ellipse reflecting within-group variation and a visual test of significance.

A related question is “how well are the penguin species distinguished by these body size measures?” Here, the relevant model is linear discriminant analysis (LDA), where **species** plays the role of the response in the model,

```
peng.lda <- MASS:lda( species ~ cbind(bill_length, bill_depth, flipper_length, body_mass),
  data=peng)
```

Both MANOVA and LDA depend on the assumption that the variances and correlations between the variables are the same for all groups. This assumption can be tested and visualized using the methods in Chapter 12.

3.10.1 Visual thinning

What can you do if there are even more variables than in these examples? If what you want is a high-level, zoomed-out display summarizing the pairwise relations more strongly, you can apply the idea of visual thinning to show only the most important features.

This example uses data on the rate of various crimes in the 50 U.S. states from the United States Statistical Abstracts, 1970, used by Hartigan (1975a) and Friendly (1991). These are ordered in the dataset roughly by seriousness of crime or from crimes of violence to property crimes.

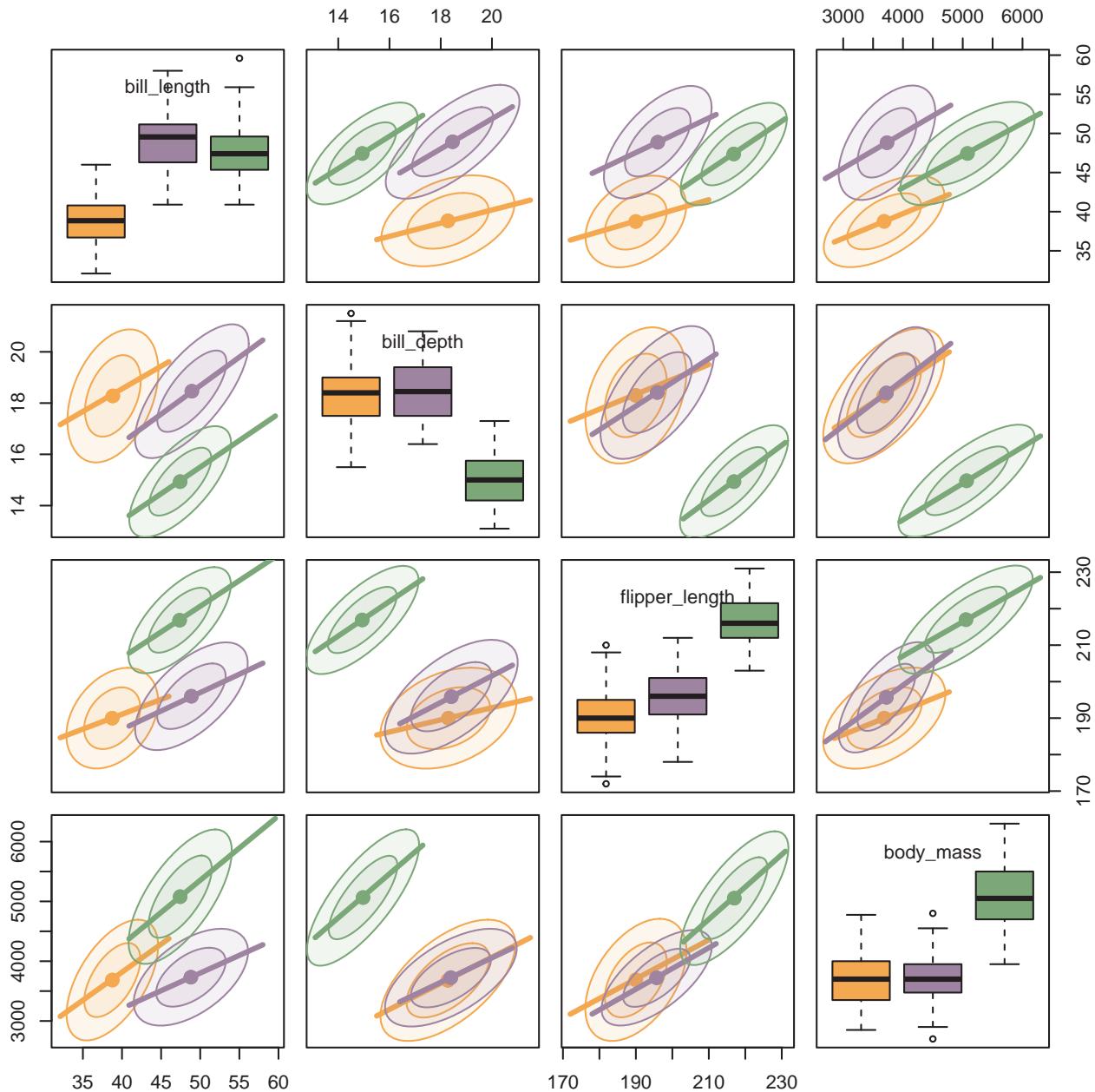


Figure 3.31: Scatterplot matrix of the quantitative variables in the penguins dataset, stratified by species.

```

data(crime, package = "ggbio")
str(crime)
#> 'data.frame': 50 obs. of 10 variables:
#> $ state    : chr "Alabama" "Alaska" "Arizona" "Arkansas" ...
#> $ murder   : num 14.2 10.8 9.5 8.8 11.5 6.3 4.2 6 10.2 11.7 ...
#> $ rape     : num 25.2 51.6 34.2 27.6 49.4 42 16.8 24.9 39.6 31.1 ...
#> $ robbery  : num 96.8 96.8 138.2 83.2 287 ...
#> $ assault  : num 278 284 312 203 358 ...
#> $ burglary: num 1136 1332 2346 973 2139 ...

```

```
#> $ larceny : num 1882 3370 4467 1862 3500 ...
#> $ auto      : num 281 753 440 183 664 ...
#> $ st        : chr "AL" "AK" "AZ" "AR" ...
#> $ region   : Factor w/ 4 levels "Northeast","South",...: 2 4 4 2 4 4 1 2 2 2 ...
```

Figure 3.32 displays the scatterplot matrix for these seven variables, using only the regression line and data ellipse to show the linear relation and the loess smooth to show potential non-linearity. To make this even more schematic, the axis tick marks and labels are also removed using the `par()` settings `xaxt = "n"`, `yaxt = "n"`.

```
crime |>
  select(where(is.numeric)) |>
  scatterplotMatrix(
    plot.points = FALSE,
    ellipse = list(levels = 0.68, fill=FALSE),
    smooth = list(spread = FALSE,
                  lwd.smooth=2, lty.smooth = 1, col.smooth = "red"),
    cex.labels = 2,
    xaxt = "n", yaxt = "n")
```

We can see that all pairwise correlations are positive, pairs closer to the main diagonal tend to be more highly correlated and in most cases the nonparametric smooth doesn't differ much from the linear regression line. Exceptions to this appear mainly in the columns for `robbery` and `auto` (auto theft).

3.11 Corrrgrams

What if you want to summarize the data even further simple visual thinning. For example with many variables you might want to show only the value of the correlation for each pair of variables, but do so in a way to help see patterns in the correlations that would be invisible in just a table.

A **corrgram** (Friendly, 2002) is a visual display of a correlation matrix, where the correlation can be rendered in a variety of ways to show the direction and magnitude: circular “pac-man” (or pie) symbols, ellipses, colored vars or shaded rectangles, as shown in Figure 3.33.

Another aspect is that of **effect ordering** (Friendly & Kwan, 2003), ordering the levels of factors and variables in graphic displays to make important features most apparent. For variables, this means that we can arrange the variables in a matrix-like display in such a way as to make the pattern of relationships easiest to see. Methods to achieve this include using principal components and cluster analysis to put the most related variables together as described in Chapter 4.

In R, these diagrams can be created using the `corrgram` (Wright, 2021) and `corrplot` (Wei & Simko, 2024) packages, with different features. `corrgram::corrgram()` is closest to Friendly (2002), in that it allows different rendering functions for the lower, upper and diagonal panels as illustrated in Figure 3.33. For example, a corrgram similar to Figure 3.32 can be produced as follows (not shown here):

```
crime |>
  select(where(is.numeric)) |>
  corrgram(lower.panel = panel.ellipse,
            upper.panel = panel.ellipse,
            diag.panel = panel.density)
```

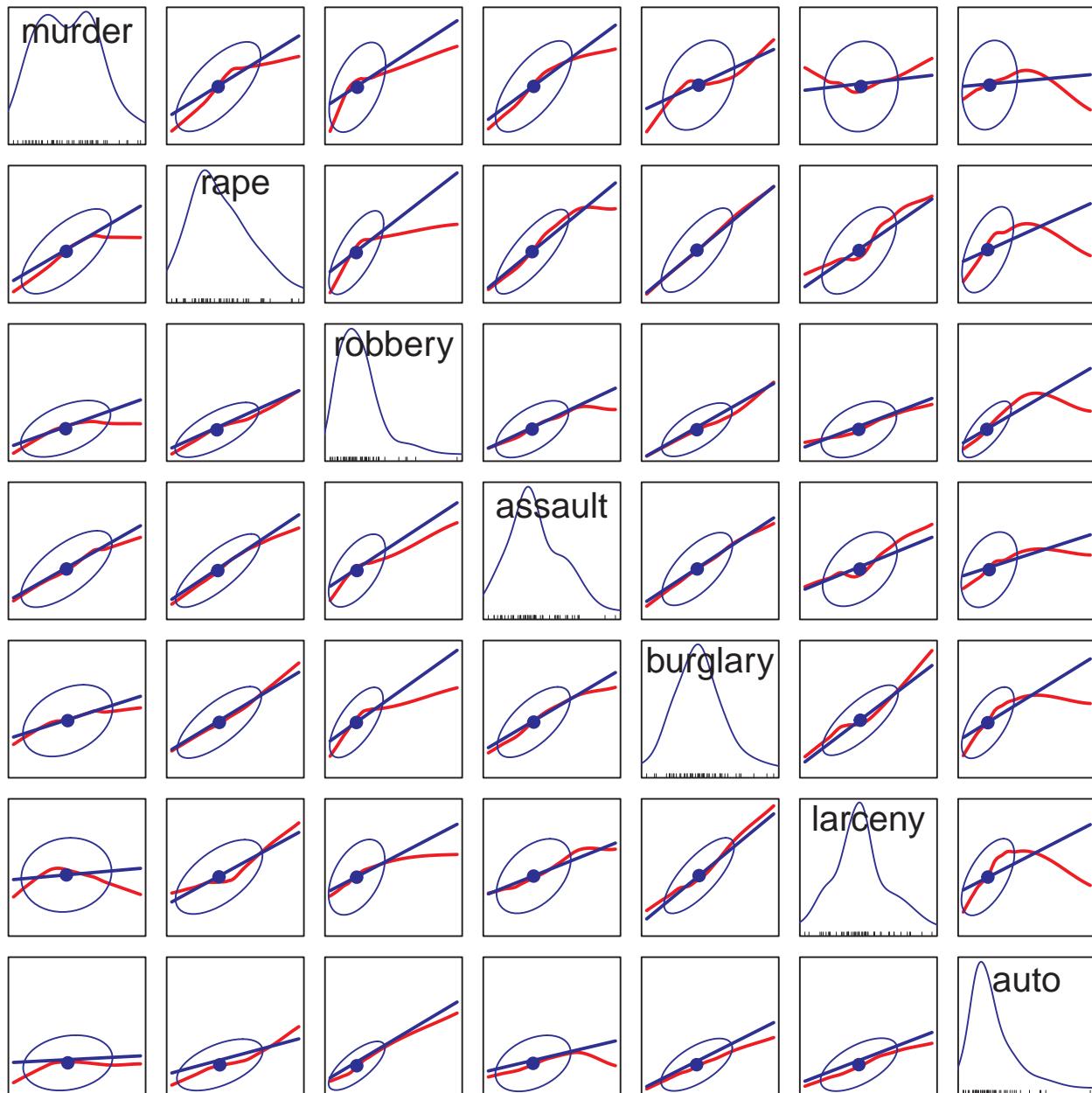


Figure 3.32: Visual thinning: Scatterplot matrix of the crime data, showing only high-level summaries of the linear and nonlinear relations between each pair of variables.

With the `corrplot` package, `corrplot()` provides the rendering methods `c("circle", "square", "ellipse", "number", "shade", "color", "pie")`, but only one can be used at a time. The function `corrplot.mixed()` allows different options to be selected for the lower and upper triangles. The iconic rendering shape is colored with a gradient in relation to the correlation value. For comparison, Figure 3.34 uses ellipses below the diagonal and filled pie charts below the diagonal using a gradient of the fill color in both cases.

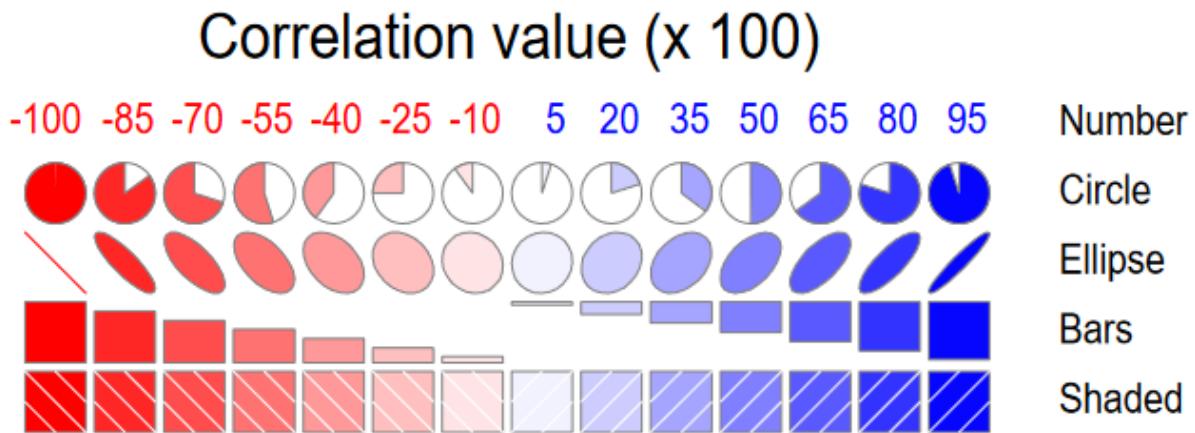


Figure 3.33: Corrgrams: Some renderings for the value of a correlation in a corrgram display, conveying sign and magnitude in different ways.

```
crime.cor <- crime |>
  dplyr::select(where(is.numeric)) |>
  cor()

corrplot.mixed(crime.cor,
  lower = "ellipse",
  upper = "pie",
  tl.col = "black",
  tl.srt = 0,
  tl.cex = 1.25,
  addCoef.col = "black",
  addCoefasPercent = TRUE)
```

The combination of renderings shown in Figure 3.34 is instructive. Small differences among correlation values are easier to see with the pie symbols than with the ellipses; for example, compare the values for murder with larceny and auto theft in row 1, columns 6-7 with those in column 1, rows 6-7, where the former are easier to distinguish. The shading color adds another visual cue.

The variables in Figure 3.32 and Figure 3.34 are arranged by their order in the dataset, which is not often the most useful. A better idea is to arrange the variables so that the most highly correlated variables are adjacent.

A general method described in Section 4.5 orders the variables according to the angles of the first two eigenvectors from a principal components analysis (PCA) around a unit circle. The function `corrMatOrder()` provides several methods (`order = c("AOE", "FPC", "hclust", "alphabet")`) for doing this, and PCA ordering is `order = "AOE"`. Murder and auto theft are still first and last, but some of the intermediate crimes have been rearranged.

```
ord <- corrMatOrder(crime.cor, order = "AOE")
rownames(crime.cor)[ord]
#> [1] "murder"    "assault"   "rape"      "robbery"   "burglary"
#> [6] "larceny"   "auto"
```

Using this ordering in `corrplot()` produces Figure 3.35.

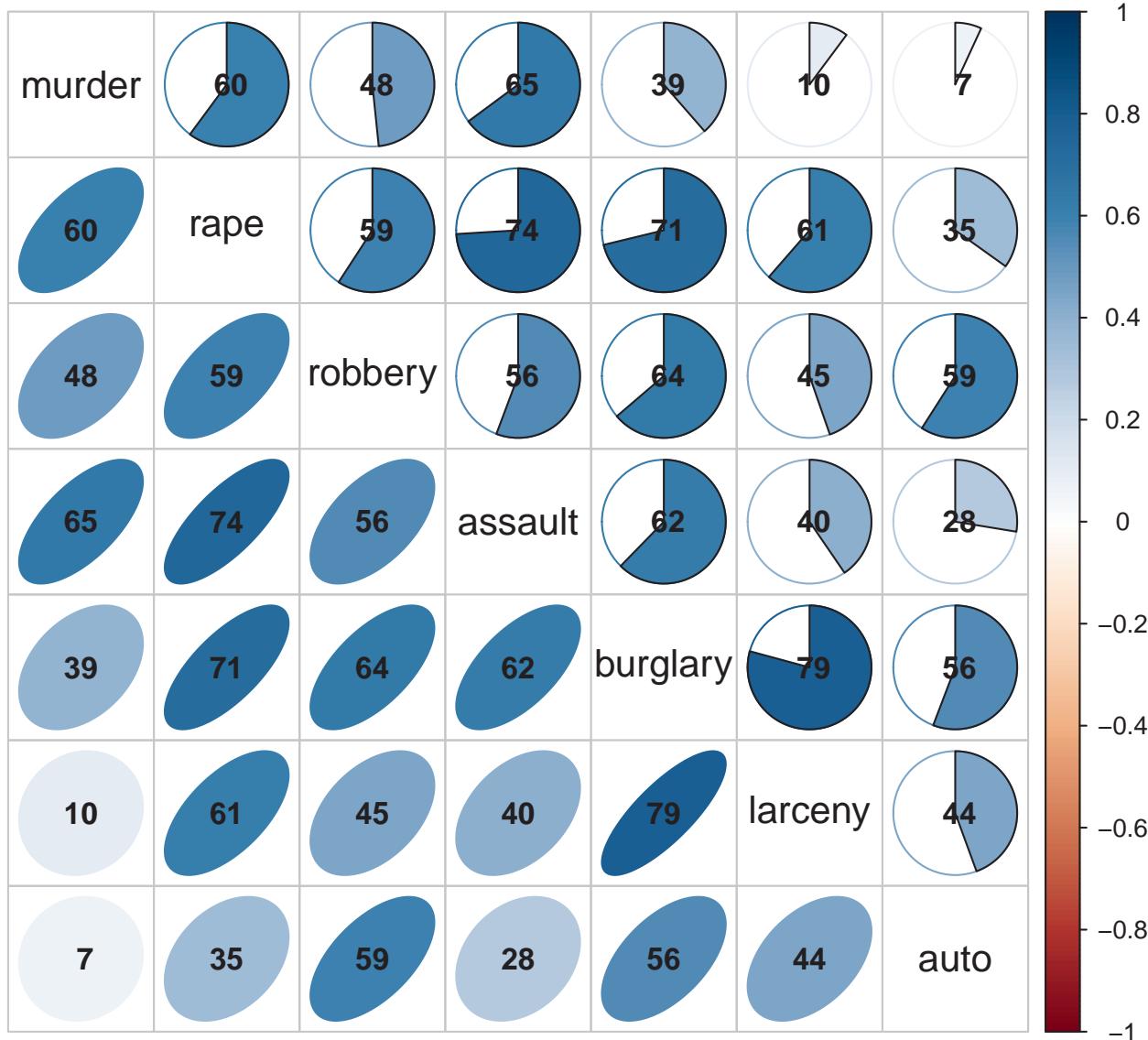


Figure 3.34: Mixed corrplot of the `crime` data, showing the correlation between each pair of variables with an ellipse (lower) and a pie chart symbol (upper), all shaded in proportion to the correlation value, also shown numerically.

```
corrplot.mixed(crime.cor,
  order = "AOE",
  lower = "ellipse",
  upper = "ellipse",
  tl.col = "black",
  tl.srt = 0,
  tl.cex = 1.25,
  addCoef.col = "black",
  addCoefasPercent = TRUE)
```

In this case, where the correlations among the crime variables are all positive, the effect of variable re-ordering is subtle, but note that there is now a slightly pronounced pattern of highest correlations near the diagonal,

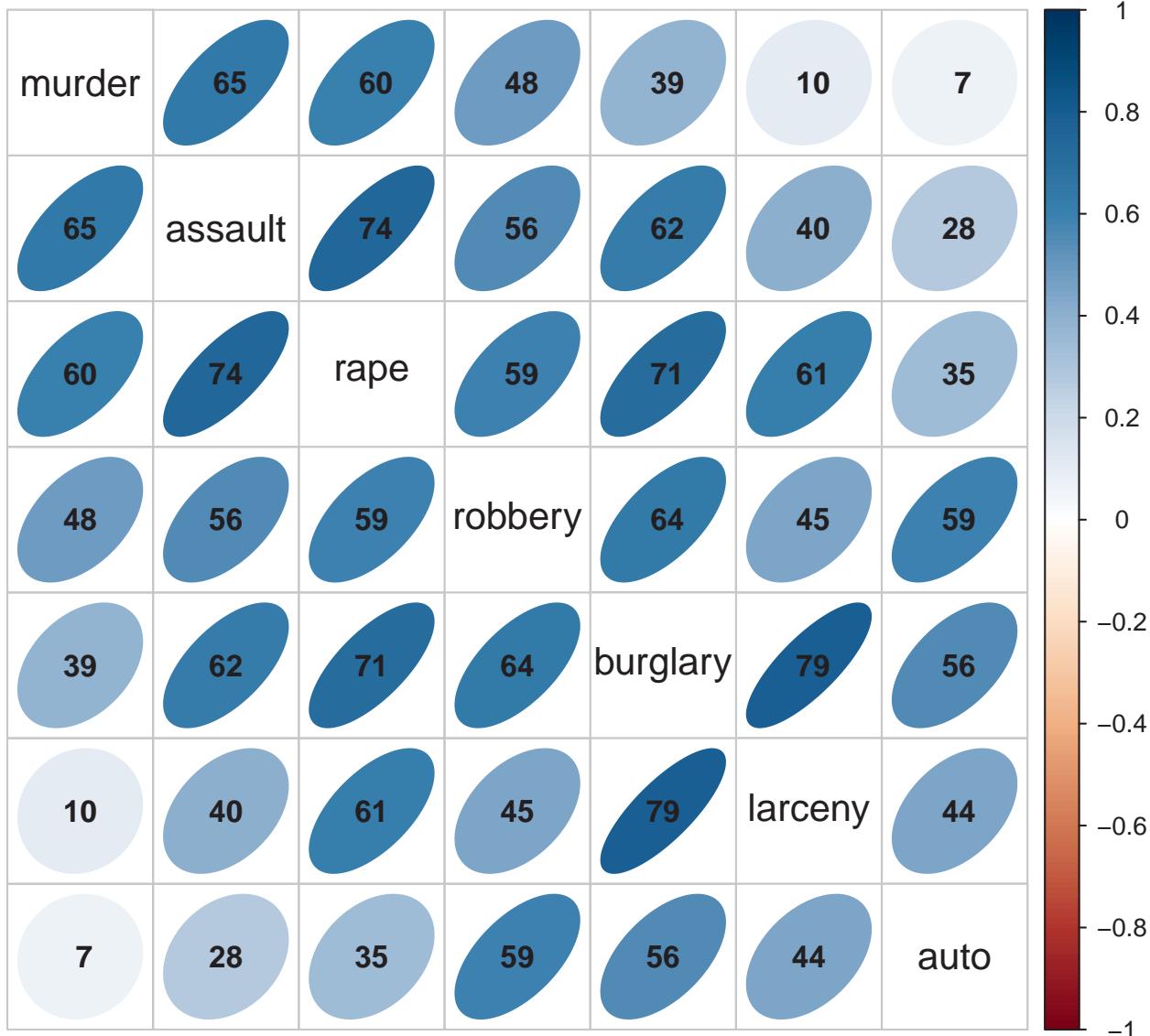


Figure 3.35: Corrplot of the `crime` data with the variables reordered according to the angles of variable eigenvectors. Correlations are rendered with ellipses shaded in proportion to their magnitude.

and decreasing away from the diagonal. Figure 4.25 and Figure 4.27 in Section 4.5 provide a more dramatic example of variable ordering using this method.

Variations of corrgrams are worthy replacements for a numeric table of correlations, which are often presented in publications only for archival value. Including the numeric value (rounded here, for presentation purposes), makes this an attractive alternative to boring tables of correlations.

3.12 Generalized pairs plots

When a dataset contains one or more discrete variables, the traditional pairs plot cannot cope, because the discrete categories would plot as many overlaid points. This cannot be represented using only color and/or point symbols in a meaningful scatterplot.

But the associations between categorical variables in a frequency table *can* be shown in *mosaic displays* (Friendly, 1994), using an array of tiles whose areas are depict the cell frequencies. For an n -way frequency, an analog of the scatterplot matrix uses mosaic plots for each pair of variables. The **vcd** package (Meyer et al., 2024) implements very general `pairs()` methods for "table" objects and **vcdExtra** (Friendly, 2025b) extends this to wide classes of loglinear models (Friendly, 1999). See Friendly (1999) and my book *Discrete Data Analysis with R* (Friendly & Meyer, 2016) for mosaic plots and mosaic matrices.

For example, we can tabulate the distributions of penguin species by sex and the island where they were observed using `xtabs()`. `ftable()` prints this three-way table more compactly. (In this example, and what follows in the chapter, I've changed the labels for sex from ("f", "m") to ("Female", "Male").)

```
# use better labels for sex
peng <- peng |>
  mutate(sex = factor(sex, labels = c("Female", "Male")))
peng.table <- xtabs(~ species + sex + island, data = peng)

ftable(peng.table)
#>           island Biscoe Dream Torgersen
#> species   sex
#> Adelie     Female      22    27     24
#>          Male       22    28     23
#> Chinstrap  Female      0    34     0
#>          Male       0    34     0
#> Gentoo    Female      58     0     0
#>          Male       61     0     0
```

We can see immediately that the penguin species differ by island: only Adelie were observed on all three islands; Biscoe Island had no Chinstraps and Dream Island had no Gentoos.

`vcd::pairs()` produces all pairwise mosaic plots, as shown in Figure 3.36. The diagonal panels show the one-way frequencies by width of the divided bars. Each off-diagonal panel shows the bivariate counts, breaking down each column variable by splitting the bars in proportion to a second variable. Consequently, the frequency of each cell is represented by its' area. The purpose is to show the **pattern of association** between each pair, and so, the tiles in the mosaic are shaded according to the signed standardized residual, $d_{ij} = (n_{ij} - \hat{n}_{ij})/\sqrt{\hat{n}_{ij}}$ in a simple $\chi^2 = \sum_{ij} d_{ij}^2$ test for association— blue where the observed frequency n_{ij} is significantly greater than expected \hat{n}_{ij} under independence, and red where it is less than expected. The tiles are unshaded when $|d_{ij}| < 2$.

```
library(vcd)
pairs(peng.table, shade = TRUE,
      lower_panel_args = list(labeling = labeling_values()),
      upper_panel_args = list(labeling = labeling_values()))
```

The shading patterns in cells (1,3) and (3,1) of Figure 3.36 show what we've seen before in the table of frequencies: The distribution of species varies across island because on each island one or more species did not occur. Row 2 and column 2 show that sex is nearly exactly proportional among species and islands,

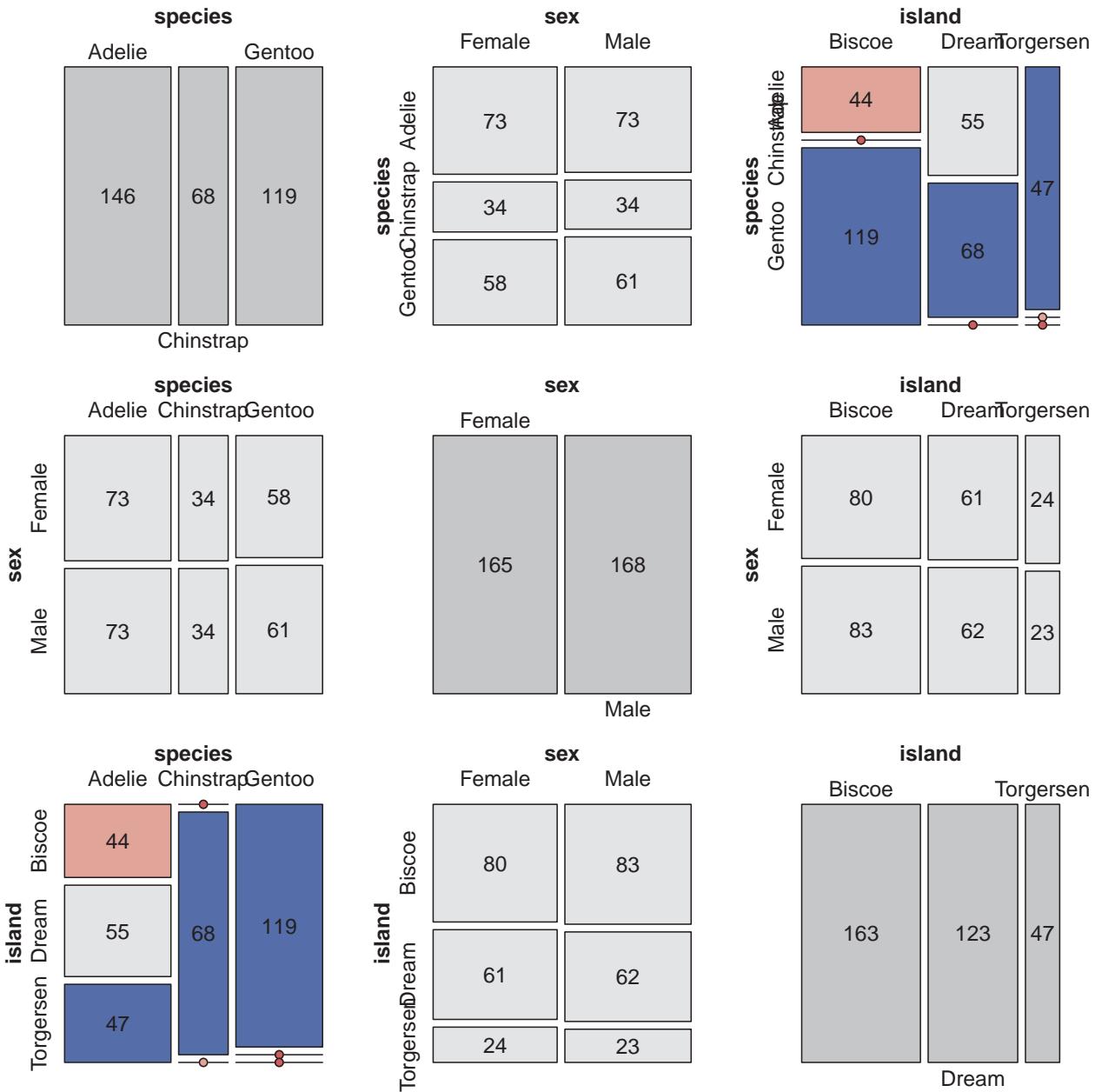


Figure 3.36: Mosaic pairs plot for the combinations of species, sex and island. Diagnonal plots show the marginal frequency of each variable by the width of each rectangle. Off-diagonal mosaic plots subdivide by the conditional frequency of the second variable, shown numerically in the tiles.

indicating independence, $\text{sex} \perp \{\text{species}, \text{island}\}$. More importantly, mosaic pairs plots can show, at a glance, all (bivariate) associations among multivariate categorical variables.

The next step, by John Emerson and others (Emerson et al., 2013) was to recognize that combinations of continuous and discrete, categorical variables could be plotted in different ways.

- Two continuous variables can be shown as a standard scatterplot of points and/or bivariate density contours, or simply by numeric summaries such as a correlation value;
- A pair of one continuous and one categorical variable can be shown as side-by-side boxplots or violin plots, histograms or density plots;

- Two categorical variables could be shown in a mosaic plot or by grouped bar plots.

In the `ggplot2` framework, these displays are implemented using the `ggpairs()` function from the `GGally` package (Schloerke et al., 2025) . This allows different plot types to be shown in the lower and upper triangles and in the diagonal cells of the plot matrix. As well, aesthetics such as color and shape can be used within the plots to distinguish groups directly. As illustrated below, you can define custom functions to control exactly what is plotted in any panel.

The basic, default plot shows scatterplots for pairs of continuous variables in the lower triangle and the values of correlations in the upper triangle. A combination of a discrete and continuous variables is plotted as histograms in the lower triangle and boxplots in the upper triangle. Figure 3.37 includes `sex` to illustrate the combinations.

```
ggpairs(peng, columns=c(3:6, 7),
        aes(color=species, alpha=0.5),
        progress = FALSE) +
  theme_penguins() +
  theme(axis.text.x = element_text(angle = -45))
```

To my eye, printing the values of correlations in the upper triangle is often a waste of graphic space. But in this example the correlations show something peculiar and interesting if you look closely: In all pairs among the penguin size measurements, there are positive correlations within each species, as we can see in Figure 3.31. Yet, in three of these panels, the overall correlation ignoring species is negative. For example, the overall correlation between bill depth and flipper length is $r = -0.579$ in row 2, column 3; the scatterplot in the diagonally opposite cell, row 3, column 2 shows the data. These cases, of differing signs for an overall correlation, ignoring a group variable and the within group correlations are examples of **Simpson's Paradox**, explored later in Section 6.4.2.

The last row and column, for `sex` in Figure 3.37, provides an initial glance at the issue of sex differences among penguin species that motivated the collection of these data. We can go further by also examining differences among species and island, but first we need to understand how to display exactly what we want for each pairwise plot.

`ggpairs()` is extremely general in that for each of the `lower`, `upper` and `diag` sections you can assign any of a large number of built-in functions (of the form `ggally_NAME`), or your own custom function for what is plotted, depending on the types of variables in each plot.

- `continuous`: both X and Y are continuous variables, supply this as the NAME part of a `ggally_NAME()` function or the name of a custom function;
- `combo`: one X or Y variable is discrete while the other is continuous, using the same convention;
- `discrete`: both X and Y are discrete variables.

The defaults, which were used in Figure 3.37, are:

```
upper = list(continuous = "cor",           # correlation values
            combo = "box_no_facet",      # boxplots
            discrete = "count")        # rectangles ~ count
lower = list(continuous = "points",         # just data points
            combo = "facethist",       # faceted histograms
            discrete = "facetbar")     # faceted bar plots
diag  = list(continuous = "densityDiag",    # density plots
            discrete = "barDiag")      # bar plots
```

Thus, `ggpairs()` uses `ggally_cor()` to print the correlation values for pairs of continuous variables in the upper triangle, and uses `ggally_points()` to plot scatterplots of points in the lower portion. The

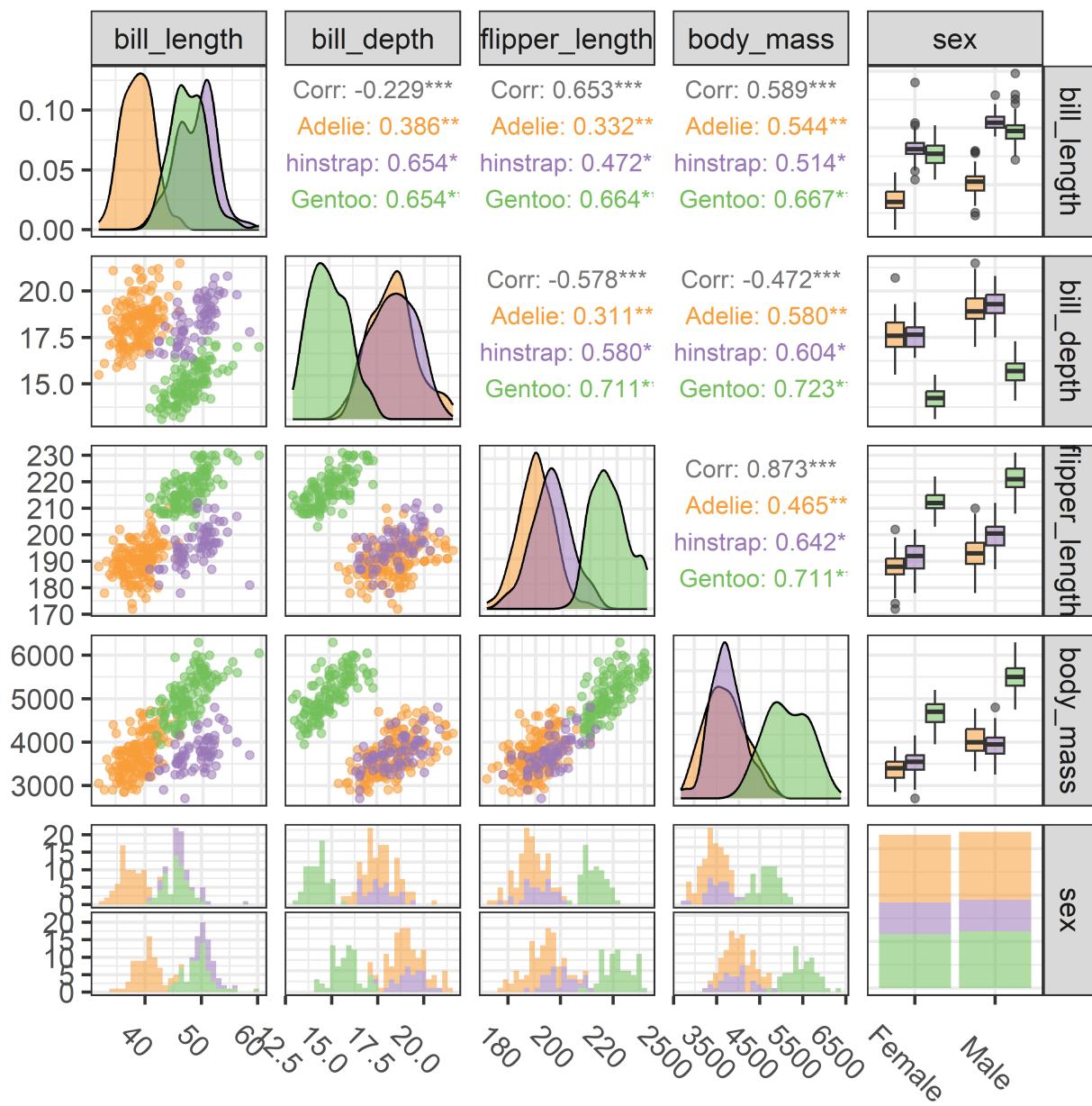


Figure 3.37: Basic `ggpairs()` plot of penguin size variables and sex, stratified by species.

diagonal panels as shown as density plots (`ggally_densityDiag()`) for continuous variables but as bar plots (`ggally_barDiag()`) for discrete factors.

See the vignette, [ggally_plots](#) for an illustrated list of available high-level plots. For our purpose here, which is to illustrate enhanced displays, note that for scatterplots of continuous variables, there are two functions which plot the points and also add a smoother, `_lm` or `_loess`.

```
ls(getNamespace("GGally")) |>
  stringr::str_subset("^ggally_smooth_")
#> [1] "ggally_smooth_lm"    "ggally_smooth_loess"
```

A customized display for scatterplots of continuous variables can be any function that takes `data` and `mapping` arguments and returns a "ggplot" object. The `mapping` argument supplies the aesthetics, e.g., `aes(color=species, alpha=0.5)`, but only if you wish to override what is supplied in the `ggpairs()` call.

Here is a function, `my_panel()` that plots the data points, regression line and loess smooth:

```
my_panel <- function(data, mapping, ...){
  p <- ggplot(data = data, mapping = mapping) +
    geom_point() +
    geom_smooth(method=lm, formula = y ~ x, se = FALSE, ...) +
    geom_smooth(method=loess, formula = y ~ x, se = FALSE, ...)
  p
}
```

For this example, I want only simple summaries of for the scatterplots, so I don't want to plot the data points, but do want to add the regression line and a data ellipse.

```
my_panel1 <- function(data, mapping, ...){
  p <- ggplot(data = data, mapping = mapping) +
    geom_smooth(method=lm, formula = y ~ x, se = FALSE, ...) +
    stat_ellipse(geom = "polygon", level = 0.68, ...)
  p
}
```

Then, to show what can be done, Figure 3.38 uses `my_panel1()` for the scatterplots in the 4 x 4 block of plots in the upper left. The combination of the continuous body size measures and the discrete factors `species`, `island` and `sex` are shown in upper triangle by boxplots but by faceted histograms in the lower portion. The factors are shown as rectangles with area proportional to count (poor-man's mosaic plots) above the diagonal and as faceted bar plots below.

```
ggpairs(peng, columns=c(3:6, 1, 2, 7),
        mapping = aes(color=species, fill = species, alpha=0.2),
        lower = list(continuous = my_panel1),
        upper = list(continuous = my_panel1),
        progress = FALSE) +
  theme_penguins() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  theme(axis.text.x = element_text(angle = -45))
```

There is certainly a lot going on in Figure 3.38, but it does show a high-level overview of all the variables (except `year`) in the penguins dataset. It is probably easiest to "read" this figure by focusing on the four blocks for the combinations of 4 continuous and 3 categorical measures. In the upper left block, visual thinning of the scatterplots, showing only the data ellipses and regression lines gives a simple view as it did in Figure 3.31.

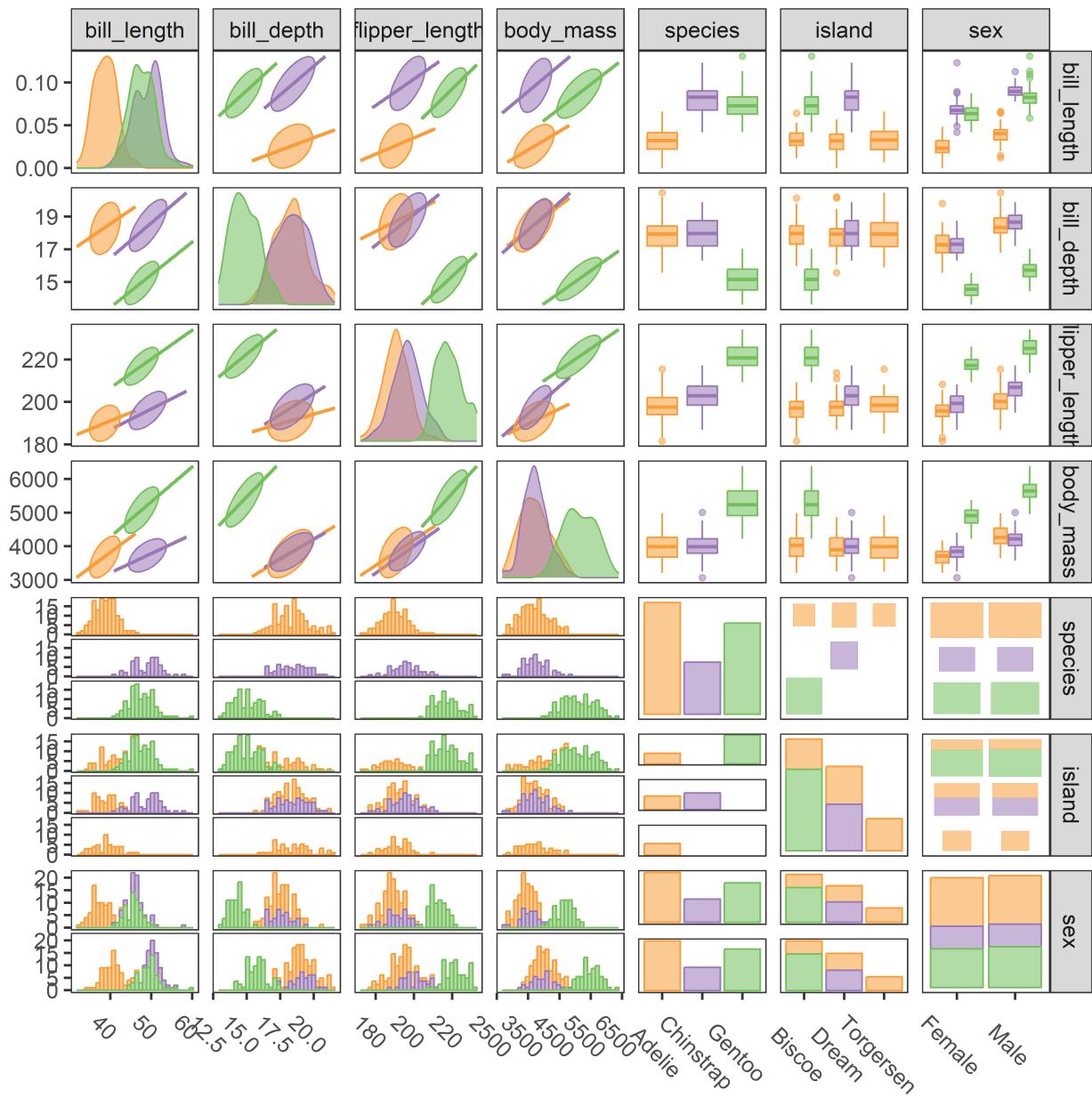


Figure 3.38: Customized `ggpairs()` plot of penguin size variables, together with species, island and sex.

3.13 Parallel coordinate plots

As we have seen above, scatterplot matrices and generalized pairs plots extend data visualization to multivariate data, but these variables share one 2D space, so resolution decreases as the number of variable increase. You need a very large screen or sheet of paper to see more than, say 5-6 variables with any clarity.

Parallel coordinate plots are an attractive alternative, with which we can visualize an arbitrary number of variables to get a visual summary of a potentially high-dimensional dataset, and perhaps recognize outliers and clusters in the data in a different way. In these plots, each variable is shown on a separate, parallel axis.

A multivariate observation is then plotted by connecting their respective values on each axis with lines across all the axes.

The geometry of parallel coordinates is interesting, because what is a point in n -dimensional (Euclidean) *data* space becomes a line in the *projective* parallel coordinate space with n axes, and vice-versa: lines in parallel coordinate space correspond to points in data space. Thus, a collection of points in data space map to lines that intersect in a point in projective space. What this does is to map n -dimensional relations into 2D patterns we can see in a parallel coordinates plot.

History Corner

Those who don't know history are doomed to plagiarize it —The author

The theory of projective geometry originated with the French mathematician Maurice d'Ocagne (1885) who sought a way to provide graphic calculation of mathematical functions with alignment diagrams or *nomograms* using parallel axes with different scales. A three-variable equation, for example, could be solved using three parallel axes, where known values could be marked on their scales, a line drawn between them, and an unknown read on its scale at the point where the line intersects that scale.

Henry Gannet (1880), in work preceding the *Statistical Atlas of the United States* for the 1890 Census (Gannett, 1898), is widely credited with being the first to use parallel coordinates plots to show data, in his case, to show the [rank ordering of US states](#) by 10 measures including population, occupations, wealth, manufacturing, agriculture and so on.

However, both d'Ocagne and Gannet were far preceded in this by Andre-Michel Guerry (1833) who used this method to show how the rank order of various crimes changed with age of the accused. See Friendly (2022), Figure 7 for his version and for an appreciation of the remarkable contributions of this amateur statistician to the history of data visualization.

The use of parallel coordinates for display of multidimensional data was rediscovered by Alfred Inselberg (1985) and extended by Edward Wegman (1990), neither of whom recognized the earlier history. Somewhat earlier, David Andrews (1972) proposed mapping multivariate observations to smooth Fourier functions composed of alternating $\sin()$ and $\cos()$ terms. And in my book, *SAS System for Statistical Graphics* (Friendly, 1991), I implemented what I called *profile plots* without knowing their earlier history as parallel coordinate plots.

Parallel coordinate plots present a challenge for graphic developers, in that they require a different way to think about plot construction for multiple variables, which can be quantitative, as in the original idea, or categorical factors, all to be shown along parallel axes.

Here, I use the `ggpcp` package (Hofmann et al., 2022), best described in VanderPlas et al. (2023), who also review the modern history.⁷ This takes some getting used to, because they develop `pcp_*` extensions of the `ggplot2` grammar of graphics framework to allow:

- `pcp_select()`: selections of the variables to be plotted and their horizontal order on parallel axes,
- `pcp_scale()`: methods for scaling of the variables to each axis,
- `pcp_arrange()`: methods for breaking ties in factor variables to space them out.

Then, it provides `geom_pcp_*` functions to control the display of axes with appropriate aesthetics, labels for categorical factors and so forth. Figure 3.39 illustrates this type of display, using sex and species in addition to the quantitative variables for the penguin data.

```
peng |>
  pcp_select(bill_length:body_mass, sex, species) |>
  pcp_scale(method = "uniminmax") |>
  pcp_arrange() |>
```

⁷Other implementations of parallel coordinate plots in R include: `MASS:::parcoord()`, `GGally:::ggparcoord()` and `PairViz:::pcp()`. The `ggpcp` version used here is the most general.

```
ggplot(aes_pcp()) +
  geom_pcp_axes() +
  geom_pcp(aes(colour = species), alpha = 0.8, overplot = "none") +
  geom_pcp_labels() +
  scale_colour_manual(values = peng.colors()) +
  labs(x = "", y = "") +
  theme(axis.title.y = element_blank(), axis.text.y = element_blank(),
        axis.ticks.y = element_blank(), legend.position = "none")
```

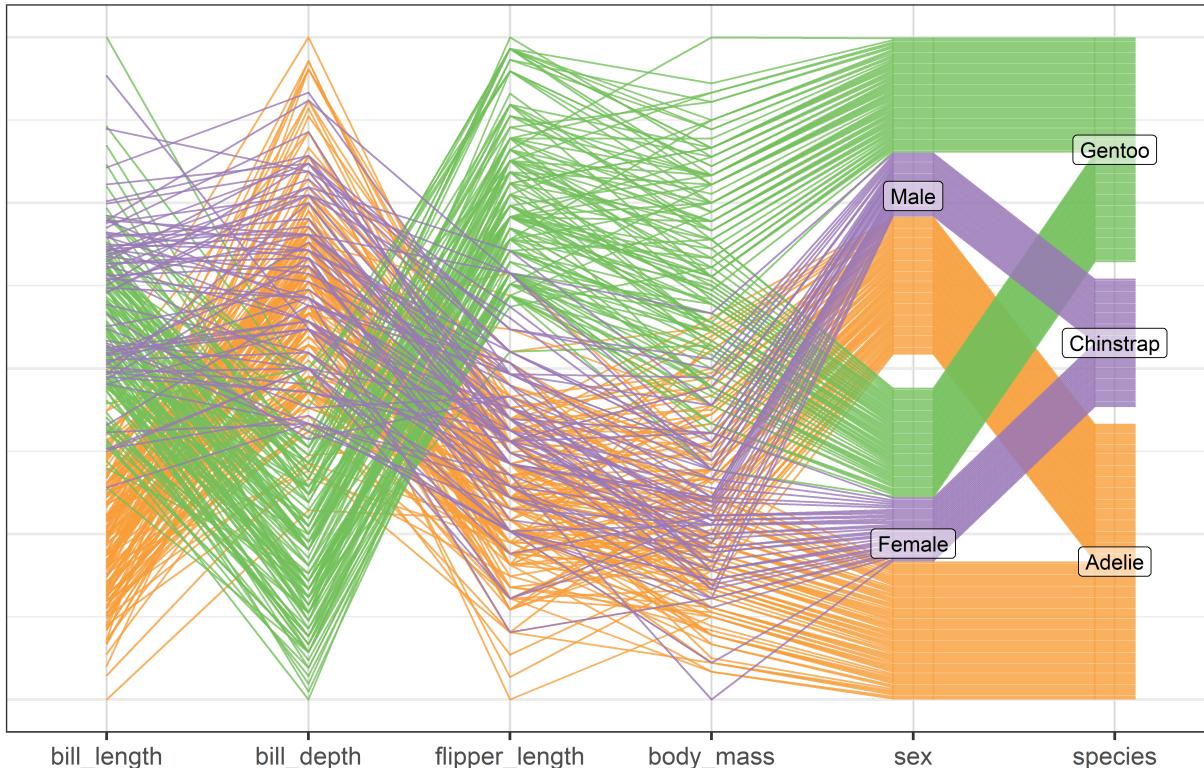


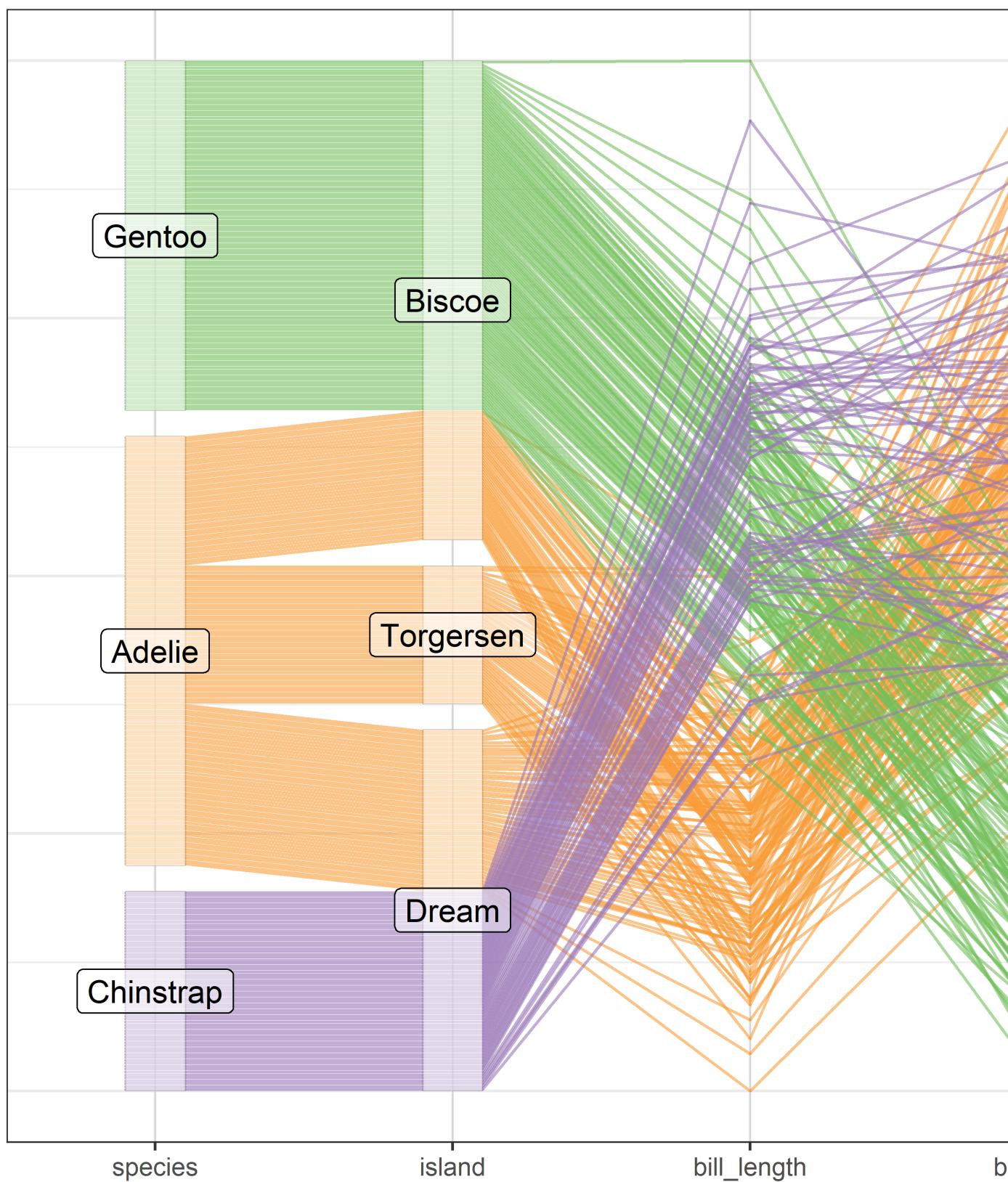
Figure 3.39: Parallel coordinates plot of penguin size variables, together with sex and species.

Rearranging the order of variables and the ordering of factor levels can make a difference in what we can see in such plots. For a simple example (following VanderPlas et al. (2023)), we reorder the levels of species and islands to make it clearer which species occur on each island.

```
peng1 <- peng |>
  mutate(species = factor(species, levels = c("Chinstrap", "Adelie", "Gentoo"))) |>
  mutate(island = factor(island, levels = c("Dream", "Torgersen", "Biscoe")))

peng1 |>
  pcp_select(species, island, bill_length:body_mass) |>
  pcp_scale() |>
  pcp_arrange(method = "from-left") |>
  ggplot(aes_pcp()) +
  geom_pcp_axes()
```

```
geom_pcp(aes(colour = species), alpha = 0.6, overplot = "none") +  
  geom_pcp_boxes(fill = "white", alpha = 0.5) +  
  geom_pcp_labels() +  
  scale_colour_manual(values = peng.colors()[c(2,1,3)]) +  
  theme_bw() +  
  labs(x = "", y = "") +  
  theme(axis.text.y = element_blank(),  
        axis.ticks.y = element_blank(),  
        legend.position = "none")
```



The order of variables in this plot emphasizes the relation between penguin species and the island where they were observed and then shows the values of the quantitative body size measurements. More generally, quantitative variables can, and probably should, be ordered to place the most highly correlated variables adjacently to minimize the degree of crossing lines from one variable to the next (Martí & Laguna, 2003). When variables are highly *negatively* correlated (such as `bill_depth` and `flipper_length` here), crossings can be reduced simply by reversing the scale of one of the variables, e.g., by plotting `-bill_depth`.

3.14 Animated tours

In the mid 17th to early 19th-century the **Grand Tour** became a coming-of-age custom for young Europeans (mainly British nobility and landed gentry) of sufficient rank and means to undertake a journey to the principal sites of Europe (Paris, Geneva, Rome, Athens, ...) to complete their education by learning something of the cultural legacies in history, art, and music from antiquity to the Renaissance. Thereby, they could gain a wider appreciation of history and be prepared to play a role in polite society or in their chosen endeavors.

Travels in high-dimensional data space might be less thrilling than a journey from London through Paris and Milan to Rome. Yet, in both cases it is useful to think of the path taken, and what might be seen along the way. But there are different kinds of tours. We might simply take a meandering tour, exploring all the way, or want to plan a tour to see the most interesting sites in travel or have a tour guided by an expert. Similarly in data space, we might travel randomly to see what we can find or be guided to find interesting features such as clusters, outliers or non-linear relations in data.

Following the demonstration in PRIM-9 (?@sec-discoveries) of exploring multidimensional data space by rotation Asimov (1985) developed the idea of the *grand tour*, a computer method for viewing multivariate statistical data via orthogonal projections onto an animated sequence of low-dimensional subspaces, like a movie. In contrast to a scatterplot matrix which shows a static view of a data cloud projected onto all pairwise variable axes, a statistical tour is like the view of an eye moving smoothly in high-dimensional space, capturing what it sees from a given location onto the 2-d plane of the computer screen.

More generally, statistical tours are a type of dynamic projections onto orthogonal axes (called a *basis*) that embed data in a p -dimensional space into a d -dimensional viewing subspace. Typically, $d = 2$, and the result is displayed as scatterplots, together with vectors representing the projections of the data variables in this space. But the projected data can be rendered in 1-d as densities or histograms, or in other number of dimensions as glyphs, or even as parallel coordinate plots. The essential idea is that we can define, and animate, a *tour path* as a smooth sequence of such projections over small changes to the projection basis, which gives the orientation of the data in the viewing space.

3.14.1 Projections

The idea of a projection is fundamental to touring methods and other visualizations of high-D data, so it is useful to understand what a projection is. Quite simply, you can think of a projection as the shadow of an object or cloud of points. This is nicely illustrated by the cover image (Figure 3.40) used for Douglas Hofstadter's (1979) *Gödel, Bach and Escher* which shows 3D solid shapes illuminated by light sources so their shadows form the letters G, B and E projected onto the planes formed by pairs of the three coordinate axes. The set of three 2D views is essentially the same that we see in a scatterplot matrix, where a 3D dataset is portrayed by the set of shadows of the points on planes formed by pairs of coordinate axes.

In the simplest case, a data point $\mathbf{x} = (x_1, x_2)$ in two dimensions can be represented geometrically as a vector from the origin as shown in Figure 3.41. This point can be projected on any one-dimensional axis \mathbf{p} by dropping a line perpendicular to \mathbf{p} , which is the idea of a shadow. Mathematically, this is calculated as the product $\mathbf{x}^T \mathbf{p} = x_1 p_1 + x_2 p_2$ and suitably normalized to give the correct length. ...

More generally, a projection of an $(n \times p)$ data matrix \mathbf{X} representing n observations in p dimensions onto a

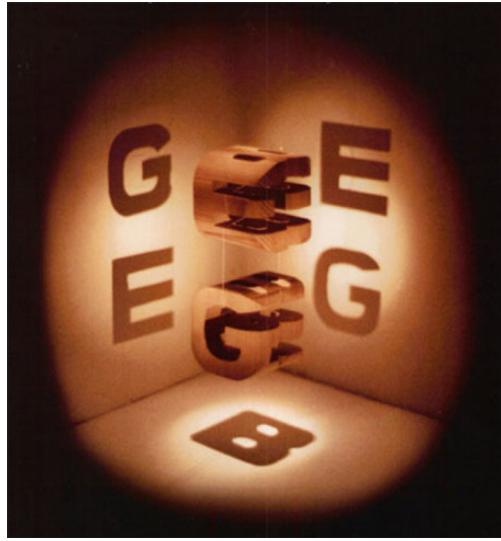


Figure 3.40: The cover image from Hofstadter (1979) illustrates how projections are shadows of an object cast by a light from a given direction.

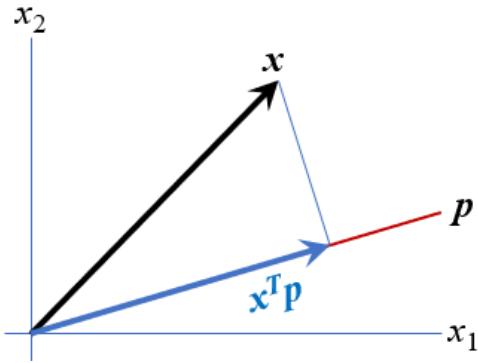


Figure 3.41: Projection of a point \mathbf{x} onto a direction or axis \mathbf{p} .

d -dimensional viewing space $\mathbf{Y}_{n \times d}$ is represented by a $p \times d$ projection matrix \mathbf{P} as $\mathbf{Y} = \mathbf{X}\mathbf{P}$, where the columns of \mathbf{P} are orthogonal and of unit length, i.e., $\mathbf{P}^T\mathbf{P} = \mathbf{I}_{(d \times d)}$.

For example, to project a data matrix \mathbf{X} in three dimensions onto a 2D plane, we would multiply it by a (3×2) orthonormal matrix \mathbf{P} . The matrix \mathbf{P}_1 below simply selects the first two columns of \mathbf{X} .⁸

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 10 \\ 0 & 10 & 0 \\ 0 & 10 & 10 \\ 10 & 0 & 0 \\ 10 & 0 & 10 \\ 10 & 10 & 0 \\ 10 & 10 & 10 \end{bmatrix}_{8 \times 3} ; \mathbf{P}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}_{3 \times 2} \Rightarrow \mathbf{Y} = \mathbf{X} \mathbf{P}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 10 \\ 0 & 10 \\ 10 & 0 \\ 10 & 0 \\ 10 & 10 \\ 10 & 10 \end{bmatrix}_{8 \times 2}$$

An oblique projection using all three dimensions is given by \mathbf{P}_2 below. This produces a new 2D view in \mathbf{Y} :

⁸This example was modified from one used by D. Cook et al. (2008).

$$\mathbf{P}_2 = \begin{bmatrix} 0.71 & -0.42 \\ 0.71 & 0.42 \\ 0 & 0.84 \end{bmatrix}_{3 \times 2} \Rightarrow \mathbf{Y} = \mathbf{X} \mathbf{P}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 8.4 \\ 7.1 & 4.2 \\ 7.1 & 12.6 \\ 7.1 & -4.2 \\ 7.1 & 4.2 \\ 14.2 & 0 \\ 14.2 & 8.4 \end{bmatrix}$$

The columns in \mathbf{Y} are simply the linear combinations of those of \mathbf{X} using the weights in each column of \mathbf{P}_2

$$\begin{aligned}\mathbf{y}_1 &= 0.71\mathbf{x}_1 + 0.71\mathbf{x}_2 + 0\mathbf{x}_3 \\ \mathbf{y}_2 &= -0.42\mathbf{x}_1 + 0.42\mathbf{x}_2 + 0.84\mathbf{x}_3\end{aligned}$$

```
vals <- c(0, 10)
X <- expand.grid(x1 = vals, x2=vals, x3=vals) |> as.matrix()

# project on just x1, x2 plane
P1 <- rbind(diag(2), c(0,0))
Y1 <- X %*% P1

# oblique projection
P2 <- matrix(c(0.71, 0.71, 0, -0.42, .42, 0.84), ncol=2)
Y2 <- X %*% P2
```

In this example, the matrix \mathbf{X} consists of 8 points at the vertices of a cube of size 10, as shown in Figure 3.42 (a). The projections $\mathbf{Y}_1 = \mathbf{P}_1\mathbf{X}$ and $\mathbf{Y}_2 = \mathbf{P}_2\mathbf{X}$ are shown in panels (b) and (c). To make it easier to relate the points in different views, shapes and colors are assigned so that each point has a unique combination of these attributes.⁹

```
pch <- rep(15:18, times = 2)
colors <- c("red", "blue", "darkgreen", "brown")
col <- rep(colors, each = 2)
data.frame(X, pch, col)
#>   x1 x2 x3 pch      col
#> 1  0  0  0  15     red
#> 2 10  0  0  16     red
#> 3  0 10  0  17    blue
#> 4 10 10  0  18    blue
#> 5  0  0 10  15 darkgreen
#> 6 10  0 10  16 darkgreen
#> 7  0 10 10  17   brown
#> 8 10 10 10  18   brown
```

But, if we are traveling in the projection space of \mathbf{Y} , we need some signposts to tell us how the new dimensions relate to those of \mathbf{X} . The answer is provided simply by plotting the rows of \mathbf{P} as vectors, as shown in Figure 3.43. In these plots, each row of \mathbf{P}_1 and \mathbf{P}_2 appears as a vector from the origin. Its direction shows the contribution each of $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ make to the new coordinates \mathbf{y}_1 and \mathbf{y}_2 .

⁹Plot shapes given by `pch = 15:18` correspond to: filled square (15), filled circle (16), filled triangle point-up (17), filled diamond (18).

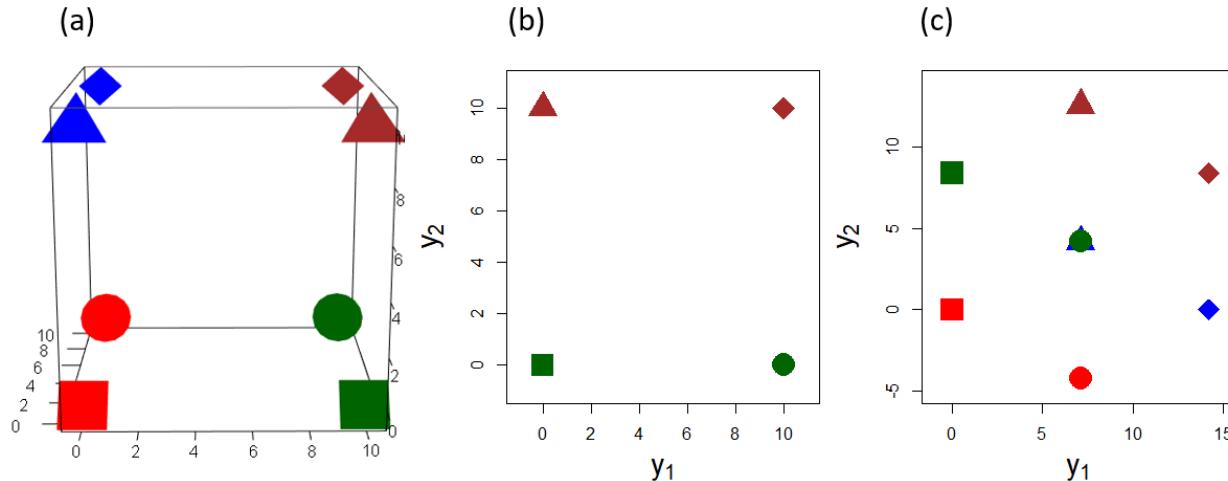


Figure 3.42: Projection example: (a) The 8 points in \mathbf{X} form a cube of size 10; (b) the projection by \mathbf{P}_1 is the view ignoring \mathbf{x}_3 (two points coincide at each vertex); (c) the projection by \mathbf{P}_2 is an oblique view.

In \mathbf{P}_1 , the projected variable y_1 is related only to \mathbf{x}_1 , while y_2 is related only to \mathbf{x}_2 ; \mathbf{x}_3 makes no contribution, and appears at the origin. However in the projection given by \mathbf{P}_2 , \mathbf{x}_1 and \mathbf{x}_2 make the same contribution to y_1 , while \mathbf{x}_3 has no contribution to that horizontal axis. The vertical axis, y_2 here is completely aligned with \mathbf{x}_3 ; \mathbf{x}_1 and \mathbf{x}_2 have vertical components that are half of that for \mathbf{x}_3 in absolute value.

```
library(matlib)
op <- par(mar=c(4, 5, 1, 1)+.1)
xlim <- ylim <- c(-1.1, 1.1)
axes.x <- c(-1, 1, NA, 0, 0)
axes.y <- c(0, 0, NA, -1, 1)
labs <- c(expression(x[1]), expression(x[2]), expression(x[3]))
plot(xlim, ylim, type = "n", asp=1,
      xlab = expression(y[1]), ylab = expression(y[2]),
      cex.lab = 1.8)
circle(0, 0, 1, col = adjustcolor("skyblue", alpha = 0.2))
lines(axes.x, axes.y, col = "grey")
vectors(P1, labels = labs, cex.lab = 1.8, lwd = 3, pos.lab = c(4, 2, 1))

plot(xlim, ylim, type = "n", asp=1,
      xlab = expression(y[1]), ylab = expression(y[2]),
      cex.lab = 1.8)
circle(0, 0, 1, col = adjustcolor("skyblue", alpha = 0.2))
lines(axes.x, axes.y, col = "grey")
vectors(P2, labels = labs, cex.lab = 1.8, lwd = 3)
par(op)
```

3.14.1.1 Vector lengths

In Figure 3.43, the **lengths** of the \mathbf{x} vectors reflect the relative degree to which each variable is represented in the space of the projection, and this is important for interpretation. For the \mathbf{P}_1 projection, \mathbf{x}_3 is of length 0, while \mathbf{x}_1 and \mathbf{x}_2 fill the unit circle. In the projection given by \mathbf{P}_2 , all three \mathbf{x} are approximately the same length.

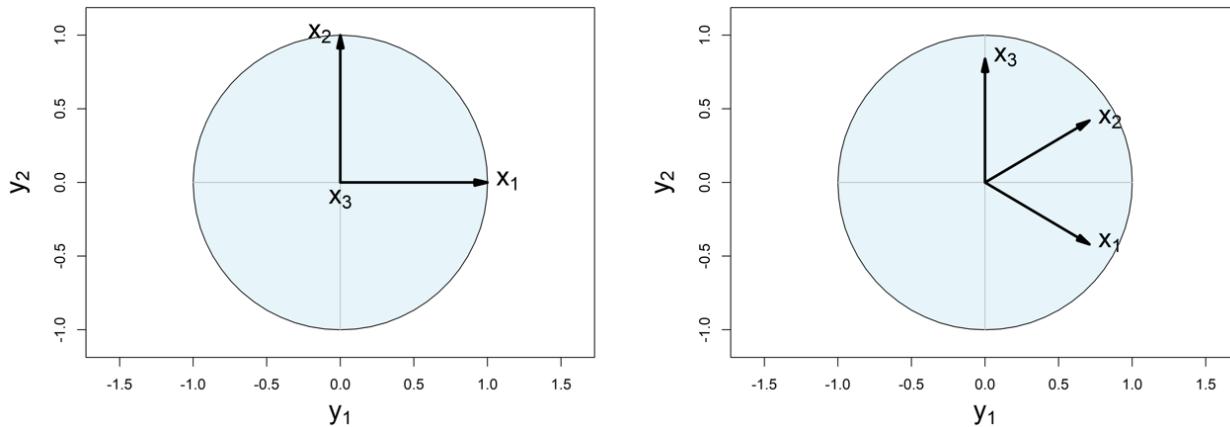


Figure 3.43: Variable vectors: Data variables viewed as vectors in the space of their projections. The angles of the \mathbf{x} vectors with respect to the \mathbf{y} coordinate axes show their relative contributions to each. The lengths of the \mathbf{x} vectors show the relative degree to which they are represented in the space of \mathbf{y} s. Left: the $\mathbf{P1}$ projection; right: the $\mathbf{P2}$ projection.

In algebra, the length of a vector \mathbf{x} is $\|\mathbf{x}\| = (\mathbf{x}^T \mathbf{x})^{1/2} = \sqrt{\sum x_i^2}$, the Euclidean distance of the tip of the vector from the origin. In R, we calculate the lengths of row vectors in a projection matrix by transposing and using `matlib::len()`.

```
P1 |> t() |> matlib::len()
#> [1] 1 1 0
P2 |> t() |> matlib::len()
#> [1] 0.825 0.825 0.840
```

3.14.1.2 Joint-views

To interpret such projections, we want to see **both** the projected data and the signposts that tell us where we are in relation to the original variables. To do this, we can overlay the variable vectors represented by the rows of the projection matrix \mathbf{P} onto plots like Figure 3.42 (b) and Figure 3.42 (c) to see how the axes in a projection relate to those in the data. To place these together on the same plot, we can either center the columns of \mathbf{Y} at their means or shift the the columns of \mathbf{P} to `colMeans(Y)`. It is only the directions of the vectors that matters, so we are free to scale their lengths by any convenient factor.

```
Y2s <- scale(Y2, scale=FALSE)           # center Y2
plot(Y2s, cex = 3,
      asp = 1,
      pch = pch, col = col,
      xlab = expression(y[1]), ylab = expression(y[2]),
      xlim = c(-10, 10), ylim = c(-10, 10), cex.lab = 1.8)
r <- 7
vecs <- (r*diag(3) %*% P2)
vectors(vecs, labels = labs, cex.lab = 1.8, lwd = 2)
vectors(-vecs, labels = NULL, lty = 1, angle = 1, col = "gray")
```

The plot in Figure 3.44 illustrates this, centering \mathbf{Y} , and multiplying the vectors in \mathbf{P} by 7. To check your understanding, try to see if you can relate what is shown in this plot to the 3D plot in Figure 3.42 (a).

The idea of viewing low-dimensional projections of data together with vectors representing the contributions of

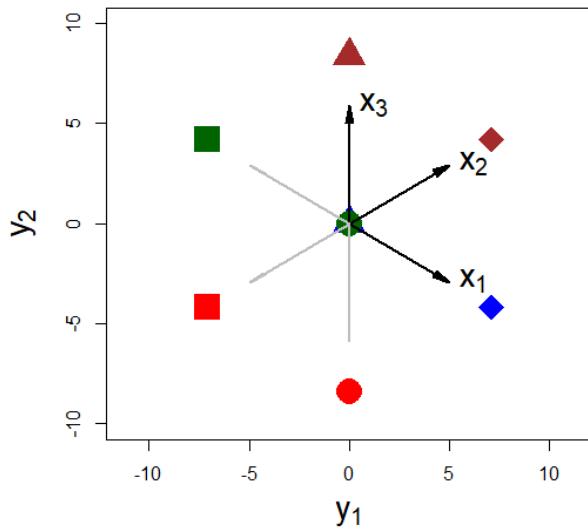


Figure 3.44: The \mathbf{P}_2 projection of the data showing vectors for the original variables in the space of \mathbf{Y} .

the original variables to the dimensions shown in a display is also the basis of **biplot** techniques (Section 4.3) we will use in relation to principal components analysis.

3.14.2 Touring methods

The trick of statistical touring methods is to generate a smooth sequence of interpolated projections $\mathbf{P}_{(t)}$ indexed by time t , $\mathbf{P}_{(1)}, \mathbf{P}_{(2)}, \mathbf{P}_{(3)}, \dots, \mathbf{P}_{(T)}$. This gives a path of views $\mathbf{Y}_{(t)} = \mathbf{XP}_{(t)}$, that can be animated in successive frames, as shown schematically in Figure 3.45.

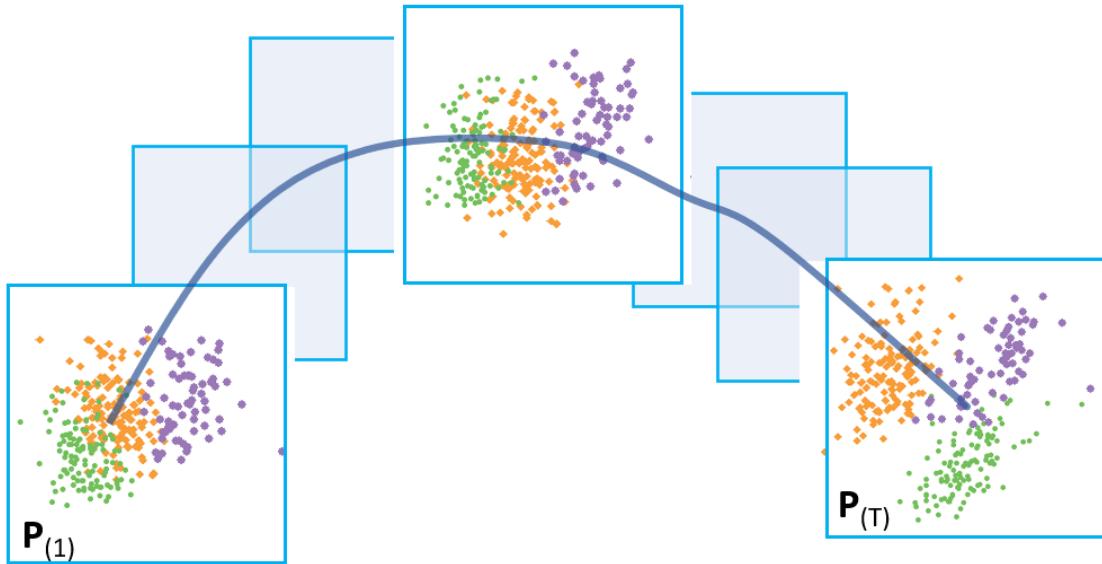


Figure 3.45: Interpolations: Illustration of a grand tour of interpolations of projection planes showing 2D scatterplots of the Penguin dataset. The sequence of views moves smoothly from an initial frame $\mathbf{P}(1)$ to a final frame $\mathbf{P}(T)$ where the penguin species are widely separated.

Asimov's (1985) original idea of the grand tour was that of a random path, picking orthogonal projections

$\mathbf{P}_{(i)}$ at random. Given enough time, the grand tour gives a space-filling path and would eventually show every possible projection of the data. But it does so smoothly, by interpolating from one projection to the next. In the travel analogy, the path by road from London to Paris might go smoothly through Kent to Dover, thence via Amiens and Beauvais before reaching Paris. By air, the tour would follow a smoother *geodesic* path, and this is what the grand tour does. The sense in watching an animation of a statistical grand tour is that of continuous motion. The grand tour algorithm is described in detail by Buja et al. (2005) and D. Cook et al. (2008).

3.14.2.1 Guided tours

The next big idea was that rather than traveling randomly in projection space one could take a *guided tour*, following a path that leads to “interesting projections”, such as those that reveal clusters, gaps in data space or outliers. This idea, called *projection pursuit* (D. Cook et al., 1995), works by defining a measure of interestingness of a data projection. In a guided tour, the next projection is chosen to increase that index, so over time the projection moves toward one that is maximizes that index.

In the time since Asimov (1985), there have been many implementations of touring visualization methods. XGobi (Swayne et al., 1998) for X-Windows displays on Linux systems provided a test-bed for dynamic, interactive graphic methods; its successor, GGobi (D. Cook & Swayne, 2007; Swayne et al., 2003) extended the range of touring methods to include a wider variety of projection pursuit indices.

3.14.2.2 tourr package

The current state of art is best captured in the `tourr` package for R (Wickham et al., 2011; Wickham & Cook, 2025). It defines a tour to consist of three components:

- **data:** An $(n \times p)$ numerical data matrix to be viewed.
- **path:** A tour path function that produces a smoothed sequence of projection matrices $\mathbf{P}_{(p \times d)}$ in d . dimensions, for example `grand_tour(d = 2)` or `guided_tour(index = holes)`.
- **display:** A function that renders the projected data, for example `display_xy()` for a scatterplot, `display_depth()` for a 3D plot with simulated depth, or `display_pcp()` for a parallel coordinates plots

This very nicely separates the aspects of a tour, and allows one to think of and define new tour path methods and display methods. The package defines two general tour functions: `animate()` produces a real-time animation on a display device and `render()` saves image frames to disk, such as a .gif file.

```
animate(data, tour_path, display_method)
render(data, tour_path, display_method)
```

The `tourr` package provides a wide range of tour path methods and display methods:

```
# tour path methods
grep("_tour$", ls.str("package:tourr"), value = TRUE)
#> [1] "dependence_tour"      "frozen_guided_tour"
#> [3] "frozen_tour"          "grand_tour"
#> [5] "guided_anomaly_tour" "guided_section_tour"
#> [7] "guided_tour"          "little_tour"
#> [9] "local_tour"           "new_tour"
#> [11] "planned_tour"        "planned2_tour"
#> [13] "radial_tour"

# display methods
grep("display_", ls.str("package:tourr"), value = TRUE)
#> [1] "display_andrews"    "display_density2d" "display_depth"
#> [4] "display_dist"       "display_faces"     "display_groupxy"
```

```
#> [7] "display_idx"      "display_image"    "display_pca"
#> [10] "display_pcp"     "display_sage"     "display_scatmat"
#> [13] "display_slice"   "display_stars"   "display_stereo"
#> [16] "display_trails"  "display_xy"
```

Tour path methods take a variety of optional arguments to specify the detailed behavior of the method. For example, most allow you to specify the number of dimension (`d =`) of the projections. The `guided_tour()` is of particular interest here.

```
args(guided_tour)
#> function (index_f, d = 2, cooling = 0.99, max.tries = 25, max.i = Inf,
#>   search_f = search_geodesic, n_jellies = 30, n_sample = 100,
#>   alpha = 0.5, ...)
#> NULL
```

In this, `index_f` specifies a function that the method tries to optimize on its path and package defines four indices:

- Holes (`holes()`): This is sensitive to projections with separated clusters of points, with few points near the origin
- Central mass (`cmass()`): Sensitive to projections with lots of points in the center, but perhaps with some outliers
- Linear discriminant analysis (`lda_pp()`): For data with a grouping factor, optimizes a measure of separation of the group means as in MANOVA or linear discriminant analysis.
- PDA analysis (`pda_pp()`): A penalized version of `lda_pp()` for cases of large p relative to sample size n (E.-K. Lee & Cook, 2009).

In addition, there is now a `guided_anomaly_tour()` that looks for the best projection of observations that are outside the data ellipsoid, finding a view showing observations with large Mahalanobis distances from the centroid.

3.14.2.3 Penguin tours

Penguins are a traveling species. They make yearly travels inland to breeding sites in early spring, repeating the patterns of their ancestors. Near the beginning of summer, adult penguins and their chicks return to the sea and spend the rest of the summer feeding there (Black et al., 2018). If they were also data scientists, they might wonder about the relations among among their cousins of different species and take a tour of their measurements...

For example, using the Penguins dataset, the following calls produce grand tours in 2, 3, and 4 dimensions. The 2D tour is displayed as a scatterplot, the 3D tour using simulated depth as shown by variation in point size and transparency, and the 4D tour is shown using a parallel coordinate plot.

```
data(peng, package = "heplots")
peng_scaled <- scale(peng[,3:6])
colnames(peng_scaled) <- c("BL", "BD", "FL", "BM")

animate(peng_scaled, grand_tour(d = 2), display_xy())
animate(peng_scaled, grand_tour(d = 3), display_depth())
animate(peng_scaled, grand_tour(d = 4), display_pcp())
```

To illustrate, I'll start with a grand tour designed to explore this 4D space of penguins. I'll abbreviate the variables to two characters, “BL” = `bill_length`, “BD” = `bill_depth`, “FL” = `flipper_length`, and “BM” = `body_mass` and identify the penguin species using point shape (`pch`) and color (`col`).

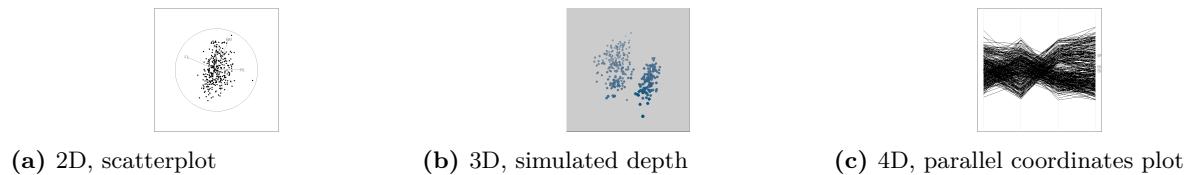


Figure 3.46: Grand tours of the penguin dataset in 2, 3, and 4 dimensions using different `display_*`() methods.

As you watch this pay attention to the separation of the species and any other interesting features. What do you see?

```
data(peng, package = "heplots")
peng_scaled <- scale(peng[,3:6])
colnames(peng_scaled) <- c("BL", "BD", "FL", "BM")

pch <- c(15, 16, 17)[peng$species]
cex = 1.2

set.seed(1234)
animate(peng_scaled,
        tour_path = grand_tour(d=2),
        display_xy(col = peng$species,
                   palette = peng.colors("dark"),
                   pch = pch, cex = cex,
                   axis.col = "black",
                   axis.text.col = "black",
                   axis.lwd = 1.5))
```

Figure 3.47 shows three frames from this movie. The first (a) is the initial frame that shows the projection in the plane of bill depth and bill length. The variable vectors indicate that bill length differentiates Adelie penguins from the others. In frame (b), the three species are widely separated, with bill depth distinguishing Gentoo from the others. In frame (c) the three species are largely mixed, but two points stand out as outliers, with exceptionally long bills compared to the rest.

Let's take the penguins on a guided tour, trying to find views that show the greatest separations among the penguin species; that is, a guided tour, optimizing the `lda_pp()` index.

```
set.seed(1234)
animate(peng_scaled,
        guided_tour(lda_pp(peng$species)),
        display_xy(col = peng$species,
                   palette = peng.colors("dark"),
                   pch = pch,
                   cex = cex)
)
```

TODO: I'm trying to balance what will/can be shown in the HTML version vs. the printed PDF. Needs text here specifically for the PDF version.

These examples are intended to highlight what is possible with dynamic graphics for exploring high-dimensional data visually. D. Cook & Laa (2024) extend the discussion of these methods from D. Cook & Swayne (2007)

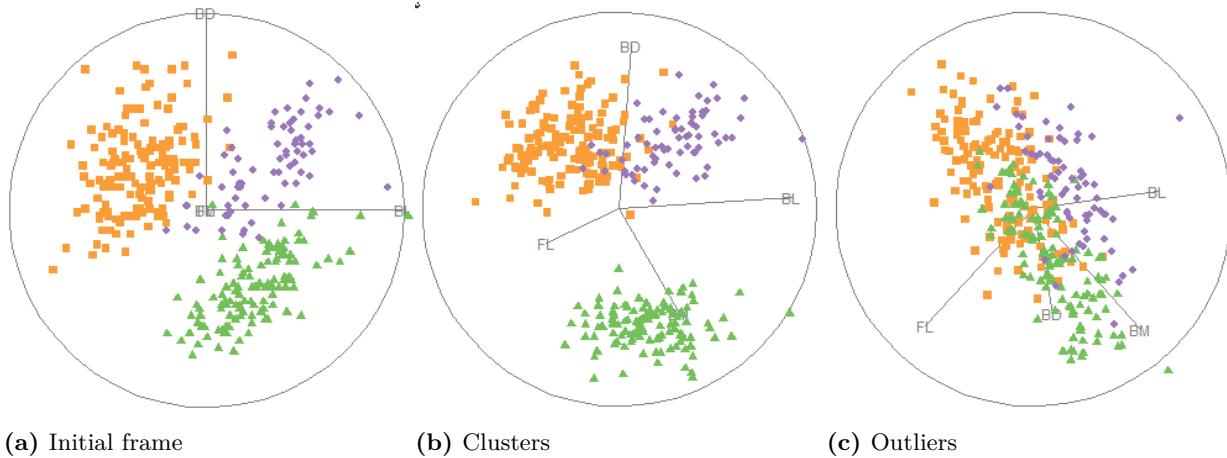


Figure 3.47: Three frames from the grand tour of the Penguin data. (a) The initial frame is the projection showing only BD and BL, where bill length conveniently separates Adelie from the other two species. (b) A frame that shows the three species more widely separated. (c) A frame that shows two outliers with very large bills.

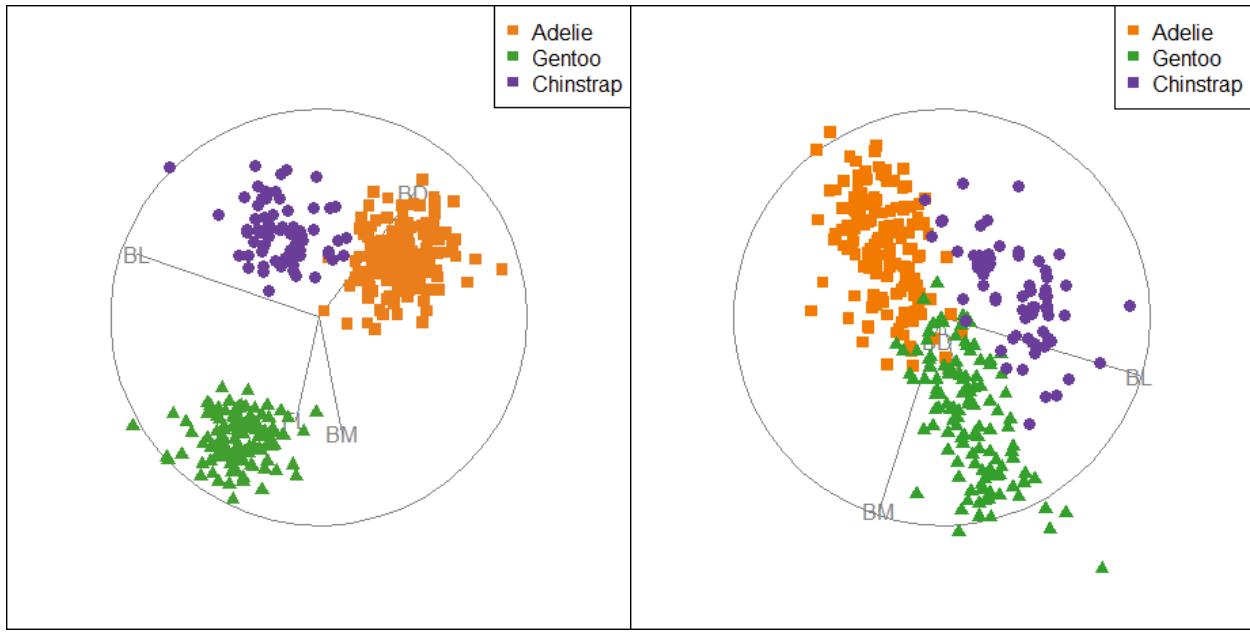


Figure 3.48: Guided tours: These figures show the final frame in the animations of guided tours designed to find the projection that optimize an index. (a) The `lda_pp()` criterion optimizes the separation of the means relative to within-group variation. (b) The `anomalies_index()` optimizes the average Mahalanobis distance of points from the centroid

(which used Ggobi) to the `tourrr` package. They illustrate dimension reduction, various cluster analysis methods, trees and random forests and some machine-learning techniques.

Ideally, we should be able interact with a tour,

- pausing when we see something interesting and saving the view for later analysis;
- selecting or highlighting unusual points,

- changing tour methods or variables displayed on the fly, and so forth.

Some packages that provide these capabilities are: **detourr** (Hart & Wang, 2022) **liminal** (S. Lee, 2021) and **langevitour** (Harrison, 2023, 2025) The **loon** package (Waddell & Oldford, 2023) is a general toolkit that enables highly interactive data visualization. It provides a **loon.tour** package (Xu & Oldford, 2021) for using touring methods within the **loon** environment.

3.15 Network diagrams

A major theme throughout this chapter has been to understand how to extend data visualization from simple bivariate scatterplots to increasingly more complex situations with larger datasets. With a moderate number of variables, techniques such as smoothing, summarizing with data ellipses and fitted curves, and visual thinning can be used to tame “big N ” datasets with thousands of cases.

However “big p ” datasets, with more than a moderate number (p) of variables still remain a challenge. It is hard to see how the more advanced methods (corrgrams, parallel coordinate) described earlier could cope with $p = 20, 50, 100, 500, \dots$ variables. At some point, each of these begins to break down for the purpose of visualizing associations among many variables. We are forced to thin the information presented in graphs more and more as the number of variables increases.

It turns out that there is a way to increase the number of variables displayed dramatically, if we are mainly interested in the pairwise correlations for reasonably normally distributed data. A graphical **network diagram** portrays variables by *nodes* (vertices), connected by (weighted) *edges* whose properties reflect the strength of connections between pairs, such as a correlation. Such diagrams can reveal properties not readily seen by other means.

As an example consider Figure 3.49, which portrays the correlations among 25 self-report items reflecting 5 factors (the “Big Five”) considered in personality psychology to represent the dominant aspects of all of personality. These factors are easily remembered by the acronym **OCEAN**: Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism. The dataset, **bfi**, contains data from an online sample of $n = 2800$ with 5 items for each scale.

In this figure (taken from Rodrigues (2021)), the item nodes are labeled according to the OCEAN factor they are assumed to measure. For 25 items, there are $25 \times 24/2 = 300$ correlations, way too much to see. A clearer picture arises when we reduce the number of edges shown according to some criterion. Here, edges are drawn *only* between nodes where the correlation is considered important by a method (“glasso” = graphical LASSO) designed to make the graph optimally sparse.

The edges shown in Figure 3.49 reflect the Pearson correlation between a given pair of items by the visual attributes of color and line style: magnitude is shown by both the thickness and transparency of the edge; the sign of the correlation is shown by color and line type: solid blue for positive correlations and dashed red for negative ones.

According to some theories, the five personality factors should be largely non-overlapping, so there should not be many edges connecting items of one factor with those of another. Yet, there are quite a few cross-factor connections in Figure 3.49, so perhaps the theory is wrong, or, more likely, the 25 items are not good representatives of these underlying dimensions. The network diagram shown here is a visual tool for thought and refinement. See Costantini et al. (2015) for a tutorial on network analysis of personality data in R.

Network diagrams stem from mathematical graph theory (Bondy & Murty, 2008; West, 2001) of the abstract properties of nodes and edges used to represent pairwise relationships. These can be used to model many types of relations and processes in physical, biological, social and other sciences, where such properties as connectedness, centrality, cliques of connected nodes and so forth provide a vocabulary used to understand and explain complex systems.

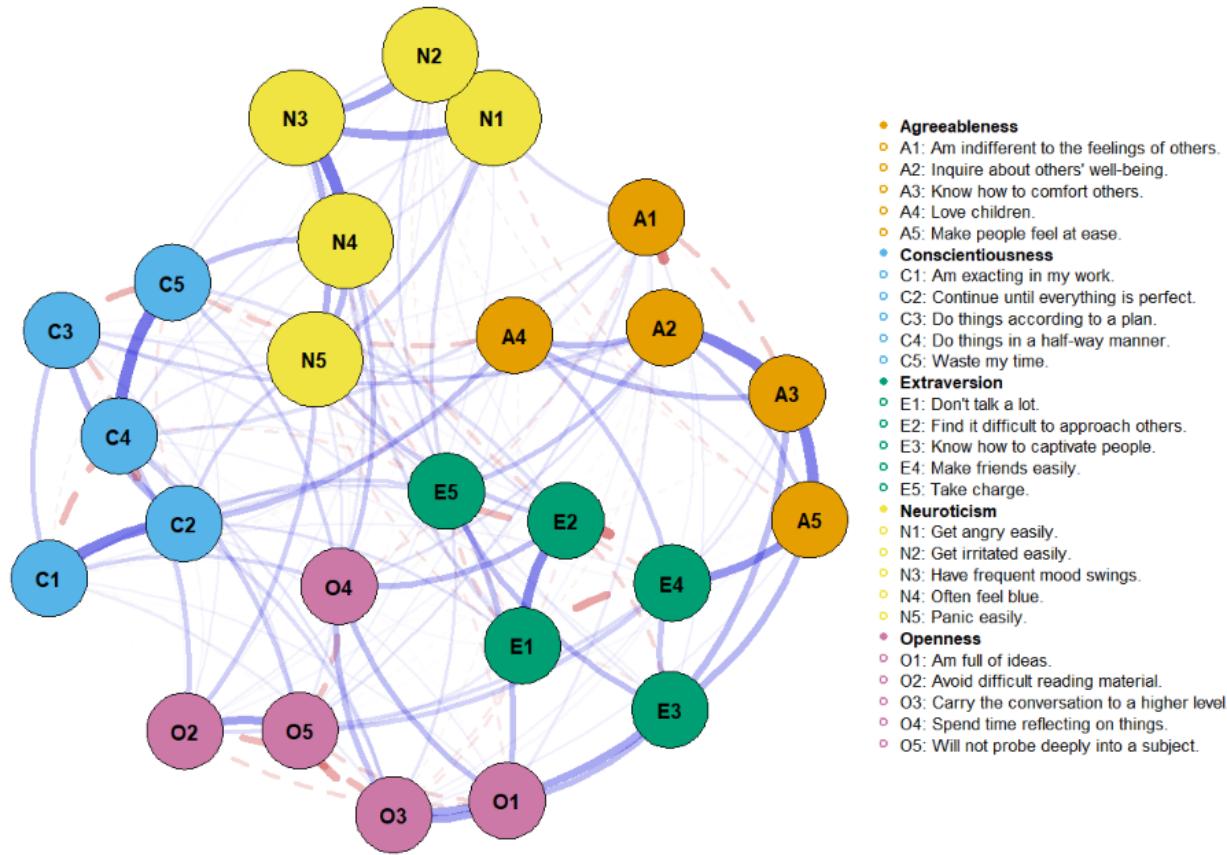


Figure 3.49: Network diagram of the correlations among 25 items from a Big-Five personality scale, 5 items for each scale. The magnitude of a correlation is shown by the thickness and transparency of the edge between two item nodes. The sign of a correlation is shown by edge color and style: solid blue for positive and dashed red for negative. *Source: Rodrigues (2021)*

For one example, Grandjean (2016) used network analysis to study the connections among 2500 Twitter users (nodes) who identified as belonging to a “digital humanities” community from the relations (edges) of who follows whom. Grandjean also used these methods to study the relationships among [characters in Shakespeare’s tragedies](#) in terms of the characters (nodes) and edges representing how often they appeared in the same scene.

The wide applicability of these ideas has led to what is now called *network science* (Barab’asi, 2016) encompassing computer networks, biological networks, cognitive and semantic networks, and social networks. Recent developments in psychology led to a framework of *network psychometrics* (Ivoranu et al., 2022), where, for example, symptoms of psychopathology (phobias, anxiety, substance abuse) can be conceptualized as an interconnected network of clusters and studied for possible causal relations (Robinaugh et al., 2019).

Because a network diagram can potentially reflect hundreds of variables, various [graph layout algorithms](#) have been developed to automatically position the nodes so as to generate aesthetically pleasing network visualizations that emphasize important structural properties, like clusters and central nodes, while minimizing visual clutter (many crossing lines) to promote understandability and usability.

There are quite a few R packages for constructing network diagrams, both static and dynamic / interactive, and these differ considerably in how the information required for a graph is structured as R objects, and the flexibility to produce attractive graphs. Among these, [igraph](#) (Csárdi et al., 2024) structures the data as a dataset of vertices and edges with properties

-> packages: qgraph, ...

3.15.1 Crime data

For the present purposes, let's see what network diagrams can tell us about the crime data analyzed earlier. Here, I first reorder the variables as in Figure 3.35. In the call to `qgraph()`, the argument `minimum = "sig"` says to show only the edges for significant correlations (at $\alpha = 0.01$ here). In Figure 3.50, the variable nodes are positioned around a circle (`layout = "circle"`), which is the default.

```
library(qgraph)
ord <- corrMatOrder(crime.cor, order = "AOE")
rownames(crime.cor) [ord]
#> [1] "murder"    "assault"    "rape"       "robbery"    "burglary"
#> [6] "larceny"    "auto"
crime.cor <- crime.cor[ord, ord]

# "association graph": network of correlations
qgraph(crime.cor,
        title = "Crime data:\ncorrelations", title.cex = 1.5,
        graph = "cor",
        layout = "circle",
        minimum = "sig", sampleSize = nrow(crime), alpha = 0.01,
        color = grey(.9), vsizer = 12,
        labels = rownames(crime.cor),
        posCol = "blue")
```

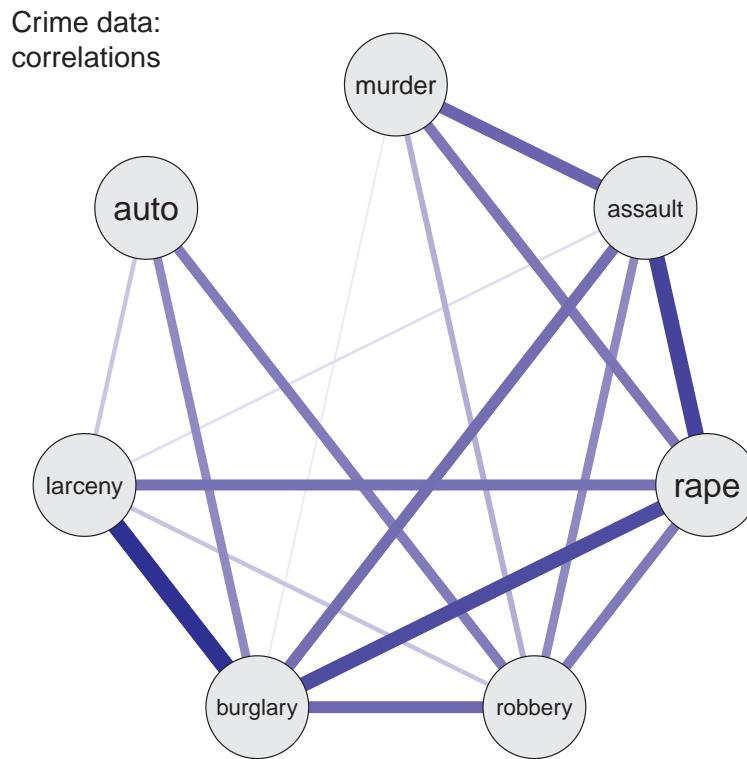


Figure 3.50: Network diagram depicting the correlations among the crime variables. Only edges for correlations that are significant at the $\alpha = 0.01$ level are displayed.

In this figure, you can see the group of property crimes (auto theft, larceny, burglary) at the left separated from the violent crimes against persons at the right.

3.15.2 Partial correlations

Among the more important statistical applications of network graph theory is the idea that you can also use them to study the the *partial* (conditional) associations among variables with the contributions of all other variables removed in what are called Graphical Gaussian Models (GGMs) (Højsgaard et al., 2012; Lauritzen, 1996). In a network diagram of these partial associations,

- The edges between nodes represent the *partial correlations* between those variables.
- The absence of an edge between two nodes indicates their variables are *conditionally independent*, given the other variables.

So, whereas a network diagram of correlations shows *marginal associations* ignoring other variables, one of partial correlations allows you to visualize the *direct* relationship between each pair of variables, removing the indirect effects that might be mediated through all other variables.

For a set of variables $X = \{x_1, x_2, \dots, x_p\}$, the partial correlation between x_i and x_j , controlling for all other variables $Z = X \setminus \{x_i, x_j\} = x_{\text{others}}$ is equivalent to the correlation between the *residuals* of the linear regressions of x_i on all other \mathbf{Z} and x_j on \mathbf{Z} . (The notation $X \setminus \{x_i, x_j\}$ is read as “ X without the set $\{x_i, x_j\}$ ”).

Mathematically, let \hat{x}_i and \hat{x}_j be the predicted values from the linear regressions of x_i on \mathbf{Z} and of x_j on \mathbf{Z} , respectively. The partial correlation p_{ij} between x_i and x_j controlling for \mathbf{Z} is given by:

$$p_{x_i, x_j | \mathbf{Z}} = r(x_i, x_j | \text{others}) = \text{cor}[(x_i - \hat{x}_i), (x_j - \hat{x}_j)] \quad (3.3)$$

But, rather than running all these linear regressions, they can all be computed from the inverse of the correlation matrix (Whittaker, 1990, Ch. 5), a relation first noted by Dempster (1972). Let \mathbf{R} be the correlation matrix of the variables. Then, the matrix \mathbf{P} of partial correlations can be obtained from the negative inverse, $-\mathbf{R}^{-1}$, standardized to a correlation matrix by dividing by the square root of product of its diagonal elements,

$$P_{ij} = -\frac{R_{ij}^{-1}}{\sqrt{(R_{ii}^{-1} \cdot R_{jj}^{-1})}} .$$

The practical implications of this are:

- If a partial correlation is close to zero, it suggests the relationship between two variables is primarily mediated through other variables.
- Non-zero partial correlations indicate a direct relationship that persists after controlling for other variables.

Figure 3.51 shows the partial correlation network for the crime data, using the `qgraph()` argument `graph = "pcor"`. To provide a more interpretable result, the argument `layout = "spring"` positions the nodes using a force-embedded algorithm where edges act like springs, pulling connected nodes together and unconnected nodes repel each other, pushing them apart.

```
qgraph(crime.cor,
       title = "Crime data:\npartial correlations", title.cex = 1.5,
       graph = "pcor",
       layout = "spring", repulsion = 1.2,
       minimum = "sig", sampleSize = nrow(crime), alpha = 0.05,
       color = grey(.9), vsize = 14,
```

```
labels = rownames(crime.cor),
edge.labels = TRUE, edge.label.cex = 1.7,
posCol = "blue")
```

Crime data:
partial correlations

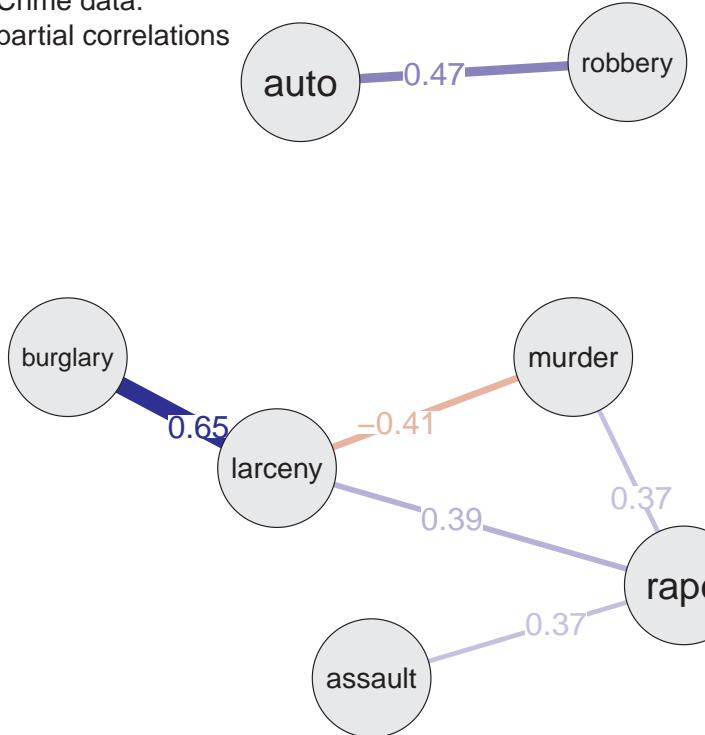


Figure 3.51: Network diagram of partial correlations among the crime variables, controlling for all others. Variable nodes have been positioned by a “spring” layout method ...

Figure 3.51 shows that, once all other crime variables are controlled for each pair, there remain only a few partial correlations at the $\alpha = 0.05$ level. Of these, only the largest three in absolute value are significant at $\alpha = 0.01$.

Thus, once all other variables are taken into account, what remains is mainly a strong positive association between burglary and larceny and a moderate one between auto theft and robbery. There also remains a moderate negative correlation between murder and larceny. The spring layout makes it clear that, with suppression of weak edges, auto theft and robbery form a cluster separated from the other variables.

3.15.3 Visualizing partial correlations

Just as you can visualize *marginal* association between variables in a scatterplot, you can also visualize *conditional* association. A **partial variables plot** is simply a scatterplot of the partial residuals $e_i = (x_i - \hat{x}_i)$ from a regression of x_i on the other variables Z against those $e_j = (x_j - \hat{x}_j)$ for another variable x_j .

In this, you can use all the bells and whistles of standard scatterplots (regression lines, smooths, data ellipses, ...) to listen more attentively to the story partial association has to tell. The function `pvPlot()` calculates the partial residuals and then calls `car::dataEllipse()` for display. The five most “unusual” observations by Mahalanobis D^2 are identified with their abbreviated state labels. Figure 3.52 shows these plots for the variable pairs with the two largest partial correlations.

```

source("R/pvPlot.R")
# select numeric, make `st` into rownames
crime.num <- crime |>
  tibble::column_to_rownames("st") |>
  dplyr::select(where(is.numeric))

pvPlot(crime.num, vars = c("burglary", "larceny"),
       id = list(n=5),
       cex.lab = 1.5)
pvPlot(crime.num, vars = c("robbery", "auto"),
       id = list(n=5),
       cex.lab = 1.5)

```

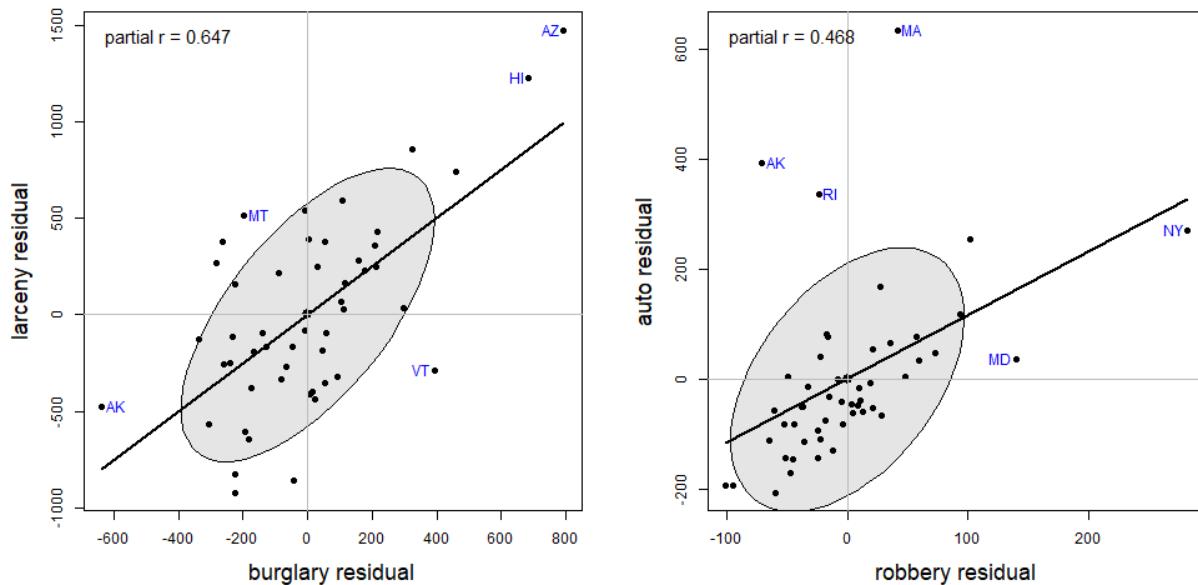


Figure 3.52: Partial variables plots for burglary and larceny (left) and for robbery and auto theft (right) in the network diagram for partial correlations of the crime variables.

In the pvPlot for burglary and larceny, you can see that the high partial correlation is largely driven by the extreme points at the left and right sides. Once all other variables are taken into account, Arizona (AZ) and Hawaii (HI) have larger incidence of both crimes, while Arkansas (AK) are smaller on both.

In the pvPlot for robbery and auto theft, New York stands out as an influential, high-leverage point (see Section 6.6); Massachusetts (MA) is noteworthy because auto theft in that state is considerably higher than what would be predicted from all other variables.

3.16 What have we learned?

This chapter provides a comprehensive toolkit for visualizing multivariate relationships, transforming the humble scatterplot into a powerful engine for data exploration and discovery.

- **Enhance your plots:** The basic scatterplot becomes far more informative when we add **smoothers**

(regression lines, loess curves), **stratifiers** (grouping by color, shape, or panels), and **data ellipses** that capture correlation, variance, and regression relationships in a single elegant geometric form. These annotations help turn static dots into informative stories about relationships in your data.

- **Data ellipses are a visualization Multi-Tool:** These simple geometric summaries encode a remarkable amount of information—means, standard deviations, correlation, regression slopes, and confidence regions—all in one visual package. When your data are roughly bivariate normal, the ellipse becomes a sufficient summary, telling you everything you need to know about the relationship between two variables. Relaxing the normality assumption, robust methods like the bagplot can serve nearly as well.
- **Simpson’s Paradox lurks everywhere:** The grouping variables you ignore can completely reverse the relationships you see, once you properly account for the grouping variables. Marginal correlations can be negative while all within-group correlations—conditional on the grouping variable—are positive (or vice versa). This fundamental lesson reminds us that **context matters**—always consider what variables you might be overlooking.
- **Scatterplot matrices scale gracefully:** When you have multiple variables, pairs plots let you see all pairwise relationships simultaneously. Add **visual thinning** (removing points, keeping only smoothers and ellipses) and **effect ordering** (arranging variables by similarity) to handle even larger numbers of variables while maintaining interpretability.
- **Generalized pairs plots bridge data types:** Modern extensions handle mixtures of continuous and categorical variables elegantly, using appropriate plot types for each combination—scatterplots for continuous pairs, box plots for continuous-categorical combinations, and mosaic plots for categorical pairs. This unified framework means no variable gets left behind.
- **Parallel coordinates reveal high-dimensional patterns:** When scatterplot matrices reach their limits, parallel coordinate plots let you visualize dozens of variables simultaneously. Each observation becomes a connected line across parallel axes, revealing clusters, outliers, and multivariate patterns that would be invisible in traditional 2D projections.
- **Tours provide dynamic exploration:** Animated statistical tours take you on guided journeys through high-dimensional data space, smoothly rotating through different 2D projections. Whether taking a random grand tour or a guided tour optimized for specific features (clusters, outliers, group separation), these methods reveal structures hidden in static displays.
- **Network diagrams conquer “big p” data:** When correlations become too numerous to display traditionally, network graphs show variables as nodes connected by edges representing associations. Partial correlation networks reveal **direct** relationships between variables with all other influences removed—the difference between marginal and conditional independence becomes visually apparent.
- **Visual thinning is a superpower:** As datasets grow in size and complexity, strategic removal of visual elements (points, axes, labels) while retaining essential summaries (trends, ellipses, connections) lets you maintain clarity and insight. Less can indeed be more when every remaining element carries a greater share of graphic information.

Having understood these lessons of multivariate thinking and visualization in *data space*, you are now prepared to take the next step in Chapter 4 to consider how to enhance your understanding of multivariate data using **multivariate juicers** which project high-dimensional data into a lower-dimensional space in which important effects can be more easily seen.

3.17 Exercises

Exercise 3.1. Using the `Salaries` dataset, create one or more plots to compare different smoothing methods for the relationship between `yrs.since.phd` and `salary` shown in Figure 3.5. Include linear regression, quadratic polynomial, and loess smoothers,

```
library(ggplot2)
data(Salaries, package = "carData")
# Your code here
```

Exercise 3.2. One alternative to a loess smooth, which allows a `span` argument to control the degree of smoothing is a **natural spline**, that can be used in `geom_smooth()` using the argument `formula = y ~ splines::ns(x, df=)`, where `df` is the equivalent number of degrees of freedom for the spline smoother. Re-do Exercise 3.1, but trying out this smoothing method for several values of `df`.

Package summary

For development, keep track of the packages used in each chapter.

19 packages used here: `broom`, `car`, `carData`, `corrgram`, `corrplot`, `dplyr`, `GGally`, `ggdensity`, `gggda`, `ggpctp`, `ggplot2`, `grid`, `heplots`, `knitr`, `patchwork`, `qgraph`, `tidyverse`, `tourr`, `vcd`

4

Dimension Reduction

4.1 Flatland and Spaceland

It is high time that I should pass from these brief and discursive notes about Flatland to the central event of this book, my initiation into the mysteries of Space. THAT is my subject; all that has gone before is merely preface — Edwin Abbott, *Flatland*, p. 57.

There was a cloud in the sky above *Flatland* one day. But it was a huge, multidimensional cloud of sparkly points that might contain some important message, perhaps like the hidden EUREKA (Figure 3), or perhaps forecasting the upcoming harvest, if only Flatlanders could appreciate it.

A leading citizen, A SQUARE, who had traveled once to Spaceland and therefore had an inkling of its majesty beyond the simple world of his life in the plane looked at that cloud and had a brilliant thought, an OMG moment:

“Oh, can I, in my imagination, rotate that cloud and squeeze its juice so that it rains down on Flatland with greatest joy?”

As it happened, our Square friend, although he could never really *see* in three dimensions, he could now at least *think* of a world described by **height** as well as breadth and width, and think of the **shadow** cast by a cloud as something mutable, changing size and shape depending on its’ orientation over Flatland.

And what a world it was, inhabited by Pyramids, Cubes and wondrous creatures called Polyhedrons with many *Corners*, *Faces* and *Edges*. Not only that, but all those Polyhedra were forced in Spaceland to obey a magic formula: $C + F - E = 2$.¹ How cool was that!

Indeed, there were even exalted Spheres, having so many faces that its surface became as smooth as a baby’s bottom with no need for pointed corners or edges, just as Circles were the smoothest occupants of his world with far too many sides to count. It was his dream of a Sphere passing through Flatland (Figure 1) that first awakened him to a third dimension.

He also marveled at Ellipsoids, as smooth as Spheres, but in Spaceland having three natural axes of different extent and capable of being appearing fatter or slimmer when rotated from different views. An Ellipsoid had magical properties: it could appear as so thin in one or more dimensions that it became a simple 2D ellipse, or a 1D line, or even a 0D point (Friendly et al., 2013).

All of these now arose in Square’s richer 3D imagination. And, all of this came from just one more dimension than his life in Flatland.

4.1.1 Multivariate juicers

Up to now, we have also been living in Flatland. We have been trying to understand data in **data space** of possibly many dimensions, but confined to the 2D plane of a graph window. Scatterplot matrices and

¹This is Euler’s (1758) formula, which states that any convex polyhedron must obey the formula $V + F - E = 2$ where V is the number of vertexes (corners), F is the number of faces and E is the number of edges. For example, a tetrahedron or pyramid has $(V, F, E) = (4, 4, 6)$ and a cube has $(V, F, E) = (8, 6, 12)$. Stated in words, for all solid bodies confined by planes, the sum of the number of vertexes and the number of faces is two less than the number of edges.

parallel coordinate plots provided some relief. The former did so by **projecting** the data into sets of 2D views in the coordinates of data space; the latter did so by providing multiple axes in a 2D space along which we could trace the paths of individual observations.

This chapter is about seeing data in a different projection, a low-dimensional (usually 2D) space that squeezes out the most juice from multidimensional data for a particular purpose (Figure 4.1), where what we want to understand can be more easily seen.

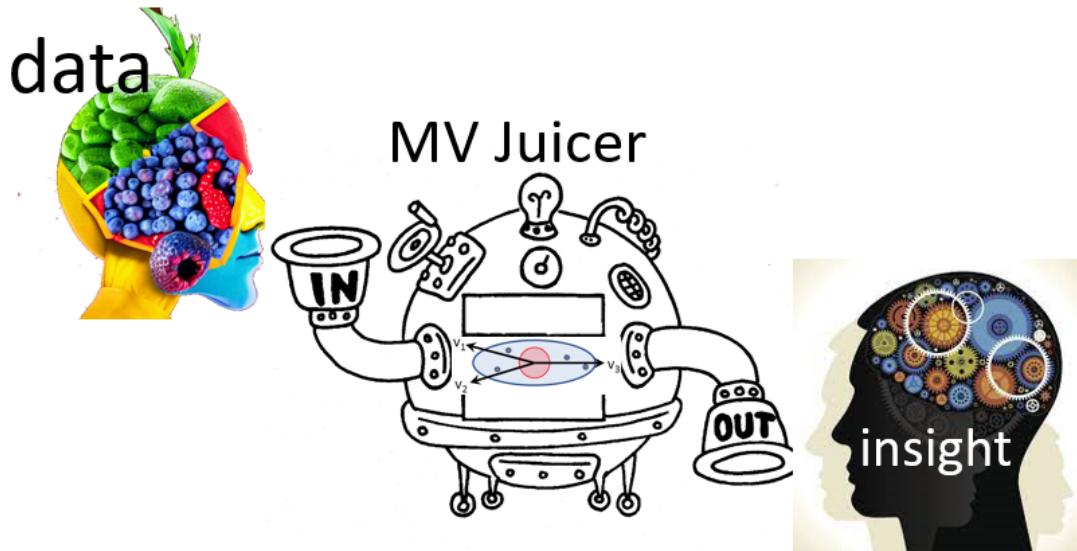


Figure 4.1: A multivariate juicer takes data from possibly high-dimensional data space and transforms it to a lower-dimensional space in which important effects can be more easily seen.

Here, I concentrate on **principal components analysis** (PCA), whose goal reflects A Square's desire to see that sparkly cloud of points in nD space in the plane showing the greatest variation (squeezing the most juice) among all other possible views. This appealed to his sense of geometry, but left him wondering how the variables in that high-D cloud were related to the dimensions he could see in a best-fitting plane.

The idea of a **biplot**, showing the data points in the plane, together with thick pointed arrows—variable vectors—in one view is the other topic explained in this chapter (Section 4.3). The biplot is the simplest example of a multivariate juicer. The essential idea is to project the cloud of data points in n dimensions into the 2D space of principal components and simultaneously show how the original variables relate to this space. For exploratory analysis to get an initial, incisive view of a multivariate dataset, a biplot is often my first choice.

i Looking ahead

I'm using the term *multivariate juicer* here to refer the wider class of **dimension reduction** techniques, used for various purposes in data analysis and visualization. PCA is the simplest example and illustrates the general ideas.

The key point is that these methods are designed to transform the data into a low-dimensional space for a particular goal or purpose. In PCA, the goal is to extract the greatest amount of total variability in the data. In the context of univariate multiple regression, the goal is often to reduce the number of predictors necessary to account for an outcome variable, called *feature extraction* in the machine learning literature.

When the goal is to best distinguish among groups **discriminant analysis** finds uncorrelated weighted

sums of predictors on which the means of groups are most widely separated in a reduced space of hopefully fewer dimensions.

The methods I cover in this book are all linear methods, but there is also a wide variety of non-linear dimension reduction techniques.

Packages

In this chapter I use the following packages. Load them now:

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(patchwork)
library(ggbiplot)
library(FactoMineR)
library(factoextra)
library(car)
library(ggpubr)
library(matlib)
```

4.2 Principal components analysis (PCA)

When Francis Galton (1886) first discovered the idea of regression toward the mean and presented his famous diagram (Figure 3.10), he had little thought that he had provided a window to a higher-dimensional world, beyond what even A Square could imagine. His friend, Karl Pearson (1896) took that idea and developed it into a theory of regression and a measure of correlation that would bear his name, Pearson's r .

But then Pearson (1901) had a further inspiration, akin to that of A Square. If he also had a cloud of sparkly points in $2, 3, 4, \dots, p$ dimensions, could he find a point ($0D$), or line ($1D$), or plane ($2D$), or even a hyperplane (nD) that best summarized — squeezed out the most juice—from multivariate data? This was the first truly multivariate problem in the history of statistics (Friendly & Wainer, 2021, p. 186).

The best $0D$ point was easy—it was simply the centroid, the means of each of the variables in the data, $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_p)$, because that was “closest” to the data in the sense of minimizing the sum of squared differences, $\sum_i \sum_j (x_{ij} - \bar{x}_j)^2$. In higher dimensions, his solution was also an application of the method of least squares, but he argued it geometrically and visually as shown in Figure 4.2.

For a $1D$ summary, the line of best fit to the points P_1, P_2, \dots, P_n is the line that goes through the centroid and made the average squared length of the *perpendicular* segments from those points to a line as small as possible. This was different from the case in linear regression, for fitting y from x , where the average squared length of the *vertical* segments, $\sum_i (y_i - \hat{y}_i)^2$ was minimized by least squares.

He went on to prove the visual insights from simple smoothing of Galton (1886) (shown in Figure 3.10) regarding the regression lines of $y \sim x$ and $x \sim y$. More importantly, he proved that the cloud of points is captured, for the purpose of finding a best line, plane or hyperplane, by the ellipsoid that encloses it, as seen in his diagram, Figure 4.3. The major axis of the 2D ellipse is the line of best fit, along which the data points have the smallest average squared distance from the line. The axis at right angles to that—the minor axis—is labeled “line of worst fit” with the largest average squared distance.

Even more importantly—and this is the basis for PCA—he recognized that the two orthogonal axes of the ellipse gave new coordinates for the data which were uncorrelated, whatever the correlation of x and y .

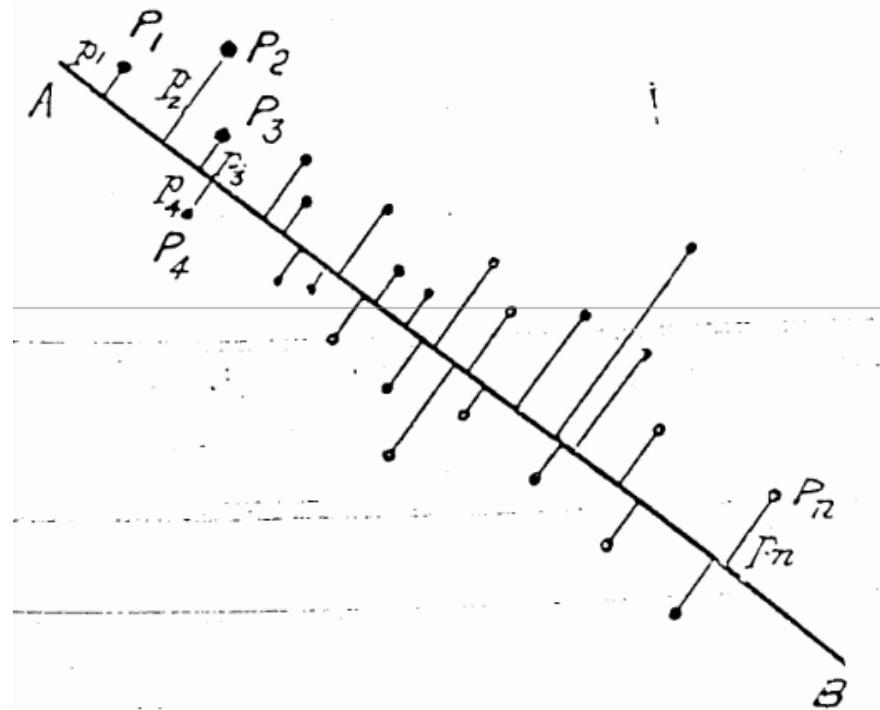


Figure 4.2: Karl Pearson's (1901) geometric, visual argument for finding the line or plane of closest fit to a collection of points, P_1, P_2, P_3, \dots

Physically, the axes of the correlation type-ellipse are the directions of independent and uncorrelated variation. — Pearson (1901), p. 566.

It was but a small step to recognize that for two variables, x and y :

- The line of best fit, the major axis (PC1) had the greatest variance of points projected onto it.
- The line of worst fit, the minor axis (PC2), had the least variance.
- These could be seen as a rotation of the data space of (x, y) to a new space (PC1, PC2) with uncorrelated variables.
- The total variation of the points in data space, $\text{Var}(x) + \text{Var}(y)$, being unchanged by rotation, was equally well expressed as the total variation $\text{Var}(\text{PC1}) + \text{Var}(\text{PC2})$ of the scores on what are now called the principal component axes.

It would have appealed to Pearson (and also to A Square) to see these observations demonstrated in a 3D video. `?@fig-pca-animation` shows a 3D plot of the variables `Sepal.Length`, `Sepal.Width` and `Petal.Length` in Edgar Anderson's `iris` data, with points colored by species and the 95% data ellipsoid. This is rotated smoothly by interpolation until the first two principal axes, PC1 and PC2 are aligned with the horizontal and vertical dimensions. Because this is a rigid rotation of the cloud of points, the total variability is obviously unchanged.

4.2.1 PCA by springs

Before delving into the mathematics of PCA, it is useful to see how Pearson's problem, and fitting by least squares generally, could be solved in a physical realization.

From elementary statistics, you may be familiar with a physical demonstration that the mean, \bar{x} , of a sample is the value for which the sum of deviations, $\sum_i(x_i - \bar{x})$ is zero, so the mean can be visualized as the point of balance on a line where those differences $(x_i - \bar{x})$ are placed. Equally well, there is a physical realization of

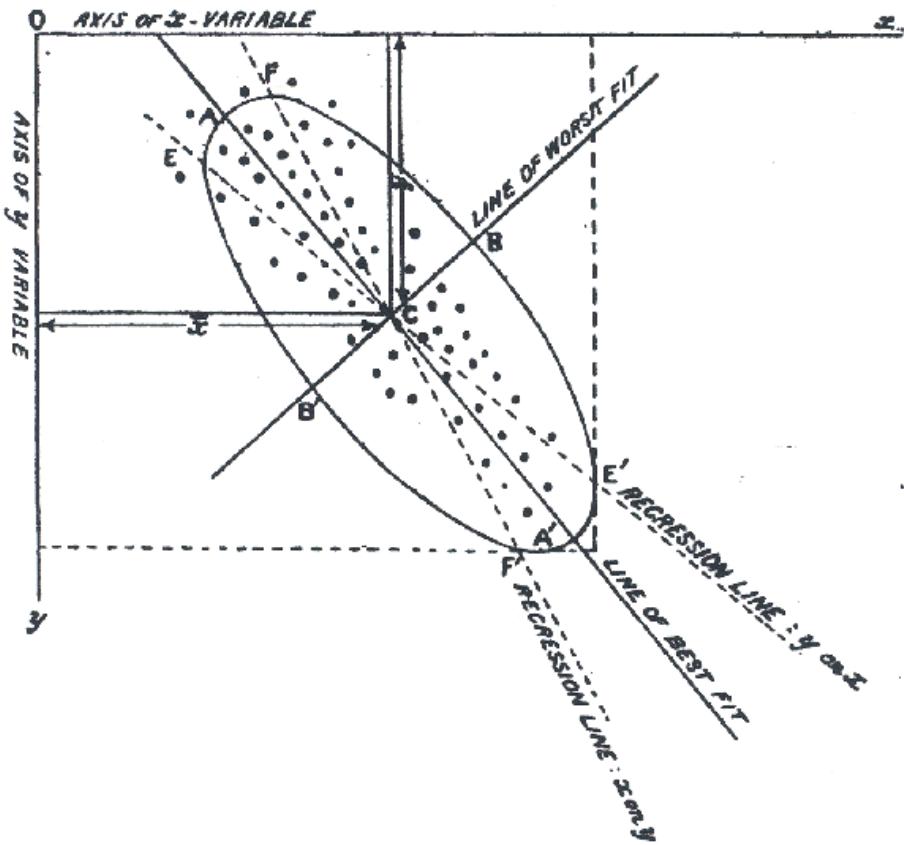


Figure 4.3: Karl Pearson's diagram showing the elliptical geometry of regression and principal components analysis . . . *Source:* Pearson (1901), p. 566.

the mean as the point along an axis where weights connected by springs will minimize the sum of squared differences, because springs with a constant stiffness, k , exert forces proportional to $k(x_i - \bar{x})^2$. That's the reason it is useful as a measure of central tendency: it minimizes the average squared error.

In two dimensions, imagine that we have points, (x_i, y_i) and these are attached by springs of equal stiffness k , to a line anchored at the centroid, (\bar{x}, \bar{y}) as shown in Figure 4.4. If we rotate the line to some initial position and release it, the springs will pull the line clockwise or counterclockwise and the line will bounce around until the forces, proportional to the squares of the lengths of the springs, will eventually balance out at the position (shown by the red fixed line segments at the ends). This is the position that minimizes the the sum of squared lengths of the connecting springs, and also minimizes the kinetic energy in the system.

If you look closely at Figure 4.4 you will see something else: When the line is at its final position of minimum squared length and energy, the positions of the red points on this line are spread out furthest, i.e., have **maximum** variance. Conversely, when the line is at right angles to its final position (shown by the black line at 90°) the projected points have the smallest possible variance.

Figure 4.4: Animation of PCA fitted by springs. The blue data points are connected to their projections on the red line by springs perpendicular to that line. From an initial position, the springs pull that line in proportion to their squared distances, until the line finally settles down to the position where the forces are balanced and the minimum is achieved. *Source:* Amoeba, <https://bit.ly/46tAicu>.

4.2.2 Mathematics and geometry of PCA

As the ideas of principal components developed, there was a happy marriage of Galton's geometrical intuition and Pearson's mathematical analysis. The best men at the wedding were ellipses and higher-dimensional ellipsoids. The bridesmaids were eigenvectors, pointing in as many different directions as space would allow, each sized according to their associated eigenvalues. Attending the wedding were the ghosts of uncles, Leonhard Euler, Jean-Louis Lagrange, Augustin-Louis Cauchy and others who had earlier discovered the mathematical properties of ellipses and quadratic forms in relation to problems in physics.

The key idea in the statistical application was that, for a set of variables $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$, the $p \times p$ covariance matrix \mathbf{S} could be expressed **exactly** as a matrix product involving a matrix \mathbf{V} , whose columns are *eigenvectors* (\mathbf{v}_i) and a diagonal matrix $\mathbf{\Lambda}$, whose diagonal elements (λ_i) are the corresponding *eigenvalues*.

To explain this, it is helpful to use a bit of matrix math:

$$\mathbf{S}_{p \times p} = \mathbf{V}_{p \times p} \quad \mathbf{\Lambda}_{p \times p} \quad \mathbf{V}_{p \times p}^T \quad (4.1)$$

$$= (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_p^T \end{pmatrix} \quad (4.2)$$

$$= \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{v}_2 \mathbf{v}_2^T + \cdots + \lambda_p \mathbf{v}_p \mathbf{v}_p^T \quad (4.3)$$

{#eq-S-eigen}

In this equation,

1. The last line follows because $\mathbf{\Lambda}$ is a diagonal matrix, so \mathbf{S} is expressed as a sum of outer products of each \mathbf{v}_i with itself, times the eigenvalue λ_i .
2. The columns of \mathbf{V} are the eigenvectors of \mathbf{S} . They are orthogonal and of unit length, so $\mathbf{V}^T \mathbf{V} = \mathbf{I}$ and thus they represent orthogonal (uncorrelated) directions in data space.
3. The columns \mathbf{v}_i are the weights applied to the variables to produce the scores on the principal components. For example, the first principal component is the weighted sum:

$$\text{PC}_1 = v_{11} \mathbf{x}_1 + v_{12} \mathbf{x}_2 + \cdots + v_{1p} \mathbf{x}_p .$$

4. The matrix of *all* scores on the principal components can be calculated by multiplying the data matrix \mathbf{X} by the eigenvectors, $\mathbf{PC} = \mathbf{X}\mathbf{V}$.
5. The eigenvalues, $\lambda_1, \lambda_2, \dots, \lambda_p$ are the variances of the components, because $\mathbf{v}_i^T \mathbf{S} \mathbf{v}_i = \lambda_i$.
6. It is usually the case that the variables $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$ are linearly independent, which means that none of these is an exact linear combination of the others. In this case, all eigenvalues λ_i are positive and the covariance matrix \mathbf{S} is said to have **rank** p . (Rank is the number of non-zero eigenvalues.)
7. Here is a key fact: If, as usual, the eigenvalues are arranged in order, so that $\lambda_1 > \lambda_2 > \cdots > \lambda_p$, then the first d components give a d -dimensional approximation to \mathbf{S} , which accounts for $\sum_i^d \lambda_i$ of the $\sum_i^p \lambda_i$ total variance, usually interpreted as the proportion, $(\sum_i^d \lambda_i) / (\sum_i^p \lambda_i)$.

For the case of two variables, \mathbf{x}_1 and \mathbf{x}_2 Figure 4.5 shows the transformation from data space to component space. The eigenvectors, $\mathbf{v}_1, \mathbf{v}_2$ are the major and minor axes of the data ellipse, whose lengths are the square roots $\sqrt{\lambda_1}, \sqrt{\lambda_2}$ of the eigenvalues.

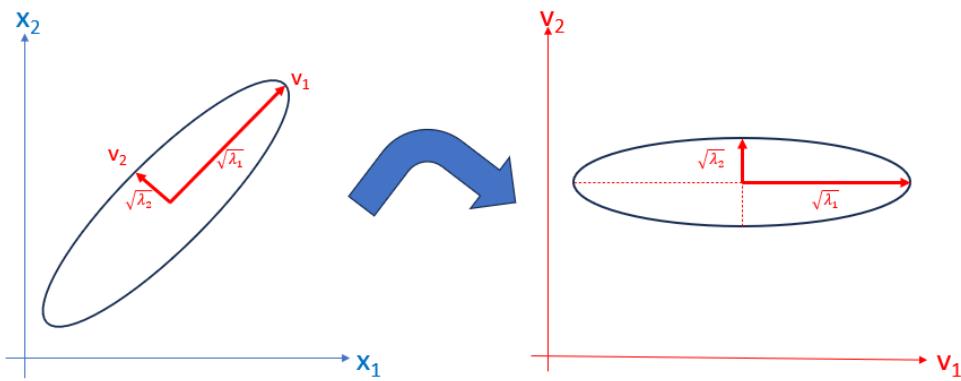


Figure 4.5: Geometry of PCA as a rotation from data space to principal component space, defined by the eigenvectors v_1 and v_2 of a covariance matrix

4.2.2.1 Example: Workers' experience and income

For a small example, consider the relation between years of experience and income in a small (contrived) sample ($n = 10$) of workers in a factory. The dataset `workers` contains these and other variables. In a wider context, we might want to fit a regression model to predict `Income`, but here we focus on a PCA of just these two variables.

```
data(workers, package = "matlib")
head(workers)

#>           Income Experience Skill Gender
#> Abby        20          0    2 Female
#> Betty       35          5    5 Female
#> Charles     40          5    8  Male
#> Doreen      30         10    6 Female
#> Ethan        50         10   10  Male
#> Francie     50         15    7 Female
```

Let's start with a simple scatterplot of `Income` vs. `Experience`, with points labeled by `Name` (and colored by `Gender`). There's a fairly strong correlation ($r = 0.853$). How does a PCA capture this?

```
vars <- c("Experience", "Income")
plot(workers[, vars],
      pch = 16, cex = 1.5,
      cex.lab = 1.5)
text(workers[, vars],
      labels = rownames(workers),
      col = ifelse(workers$Gender == "Female", "red", "blue"),
      pos = 3, xpd = TRUE)
```

To carry out a PCA of these variables, first calculate the vector of means (\bar{x}) and covariance matrix S .

```
mu <- colMeans(workers[, vars]) |> print()
#> Experience      Income
#>      15.5        46.5
S <- cov(workers[, vars]) |> print()
```

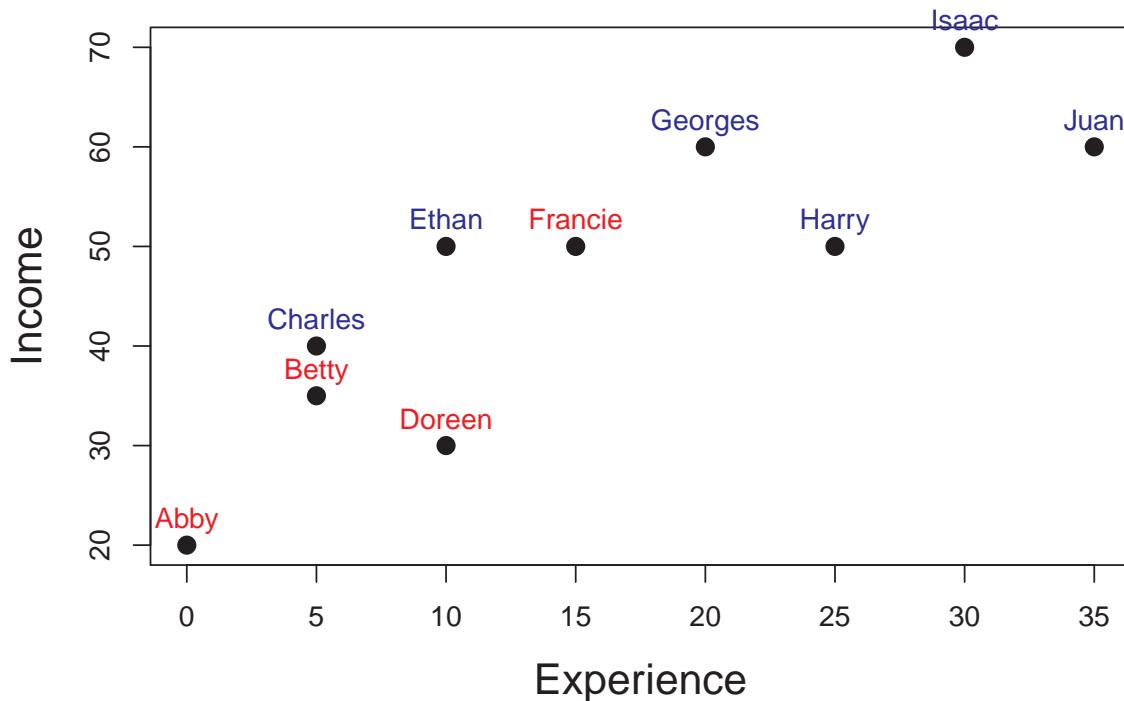


Figure 4.6: Scatterplot of Income vs. Experience for the `workers` data.

```
#>           Experience Income
#> Experience      136    152
#> Income          152    234
```

The eigenvalues and eigenvectors of \mathbf{S} are calculated by `eigen()`. This returns a list with components `values` for the λ_i and `vectors` for \mathbf{V} .

```
S.eig <- eigen(S)
Lambda <- S.eig$values |> print()
#> [1] 344.3 25.1
V <- S.eig$vectors |> print()
#> [,1]  [,2]
#> [1,] 0.589 -0.808
#> [2,] 0.808  0.589
```

From this, you can verify the points above regarding the relations between variances of the variables and the eigenvalues:

```
#total variances of the variables = sum of eigenvalues
sum(diag(S))
#> [1] 369
sum(Lambda)
#> [1] 369

# percent of variance of each PC
100 * Lambda / sum(Lambda)
#> [1] 93.2 6.8
```

Using these, you can express the eigenvalue decomposition of \mathbf{S} in `?@eq-S-eigen` with `latexMatrix()` and `Eqn` from the `matlib` package (Friendly et al., 2024) as:

```
options(digits = 3)
rownames(S) <- colnames(S) <- c("\\small \\text{Exp}",
                                    "\\small \\text{Inc}")
spacer <- "\\phantom{0000000000000000}"
Eqn("\\mathbf{S} &= \\mathbf{V}", spacer,
     "\\mathbf{\\Lambda}", spacer,
     "\\mathbf{V}^\\top", Eqn_newline(),
     latexMatrix(S), "&=",
     latexMatrix(V), " ", diag(Lambda), " ", latexMatrix(V, transpose=TRUE),
     align = TRUE)
```

$$\mathbf{S} = \mathbf{V} \quad \mathbf{\Lambda} \quad \mathbf{V}^\top$$

$$\begin{matrix} Exp & Inc \\ Exp & \begin{pmatrix} 136 & 152 \\ 152 & 234 \end{pmatrix} \\ Inc & \end{matrix} = \begin{pmatrix} 0.589 & -0.808 \\ 0.808 & 0.589 \end{pmatrix} \begin{pmatrix} 344.3 & 0.0 \\ 0.0 & 25.1 \end{pmatrix} \begin{pmatrix} 0.589 & -0.808 \\ 0.808 & 0.589 \end{pmatrix}^\top$$

The “scores” on the principal components can be calculated (point (5) above) as $\mathbf{PC} = \mathbf{X}\mathbf{V}$:

```
PC <- as.matrix(workers[, vars]) %*% V
colnames(PC) <- paste0("PC", 1:2)
head(PC)
#>      PC1   PC2
#> Abby  16.2 11.78
#> Betty 31.2 16.57
#> Charles 35.3 19.52
#> Doreen 30.1  9.59
#> Ethan  46.3 21.37
#> Francie 49.2 17.32
```

Then, you can visualize the geometry of PCA as in Figure 4.5 (left) by plotting the data ellipse for the points, along with the PCA axes (`heplots::ellipse.axes()`). Figure 4.7 also shows the bounding box of the data ellipse, which are parallel to the PC axes and scaled to have the same “radius” as the data ellipse.

```
# calculate conjugate axes for PCA factorization
pca.fac <- function(x) {
  xx <- svd(x)
  ret <- t(xx$v) * sqrt(pmax( xx$d, 0))
  ret
}

dataEllipse(Income ~ Experience, data=workers,
            pch = 16, cex = 1.5,
            center.pch = "+", center.cex = 2,
            cex.lab = 1.5,
            levels = 0.68,
            grid = FALSE,
            xlim = c(-10, 40),
            ylim = c(10, 80),
```

```

asp = 1)
abline(h = mu[2], v = mu[1],
lty = 2, col = "grey")

# axes of the ellipse = PC1, PC2
radius <- sqrt(2 * qf(0.68, 2, nrow(workers)-1 ))
heplots::ellipse.axes(S, mu,
  radius = radius,
  labels = TRUE,
  col = "red", lwd = 2,
  cex = 1.8)

# bounding box of the ellipse
lines(spida2::ellplus(mu, S, radius = radius,
  box = TRUE, fac = pca.fac),
  col = "darkgreen",
  lwd = 2, lty="longdash")

```

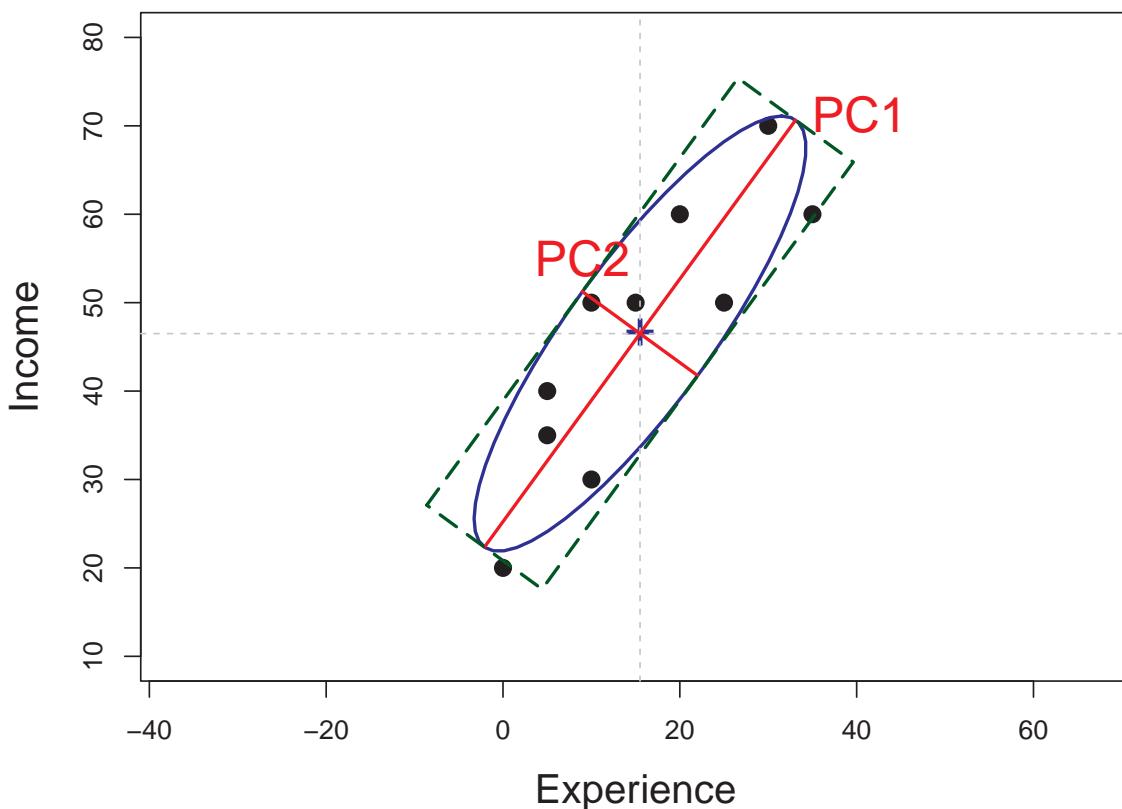


Figure 4.7: Geometry of the PCA for the `workers` data, showing the data ellipse, the eigenvectors of \mathbf{S} , whose half-lengths are the square roots $\sqrt{\lambda_i}$ of the eigenvalues, and the bounding box of the ellipse.

Finally, to preview the methods of the next section, the results calculated “by hand” above can be obtained using `prcomp()`. The values labeled “Standard deviations” are the square roots $\sqrt{\lambda_i}$ of the two eigenvalues. The eigenvectors are labeled “Rotation” because \mathbf{V} is the matrix that rotates the data matrix to produce the component scores.

```
workers.pca <- prcomp(workers[, vars]) |> print()
#> Standard deviations (1, ..., p=2):
#> [1] 18.56 5.01
#>
#> Rotation (n x k) = (2 x 2):
#>          PC1    PC2
#> Experience 0.589  0.808
#> Income     0.808 -0.589
```

4.2.3 Finding principal components

In R, PCA is most easily carried out using `stats:::prcomp()` or `stats:::princomp()` or similar functions in other packages such as `FactoMineR::PCA()`. The **FactoMineR** package (Husson et al., 2017, 2025) has extensive capabilities for exploratory analysis of multivariate data (PCA, correspondence analysis, cluster analysis).

A particular strength of FactoMineR for PCA is that it allows the inclusion of *supplementary variables* (which can be categorical or quantitative) and *supplementary points* for individuals. These are not used in the analysis, but are projected into the plots to facilitate interpretation. For example, in the analysis of the `crime` data described below, it would be useful to have measures of other characteristics of the U.S. states, such as poverty and average level of education (Section 4.3.5).

Unfortunately, although all of these functions perform similar calculations, the options for analysis and the details of the result they return differ.

The important options for analysis include:

- whether or not the data variables are **centered**, to a mean of $\bar{x}_j = 0$
- whether or not the data variables are **scaled**, to a variance of $\text{Var}(x_j) = 1$.

It nearly always makes sense to center the variables. The choice of scaling determines whether the correlation matrix is analyzed, so that each variable contributes *equally* to the total variance that is to be accounted for versus analysis of the covariance matrix, where each variable contributes its *own variance* to the total. Analysis of the covariance matrix makes little sense when the variables are measured on different scales², unless you want to interpret total variance on the scales of the different variables.

You don't need to scale your data in advance, but be aware of the options: `prcomp()` has default options `center = TRUE`, `scale. = FALSE`³ so in most cases you should specify `scale. = TRUE`. I mostly use this. The older `princomp()` has only the option `cor = FALSE` which centers the data and uses the covariance matrix, so in most cases the default is OK.

To illustrate, the analysis of the `workers` data presented above used `scale. = FALSE` by default, so the eigenvalues reflected the variances of Experience and Income. The analogous result, using standardized variables (*z-scores*) can be computed in any of the forms shown below, using either `scale. = FALSE` or standardizing first using `scale()`:

```
prcomp(workers[, vars], scale. = TRUE)
#> Standard deviations (1, ..., p=2):
#> [1] 1.361 0.383
#>
#> Rotation (n x k) = (2 x 2):
```

²For example, if two variables in the analysis are height and weight, changing the unit of height from inches to centimeters would multiply its variance by 2.54^2 ; changing weight from pounds to kilograms would divide its variance by 2.2^2 .

³The unfortunate default `scale. = FALSE` was for consistency with S, the precursor to R but in general scaling is usually advisable.

```
#>          PC1    PC2
#> Experience 0.707  0.707
#> Income      0.707 -0.707

# same as (output suppressed):
workers[, vars] |> prcomp(scale. = TRUE) |> invisible()
workers[, vars] |> scale() |> prcomp() |> invisible()
```

In this form, each of Experience and Income have variance = 1, and the "Standard deviations" reported are the square roots ($\sqrt{\lambda_i}$) of the eigenvalues λ_i of the correlation matrix \mathbf{R} . The eigenvalues of a correlation matrix always sum to p , the number of variables. This fact prompted the rough *rule of thumb* to extract principal components whose eigenvalues exceed 1.0, which is their average value, $\bar{\lambda} = (\sum^p \lambda_i)/p = p/p$.

```
prcomp(workers[, vars], scale. = TRUE)$sdev
#> [1] 1.361 0.383

# eigen values of correlation matrix
R <- cor(workers[, vars])
R.eig <- eigen(R)
Lambda <- R.eig$values |> print()
#> [1] 1.853 0.147
sum(Lambda)
#> [1] 2
```

Example: Crime data

The dataset `crime`, analysed in Section 3.11, showed all positive correlations among the rates of various crimes in the corrgram, Figure 3.34. What can we see from a PCA? Is it possible that a few dimensions can account for most of the juice in this data?

In this example, you can easily find the PCA solution using `prcomp()` in a single line in base-R. You need to specify the numeric variables to analyze by their columns in the data frame. The most important option here is `scale. = TRUE`.

```
data(crime, package = "ggbioplot")
crime.pca <- prcomp(crime[, 2:8], scale. = TRUE)
```

The tidy equivalent is more verbose, but also more expressive about what is being done. It selects the variables to analyze by a function, `is.numeric()` applied to each of the columns and feeds the result to `prcomp()`.

```
crime.pca <-
  crime |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)
```

As is typical with models in R, the result, `crime.pca` of `prcomp()` is an object of class "prcomp", a list of components, and there are a variety of methods for "prcomp" objects. Among the simplest is `summary()`, which gives the contributions of each component to the total variance in the dataset.

```
summary(crime.pca) |> print(digits=2)
#> Importance of components:
```

```
#> PC1   PC2   PC3   PC4   PC5   PC6   PC7
#> Standard deviation 2.03 1.11 0.85 0.563 0.508 0.471 0.352
#> Proportion of Variance 0.59 0.18 0.10 0.045 0.037 0.032 0.018
#> Cumulative Proportion 0.59 0.76 0.87 0.914 0.951 0.982 1.000
```

The object, `crime.pca` returned by `prcomp()` is a list of the following the following elements:

```
names(crime.pca)
#> [1] "sdev"      "rotation"   "center"    "scale"     "x"
```

Of these, for n observations and p variables,

- `sdev` is the length p vector of the standard deviations of the principal components (i.e., the square roots $\sqrt{\lambda_i}$ of the eigenvalues of the covariance/correlation matrix). When the variables are standardized, the sum of squares of the eigenvalues is equal to p .
- `rotation` is the $p \times p$ matrix of weights or **loadings** of the variables on the components; the columns are the eigenvectors of the covariance or correlation matrix of the data;
- `x` is the $n \times p$ matrix of **scores** for the observations on the components, the result of multiplying (rotating) the data matrix by the loadings. These are uncorrelated, so `cov(x)` is a $p \times p$ diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i = sdev^2$.
- `center` gives the means of the variables when the option `center.` = `TRUE` (the default)

4.2.4 Visualizing variance proportions: screeplots

For a high-D dataset, such as the crime data in seven dimensions, a natural question is how much of the variation in the data can be captured in 1D, 2D, 3D, ... summaries and views. This is answered by considering the proportions of variance accounted by each of the dimensions, or their cumulative values. The components returned by various PCA methods have (confusingly) different names, so `broom::tidy()` provides methods to unify extraction of these values.

```
(crime.eig <- crime.pca |>
  broom::tidy(matrix = "eigenvalues"))
#> # A tibble: 7 x 4
#>   PC std.dev percent cumulative
#>   <dbl>    <dbl>   <dbl>      <dbl>
#> 1     1     2.03   0.588     0.588
#> 2     2     1.11   0.177     0.765
#> 3     3     0.852  0.104     0.868
#> 4     4     0.563  0.0452    0.914
#> 5     5     0.508  0.0368    0.951
#> 6     6     0.471  0.0317    0.982
#> 7     7     0.352  0.0177    1
```

Then, a simple visualization is a plot of the proportion of variance for each component (or cumulative proportion) against the component number, usually called a **screeplot**. The idea, introduced by Cattell (1966), is that after the largest, dominant components, the remainder should resemble the rubble, or scree formed by rocks falling from a cliff.

From this plot, imagine drawing a straight line through the plotted eigenvalues, starting with the largest one. The typical rough guidance is that the last point to fall on this line represents the last component to extract, the idea being that beyond this, the amount of additional variance explained is non-meaningful. Another rule of thumb is to choose the number of components to extract a desired proportion of total variance, usually in the range of 80 - 90%.

`stats::plot(crime.pca)` would give a bar plot of the variances of the components, however `ggbioplot::ggscreeplot()` gives nicer and more flexible displays as shown in Figure 4.8.

```
p1 <- ggscreeplot(crime.pca) +
  stat_smooth(data = crime.eig |> filter(PC>=4),
              aes(x=PC, y=percent), method = "lm",
              se = FALSE,
              fullrange = TRUE) +
  theme_bw(base_size = 14)

p2 <- ggscreeplot(crime.pca, type = "cev") +
  geom_hline(yintercept = c(0.8, 0.9), color = "blue") +
  theme_bw(base_size = 14)

p1 + p2
```

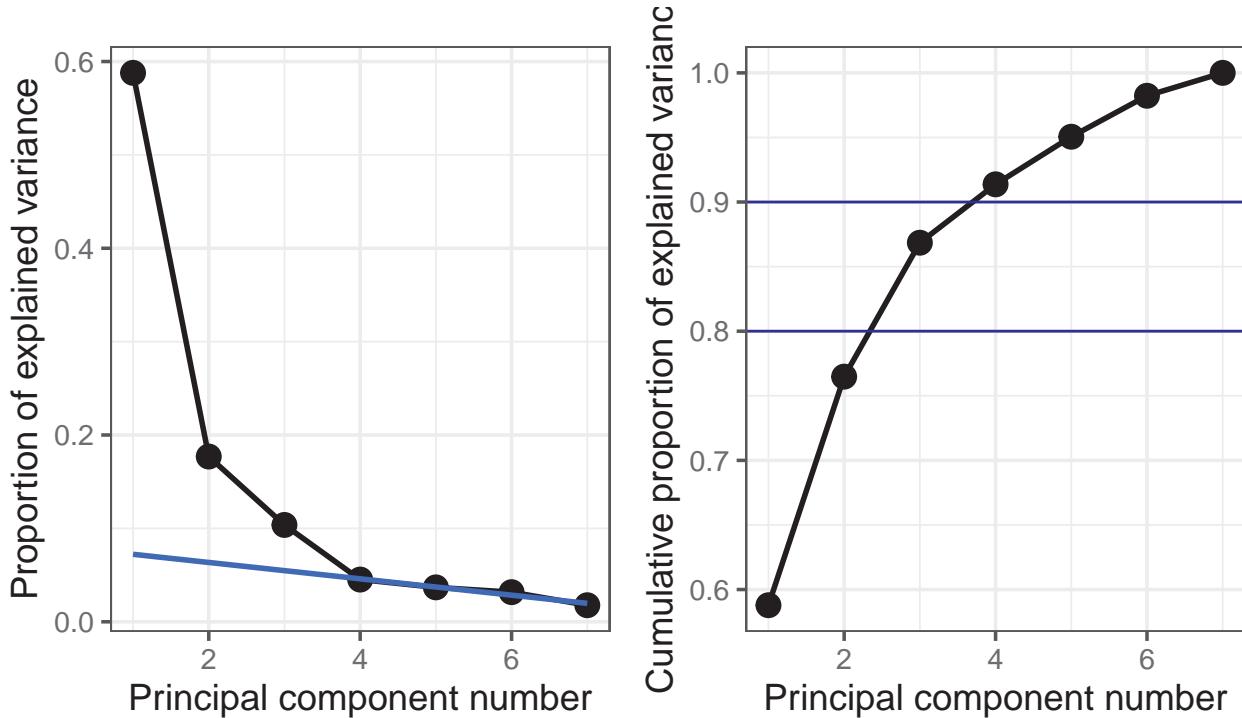


Figure 4.8: Screeplots for the PCA of the crime data. The left panel shows the traditional version, plotting variance proportions against component number, with linear guideline for the scree rule of thumb. The right panel plots cumulative proportions, showing cutoffs of 80%, 90%.

From this we might conclude that four components are necessary to satisfy the scree criterion or to account for 90% of the total variation in these crime statistics. However two components, giving 76.5%, might be enough juice to tell a reasonable story.

4.2.5 Visualizing PCA scores and variable vectors

To see and attempt to understand PCA results, it is useful to plot both the scores for the observations on a few of the largest components and also the loadings or variable vectors that give the weights for the variables in determining the principal components.

In Section 4.3 I discuss the biplot technique that plots both in a single display. However, I do this directly here, using tidy processing to explain what is going on in PCA and in these graphical displays.

Scores

The (uncorrelated) principal component scores can be extracted as `crime.pca$x` or using `purrr::pluck("x")`. As noted above, these are uncorrelated and have variances equal to the eigenvalues of the correlation matrix.

```
scores <- crime.pca |> purrr::pluck("x")
cov(scores) |> zapsmall()
#>      PC1   PC2   PC3   PC4   PC5   PC6   PC7
#> PC1  4.11  0.00  0.00  0.00  0.00  0.00  0.00
#> PC2  0.00  1.24  0.00  0.00  0.00  0.00  0.00
#> PC3  0.00  0.00  0.73  0.00  0.00  0.00  0.00
#> PC4  0.00  0.00  0.00  0.32  0.00  0.00  0.00
#> PC5  0.00  0.00  0.00  0.00  0.26  0.00  0.00
#> PC6  0.00  0.00  0.00  0.00  0.00  0.22  0.00
#> PC7  0.00  0.00  0.00  0.00  0.00  0.00  0.12
```

For plotting, it is more convenient to use `broom::augment()` which extracts the scores (named `.fittedPC*`) and appends these to the variables in the dataset.

```
crime.pca |>
  broom::augment(crime) |> head()
#> # A tibble: 6 x 18
#>   .rownames state    murder  rape robbery assault burglary larceny
#>   <chr>     <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
#> 1 1         Alabama  14.2    25.2   96.8   278.   1136.   1882.
#> 2 2         Alaska   10.8    51.6   96.8   284    1332.   3370.
#> 3 3         Arizona  9.5     34.2   138.   312.   2346.   4467.
#> 4 4         Arkansas 8.8     27.6   83.2   203.   973.    1862.
#> 5 5         California 11.5    49.4   287    358    2139.   3500.
#> 6 6         Colorado  6.3     42     171.   293.   1935.   3903.
#> # i 10 more variables: auto <dbl>, st <chr>, region <fct>,
#> #   .fittedPC1 <dbl>, .fittedPC2 <dbl>, .fittedPC3 <dbl>,
#> #   .fittedPC4 <dbl>, .fittedPC5 <dbl>, .fittedPC6 <dbl>,
#> #   .fittedPC7 <dbl>
```

Then, we can use `ggplot()` to plot any pair of components. To aid interpretation, I label the points by their state abbreviation and color them by `region` of the U.S.. A geometric interpretation of the plot requires an aspect ratio of 1.0 (via `coord_fixed()`) so that a unit distance on the horizontal axis is the same length as a unit distance on the vertical. To demonstrate that the components are uncorrelated, I also added their data ellipse.

```
crime.pca |>
  broom::augment(crime) |> # add original dataset back in
  ggplot(aes(.fittedPC1, .fittedPC2, color = region)) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_point(size = 1.5) +
  geom_text(aes(label = st), nudge_x = 0.2) +
  stat_ellipse(color = "grey") +
  coord_fixed() +
```

```
labs(x = "PC Dimension 1", y = "PC Dimension 2") +
  theme_minimal(base_size = 14) +
  theme(legend.position = "top")
```

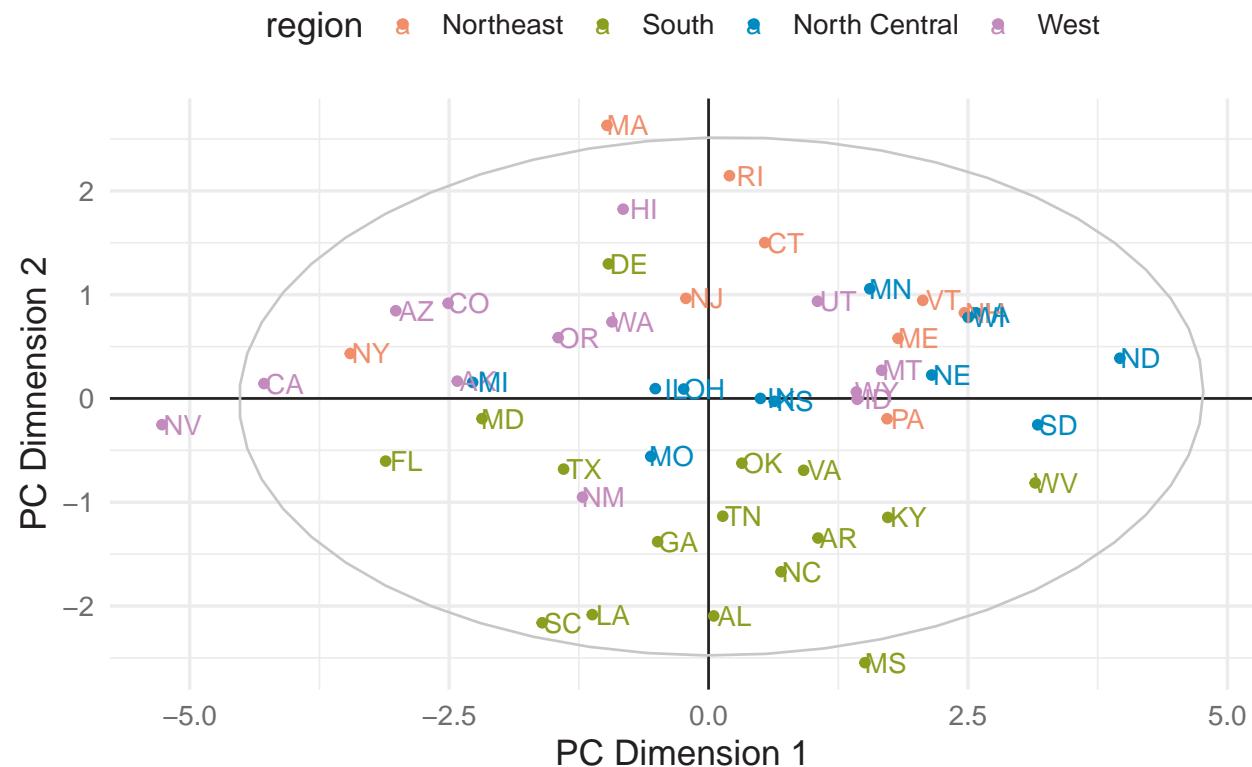


Figure 4.9: Plot of component scores on the first two principal components for the `crime` data. States are colored by `region`.

To interpret such plots, it is useful consider the observations that are a high and low on each of the axes as well as other information, such as region here, and ask how these differ on the crime statistics. The first component, PC1, contrasts Nevada and California with North Dakota, South Dakota and West Virginia. The second component has most of the southern states on the low end and Massachusetts, Rhode Island and Hawaii on the high end. However, interpretation is easier when we also consider how the various crimes contribute to these dimensions.

When, as here, there are more than two components that seem important in the scree plot, we could obviously go further and plot other pairs.

Variable vectors

You can extract the variable loadings using either `crime.pca$rotation` or `purrr::pluck("rotation")`, similar to what I did with the scores.

```
crime.pca |> purrr::pluck("rotation")
#>          PC1      PC2      PC3      PC4      PC5      PC6      PC7
#> murder   -0.300  -0.6292  -0.1782  0.2321  0.5381  0.2591  0.2676
#> rape     -0.432  -0.1694  0.2442 -0.0622  0.1885 -0.7733 -0.2965
#> robbery  -0.397  0.0422 -0.4959  0.5580 -0.5200 -0.1144 -0.0039
#> assault  -0.397 -0.3435  0.0695 -0.6298 -0.5067  0.1724  0.1917
```

```
#> burglary -0.440  0.2033  0.2099  0.0576  0.1010  0.5360 -0.6481
#> larceny   -0.357  0.4023  0.5392  0.2349  0.0301  0.0394  0.6017
#> auto       -0.295  0.5024 -0.5684 -0.4192  0.3698 -0.0573  0.1470
```

But note something important in this output: All of the weights for the first component are negative. In PCA, the directions of the eigenvectors are completely arbitrary, in the sense that the vector $-\mathbf{v}_i$ gives the same linear combination as \mathbf{v}_i , but with its' sign reversed. For interpretation, it is useful (and usually recommended) to reflect the loadings to a positive orientation by multiplying them by -1. In general, you are free to reflect any of the components for ease of interpretation, and not necessarily if all the signs are negative.

To reflect the PCA loadings (multiplying PC1 and PC2 by -1) and get them into a convenient format for plotting with `ggplot()`, it is necessary to do a bit of processing, including making the `row.names()` into an explicit variable for the purpose of labeling.

rownames in R

R software evolved over many years, particularly in conventions for labeling cases in printed output and graphics. In base-R, the convention was that the `row.names()` of a matrix or `data.frame` served as observation labels in all printed output and plots, with a default to use numbers `1:n` if there were no rownames. In `ggplot2` and the `tidyverse` framework, the decision was made that observation labels had to be an **explicit** variable in a “tidy” dataset, so it could be used as a variable in constructs like `geom_text(aes(label = label))` as in this example. This change often requires extra steps in software that uses the rownames convention.

```
vectors <- crime.pca |>
  purrr::pluck("rotation") |>
  as.data.frame() |>
  mutate(PC1 = -1 * PC1, PC2 = -1 * PC2) |>      # reflect axes
  tibble::rownames_to_column(var = "label")

vectors[, 1:3]
#>      label    PC1     PC2
#> 1    murder  0.300  0.6292
#> 2    rape    0.432  0.1694
#> 3    robbery  0.397 -0.0422
#> 4    assault  0.397  0.3435
#> 5    burglary 0.440 -0.2033
#> 6    larceny  0.357 -0.4023
#> 7    auto     0.295 -0.5024
```

Then, I plot these using `geom_segment()`, taking some care to use arrows from the origin with a nice shape and add `geom_text()` labels for the variables positioned slightly to the right. Again, `coord_fixed()` ensures equal scales for the axes, which is important because we want to interpret the angles between the variable vectors and the PCA coordinate axes.

```
arrow_style <- arrow(
  angle = 20, ends = "first", type = "closed",
  length = grid::unit(8, "pt")
)

vectors |>
```

```

ggplot(aes(PC1, PC2)) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_segment(xend = 0, yend = 0,
               linewidth = 1,
               arrow = arrow_style,
               color = "brown") +
  geom_text(aes(label = label),
            size = 5,
            hjust = "outward",
            nudge_x = 0.05,
            color = "brown") +
  ggforce::geom_circle(aes(x0 = 0, y0 = 0, r = 0.5), color = gray(.50)) +
  xlim(-0.5, 0.9) +
  ylim(-0.8, 0.8) +
  coord_fixed() +           # fix aspect ratio to 1:1
  theme_minimal(base_size = 14)

```

The variable vectors (arrows) shown in Figure 4.10 have the following interpretations:

- (1) The lengths of the variable vectors, $\|\mathbf{v}_i\| = \sqrt{\sum_j v_{ij}^2}$ give the relative proportion of variance of each variable accounted for in a two-dimensional display.
- (2) Each vector points in the direction in component space with which that variable is most highly correlated: the value, v_{ij} , of the vector for variable \mathbf{x}_i on component j reflects the correlation of that variable with the j th principal component. Thus,
 - A variable that is perfectly correlated with a component is parallel to it.
 - A variable this is uncorrelated with an component is perpendicular to it.
- (3) The angle between vectors shows the strength and direction of the correlation between those variables: the cosine of the angle θ between two variable vectors, \mathbf{v}_i and \mathbf{v}_j , which is $\cos(\theta) = \mathbf{v}_i' \mathbf{v}_j / \|\mathbf{v}_i\| \cdot \|\mathbf{v}_j\|$ gives the approximation of the correlation r_{ij} between \mathbf{x}_i and \mathbf{x}_j that is shown in this space. This means that:
 - two variable vectors that point in the same direction are highly correlated; $r = 1$ if they are completely aligned.
 - Variable vectors at right angles are approximately uncorrelated, while those pointing in opposite directions are negatively correlated; $r = -1$ if they are at 180° .

To illustrate point (1), the following indicates that almost 70% of the variance of `murder` is represented in the the 2D plot shown in Figure 4.9, but only 40% of the variance of `robbery` is captured. For point (2), the correlation of `murder` with the dimensions is 0.3 for PC1 and 0.63 for PC2. For point (3), the angle between `murder` and `burglary` looks to be about 90° , but the actual correlation is 0.39.

```

vectors |> select(label, PC1, PC2) |>
  mutate(length = sqrt(PC1^2 + PC2^2))
#>   label    PC1    PC2 length
#> 1  murder  0.300  0.6292  0.697
#> 2    rape  0.432  0.1694  0.464
#> 3  robbery  0.397 -0.0422  0.399
#> 4 assault  0.397  0.3435  0.525
#> 5 burglary 0.440 -0.2033  0.485

```

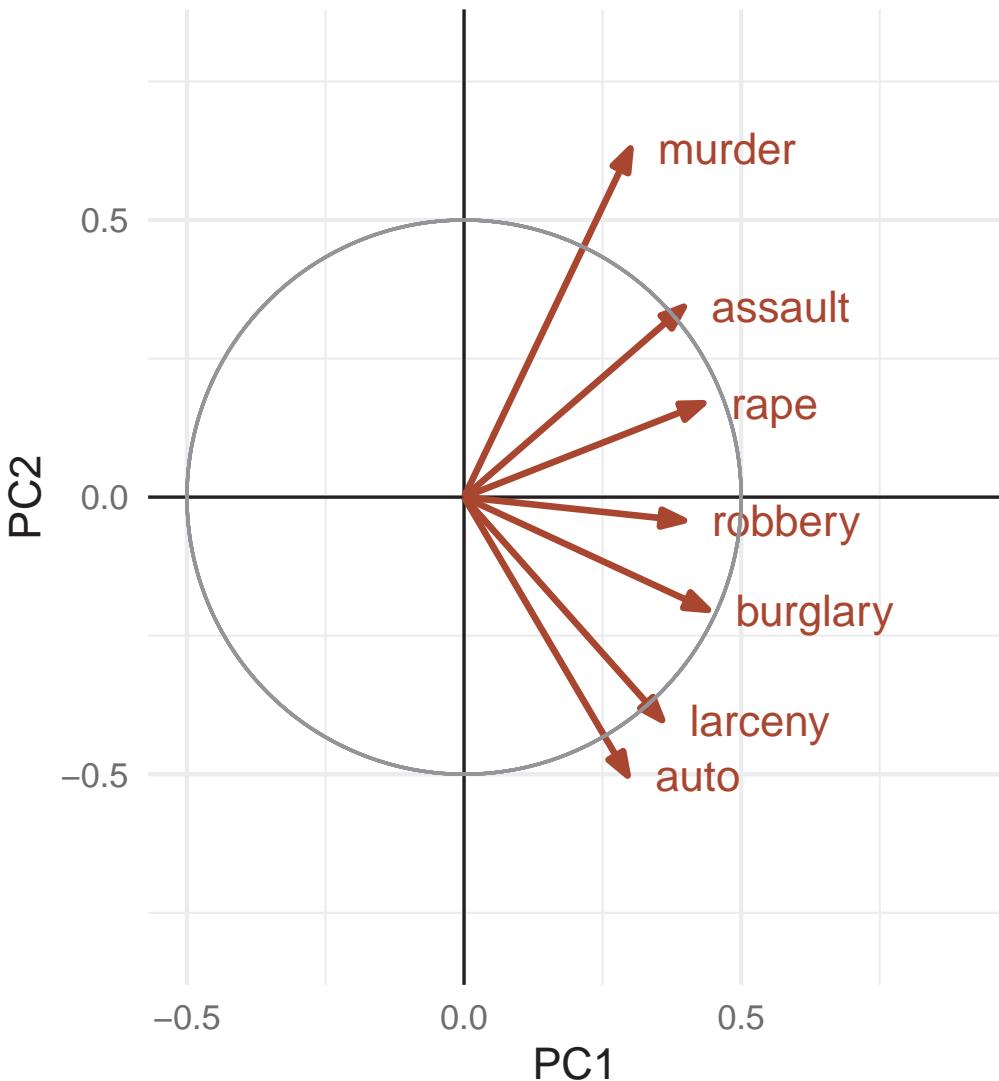


Figure 4.10: Plot of component loadings the first two principal components for the `crime` data. These are interpreted as the contributions of the variables to the components.

```
#> 6  larceny  0.357 -0.4023  0.538
#> 7      auto  0.295 -0.5024  0.583
```

4.3 Biplots

The biplot is a visual multivariate juicer. It is the simple and powerful idea that came from the recognition that you can overlay a plot of observation scores in a principal components analysis with the information of the variable loadings (weights) to give a simultaneous display that is easy to interpret. In this sense, a biplot is generalization of a scatterplot, projecting from data space to PCA space, where the observations are shown by points, as in the plots of component scores in Figure 4.9, but with the variables also shown by vectors (or scaled linear axes aligned with those vectors).

The idea of the biplot was introduced by Ruben Gabriel (1971, 1981) and later expanded in scope by Gower & Hand (1996). The book by Greenacre (2010) gives a practical overview of the many variety of biplots. Gower et al. (2011) *Understanding biplots* provides a full treatment of many topics, including how to calibrate biplot axes, 3D plots, and so forth.

Biplot methodology is far more general than I cover here. Categorical variables can be incorporated in PCA using points that represent the levels of discrete categories. Two-way frequency tables of categorical variables can be analysed using *correspondence analysis*, which is similar to PCA, but designed to account for the maximum amount of the χ^2 statistic for association; *multiple correspondence analysis* extends this to method to multi-way tables (Friendly & Meyer, 2016; Greenacre, 1984).

4.3.1 Constructing a biplot

The biplot is constructed by using the singular value decomposition (SVD) to obtain a low-rank approximation to the data matrix $\mathbf{X}_{n \times p}$ (centered, and optionally scaled to unit variances) whose n rows are the observations and whose p columns are the variables.

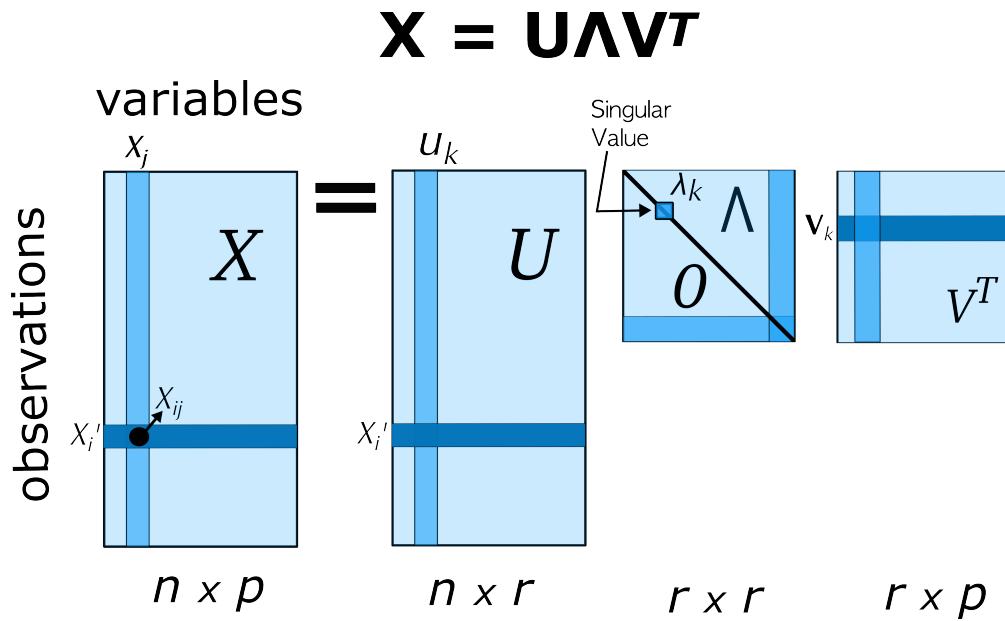


Figure 4.11: The singular value decomposition expresses a data matrix \mathbf{X} as the product of a matrix \mathbf{U} of observation scores, a diagonal matrix Λ of singular values and a matrix \mathbf{V}^T of variable weights.

Using the SVD, the matrix \mathbf{X} , of rank $r \leq p$ can be expressed *exactly* as:

$$\mathbf{X} = \mathbf{U} \Lambda \mathbf{V}^T = \sum_i^r \lambda_i \mathbf{u}_i \mathbf{v}_i^T, \quad (4.4)$$

where

- \mathbf{U} is an $n \times r$ orthonormal matrix of uncorrelated observation scores; these are also the eigenvectors of $\mathbf{X}\mathbf{X}'$.
- Λ is an $r \times r$ diagonal matrix of singular values, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r$, which are also the square roots of the eigenvalues of $\mathbf{X}\mathbf{X}'$.
- \mathbf{V} is an $r \times p$ orthonormal matrix of variable weights and also the eigenvectors of $\mathbf{X}'\mathbf{X}$.

Then, a rank 2 (or 3) PCA approximation $\widehat{\mathbf{X}}$ to the data matrix used in the biplot can be obtained from the first 2 (or 3) singular values λ_i and the corresponding $\mathbf{u}_i, \mathbf{v}_i$ as:

$$\mathbf{X} \approx \widehat{\mathbf{X}} = \lambda_1 \mathbf{u}_1 \mathbf{v}'_1 + \lambda_2 \mathbf{u}_2 \mathbf{v}'_2 .$$

The variance of \mathbf{X} accounted for by each term is λ_i^2 .

A biplot is then obtained by overlaying two scatterplots that share a common set of axes and have a between-set scalar product interpretation. Typically, the observations (rows of \mathbf{X}) are represented as points and the variables (columns of \mathbf{X}) are represented as vectors from the origin.

The **scale** factor, α allows the variances of the components to be apportioned between the row points and column vectors, with different interpretations, by representing the approximation $\widehat{\mathbf{X}}$ as the product of two matrices,

$$\widehat{\mathbf{X}} = (\mathbf{U}\boldsymbol{\Lambda}^\alpha)(\boldsymbol{\Lambda}^{1-\alpha}\mathbf{V}') = \mathbf{AB}'$$

This notation uses a little math trick involving a power, $0 \leq \alpha \leq 1$: When $\alpha = 1$, $\boldsymbol{\Lambda}^\alpha = \boldsymbol{\Lambda}^1 = \boldsymbol{\Lambda}$, and $\boldsymbol{\Lambda}^{1-\alpha} = \boldsymbol{\Lambda}^0 = \mathbf{I}$. $\alpha = 1/2$ gives the diagonal matrix $\boldsymbol{\Lambda}^{1/2}$ whose elements are the square roots of the singular values.

The choice $\alpha = 1$ assigns the singular values totally to the left factor; then, the angle between two variable vectors, reflecting the inner product $\mathbf{x}_j^\top, \mathbf{x}_{j'}$ approximates their correlation or covariance, and the distance between the points approximates their Mahalanobis distances. $\alpha = 0$ gives a distance interpretation to the column display. $\alpha = 1/2$ gives a symmetrically scaled biplot. *TODO**: Explain this better.

When the singular values are assigned totally to the left or to the right factor, the resultant coordinates are called *principal coordinates* and the sum of squared coordinates on each dimension equal the corresponding singular value. The other matrix, to which no part of the singular values is assigned, contains the so-called *standard coordinates* and have sum of squared values equal to 1.0.

4.3.2 Biplots in R

There are a large number of R packages providing biplots. The most basic, `stats::biplot()`, provides methods for "prcomp" and "princomp" objects. Among other packages, `factoextra` (Kassambara & Mundt, 2020), an extension of `FactoMineR` (Husson et al., 2025), is perhaps the most comprehensive and provides `ggplot2` graphics. In addition to biplot methods for quantitative data using PCA (`fviz_pca()`), it offers biplots for categorical data using correspondence analysis (`fviz_ca()`) and multiple correspondence analysis (`fviz_mca()`); factor analysis with mixed quantitative and categorical variables (`fviz_fAMD()`) and cluster analysis (`fviz_cluster()`). The `adegraphics` package (Dray & Siberchicot, 2025) produces lovely biplots using `lattice` graphics, but with its own analytic framework.

Here, I use the `ggbio` package (Vu & Friendly, 2024), which aims to provide a simple interface to biplots within the `ggplot2` framework. I also use some convenient utility functions from `factoextra`.

4.3.3 Example: Crime data

A basic biplot of the `crime` data, using standardized principal components and labeling the observation by their state abbreviation is shown in Figure 4.12. The correlation circle reflects the data ellipse of the standardized components. This reminds us that these components are uncorrelated and have equal variance in the display.

```
crime.pca <- reflect(crime.pca) # reflect the axes

ggbio(crime.pca,
      obs.scale = 1, var.scale = 1,
      labels = crime$st ,
      circle = TRUE,
```

```
varname.size = 4,
varname.color = "brown") +
theme_minimal(base_size = 14)
```

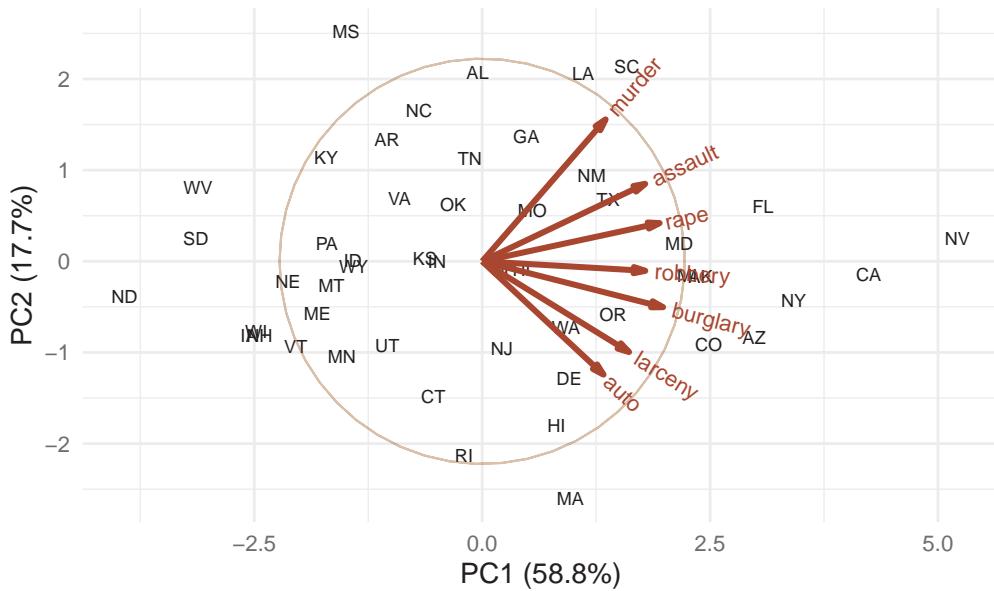


Figure 4.12: Basic biplot of the crime data. State abbreviations are shown at their standardized scores on the first two dimensions. The variable vectors reflect the correlations of the variables with the biplot dimensions.

In this dataset the states are grouped by region and we saw some differences among regions in the plot (Figure 4.9) of component scores. `ggbiplots()` provides options to include a `groups = variable`, used to color the observation points and also to draw their data ellipses, facilitating interpretation.

```
ggbiplots(crime.pca,
obs.scale = 1, var.scale = 1,
groups = crime$region,
labels = crime$st,
labels.size = 4,
var.factor = 1.4,
ellipse = TRUE,
ellipse.prob = 0.5, ellipse.alpha = 0.1,
circle = TRUE,
varname.size = 4,
varname.color = "black",
clip = "off") +
labs(fill = "Region", color = "Region") +
theme_minimal(base_size = 14) +
theme(legend.direction = 'horizontal', legend.position = 'top')
```

This plot provides what is necessary to interpret the nature of the components and also the variation of the states in relation to these. In this, the data ellipses for the regions provide a visual summary that aids interpretation.

- From the variable vectors, it seems that PC1, having all positive and nearly equal loadings, reflects a total

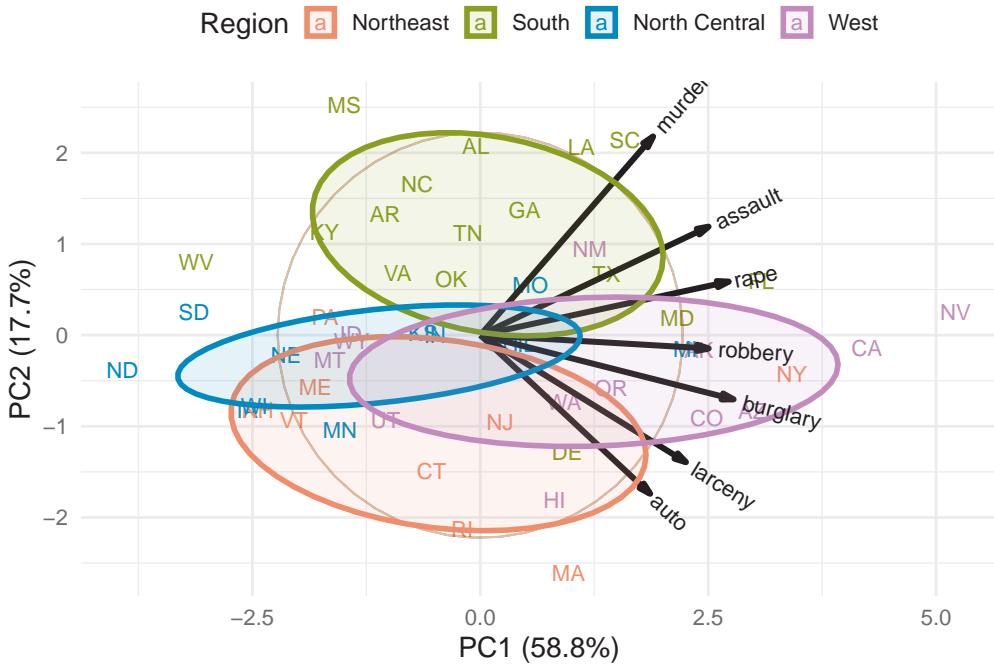


Figure 4.13: Enhanced biplot of the crime data, grouping the states by region and adding data ellipses.

or overall index of crimes. Nevada, California, New York and Florida are highest on this, while North Dakota, South Dakota and West Virginia are lowest.

- The second component, PC2, shows a contrast between crimes against persons (murder, assault, rape) at the top and property crimes (auto theft, larceny) at the bottom. Nearly all the Southern states are high on personal crimes; states in the North East are generally higher on property crimes.
- Western states tend to be somewhat higher on overall crime rate, while North Central are lower on average. In these states there is not much variation in the relative proportions of personal vs. property crimes.

Moreover, in this biplot you can interpret the value for a particular state on a given crime by considering its projection on the variable vector, where the origin corresponds to the mean, positions along the vector have greater than average values on that crime, and the opposite direction have lower than average values. For example, Massachusetts has the highest value on auto theft, but a value less than the mean. Louisiana and South Carolina on the other hand are highest in the rate of murder and slightly less than average on auto theft.

These 2D plots account for only 76.5% of the total variance of crimes, so it is useful to also examine the third principal component, which accounts for an additional 10.4%. The `choices =` option controls which dimensions are plotted.

```
ggbiplots(crime.pca,
  choices = c(1,3),
  obs.scale = 1, var.scale = 1,
  groups = crime$region,
  labels = crime$st,
  labels.size = 4,
  var.factor = 2,
  ellipse = TRUE,
  ellipse.prob = 0.5, ellipse.alpha = 0.1,
```

```

circle = TRUE,
varname.size = 4,
varname.color = "black",
clip = "off") +
labs(fill = "Region", color = "Region") +
theme_minimal(base_size = 14) +
theme(legend.direction = 'horizontal', legend.position = 'top')

```

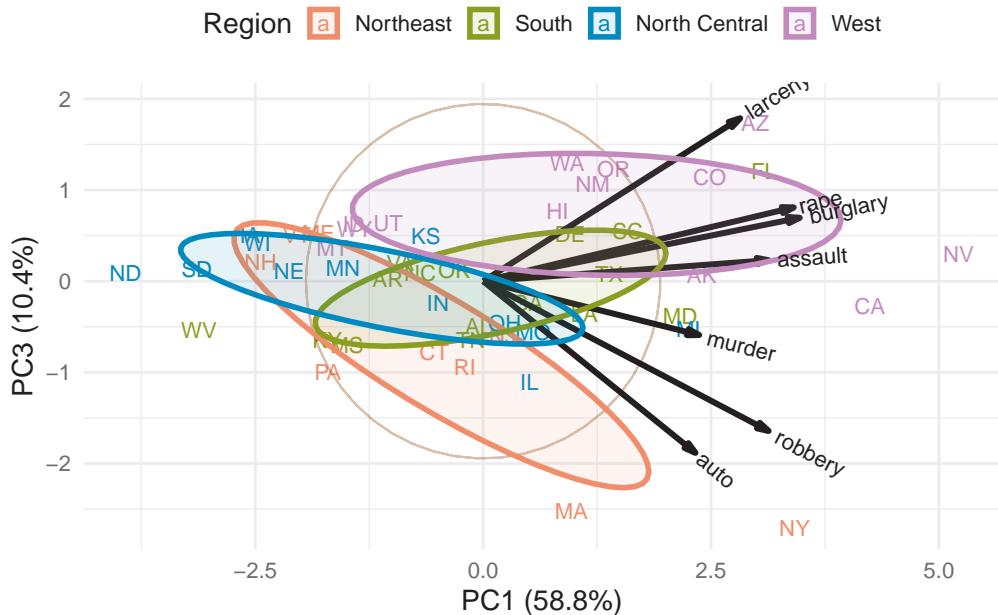


Figure 4.14: Biplot of dimensions 1 & 3 of the crime data, with data ellipses for the regions.

Dimension 3 in Figure 4.14 is more subtle. One interpretation is a contrast between larceny, which is a larceny (simple theft) and robbery, which involves stealing something from a person and is considered a more serious crime with an element of possible violence. In this plot, murder has a relatively short variable vector, so does not contribute very much to differences among the states.

4.3.4 Biplot contributions and quality

To better understand how much each variable contributes to the biplot dimensions, it is helpful to see information about the variance of variables along each dimension. Graphically, this is nothing more than a measure of the lengths of projections of the variables on each of the dimensions. `factoextra::get_pca_var()` calculates a number of tables from a "prcomp" or similar object.

```

var_info <- factoextra::get_pca_var(crime.pca)
names(var_info)
#> [1] "coord"    "cor"       "cos2"      "contrib"

```

The component `cor` gives correlations of the variables with the dimensions and `contrib` gives their variance contributions as percents, where each row and column sums to 100.

```

contrib <- var_info$contrib
cbind(contrib, Total = rowSums(contrib)) |>

```

```

rbind(Total = c(colSums(contrib), NA)) |>
  round(digits=2)
#>           Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7 Total
#> murder      9.02  39.59   3.18   5.39  28.96   6.71   7.16  100
#> rape        18.64   2.87   5.96   0.39   3.55  59.79   8.79  100
#> robbery     15.75   0.18  24.59  31.14  27.04   1.31   0.00  100
#> assault     15.73  11.80   0.48  39.67  25.67   2.97   3.68  100
#> burglary    19.37   4.13   4.41   0.33   1.02  28.73  42.01  100
#> larceny     12.77  16.19  29.08   5.52   0.09   0.16  36.20  100
#> auto         8.71  25.24  32.31  17.58  13.67   0.33   2.16  100
#> Total       100.00 100.00 100.00 100.00 100.00 100.00 100.00   NA

```

These contributions can be visualized as sorted barcharts for a given axis using `factoextra::fviz_contrib()`. The dashed horizontal lines are at the average value for each dimension.

```

p1 <- fviz_contrib(crime.pca, choice = "var", axes = 1,
                    fill = "lightgreen", color = "black")
p2 <- fviz_contrib(crime.pca, choice = "var", axes = 2,
                    fill = "lightgreen", color = "black")
p1 + p2

```

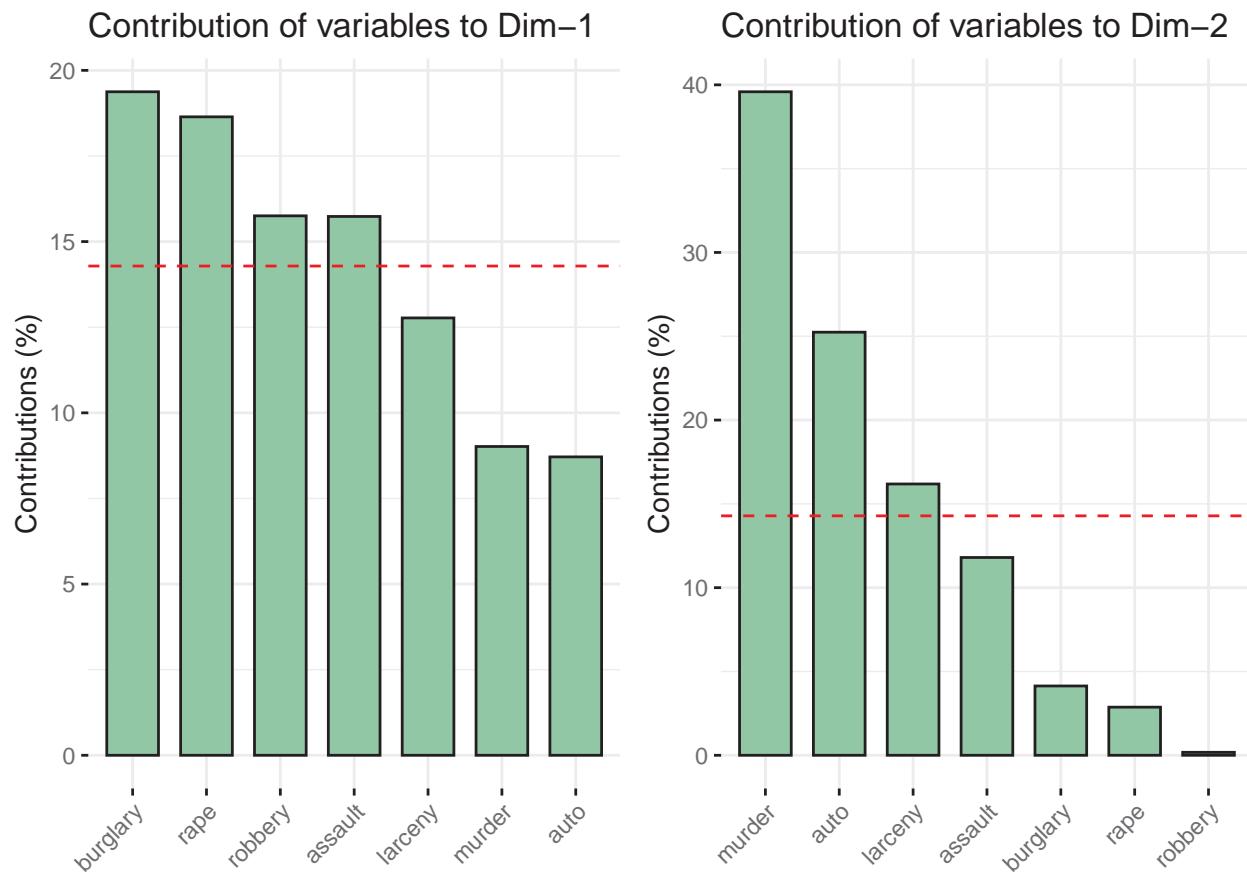


Figure 4.15: Contributions of the crime variables to dimensions 1 (left) & 2 (right) of the PCA solution

A simple rubric for interpreting the dimensions in terms of the variable contributions is to mention those that

are largest or above average on each dimension. So, burglary and rape contribute most to the first dimension, while murder and auto theft contribute most to the second.

Another useful measure is called `cos2`, the *quality* of representation, meaning how much of a variable is represented in a given component. The columns sum to the eigenvalue for each dimension. The rows each sum to 1.0, meaning each variable is completely represented on all components, but we can find the quality of a k -D solution by summing the values in the first k columns. These can be plotted in a style similar to Figure 4.15 using `factoextra::fviz_cos2()`.

```
quality <- var_info$cos2
rowSums(quality)
#>   murder      rape    robbery assault burglary larceny      auto
#>       1         1        1        1        1        1        1

colSums(quality)
#> Dim.1 Dim.2 Dim.3 Dim.4 Dim.5 Dim.6 Dim.7
#> 4.115 1.239 0.726 0.316 0.258 0.222 0.124

cbind(quality[, 1:2],
      Total = rowSums(quality[, 1:2])) |>
  round(digits = 2)
#>           Dim.1 Dim.2 Total
#> murder      0.37  0.49  0.86
#> rape        0.77  0.04  0.80
#> robbery     0.65  0.00  0.65
#> assault     0.65  0.15  0.79
#> burglary    0.80  0.05  0.85
#> larceny     0.53  0.20  0.73
#> auto        0.36  0.31  0.67
```

In two dimensions, murder and burglary are best represented; robbery and larceny are the worst, but as we saw above (Figure 4.14), these crimes are implicated in the third dimension.

4.3.5 Supplementary variables

An important feature of biplot methodology is that once you have a reduced-rank display of the relations among a set of variables, you can use other available data to help interpret what is shown in the biplot. In a sense, this is what I did above in Figure 4.13 and Figure 4.14 using `region` as a grouping variable and summarizing the variability in the scores for states with their data ellipses by region.

When we have other quantitative variables on the same observations, these can be represented as supplementary variables in the same space. Geometrically, this amounts to projecting the new variables on the space of the principal components. It is carried out by regressions of these supplementary variables on the scores for the principal component dimensions.

For example, the left panel of Figure 4.16 depicts the vector geometry of a regression of a variable \mathbf{y} on two predictors, \mathbf{x}_1 and \mathbf{x}_2 . The fitted vector, $\hat{\mathbf{y}}$, is the perpendicular projection of \mathbf{y} onto the plane of \mathbf{x}_1 and \mathbf{x}_2 . In the same way, in the right panel, a **supplementary variable** is projected into the plane of two principal component axes shown as an ellipse. The black fitted vector shows how that additional variable relates to the biplot dimensions.

For this example, it happens that some suitable supplementary variables to aid interpretation of crime rates are available in the dataset `datasets::state.x77`, which was obtained from the U.S. Bureau of the Census *Statistical Abstract of the United States* for 1977. I select a few of these below and make the state name a column variable so it can be merged with the `crime` data.

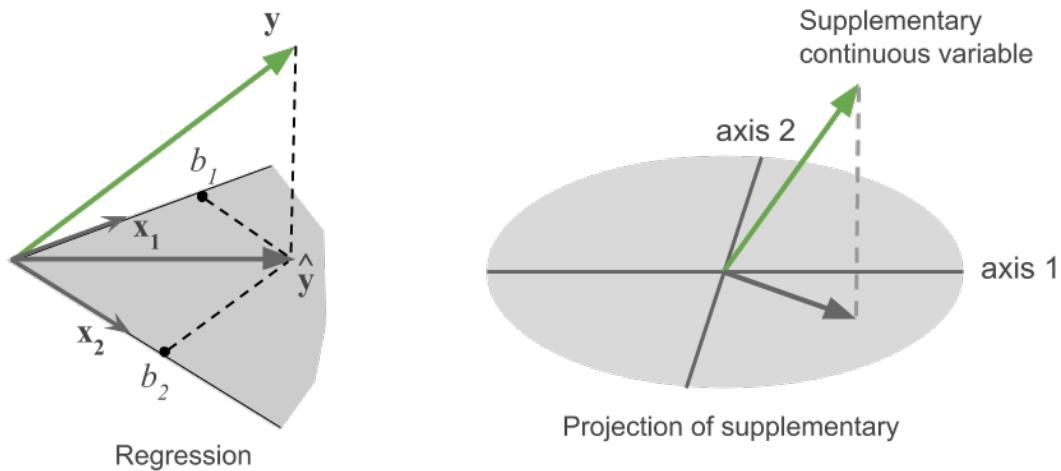


Figure 4.16: Fitting supplementary variables in a biplot is analogous (right) to regression on the principal component dimensions (left). *Source:* Aluja et al. (2018), Figure 2.11

```

supp_data <- state.x77 |>
  as.data.frame() |>
  tibble::rownames_to_column(var = "state") |>
  rename(Life_Exp = `Life_Exp`,
         HS_Grad = `HS Grad`) |>
  select(state, Income:Life_Exp, HS_Grad)

head(supp_data)
#>      state Income Illiteracy Life_Exp HS_Grad
#> 1    Alabama   3624      2.1    69.0    41.3
#> 2     Alaska   6315      1.5    69.3    66.7
#> 3    Arizona   4530      1.8    70.5    58.1
#> 4   Arkansas   3378      1.9    70.7    39.9
#> 5 California   5114      1.1    71.7    62.6
#> 6 Colorado    4884      0.7    72.1    63.9

```

Then, we can merge the `crime` data with the `supp_data` dataset to produce something suitable for analysis using `factoMineR::PCA()`.

```

crime_joined <-
  dplyr::left_join(crime[, 1:8], supp_data, by = "state")
names(crime_joined)
#> [1] "state"        "murder"       "rape"        "robbery"
#> [5] "assault"      "burglary"     "larceny"     "auto"
#> [9] "Income"       "Illiteracy"   "Life_Exp"    "HS_Grad"

```

`PCA()` can only get the labels for the observations from the `row.names()` of the dataset, so I assign them explicitly. The supplementary variables are specified by the argument `quanti.sup` as the indices of the columns in what is passed as the data argument.

```
row.names(crime_joined) <- crime$st
crime.PCA_sup <- PCA(crime_joined[,c(2:8, 9:12)],
                      quanti.sup = 8:11,
                      scale.unit=TRUE,
                      ncp=3,
                      graph = FALSE)
```

The essential difference between the result of `prcomp()` used earlier to get the `crime.pca` object and the result of `PCA()` with supplementary variables is that the `crime.PCA_sup` object now contains a `quanti.sup` component containing the coordinates for the supplementary variables in PCA space.

These can be calculated directly as the coefficients of a multivariate regression of the *standardized* supplementary variables on the PCA scores for the dimensions, with no intercept—which forces the fitted vectors to go through the origin. For example, in the plot below (Figure 4.17), the vector for Income has coordinates (0.192, -0.530) on the first two PCA dimensions.

```
reg.data <- cbind(scale(supp_data[, -1]),
                    crime.PCA_sup$ind$coord) |>
  as.data.frame()

sup.mod <- lm(cbind(Income, Illiteracy, Life_Exp, HS_Grad) ~
               0 + Dim.1 + Dim.2 + Dim.3,
               data = reg.data )

(coefs <- t(coef(sup.mod)))
#>           Dim.1   Dim.2   Dim.3
#> Income      0.192   0.530   0.0482
#> Illiteracy   0.112  -0.536   0.1689
#> Life_Exp    -0.131   0.649  -0.2158
#> HS_Grad     0.103   0.610  -0.4095
```

Note that, because the supplementary variables are standardized, these coefficients are the same as the correlations between the supplementary variables and the scores on the principal components, up to a scaling factor for each dimension. This provides a general way to relate dimensions found in other methods to the original data variables using vectors as in biplot techniques.

```
cor(reg.data[, 1:4], reg.data[, 5:7]) |>
  print() -> R
#>           Dim.1   Dim.2   Dim.3
#> Income      0.393   0.596   0.0415
#> Illiteracy   0.230  -0.602   0.1453
#> Life_Exp    -0.268   0.730  -0.1857
#> HS_Grad     0.211   0.686  -0.3524

R / coefs
#>           Dim.1   Dim.2   Dim.3
#> Income      2.05   1.12  0.861
#> Illiteracy   2.05   1.12  0.861
#> Life_Exp    2.05   1.12  0.861
#> HS_Grad     2.05   1.12  0.861
```

The `PCA()` result can then be plotted using `FactoMiner::plot()` or various `factoextra` functions like

`fviz_pca_var()` for a plot of the variable vectors or `fviz_pca_biplot()` for a biplot. When a `quanti.sup` component is present, supplementary variables are also shown in the displays.

For simplicity I use `FactoMiner::plot()` here and only show the variable vectors. For consistency with earlier plots, I first reflect the orientation of the 2nd PCA dimension so that crimes of personal violence are at the top, as in Figure 4.10.

```
# reverse coordinates of Dim 2
crime.PCA_sup <- ggbioplot::reflect(crime.PCA_sup, columns = 2)
# also reverse the orientation of coordinates for supplementary vars on Dim 2
# crime.PCA_sup$quanti.sup$coord[, 2] <- -crime.PCA_sup$quanti.sup$coord[, 2]
plot(crime.PCA_sup, choix = "var")
```

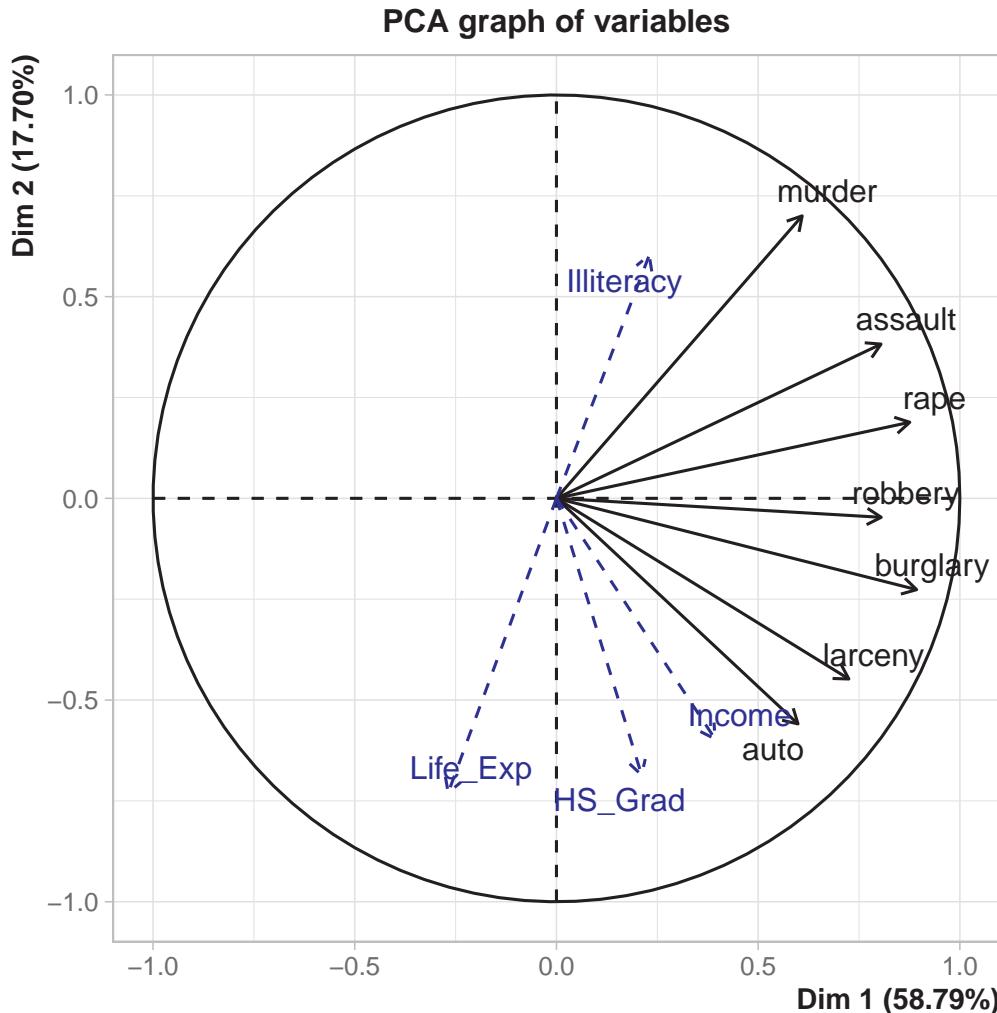


Figure 4.17: PCA plot of variables for the crime data, with vectors for the supplementary variables showing their association with the principal component dimensions.

Recall that from earlier analyses, I interpreted the dominant PC1 dimension as reflecting overall rate of crime. The contributions to this dimension, which are the projections of the variable vectors on the horizontal axis in Figure 4.10 and Figure 4.13 were shown graphically by barcharts in the left panel of Figure 4.15.

But now in Figure 4.17, with the addition of variable vectors for the supplementary variables, you can see

how income, rate of illiteracy, life expectancy and proportion of high school graduates are related to the variation in rates of crimes for the U.S. states.

On dimension 1, what stands out is that life expectancy is associated with lower overall crime, while other supplementary variable have positive associations. On dimension 2, crimes against persons (murder, assault, rape) are associated with greater rates of illiteracy among the states, which as we earlier saw (Figure 4.13) were more often Southern states. Crimes against property (auto theft, larceny) at the bottom of this dimension are associated with higher levels of income and high school graduates.

4.3.6 Example: Diabetes data

As another example, consider the data from Reaven & Miller (1979) on measures of insulin and glucose shown in Figure 4 and that led to the discovery of two distinct types of development of Type 2 diabetes ([?@sec-discoveries](#)). This dataset is available as `Diabetes`. The three groups are `Normal`, `Chemical_Diabetic` and `Overt_Diabetic`, and the (numerical) diagnostic variables are:

- `relwt`: relative weight, the ratio of actual to expected weight, given the person's height,
- `glufast`: fasting blood plasma glucose level
- `glutest`: test blood plasma glucose level, a measure of glucose intolerance
- `instest`: plasma insulin during test, a measure of insulin response to oral glucose
- `sspg`: steady state plasma glucose, a measure of insulin resistance

TODO: Should introduce 3D plots earlier, in Ch3 before Section 3.10.

First, let's try to create a 3D plot, analogous to the artist's drawing from PRIM-9 shown in Figure 5. For this, I use `car::scatter3d()` which can show data ellipsoids summarizing each group. The formula notation, `z ~ x + y` assigns the `z` variable to the vertical direction in the plot, and the `x` and `y` variable form a base plane.

```
cols <- c("darkgreen", "blue", "red")
scatter3d(sspg ~ instest + glutest, data=Diabetes,
          groups = Diabetes$group,
          ellipsoid = TRUE,
          surface = FALSE,
          col = cols,
          surface.col = cols)
```

`car::scatter3d()` uses the `rgl` package (Adler & Murdoch, 2023) to render 3D graphics on a display device, which means that it has facilities for perspective, lighting and other visual properties. You can interactively zoom in or out or rotate the display in any of the three dimensions and use `rgl::spin3d()` to animate rotations around any axes and record this a a `movie3d()`. Figure 4.18 shows two views of this plot, one from the front and one from the back. The data ellipsoids are not as evocative as the artist's rendering, but they give a sense of the relative sizes and shapes of the clouds of points for the three diagnostic groups.

The normal group is concentrated near the origin, with relatively low values on all three diagnostic measures. The chemical diabetic group forms a wing with higher values on insulin response to oral glucose (`instest`), while the overt diabetics form the other wing, with higher values on glucose intolerance (`glutest`). The relative sizes and orientations of the data ellipsoids are also informative.

Given this, what can we see in a biplot view based on PCA? The PCA of these data shows that 83% of the variance is captured in two dimensions and 96% in three. The result for 3D is interesting, in that the view from PRIM-9 shown in Figure 5 and Figure 4.18 nearly captured all available information.

```
data(Diabetes, package="heplots")

diab.pca <-
```

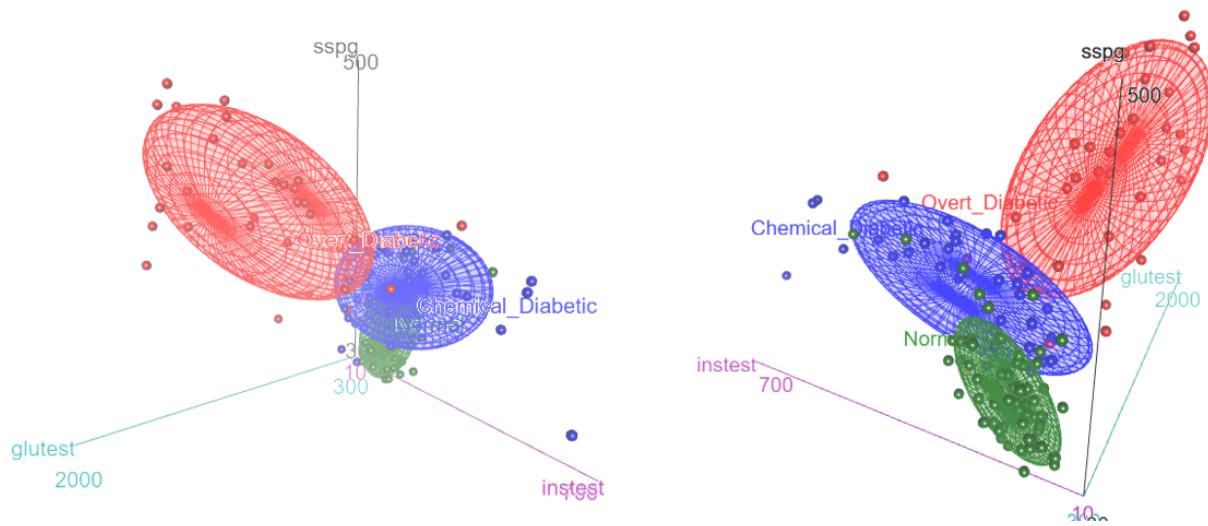


Figure 4.18: Two views of a 3D scatterplot of three main diagnostic variables in the `Diabetes` dataset. The left panel shows an orientation similar to that of Figure 5; the right panel shows a view from the back.

```
Diabetes |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)
summary(diab.pca)

## Importance of components:
##                               PC1    PC2    PC3    PC4    PC5
## Standard deviation     1.662  1.177  0.818  0.3934 0.17589
## Proportion of Variance 0.552  0.277  0.134  0.0309 0.00619
## Cumulative Proportion   0.552  0.829  0.963  0.9938 1.00000
```

A 2D biplot, with data ellipses for the groups, can be produced as before, but I also want to illustrate labeling the groups directly, rather than in a legend.

```
plt <- ggbiplot(diab.pca,
  obs.scale = 1, var.scale = 1,
  groups = Diabetes$group,
  var.factor = 1.4,
  ellipse = TRUE,
  ellipse.prob = 0.5, ellipse.alpha = 0.1,
  circle = TRUE,
  point.size = 2,
  varname.size = 4) +
  labs(fill = "Group", color = "Group") +
  theme_minimal(base_size = 14) +
  theme(legend.position = "none")
```

Then, find the centroids of the component scores and use `geom_label()` to plot the group labels.

```
scores <- data.frame(diab.pca$x[, 1:2], group = Diabetes$group)
centroids <- scores |>
```

```
group_by(group) |>
  summarize(PC1 = mean(PC1),
            PC2 = mean(PC2))

plt + geom_label(data = centroids,
                  aes(x = PC1, y = PC2,
                      label=group, color = group),
                  nudge_y = 0.2)
```

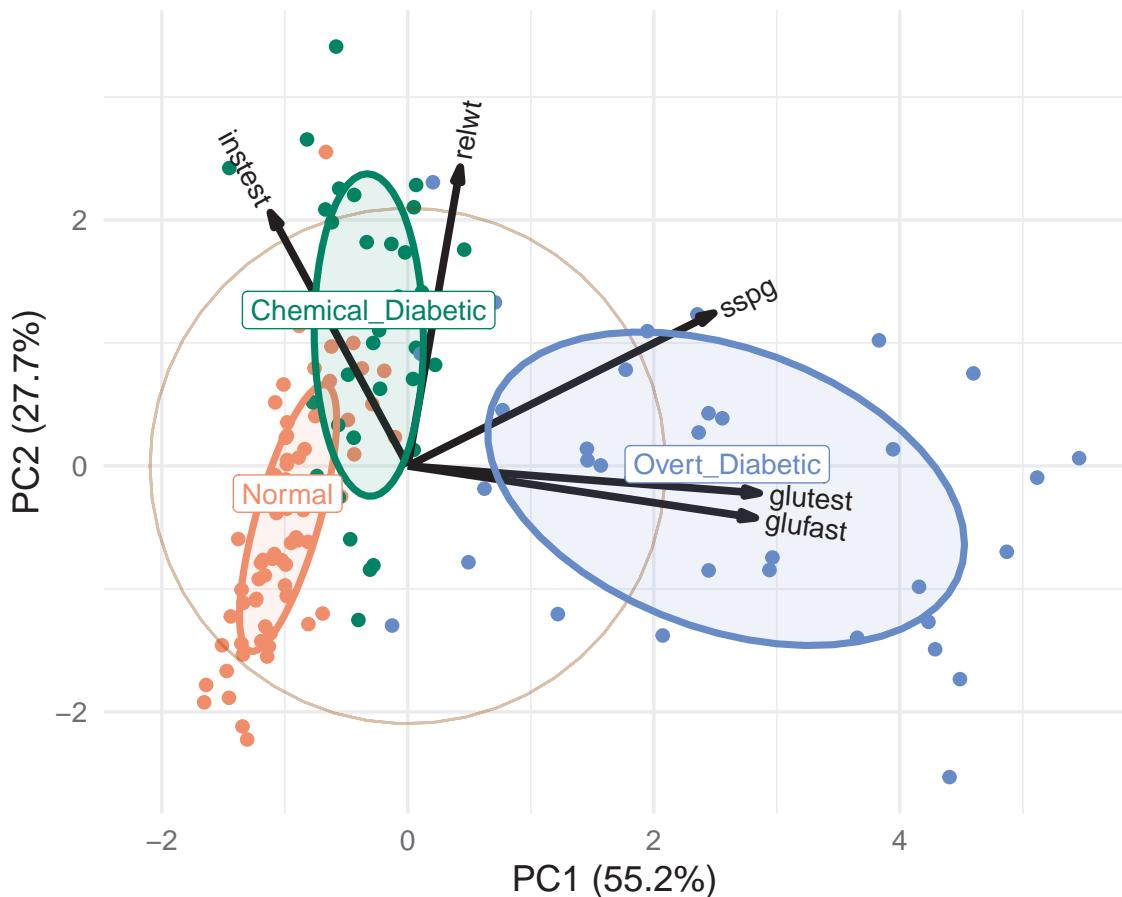


Figure 4.19: 2D biplot of the Diabetes data

What can we see here, and how does it relate to the artist's depiction in Figure 5? The variables `insteat`, `sspg` and `glutest` correspond approximately to the coordinate axes in the artist's drawing. `glutest` and `glufast` primarily separate the overt diabetics from the others. The chemical diabetics are distinguished by having larger values of insulin response (`insteat`) and are also higher in relative weight (`relwt`).

4.4 Nonlinear dimension reduction

The world of dimension reduction methods reflected by PCA is a simple and attractive one in which relationships among variable are at least approximately linear, and can be made visible in a lower-dimensional view by linear transformations and projections. PCA does an optimal job of capturing *global* linear relationships

in the data. But many phenomena defy linear description or involve *local* nonlinear relationships and clusters within the data. Our understanding of high-D data can sometimes be improved by nonlinear dimension reduction techniques.

To see why, consider the data shown in the left panel of Figure 4.20 and suppose we want to be able to separate the two classes by a line. The groups are readily seen in this simple 2D example, but there is no linear combination or projection that shows them as distinct categories. The right panel shows the same data after a nonlinear transformation to polar coordinates, where the two groups are readily distinguished by radius. Such problems arise in higher dimensions where direct visualization is far more difficult and nonlinear methods become attractive.

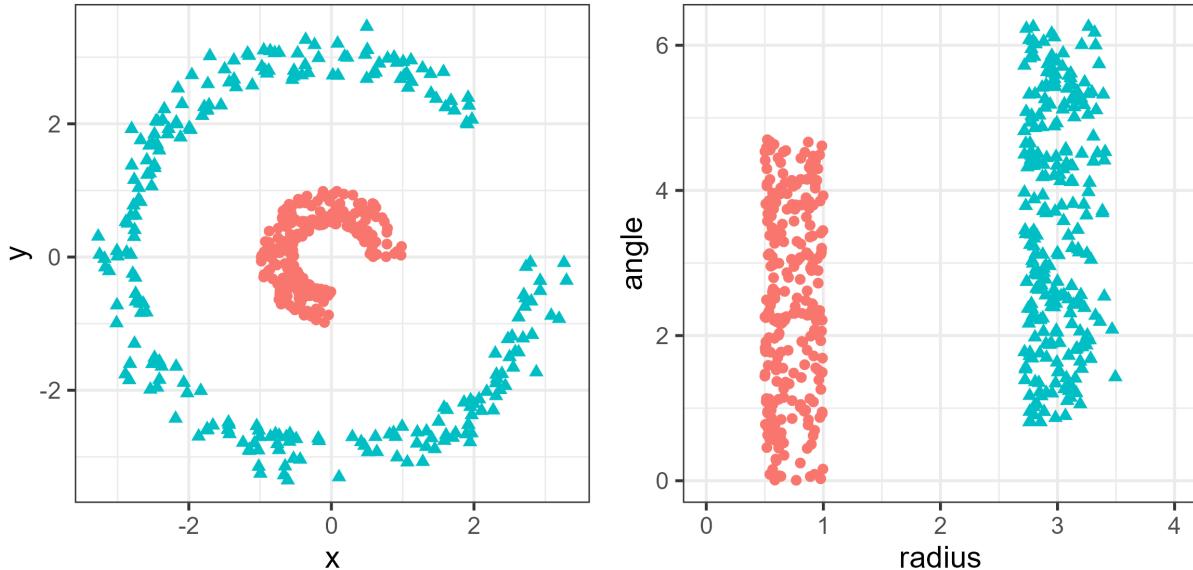


Figure 4.20: *Nonlinear patterns*: Two representations of the same data are shown. In the plot at the left, the clusters are clear to the eye, but there is no linear relation that separates them. Transforming the data nonlinearly, to polar coordinates in the plot at the right, makes the two groups distinct.

4.4.1 Multidimensional scaling

One way to break out of the “linear-combination, maximize-variance PCA” mold is to consider a more intrinsic property of points in *Spaceland*: similarity or distance. The earliest expression of this idea was in **multidimensional scaling** (MDS) by Torgerson (1952), which involved trying to determine a metric low-D representation of objects from their interpoint distances via an application of the SVD.

The break-through for nonlinear methods came from Roger Shepard and William Kruskal (Kruskal, 1964; Shepard, 1962a, 1962b) who recognized that a more general, *nonmetric* version (nMDS) could be achieved using only the *rank order* of input distances d_{ij} among objects. nMDS maps these into a low-D spatial representation of points, $\mathbf{x}_i, \mathbf{x}_j$ whose fitted distances, $\hat{d}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ matches the order of the d_{ij} as closely as possible. That is, rather than assume that the observed distances are linearly related to the fitted \hat{d}_{ij} , nMDS assumes only that their order is the same. Borg & Groenen (2005) and Borg et al. (2018) give a comprehensive overview of modern developments in MDS.

The impetus for MDS stemmed largely from psychology and the behavioral sciences, where simple experimental measures of similarity or dissimilarity of psychological objects (color names, facial expressions, words, Morse code symbols) could be obtained by direct ratings, confusions, or other tasks (Shepard et al., 1972b, 1972a). MDS was revolutionary in that it provided a coherent method to study the dimensions of perceptual and

cognitive space in applications where the explanation of a cognitive process was derived directly from an MDS solution ([Shoben, 1983](#)).

To perform nMDS, you need to calculate the matrix of distances between all pairs of observations (`dist()`). The basic function is `MASS::isoMDS()`.⁴ In the call, you can specify the number of dimensions (`k`) desired, with `k=2` as default. It returns the coordinates in a dataset called `points`.

```
diab.dist <- dist(Diabetes[, 1:5])
mds <- diab.dist |>
  MASS::isoMDS(k = 2, trace = FALSE) |>
  purrr::pluck("points")

colnames(mds) <- c("Dim1", "Dim2")
mds <- bind_cols(mds, group = Diabetes$group)
mds |> sample_n(6)

#> # A tibble: 6 x 3
#>   Dim1   Dim2 group
#>   <dbl>  <dbl> <fct>
#> 1 -213.  -42.1 Normal
#> 2  191.   47.3 Overt_Diabetic
#> 3  12.0  -63.2 Overt_Diabetic
#> 4  25.0  -38.1 Chemical_Diabetic
#> 5 774.    9.44 Overt_Diabetic
#> 6  79.0  136.  Overt_Diabetic
```

The method works by trying to minimize a measure, “Stress”, of the average difference between the fitted distances \hat{d}_{ij} and an optimal monotonic (order-preserving) transformation, $f_{\text{mon}}(d_{ij})$, of the distances in the data. Values of Stress around 5-8% and smaller are generally considered adequate.

Unlike PCA, where you can fit all possible dimensions once and choose the number of components to retain by examining the eigenvalues or variance proportions, in MDS it is necessary to fit the data for several values of `k` and consider the trade-off between goodness of fit and complexity.

```
stress <- vector(length = 5)
for(k in 1:5){
  res <- MASS::isoMDS(diab.dist, k=k, trace = FALSE)
  stress[k] <- res$stress
}
round(stress, 3)
#> [1] 17.755 3.525 0.256 0.000 0.000
```

Plotting these shows that a 3D solution is nearly perfect, while a 2D solution is certainly adequate. This plot is the MDS analog of a screeplot for PCA.

```
plot(stress, type = "b", pch = 16, cex = 2,
     xlab = "Number of dimensions",
     ylab = "Stress (%)")
```

To plot the 2D solution, I'll use `ggpubr::ggscatter()` here because it handles grouping, provides concentration ellipses and other graphical features.

⁴The `vegan` package ([Oksanen et al., 2025](#)) provides `vegan::metaMDS()` which allows a wide range of distance measures ...

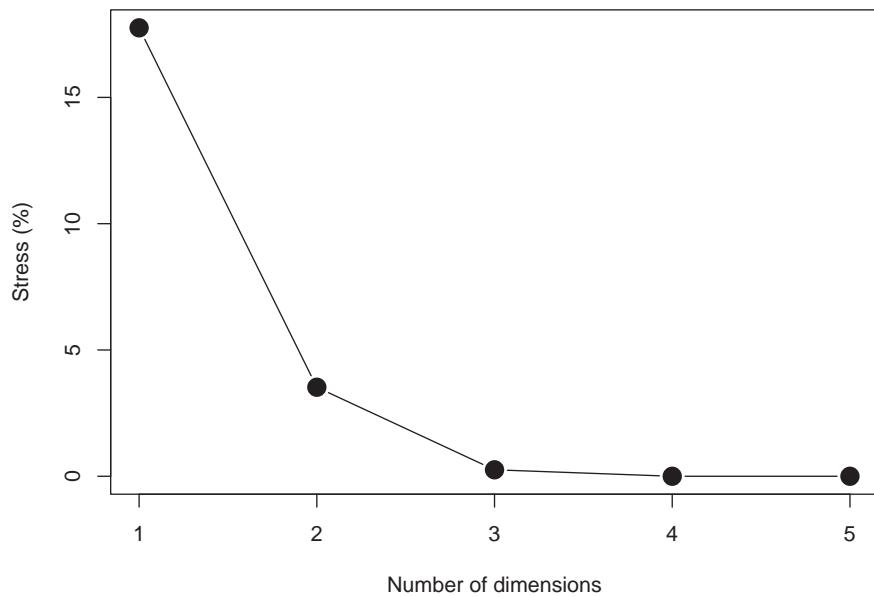


Figure 4.21: Badness of fit (Stress) of the MDS solution in relation to number of dimensions.

```
library(ggpubr)
cols <- scales::hue_pal()(3) |> rev()
mplot <-
ggscatter(mds, x = "Dim1", y = "Dim2",
          color = "group",
          shape = "group",
          palette = cols,
          size = 2,
          ellipse = TRUE,
          ellipse.level = 0.5,
          ellipse.type = "t") +
geom_hline(yintercept = 0, color = "gray") +
geom_vline(xintercept = 0, color = "gray")
```

For this and other examples using MDS, it would be nice to also show how the dimensions of this space relate to the original variables, as in a biplot. Using the idea of correlations between variables and dimensions from Section 4.3.5, I do this as shown below. Only the relative directions and lengths of the variable vectors matter, so you can choose any convenient scale factor to make the vectors fill the plot region.

```
vectors <- cor(Diabetes[, 1:5], mds[, 1:2])
scale_fac <- 500
mplot +
coord_fixed() +
geom_segment(data=vectors,
             aes(x=0, xend=scale_fac*Dim1, y=0, yend=scale_fac*Dim2),
             arrow = arrow(length = unit(0.2, "cm"), type = "closed"),
             linewidth = 1.1) +
geom_text(data = vectors,
          aes(x = 1.15*scale_fac*Dim1, y = 1.07*scale_fac*Dim2,
              label=row.names(vectors)),
```

```

nudge_x = 4,
size = 4) +
theme(legend.position = "inside",
legend.position.inside = c(.8, .8))

```

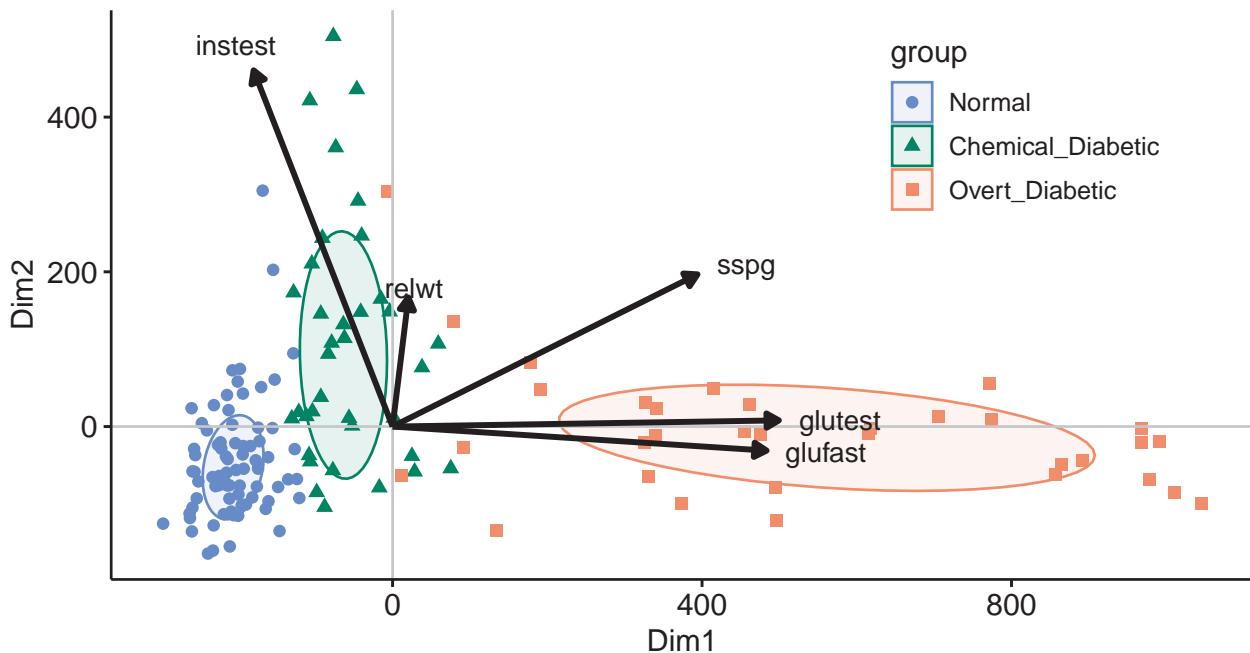


Figure 4.22: Nonmetric MDS representation of the Diabetes data. The vectors reflect the correlations of the variables with the MDS dimensions.

The configuration of the groups in Figure 4.22 is similar to that of the biplot in Figure 4.19, but the groups are more widely separated along the first MDS dimension. The variable vectors are also similar, except that *relwt* is not well-represented in the MDS solution.

4.4.2 t-SNE

With the rise of “machine learning” methods for “feature extraction” in “supervised” vs. “unsupervised” settings, a variety of new algorithms have been proposed for the task of finding low-D representations of high-D data. Among these, t-distributed Stochastic Neighbor Embedding (t-SNE) developed by Maaten & Hinton (2008) is touted as method for revealing *local structure* and clustering better in possibly complex high-D data and at different scales.

t-SNE differs from MDS in what it tries to preserve in the mapping to low-D space: Multidimensional scaling aims to preserve the distances between pairs of data points, focusing on pairs of *distant points* in the original space. t-SNE, on the other hand focuses on preserving *neighboring* data points. Data points that are close in the original data space will be tight in the t-SNE embeddings.

“The t-SNE algorithm models the probability distribution of neighbors around each point. Here, the term *neighbors* refers to the set of points which are closest to each point. In the original, high-dimensional space, this is modeled as a Gaussian distribution. In the 2-dimensional output space this is modeled as a *t*-distribution. The goal of the procedure is to find a mapping onto the 2-dimensional space that minimizes the differences between these two distributions over all points. The fatter tails of a *t*-distribution compared to a Gaussian help to spread the points more evenly in the 2-dimensional space.” (Jake Hoare, [How t-SNE works and Dimensionality Reduction](#)).

t-SNE also uses the idea of mapping distance measures into a low-D space, but converts Euclidean distances into conditional probabilities. Stochastic neighbor embedding means that t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects are assigned a higher probability while dissimilar points are assigned a lower probability.

As van der Maaten and Hinton explained: “The similarity of datapoint \mathbf{x}_j to datapoint \mathbf{x}_i is the conditional probability, $p_{j|i}$, that \mathbf{x}_i would pick \mathbf{x}_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian distribution centered at \mathbf{x}_i .” For $i \neq j$, they define:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)} .$$

and set $p_{i|i} = 0$. σ_i^2 is the variance of the normal distribution that centered on datapoint \mathbf{x}_i and serves as a tuning bandwidth so smaller values of σ_i are used in denser parts of the data space. These conditional probabilities are made symmetric via averaging, giving $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$.

t-SNE defines a similar probability distribution q_{ij} over the points \mathbf{y}_i in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map,

$$D_{\text{KL}}(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} ,$$

a measure of how different the distribution of P in the data is from that of Q in the low-D representation. The t in t-SNE comes from the fact that the probability distribution of the points \mathbf{y}_i in the embedding space is taken to be a heavy-tailed $t_{(1)}$ distribution with one degree of freedom to spread the points more evenly in the 2-dimensional space, rather than the Gaussian distribution for the points in the high-D data space.

t-SNE is implemented in the `Rtsne` package (Krijthe, 2023) which is capable of handling thousands of points in very high dimensions. It uses a tuning parameter, “perplexity” to choose the bandwidth σ_i^2 for each point. This value effectively controls how many nearest neighbors are taken into account when constructing the embedding in the low-dimensional space. It can be thought of as the means to balance between preserving the global and the local structure of the data.⁵

`Rtsne::Rtsne()` finds the locations of the points in the low-D space, of dimension `k=2` by default. It returns the coordinates in a component named `Y`. The package has no `print()`, `summary()` or plot methods, so you’re on your own.

```
library(Rtsne)
set.seed(123)
diab.tsne <- Rtsne(Diabetes[, 1:5], scale = TRUE)
df2 <- data.frame(diab.tsne$Y, group = Diabetes$group)
colnames(df2) <- c("Dim1", "Dim2", "group")
```

You can plot this as shown below:

```
p2 <- ggplot(df2, aes(x=Dim1, y=Dim2, color = group, shape=group)) +
  geom_point(size = 3) +
  stat_ellipse(level = 0.68, linewidth=1.1) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
```

⁵The usual default, `perplexity = 30` focuses on preserving the distances to the 30 nearest neighbors and puts virtually no weight on preserving distances to the remaining points. For data sets with a small number of points e.g. $n = 100$, this will uncover the global structure quite well since each point will preserve distances to a third of the data set. For larger problems, e.g., $n = 10,000$ points, using a higher perplexity value e.g. 500, will do a much better job for of uncovering the global structure. (This description comes from <https://opentsne.readthedocs.io/en/latest/parameters.html>)

```

scale_color_manual(values = cols) +
  labs(x = "Dimension 1",
       y = "Dimension 2") +
  ggtitle("tSNE") +
  theme_bw(base_size = 16) +
  theme(legend.position = "bottom")
p2

```

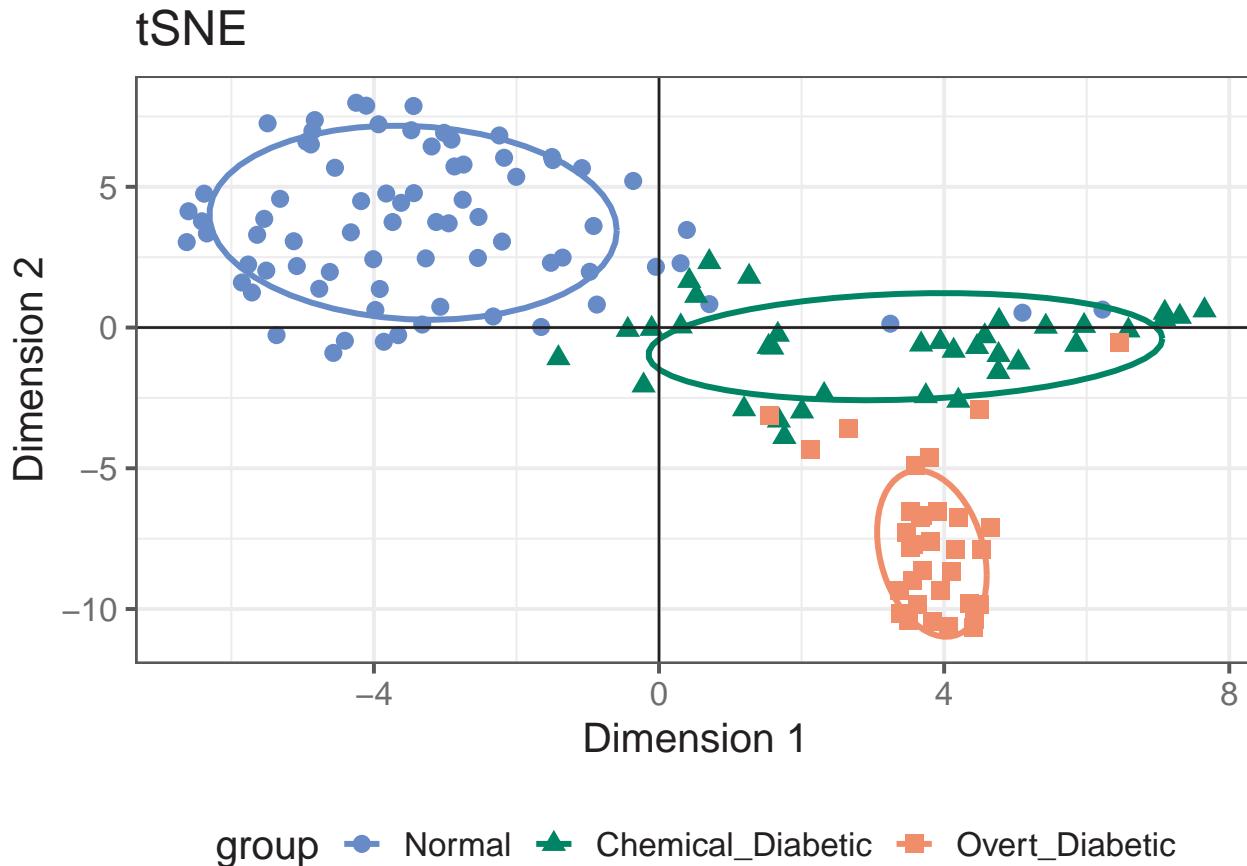


Figure 4.23: t-SNE representation of the Diabetes data.

4.4.2.1 Comparing solutions

For the Diabetes data, I've shown the results of three different dimension reduction techniques, PCA (Figure 4.19), MDS (Figure 4.22), and t-SNE (Figure 4.23). How are these similar, and how do they differ?

One way is to view them side by side as shown in Figure 4.24. To an initial glance, the t-SNE solution looks like a rotated version of the PCA solution, but there are differences in the shapes of the clusters as well.

Another way to compare these two views is to animate the transition from the PCA to the t-SNE representation by a series of smooth interpolated views. This is a more generally useful visualization technique, so it is useful to spell out the details.

The essential idea is calculate interpolated views as a weighted average of the two endpoints using a weight γ that is varied from 0 to 1.

$$\mathbf{X}_{\text{View}} = \gamma \mathbf{X}_{\text{PCA}} + (1 - \gamma) \mathbf{X}_{\text{t-SNE}}$$

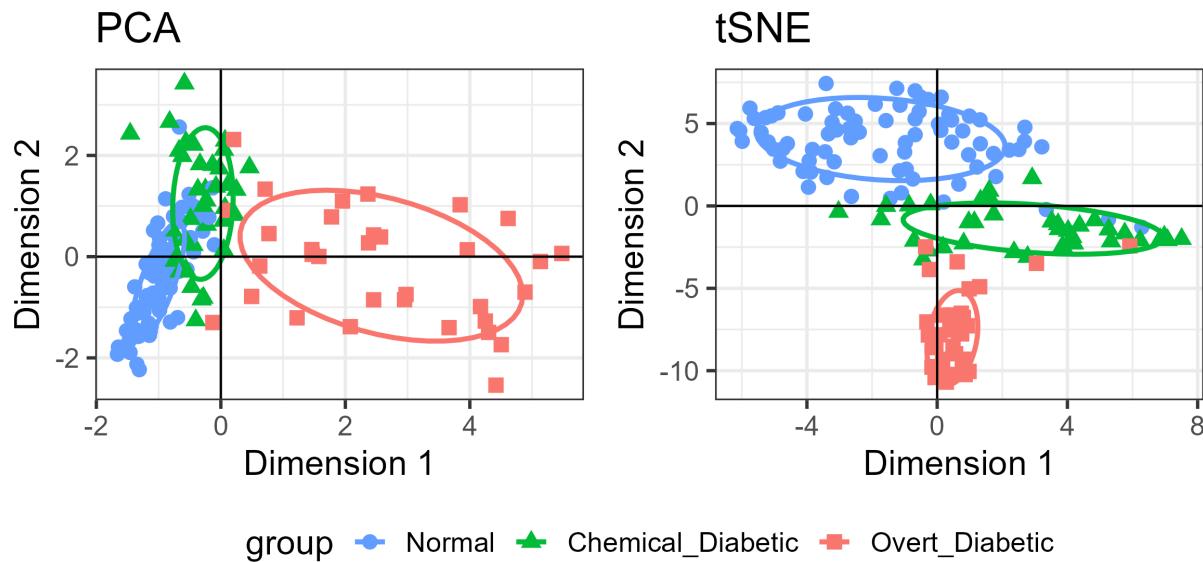


Figure 4.24: Comparison of the PCA and t-SNE 2D representations of the Diabetes data.

The same idea can be applied to other graphical features: lines, paths (ellipses), and so forth. These methods are implemented in the [gganimate](#) package ([Pedersen & Robinson, 2025](#)) .

In this case, to create an animation you can extract the coordinates for the PCA, \mathbf{X}_{PCA} , as a data.frame `df1`, and those for the t-SNE, $\mathbf{X}_{\text{t-SNE}}$ as `df2`, each with a constant `method` variable. These two are then stacked (using `rbind()`) to give a combined `df3`. The animation can then interpolate over `method` going from pure PCA to pure t-SNE.

```
diab.pca <- prcomp(Diabetes[, 1:5], scale = TRUE, rank.=2)
df1 <- data.frame(diab.pca$x, group = Diabetes$group)
colnames(df1) <- c("Dim1", "Dim2", "group")
df1 <- cbind(df1, method="PCA")

set.seed(123)
diab.tsne <- Rtsne(Diabetes[, 1:5], scale = TRUE)
df2 <- data.frame(diab.tsne$Y, group = Diabetes$group)
colnames(df2) <- c("Dim1", "Dim2", "group")
df2 <- cbind(df2, method="tSNE")

# stack the PCA and t-SNE solutions
df3 <- rbind(df1, df2)
```

Then, plot the configuration of the points and add data ellipses as before. The key thing for animating the difference between the solutions is to add `transition_states(method, ...)`, tweening from PCA to t-SNE. The `state_length` argument `transition_states()` controls the relative length of the pause between states.

This animated graphic is shown only in the online version of the book.

```
library(gganimate)
animated_plot <-
```

```

ggplot(df3, aes(x=Dim1, y=Dim2, color=group, shape=group)) +
  geom_point(size = 3) +
  stat_ellipse(level = 0.68, linewidth=1.1) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  scale_color_manual(values = cols) +
  labs(title = "PCA vs. tSNE Dimension Reduction: {closest_state}",
       subtitle = "Frame {frame} of {nframes}",
       x = "Dimension 1",
       y = "Dimension 2") +
  transition_states( method, transition_length = 3, state_length = 2 ) +
  view_follow() +
  theme_bw(base_size = 16) +
  theme(legend.position = "bottom")

animated_plot

```

4.5 Application: Variable ordering for data displays

In many multivariate data displays, such as scatterplot matrices, parallel coordinate plots and others reviewed in Chapter 3, the order of different variables might seem arbitrary. They might appear in alphabetic order, or more often in the order they appear in your dataset, for example when you use `pairs(mydata)`. Yet, the principle of *effect ordering* (Friendly & Kwan (2003)) for variables says you should try to arrange the variables so that adjacent ones are as similar as possible.⁶

For example, the `mtcars` dataset contains data on 32 automobiles from the 1974 U.S. magazine *Motor Trend* and consists of fuel consumption (`mpg`) and 10 aspects of automobile design (`cyl`: number of cylinders; `hp`: horsepower, `wt`: weight) and performance (`qsec`: time to drive a quarter-mile). What can we see from a simple `corrplot()` of their correlations? No coherent pattern stands out in Figure 4.25.

```

data(mtcars)
library(corrplot)
R <- cor(mtcars)
corrplot(R,
         method = 'ellipse',
         title = "Dataset variable order",
         tl.srt = 0, tl.col = "black", tl.pos = 'd',
         mar = c(0,0,1,0))

```

In this display you can scan the rows and columns to “look up” the sign and approximate magnitude of a given correlation; for example, the correlation between `mpg` and `cyl` appears to be about -0.9, while that between `mpg` and `gear` is about 0.5. Of course, you could print the correlation matrix to find the actual values (-0.86 and 0.48 respectively):

⁶The general topic of arranging items (variables, factor values) in an orderly sequence is called *seriation*, and stems from methods of dating in archaeology, used to arrange stone tools, pottery fragments, and other artifacts in time order. In R, the `seriation` package (Hahsler et al., 2024) provides a wide range of techniques. . . .

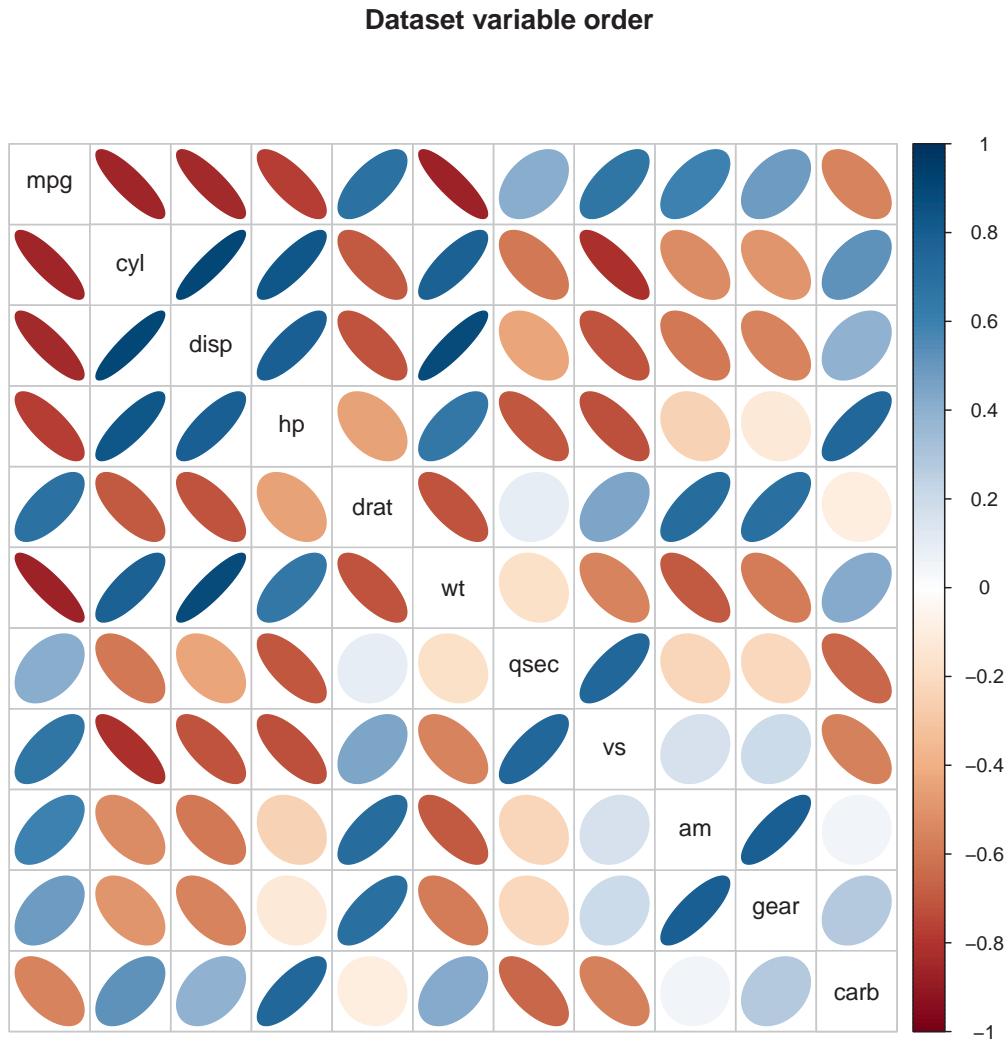


Figure 4.25: Corrplot of `mtcars` data, with the variables arranged in the order they appear in the dataset.

```
print(floor(100*R))
#>      mpg cyl disp  hp drat   wt  qsec vs am gear carb
#> mpg  100 -86 -85 -78   68 -87   41  66  59   48 -56
#> cyl   -86 100  90  83  -70  78  -60 -82 -53  -50  52
#> disp  -85  90 100  79  -72  88  -44 -72 -60  -56  39
#> hp    -78  83  79 100  -45  65  -71 -73 -25  -13  74
#> drat  68 -70 -72 -45  100 -72    9  44  71   69 -10
#> wt   -87  78  88  65  -72 100  -18 -56 -70  -59  42
#> qsec  41 -60 -44 -71    9 -18 100  74 -23  -22 -66
#> vs    66 -82 -72 -73   44 -56  74 100  16   20 -57
#> am   59 -53 -60 -25   71 -70  -23  16 100   79   5
#> gear 48 -50 -56 -13   69 -59  -22  20  79  100  27
#> carb -56  52  39  74  -10  42  -66 -57   5  27 100
```

Because the angles between variable vectors in the biplot reflect their correlations, Friendly & Kwan (2003) defined **principal component variable ordering** as the order of angles, a_i of the first two eigenvectors, $\mathbf{v}_1, \mathbf{v}_2$ around the unit circle. These values are calculated going counter-clockwise from the 12:00 position as:

$$a_i = \begin{cases} \tan^{-1}(v_{i2}/v_{i1}), & \text{if } v_{i1} > 0; \\ \tan^{-1}(v_{i2}/v_{i1}) + \pi, & \text{otherwise.} \end{cases} \quad (4.5)$$

In Equation 4.5 $\tan^{-1}(x)$ is read as “the angle whose tangent is x ”, and so the angles are determined by the tangent ratios “opposite” / “adjacent” = v_{i2}/v_{i1} in the right triangle defined by the vector and the horizontal axis.

For the `mtcars` data the biplot in Figure 4.26 accounts for 84% of the total variance so a 2D representation is fairly good. The plot shows the variables as widely dispersed. There is a collection at the left of positively correlated variables and another positively correlated set at the right.

```
mtcars.pca <- prcomp(mtcars, scale. = TRUE)
ggbiplot(mtcars.pca,
         circle = TRUE,
         point.size = 2.5,
         varname.size = 6,
         varname.color = "brown") +
  theme_minimal(base_size = 14)
```

In `corrplot()` principal component variable ordering is implemented using the `order = "AOE"` option. There are a variety of other methods based on hierarchical cluster analysis described in the [package vignette](#).

Figure 4.27 shows the result of ordering the variables by this method. A nice feature of `corrplot()` is the ability to manually highlight blocks of variables that have a similar pattern of signs by outlining them with rectangles. From the biplot, the two main clusters of positively correlated variables seemed clear, and are outlined in the plot using `corrplot::corrRect()`. What became clear in the corrplot is that `qsec`, the time to drive a quarter-mile from a dead start didn't quite fit this pattern, so I highlighted it separately.

```
corrplot(R,
         method = 'ellipse',
         order = "AOE",
         title = "PCA variable order",
         tl.srt = 0, tl.col = "black", tl.pos = 'd',
         mar = c(0,0,1,0)) |>
  corrRect(c(1, 6, 7, 11))
```

But wait, there is something else to be seen in Figure 4.27. Can you see one cell that doesn't fit with the rest?

Yes, the correlation of number of forward gears (`gear`) and number of carburetors (`carb`) in the upper left and lower right corners stands out as moderately positive (0.27) while all the others in their off-diagonal blocks are negative. This is another benefit of effect ordering: when you arrange the variables so that the most highly related variable are together, features that deviate from dominant pattern become visible.

4.6 Application: Eigenfaces

There are many applications of principal components analysis beyond the use for visualization for multivariate data covered here, that rely on its' ability as a **dimension reduction** technique, that is, to find a low-dimensional approximation to a high-dimensional dataset.

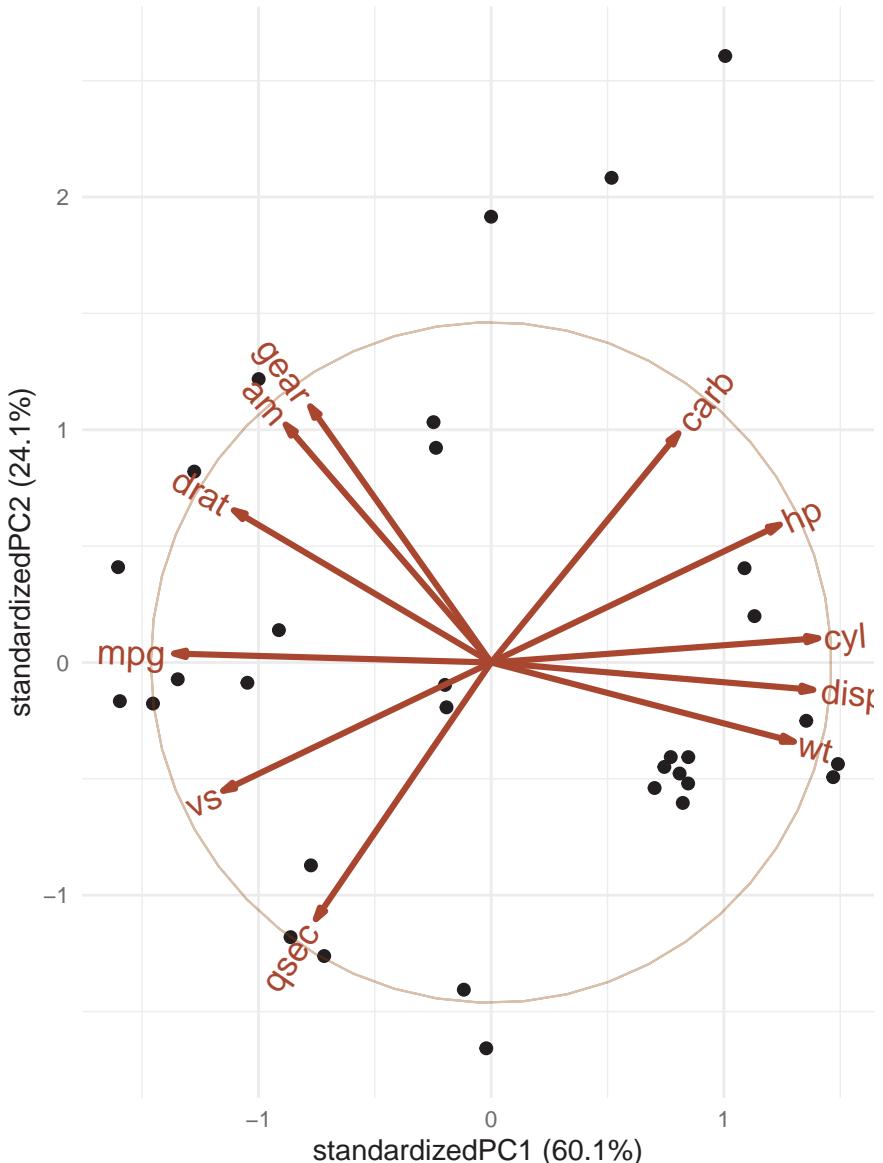


Figure 4.26: Biplot of the `mtcars` data. The order of the variables around the circle, starting from “gear” (say) arranges them so that the most similar variables are adjacent in graphical displays.

i Machine learning uses

In machine learning, for example, PCA is a method used to reduce model complexity and avoid overfitting by *feature extraction*, which amounts to fitting a response variable in a low-D space of the predictors. This is just another name for *principal components regression*, where, instead of regressing the dependent variable on all the explanatory variables directly, a smaller number principal components of the explanatory variables is used as predictors. This has the added benefit that it avoids problems of collinearity (section-ref) due to high correlations of the predictors, because the principal component scores are necessarily uncorrelated. When the goal is model explanation rather than pure prediction, it has the disadvantage that the components may be hard to interpret.

An interesting class of problems have to do with image processing, where an image of size width \times height in

PCA variable order

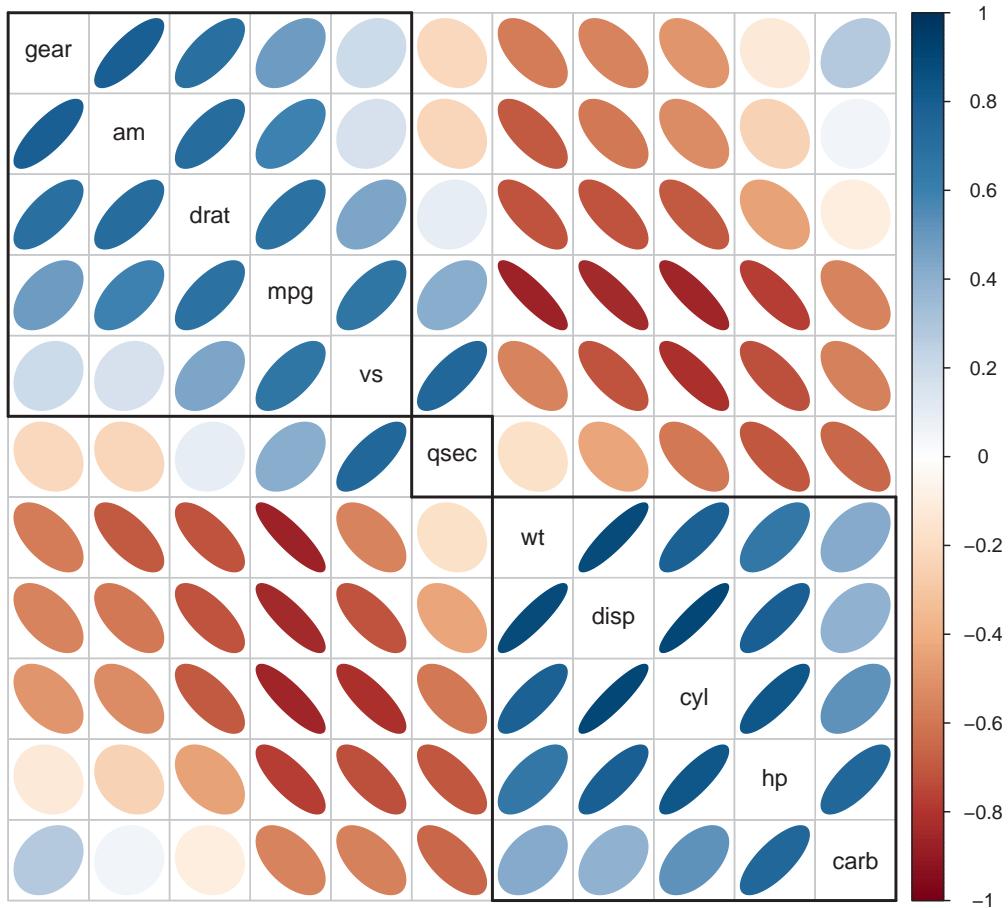


Figure 4.27: Corrplot of `mtcars` data, with the variables ordered according to the variable vectors in the biplot.

pixels can be represented by a $w \times h$ array of greyscale values x_{ij} in the range of $[0, 1]$ or $h \times w \times 3$ array x_{ijk} of (red, green, blue) color values. For example a single 640×640 photo is comprised of about 400K pixels in B/W and 1200K pixels in color.

The uses here include

- **Image compression:** a process applied to a graphics file to minimize its size in bytes for storage or transmission, without degrading image quality below an acceptable threshold
- **image enhancement:** improving the quality of an image, with applications in Computer Vision tasks, remote sensing, and satellite imagery.
- **facial recognition:** classifying or matching a facial image against a large corpus of stored images.

When PCA is used on facial images, you can think of the process as generating **eigenfaces** (Turk & Pentland (1991)) a representation of the pixels in the image in terms of an eigenvalue decomposition. Dimension reduction means that a facial image can be considerably compressed by removing the components associated with small dimensions.

As an example, consider the black and white version of the Mona Lisa shown in Figure 4.28. The idea and code for this example is adapted from this [blog post](#) by Kieran Healy.⁷



Figure 4.28: 640 x 954 black and white image of the *Mona Lisa*. Source: [Wikipedia](#)

It would take too long to explain the entire method, so I'll just sketch the essential parts here. The complete script for this example is contained in [PCA-MonaLisa.R](#). . . .

TODO: Show the necessary parts, including the screeplot.

An image can be imported using `imager::load.image()` which creates a "cimg" object, a 4-dimensional array with dimensions named `x`, `y`, `z`, `c`. `x` and `y` are the usual spatial dimensions, `z` is a depth dimension (which would correspond to time in a movie), and `c` is a color dimension containing R, G, B values.

```
library(imager)
img <- imager::load.image(here::here("images", "MonaLisa-BW.jpg"))
dim(img)
#> [1] 640 954    1    1
```

An `as.data.frame()` method converts this to a data frame with `x` and `y` coordinates. Each `x-y` pair is a location in the 640 by 954 pixel grid, and the `value` is a grayscale value ranging from zero to one.

```
img_df_long <- as.data.frame(img)
head(img_df_long)
#>   x y value
#> 1 1 1 0.431
#> 2 2 1 0.337
#> 3 3 1 0.467
```

⁷<https://kieranhealy.org/blog/archives/2019/10/27/reconstructing-images-using-pca/>

```
#> 4 4 1 0.337
#> 5 5 1 0.376
#> 6 6 1 0.361
```

However, to do a PCA we will need a matrix of data in wide format containing the grayscale pixel values. We can do this using `tidy::pivot_wider()`, giving a result with 640 rows and 954 columns.

```
img_df <- pivot_wider(img_df_long,
                       names_from = y,
                       values_from = value) |>
  select(-x)
dim(img_df)
#> [1] 640 954
```

Mona's PCA is produced from this `img_df` with `prcomp()`:

```
img_pca <- img_df |>
  prcomp(scale = TRUE, center = TRUE)
```

With 955 columns, the PCA comprises 955 eigenvalue/eigenvector pairs. However, the rank of a matrix is the smaller of the number of rows and columns, so only 640 eigenvalues can be non-zero. Printing the first 10 shows that the first three dimensions account for 46% of the variance and we only get to 63% with 10 components.

```
img_pca |>
  broom::tidy(matrix = "eigenvalues") |> head(10)
#> # A tibble: 10 x 4
#>   PC std.dev percent cumulative
#>   <dbl>    <dbl>    <dbl>      <dbl>
#> 1 1     14.1  0.209      0.209
#> 2 2     11.6  0.141      0.350
#> 3 3     10.1  0.107      0.457
#> 4 4     7.83  0.0643     0.522
#> 5 5     6.11  0.0392     0.561
#> 6 6     4.75  0.0237     0.585
#> 7 7     3.70  0.0143     0.599
#> 8 8     3.52  0.0130     0.612
#> 9 9     3.12  0.0102     0.622
#> 10 10   2.86  0.00855    0.631
```

Figure 4.29 shows a screeplot of proportions of variance. Because there are so many components and most of the information is concentrated in the largest dimensions, I've used a `log10()` scale on the horizontal axis. Beyond 10 or so dimensions, the variance of additional components looks quite tiny.

```
ggscreeplot(img_pca) +
  scale_x_log10()
```

Then, if \mathbf{M} is the 640×955 matrix of pixel values, a best approximation $\widehat{\mathbf{M}}_k$ using k dimensions can be obtained as $\widehat{\mathbf{M}}_k = \mathbf{X}_k \mathbf{V}_k^T$ where \mathbf{X}_k are the principal component scores and \mathbf{V}_k are the eigenvectors corresponding to the k largest eigenvalues. The function `approx_pca()` does this, and also undoes the scaling and centering carried out in PCA.

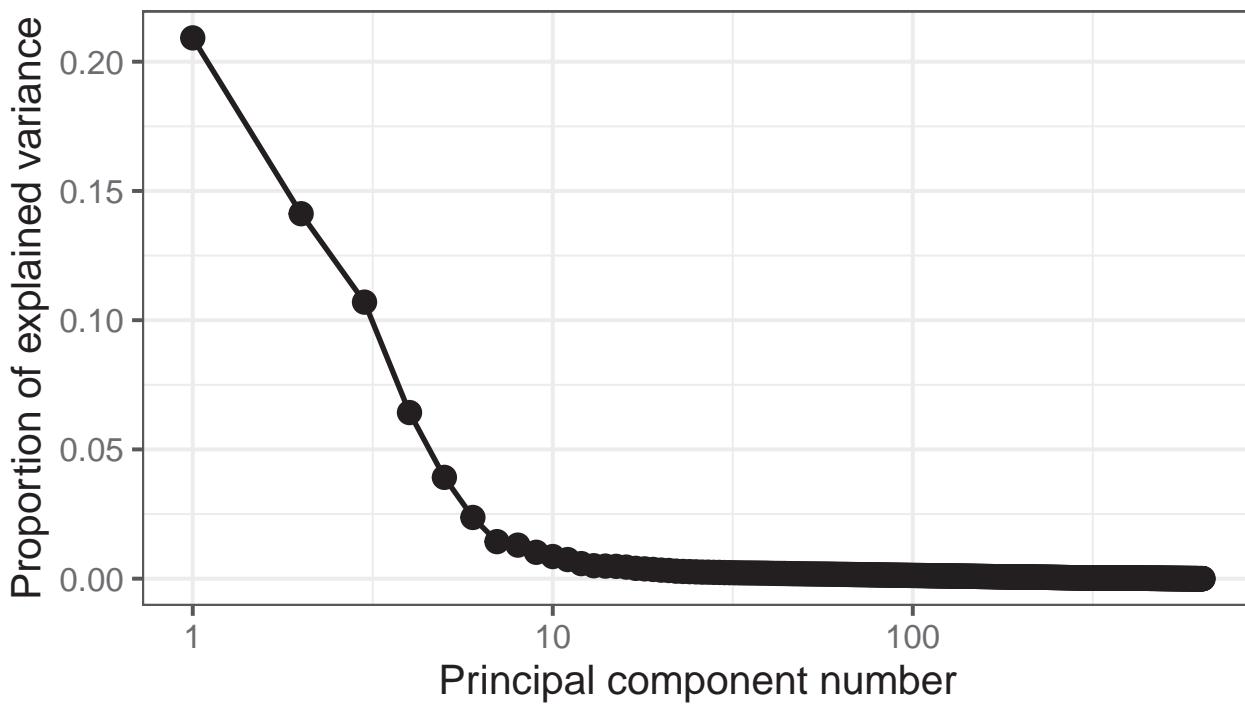


Figure 4.29: Screeplot of the variance proportions in the Mona Lisa PCA.

```

approx_pca <- function(n_comp = 20, pca_object = img_pca){
  ## Multiply the matrix of rotated data (component scores) by the transpose of
  ## the matrix of eigenvectors (i.e. the component loadings) to get back to a
  ## matrix of original data values

  recon <- pca_object$x[, 1:n_comp] %*% t(pca_object$rotation[, 1:n_comp])

  ## Reverse any scaling and centering that was done by prcomp()
  if(all(pca_object$scale != FALSE)){
    ## Rescale by the reciprocal of the scaling factor, i.e. back to
    ## original range.
    recon <- scale(recon, center = FALSE, scale = 1/pca_object$scale)
  }
  if(all(pca_object$center != FALSE)){
    ## Remove any mean centering by adding the subtracted mean back in
    recon <- scale(recon, scale = FALSE, center = -1 * pca_object$center)
  }

  ## Make it a data frame that we can easily pivot to long format
  ## for drawing with ggplot
  recon_df <- data.frame(cbind(1:nrow(recon), recon))
  colnames(recon_df) <- c("x", 1:(ncol(recon_df)-1))

  ## Return the data to long form
  recon_df_long <- recon_df |>
    tidyverse::pivot_longer(cols = -x,
                           names_to = "y",

```

```

            values_to = "value") |>
mutate(y = as.numeric(y)) |>
arrange(y) |>
as.data.frame()

recon_df_long
}

```

Finally, the recovered images, using 2, 3 , 4, 5, 10, 15, 20, 50, and 100 principal components can be plotted using ggplot. In the code below, the `approx_pca()` function is run for each of the 9 values specified by `n_pcs` giving a data frame `recovered_imgs` containing all reconstructed images, with variables `x`, `y` and `value` (the greyscale pixel value).

```

n_pcs <- c(2:5, 10, 15, 20, 50, 100)
names(n_pcs) <- paste("First", n_pcs, "Components", sep = "_")

recovered_imgs <- map_dfr(n_pcs,
                           approx_pca,
                           .id = "pcs") |>
mutate(pcs = stringr::str_replace_all(pcs, "_", " "),
       pcs = factor(pcs, levels = unique(pcs), ordered = TRUE))

```

In `ggplot()`, each is plotted using `geom_raster()`, using `value` to as the fill color. A quirk of images imported to R is that origin is taken as the upper left corner, so the Y axis scale needs to be reversed. The 9 images are then plotted together using `facet_wrap()`.

```

p <- ggplot(data = recovered_imgs,
             mapping = aes(x = x, y = y, fill = value))
p_out <- p + geom_raster() +
scale_y_reverse() +
scale_fill_gradient(low = "black", high = "white") +
facet_wrap(~ pcs, ncol = 3) +
guides(fill = "none") +
labs(title = "Recovering Mona Lisa from PCA of her pixels") +
theme(strip.text = element_text(face = "bold", size = rel(1.2)),
      plot.title = element_text(size = rel(1.5)))

p_out

```

The result, in Figure 4.30 is instructive about how much visual information is contained in lower-dimensional reconstructions, or conversely, how much the image can be compressed by omitting the many small dimensions.

In this figure, with 4-5 components most people would recognize this as a blurry image of the world's most famous portrait. It is certainly clear that this is the Mona Lisa with 10–15 components. Details of the portrait and background features become recognizable with 20–50 components, and with 100 components it compares favorably with the original in Figure 4.28. In numbers, the original 640×955 image is of size 600 Kb. The 100 component version is only 93 Kb, 15.6% of this.

4.7 Elliptical insights: Outlier detection

The data ellipse (Section 3.2), or ellipsoid in more than 2D is fundamental in regression. But, as Pearson showed, it is also key to understanding principal components analysis, where the principal component directions are simply the axes of the ellipsoid of the data. As such, observations that are unusual in data space may not stand out in univariate views of the variables, but will stand out in principal component space, usually on the *smallest* dimension.

As an illustration, I created a dataset of $n = 100$ observations with a linear relation, $y = x + \mathcal{N}(0, 1)$ and then added two discrepant points at $(1.5, -1.5)$, $(-1.5, 1.5)$.

```
set.seed(123345)
x <- c(rnorm(100), 1.5, -1.5)
y <- c(x[1:100] + rnorm(100), -1.5, 1.5)
```

When these are plotted with a data ellipse in Figure 4.31 (left), you can see the discrepant points labeled 101 and 102, but they do not stand out as unusual on either x or y . The transformation to from data space to principal components space, shown in Figure 4.31 (right), is simply a rotation of (x, y) to a space whose coordinate axes are the major and minor axes of the data ellipse, (PC_1, PC_2) . In this view, the additional points appear a univariate outliers on the smallest dimension, PC_2 .

To see this more clearly, `?@fig-outlier-animation` shows an animation of the rotation from data space to PCA space. This uses `heplots::interpPlot()` to interpolate linearly from the positions of the points in data space to their locations in PCA space.

4.7.1 Example: Penguin data

In Section 3.9.2 we examined the questions of multivariate normality and outliers for the penguin data. From a χ^2 QQ plot (Figure 3.26) of the Mahalanobis D^2 values, three Penguins (283, 10, 35) were identified as noteworthy, deserving a closer look to see why they are unusual. It was pointed out (Figure 3.27) that 2D plots of the data variables were only partially revealing. Let's see where they appear in biplots.

First, I find the noteworthy points with the three the largest D^2 values as before:

```
data(peng, package="heplots")

# find potential multivariate outliers
DSQ <- heplots::Mahalanobis(peng[, 3:6])
noteworthy <- order(DSQ, decreasing = TRUE)[1:3] |> print()
#> [1] 283 10 35
```

The PCA shows that the first two components account for 88% of variance, so this is probably an adequate representation of the overall structure of our penguins:

```
peng.pca <- prcomp(
  ~ bill_length + bill_depth + flipper_length + body_mass,
  data=peng, scale. = TRUE
)
summary(peng.pca)
#> Importance of components:
#>                               PC1    PC2    PC3    PC4
#> Standard deviation     1.657 0.882 0.6072 0.328
```

```
#> Proportion of Variance 0.686 0.195 0.0922 0.027
#> Cumulative Proportion 0.686 0.881 0.9730 1.000
```

Figure 4.32 gives the biplot for the first two dimensions. It can be seen that:

- PC1 is largely determined by flipper length and body mass. We can interpret this as an overall measure of **penguin size**. On this dimension, Gentoos are the largest, by quite a lot, compared with Adelie and Chinstrap.
- PC2 is mainly determined by variation in the two beak variables: bill length and depth. Chinstrap are lower than the other two species on bill length and depth, but bill length further distinguishes the Gentoos from the Adelies. A penguin biologist could almost certainly provide an explanation, but I'll call this **beak shape**.
- But, our three suspected outliers are well-within the bulk of their species.

That's the point of this exercise. The projection of the data into the space that accounts for the greatest **total** variance usually does not reveal a few unusual points.

```
source("R/penguin/penguin-colors.R")
# create vector of labels, blank except for the noteworthy
lab <- 1:nrow(peng)
lab <- ifelse(lab %in% noteworthy, lab, "")

ggbiplot(peng.pca,
  choices = 1:2,
  groups = peng$species,
  ellipse = TRUE, ellipse.alpha = 0.1,
  circle = TRUE,
  var.factor = 1,
  geom.ind = c("point", "text"),
  point.size = 1,
  labels = lab, labels.size = 6,
  varname.size = 5,
  clip = "off") +
  theme_minimal(base_size = 14) +
  theme_penguins("dark") +
  scale_shape_discrete() +
  theme(legend.direction = 'horizontal', legend.position = 'top')
```

Now, plotting dimensions 3 and 4 gives Figure 4.33. Dimension 3, accounting for 9%, is largely determined by a contrast of bill length with bill depth and body mass, while dimension 4 involves a contrast between body mass and flipper length. The Chinstraps here have the longest, straightest beaks.

Recall that the perpendicular projection of observation i on the vector for variable j gives an approximation of \hat{x}_{ij} shown in that space. Our friend Cyrano (case 283), the only true multivariate outlier, lies at the extreme ends of both dimensions, with his exceptionally long, straight bill. Case 10 (Hook Nose) stands out at the high end of dimension 3 with a highly curved beak. case 35 is at the high end of dimension 4, so probably is much heavier than most and has short flippers.

4.8 What have we learned?

Welcome to the world of **multivariate juicers**—those magical tools that squeeze the most meaningful information from high-dimensional data clouds! This chapter has taken us on a journey from Flatland to Spaceland, revealing how dimension reduction methods can transform overwhelming complexity into interpretable insights.

- **PCA is your geometric friend, helping you compress N-dimensional data:** Principal Components Analysis finds the directions of maximum variance in your data, creating uncorrelated orthogonal dimensions that capture the most “juice” from your multivariate cloud. Think of it as finding the best viewpoint to see a 3D sculpture when you can only look at a 2D photograph—PCA rotates and compresses your data to show you the best 2D viewing angle.
- **Biplots are visualization gold, helping you view compressed N-dimensional data:** These elegant displays build off of PCA by simultaneously showing both your observations (as points) and your variables (as vectors) in the same reduced space. The magic lies in the interpretation: variable vectors pointing in similar directions are correlated, and you can read approximate values by projecting points onto variable vectors. It’s like having X-ray vision for multivariate relationships!
- **Eigenvalues tell the variance story:** The screeplot becomes your guide for deciding how many dimensions to keep. Look for the “elbow” where the eigenvalues start to resemble scree (rubble) rather than meaningful signal. Generally, 80-90% cumulative variance gives you a solid foundation for interpretation.
- **Supplementary variables enhance interpretation:** Once you’ve found your reduced-dimension view, you can project additional variables into the space to aid interpretation—like adding helpful annotations to a map. This technique bridges the gap between statistical discovery and domain knowledge.
- **Nonlinear methods reveal hidden structures:** When relationships aren’t linear, techniques like multidimensional scaling (MDS) and t-SNE can uncover patterns that PCA might miss. These methods focus on preserving local neighborhoods and distances rather than global variance, often revealing clusters and nonlinear manifolds lurking in your data.
- **Variable ordering creates visual clarity:** When similar variables are placed adjacent to each other, patterns emerge and anomalies become visible—it’s like organizing a messy bookshelf by subject. In biplots, the angles of variable vectors provide a natural ordering that are used to make correlation matrices and other displays much more interpretable.
- **Outlier detection gets multidimensional power:** Points that seem normal in individual variables can reveal themselves as true multivariate outliers when viewed in principal component space, especially along the smallest dimensions. The data ellipse becomes your guide to understanding what’s typical versus what deserves a closer look.
- **Real applications abound:** From compressing the Mona Lisa using eigenfaces to understanding crime patterns across U.S. states, dimension reduction methods bridge the gap between statistical technique and practical insight. These aren’t just mathematical curiosities—they’re essential tools for making sense of our increasingly high-dimensional world.

Recovering Mona Lisa from PCA of her pixels

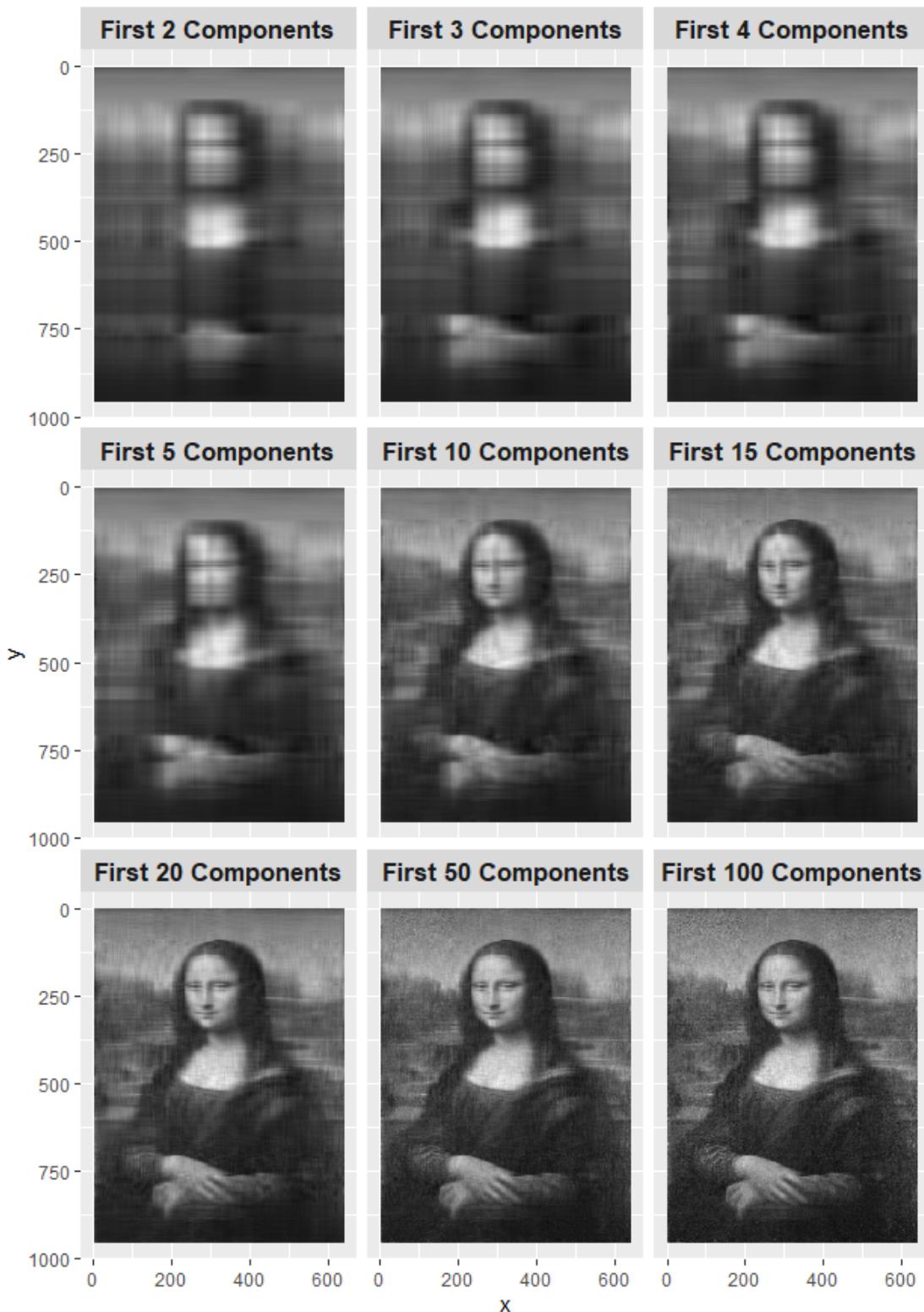


Figure 4.30: Re-construction of the Mona Lisa using 2, 3 , 4, 5, 10, 15, 20, 50, and 100 principal components.

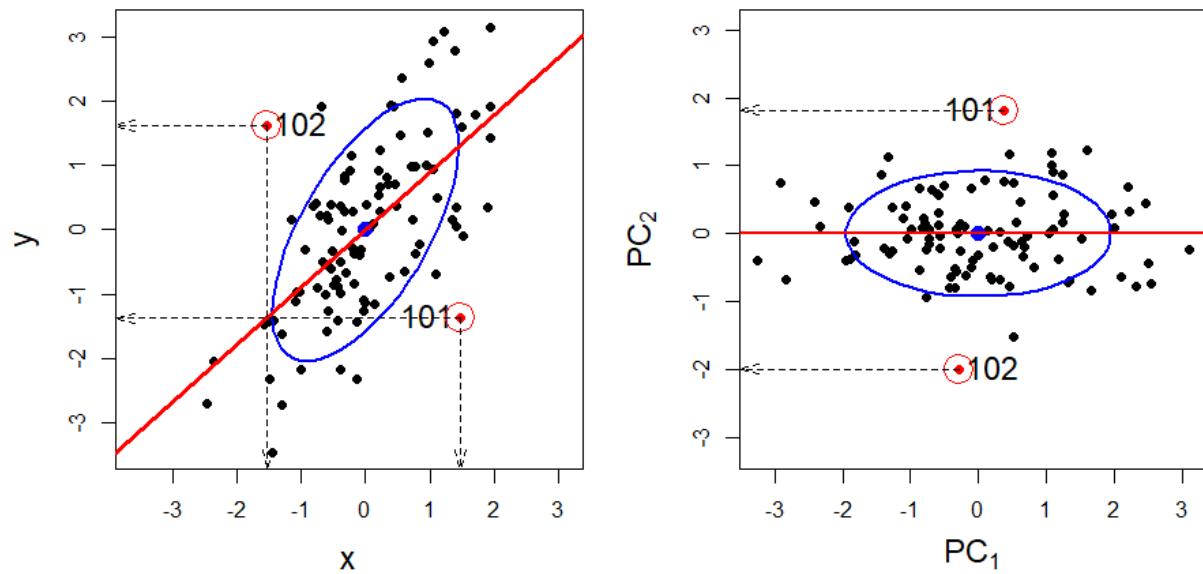


Figure 4.31: Outlier demonstration: The left panel shows the original data and highlights the two discrepant points, which do not appear to be unusual on either x or y . The right panel shows the data rotated to principal components, where the labeled points stand out on the smallest PCA dimension.

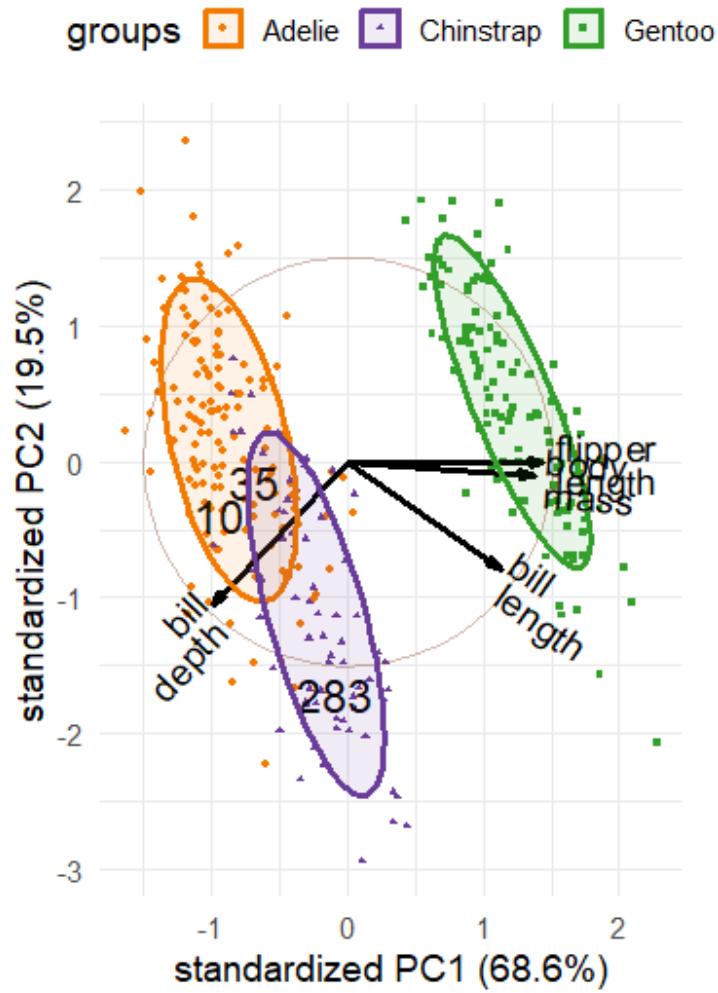


Figure 4.32: Biplot of the first two dimensions of the Penguin data. The points for the three noteworthy cases are labeled, but none of these appear to be unusual in this view.

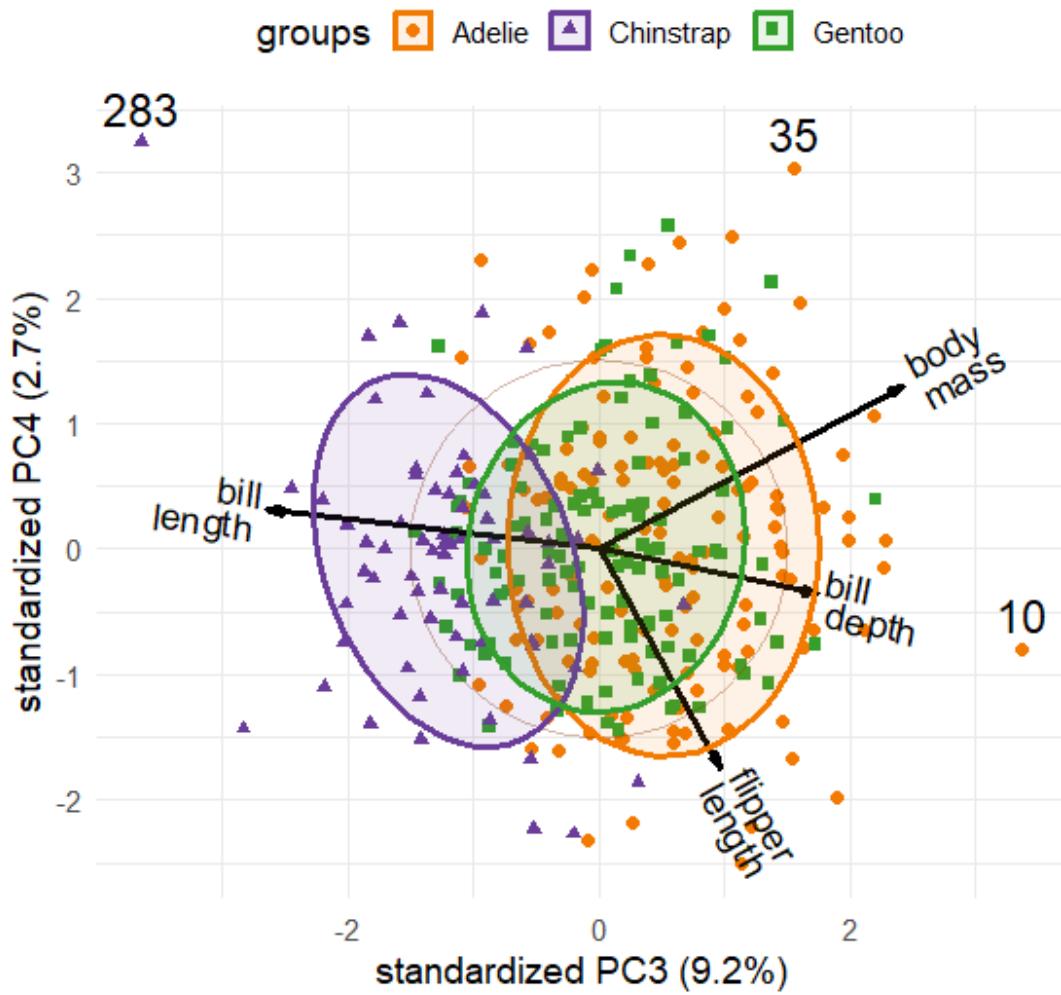


Figure 4.33: Biplot of dimensions 3-4 of the Penguin data. The three noteworthy birds stand out in this view.

Part III

Univariate Linear Models

5

Overview of Linear models

Although this book is primarily about multivariate models, it is useful to have an overview of the range of available techniques for univariate response models to see their uses and to appreciate how easily univariate models generalize to multivariate ones. Hence, this chapter reviews the characteristics of the standard univariate methods for explaining or predicting a single outcome variable from a set of predictors.

The key ideas are:

- For a single quantitative outcome variable \mathbf{y} , methods of linear regression and analysis of variance (ANOVA) are comprised within a single framework of the **general linear model** (GLM). Regression and ANOVA differ only in that the predictors in the former are quantitative, while those in the latter are discrete factors. They can all be fit using `lm()`.
- These models extend directly to the multivariate case of $q > 1$ outcomes, $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q)$, and are also fit using `lm()`.
- A binary outcome $y = (0, 1)$ and categorical outcomes like marital status (“never married”, “married”, “separated”, “divorced”) can be handled within a different extension, the **generalized** linear model as logistic regression or multinomial regression, fit using `glm()`.
- All of these models involve linear combinations of predictors (weighted sums) fit to optimize some criterion, for example minimizing some function of the residuals or maximizing some measure of fit.
- Models and data can be more easily understood with graphics, and many statistical ideas have a visual representation in geometry.

Figure 5.1 summarizes a variety of methods for linear models, classified by number of predictors and number of response variables, and whether these are quantitative vs. discrete. For the present purposes, the key columns are the first two, for the case of one or more quantitative outcome variables.

When the *predictors* are also quantitative, simple regression ($p = 1$) generalizes to multivariate regression with two or more outcomes ($q > 1$). For example we might want to predict weight and body mass index *jointly* from a person’s height.

The situation is more interesting when there are $p > 1$ predictors. The most common multivariate generalization is *multivariate multiple regression* (MMRA), where each outcome is regressed on the predictors, as if done separately for each outcome, but using multivariate tests that take correlations among the predictors into account. Other methods for this case include canonical correlation analysis, which tries to explain all relations between \mathbf{Y} and a set of \mathbf{x} s through maximally correlated linear combinations of each.

When the predictor variables are all discrete or categorical, such as gender or level of education, methods like the simple *t*-test, one-way ANOVA and factorial ANOVA with $q = 1$ outcome measures all have simple extensions to the case of $q > 1$ outcomes.

History Corner

Why are there so many different names for regression vs. ANOVA concepts, statistics and techniques? In regression, we use notation like x_1, x_2, \dots to refer to *predictors* in a model, while in ANOVA, factors A, B, \dots are called *main effects*. In regression applications, we often test *linear hypotheses*, are interested

Response variables: $\mathbf{Y} = (y_1, \dots y_q)$

	Quantitative		Discrete	
	$q=1$	$q>1$	$q=1$	$q>1$
Quantitative	Simple regression	Multivariate regression	Simple logistic regression	
	Multiple regression	Multivariate regression Canonical corr. Partial corr.	Mult. logistic regression Discriminant analysis	Multivariate logistic regression
Discrete	t-test 1-way ANOVA	Hotelling T ² 1-way MANOVA	Simple χ^2	Loglinear models
	Factorial ANOVA	Factorial MANOVA	Logit models Loglinear models	Loglinear models

Figure 5.1: Techniques for linear models classified by number of predictors and number of response variables, and whether these are quantitative vs. discrete

in *coefficients* and evaluate a model with an R^2 statistic, while in ANOVA we may test *contrasts* among factor levels, and use F -tests to evaluate models.

Well, like twins separated at birth, they grew up in homes in different places and with different parents, who were each free to choose their own names, not recognizing their shared DNA.

Methods of regression began in evolutionary biology with Francis Galton's (1886, 1889) studies of heritability of traits, trying to understand how strongly the physical characteristics of one generation of living things resembled those in the next generation. From a study of the diameters of sweet peas in parent plants and their size in the next generation, and another on the relationship between heights of human parents and their offspring, he developed the fundamental ideas of regression. Karl Pearson (1896)

In contrast, analysis of variance methods were raised on farms, notably the Rothamsted Experimental Station, where R. A. Fisher analyzed vast amounts of data on crop experiments designed to determine the conditions (soil condition, fertilizer treatments) that gave the greatest yields, while controlling for extraneous determiners (plots of planting). With multiple factors determining the outcome, Fisher (1923), in an experiment on yields of different varieties of potatoes given various manure treatments, devised the method of breaking down the total variance into portions attributable to each factor and presented the first ANOVA table. The method became well-known after Fisher's (1925b) *Statistical Methods for Research Workers*.

The great synthesis of regression and ANOVA did not take place until the 1960s. At that time, methods for computing were beginning to move from programmable desk calculators to mainframe computers, largely using a collection of separate FORTRAN programs, designed for regression, one-way ANOVA, two-way ANOVA, models with interactions, and so forth. To complete an analysis, as a graduate student I often had to use three or more different programs.

Then, something remarkable happened on two fronts: theory and computation. First, in quick succession

textbooks by Scheffé (1960), Graybill (1961), Winer (1962) ... began to layout a general theory of linear models that encompassed all of these separate models, giving the “General Linear Model” a well-deserved name. Second, two symposiums, one at IBM Yorkdown Heights (IBM (1965)) and the other at the University of Georgia (Bashaw & Findley (1967)) resulted in the first general programs to handle all these cases in an understandable way.

A bit of matrix algebra thrown into the mix showed that most of the ideas for univariate models could be extended to multiple response variables, and so the “Multivariate Linear Model” was born. R. Darrell Bock (Bock, 1963, 1964) sketched a flowchart of the computational steps, which was implemented at the University of Chicago by Jeremy Finn (1967) in the MULTIVARIANCE program. A group at the University of North Carolina headed by Elliot Cramer developed their MANOVA program (Clyde et al., 1966) and Willard Dixon (1965) at UCLA developed the BMD programs incorporating these ideas. The ANOVA and regression twins had finally become part of a larger family.

Packages

In this chapter I use the following packages. Load them now:

```
library(ggplot2)
```

TODO: This stuff on linear combinations seems out of place here. Where to move it?

5.1 The General Linear Model

To establish notation and terminology, it is worthwhile to state the the general linear model formally. For convenience, I use vector and matrix notation. This expresses a response variable, $\mathbf{y} = (y_1, y_2, \dots, y_n)^T$ for n observations, as a sum of terms involving p regressors, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$, each of length n .

$$\begin{aligned}\mathbf{y} &= \beta_0 + \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \cdots + \beta_p \mathbf{x}_p + \epsilon \\ &= [\mathbf{1}, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p] \boldsymbol{\beta} + \epsilon\end{aligned}$$

{#eq-glm}

or, expressed in matrices,

$$\mathbf{y}_{n \times 1} = \mathbf{X}_{n \times (p+1)} \boldsymbol{\beta}_{(p+1) \times 1} + \epsilon$$

The matrix \mathbf{X} is called the *model matrix* and contains the numerical representations of the predictor variables called *regressors*. The essential thing about a linear model is that it is linear in the **parameters** β_i . That is, the predicted value of \mathbf{y} is a linear combination of some \mathbf{x}_i with weights β_i . An example of a nonlinear model is the exponential growth model, $y = \beta_0 + e^{\beta_1 x}$, where the parameter β_1 appears as an exponent.¹

- These can be quantitative variables like `age`, `salary` or `years` of education. But they can also be transformed versions, like `sqrt(age)` or `log(salary)`.
- A quantitative variable can be represented by more than one model regressor, for example if it is expressed as a polynomial like `poly(age, degree=2)` or a natural spline like `ns(salary, df=5)`. The model matrix portion for such terms contains one column for each degree of freedom (`df`) and there are `df` coefficients in the corresponding portion of $\boldsymbol{\beta}$.

¹Taking logarithms of both sides would yield the linear model, $\log(y) = c + \beta_1 x$.

- A categorical or discrete predictor– a factor variable in R– with d levels is expressed as $d - 1$ columns in **X**. Typically these are contrasts or comparisons between a baseline or reference level and each of the remaining ones, but any set of $d - 1$ linearly independent contrasts can be used by assigning to **contrasts(factor)**. For example, **contrasts(factor) <- contr.treatment(4)** for a 4-level factor assigns 3 contrasts representing comparisons with a baseline level, typically the first (in alphabetic order). For an *ordered* factor, such as one for political knowledge with levels “low”, “medium”, “high”, **contrasts.poly()** returns the coefficients of orthogonal polynomial contrasts representing linear and quadratic trends.
- Interactions between predictors are represented as the direct products of the corresponding columns of **X**. This allows the effect of one predictor on the response to depend on values of other predictors. For example, the interaction of two quantitative variables, **x₁, x₂** is represented by the product **x₁ × x₂**. More generally, for variables or factors *A* and *B* with degrees of freedom df_A and df_B the regressors in **X** are the $df_A \times df_B$ products of each column for *A* with each column for *B*.

5.1.1 Model formulas

Statistical models in R, such as those fit by **lm()**, **glm()** and many other modelling function in R are expressed in a simple notation that was developed by Wilkinson & Rogers (1973) for the GENSTAT software system at the Rothamsted Research Station. It solves the problem of having a compact way to specify any model consisting of any combinations of quantitative and discrete factor variables, interactions of these and arbitrary transformations of these.

In this, a **model formula** take the forms

```
response ~ terms
response ~ term1 + term2 + ...
```

where the left-hand side, **response** specifies the response variable in the model and the right-hand side specifies the **terms** in the model specifying the columns in the **X** matrix of **?@eq-glm**; the coefficients β are implied and not represented explicitly in the formula.

The notation **y ~ x** is read as “**y is modeled by x**”. The left-hand side is usually a variable name (such as **height**), but it could be an expression that evaluates to the the response, such as **log(salary)** or **weight/height^2** which represents the body mass index.

On the right-hand side (RHS), the usual arithmetic operator, **+**, **-**, *****, **/**, **^** have special meanings as described below. The most fundamental is that **y ~ a + b** is interpreted as “**y is modeled by a and b**”; that is, the sum of linear terms for **a** and **b**.

Some examples for regression-like models using only quantitative variables, **x**, **x₁**, **x₂**, **x₃**, ... are shown below:

```
y ~ x                      # simple linear regression
y ~ x - 1                   # no intercept: regression through the origin
y ~ x + I(x^2)              # quadratic model
y ~ poly(x, 3)              # cubic model
y ~ x1 * x2                 # crossing: x1 + x2 + x1 : x2
y ~ x1 + x2 + x3            # multiple regression
y ~ (x1 + x2 + x3)^2        # response surface: all quadratics & two-way interactions
log(y) ~ x1 + poly(x, 2)     # arbitrary transformation of response
y1 + y2 ~ x1 + x2 + x3      # response is sum of y1 and y2
```

The intercept β_0 is automatically included in the model without need to specify it explicitly. The minus sign, **-** on the right-hand side removes terms from the model, so a model with no intercept $\beta_0 = 0$ can be specified as **y ~ X -1** (or perhaps more naturally, **y ~ 0 + X**).

Function calls on the RHS, such as `poly(x, 3)` are evaluated directly, but to use a special model operator, like `^` must be “protected” by wrapping the term in `I()`, meaning “identity” or “inhibit”. Thus, the model $y \sim x + I(x^2)$ means the quadratic model $y = \beta_0 + \beta_1 x + \beta_2 x^2$. This differs from the model $y \sim \text{poly}(x, 2)$ in that the former uses the raw x , x^2 values (which are necessarily positively correlated) while `poly()` converts these to orthogonal polynomial scores, which are uncorrelated (and therefore free from problems of collinearity).

Example 5.1. Workers data: Regression models

For the `workers` data (Section 4.2.2.1) you can fit simple regression models predicting income from years of experience using a linear and quadratic model as follows:

```
data(workers, package = "matlib")
workers.mod1 <- lm(Income ~ Experience, data=workers)
coef(workers.mod1) |> t() |> t()
#>           [,1]
#> (Intercept) 29.16
#> Experience   1.12

workers.mod2 <- lm(Income ~ poly(Experience, 2), data=workers)
coef(workers.mod2) |> t() |> t()
#>           [,1]
#> (Intercept)      46.5
#> poly(Experience, 2)1 39.1
#> poly(Experience, 2)2 -11.2
```

It is simplest to understand these models by plotting the data overlaid with the fitted regressions. This uses `geom_smooth()` and specifies the smoothing model as `method = "lm"` with a `formula`, which is $y \sim x$ for the linear model and $y \sim \text{poly}(x, 2)$ for the quadratic.

```
ggplot(data = workers,
       aes(x = Experience, y = Income)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", formula = y ~ x,
              se = FALSE, linewidth = 2,
              color = "blue") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2),
              se = FALSE, linewidth = 2,
              color = "red")
```

The coefficients of the linear model are also easy to interpret:

$$\widehat{\text{Income}} = 29.162 + 1.119(\text{Experience}) \quad (5.1)$$

So a worker with zero years of experience can expect an income of \$29162 and this should increase by \$1119 for each additional year. However, it is not so simple to interpret the coefficients when a `poly()` term is used. Naively plugging in the coefficients for `workers.mod2` gives

$$\widehat{\text{Income}} = 46.5 + 39.111(\text{Experience}) - 11.16(\text{Experience}^2) \quad (5.2)$$

The problem is that $x = \text{Experience}$ in model `workers.mod` is represented not by the raw values, but rather by values of x and x^2 that have been made to be uncorrelated. If you really want to interpret the coefficient values in terms of years of experience, use the option `raw = TRUE` in `poly()`:

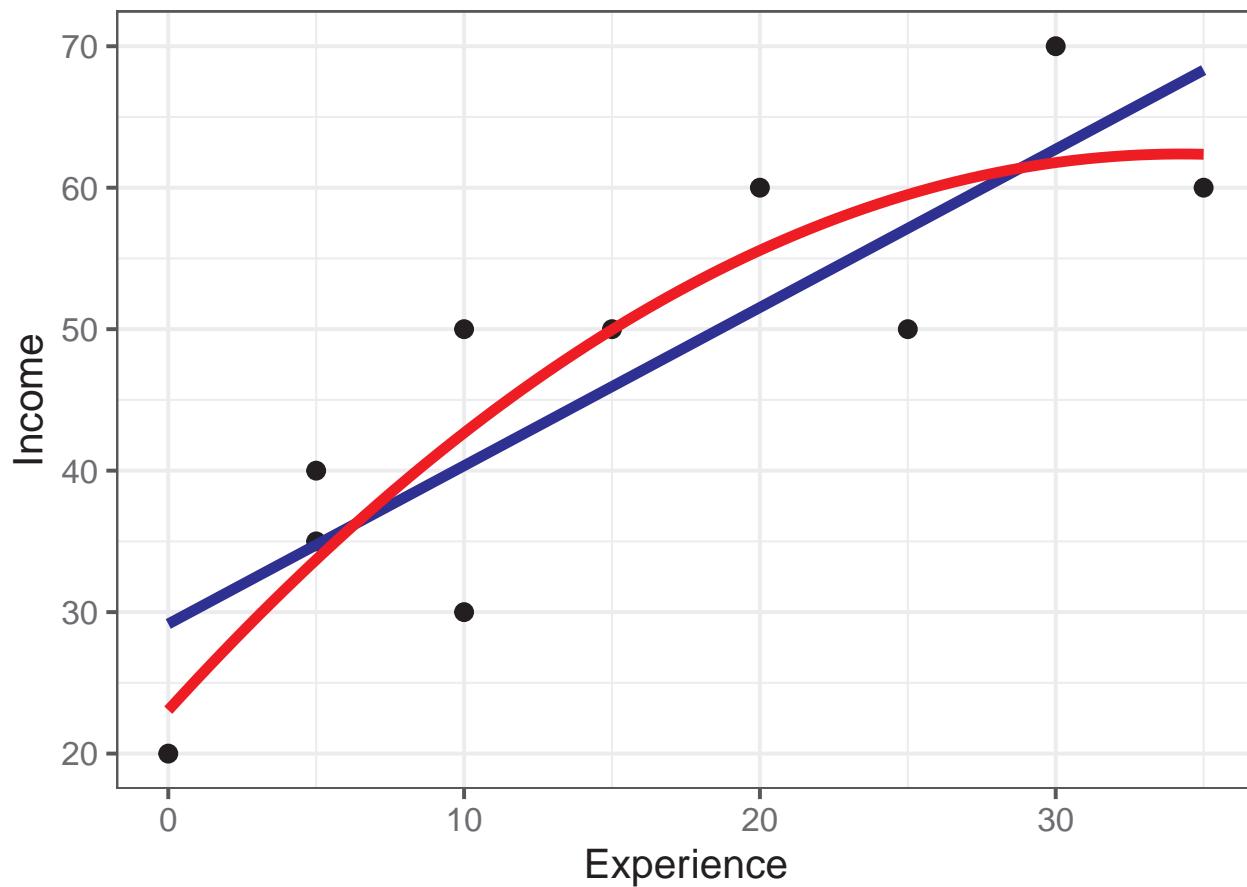


Figure 5.2: Workers data with fitted linear and quadratic models for years of experience.

```
workers.mod2b <- lm(Income ~ poly(Experience, 2, raw = TRUE),
                      data=workers) |>
  print()
#>
#> Call:
#> lm(formula = Income ~ poly(Experience, 2, raw = TRUE), data = workers)
#>
#> Coefficients:
#>              (Intercept)  poly(Experience, 2, raw = TRUE)1
#>                  23.0680                      2.2952
#> poly(Experience, 2, raw = TRUE)2
#>                  -0.0335
```

$$\widehat{\text{Income}} = 23.07 + 2.3(\text{Experience}) - 0.03(\text{Experience}^2)$$

This says that income is predicted to be \$23,068 with no experience, increase initially by \$2295, but that yearly increase decreases by \$330. Some further details of orthogonal polynomials are explained below.

5.1.1.1 Factors

Factor variables are treated specially in linear models, but have simple notations in R formulas. The following examples use A, B, C to represent discrete factors with two or more levels.

```

y ~ A                      # one-way ANOVA
y ~ A + B                  # two-way, main effects only
y ~ A * B                  # full two-way, with interaction
y ~ A + B + A:B            # same, in long-hand
y ~ x + A                  # one-way ANCOVA
y ~ (A + B + C)^2          # three-way ANOVA, incl. all two-way interactions

```

5.1.1.2 Crossing

The `*` operator has special meaning used to specify the crossing of variables and factors and `:` specifies interactions (products of variables). So, the model `y ~ x1 * x2` is expanded to give `y ~ x1 + x2 + x1:x2` and the interaction term `x1:x2` is calculated as $x_1 \times x_2$. In algebraic notation (omitting the error term) this works out to the model,

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_1 x_1 * \beta_2 x_2 \\ &= \beta_0 + (\beta_1 + \beta_2 x_2)x_1 + \beta_2 x_2 , \end{aligned}$$

which means that the coefficient for x_1 in the model is not constant for all values of x_2 , but rather changes with the value of x_2 . If $\beta_2 > 0$, the slope for x_1 increases with x_2 and vice-versa.

`y ~ A * B` for factors is similar, expanding to `y ~ A + B + A:B`, but the columns in the model matrix represent contrasts among the factor levels as described in detail below (Section 5.1.3). The main effects, `A` and `B` come from contrasts among the means of their factor levels and the interaction term `A:B` reflects *differences* among means of `A` across the levels of `B` (and vice-versa).

The model formula `y ~ x + A` specifies an ANCOVA model with different intercepts for the levels of `A`, but with a common slope for `x`. Adding an interaction of `x:A` in the model `y ~ x * A` allow separate slopes and intercepts for the groups.

5.1.1.3 Powers

The `^` exponent operator indicates *powers of a term expression* to a specified degree. Thus the term `(A + B)^2` is identical to `(A + B) * (A + B)` which expands to the main effects of `A`, `B` and their interaction, also identical to `A * B`. In general, the product of parenthesized terms expands as in ordinary algebra,

```
y ~ (A + B) * (C + D) -> A + B + C + D + A:C + A:D + B:C + B:D
```

Powers get more interesting with more terms, so `(A + B + C)^2` is the same as `(A + B + C) * (A + B + C)`, which includes main effects of `A`, `B` and `C` as well as all two-way interactions, `A:B`, `A:C` and `B:C`. The model formula `(A + B + C)^3` expands to include all two-way interactions and the three-way interaction `A:B:C`.

```
(A + B + C)^3 -> A + B + C + A:B + A:C + B:C + A:B:C
```

In this context `-` can be used to remove terms, as shown in the following examples

```

(A + B + C)^2 <-> (A + B + C)^3 - A:B:C
(A + B + C)^3 - B:C - A:B:C <-> A + B + C + A:B + A:C

```

Finally, the symbol `.` on the right-hand side specifies *all terms* in the current dataset other than the response. Thus if you have a data.frame containing `y`, `x1`, `x2`, ..., `x6`, you can specify a model with all variables except `x6` as predictors as

```
y ~ . - x6
```

To test what we've covered above,

- What do you think the model formula $y \sim .^2$ means in a data set containing variables x_1 , x_2 , x_3 , and x_4 ?
- What about the formula with $y \sim .^2 - A:B:C:D$ with factors A , B , C , D ?

You can work out questions like these or explore model formulae using `terms()` for a "formula" object. The labels of these terms can then be concatenated to a string and turned back into a formula using `as.formula()`:

```
f <- formula(y ~ (x1 + x2 + x3 + x4)^2)
terms = attr(terms(f), "term.labels")

terms |> paste(collapse = " + ")
#> [1] "x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x1:x4 + x2:x3 + x2:x4 + x3:x4"
# convert back to a formula
as.formula(sprintf("y ~ %s", paste(terms, collapse=" + ")))
#> y ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x1:x4 + x2:x3 + x2:x4 +
#>     x3:x4
```

5.1.2 Model matrices

As noted above, a model formula is used to generate the $n \times (p + 1)$ model matrix, \mathbf{X} , typically containing the column of 1s for the intercept β_0 in the model, followed by p columns representing the regressors x_1, x_2, \dots, x_p . Internally, `lm()` uses `stats::model.matrix()` and you can use this to explore how factors, interactions and other model terms are represented in a model object.

For a small example, here are a few observations representing income (`inc`) and type of occupation, taking on values `bc` (blue collar), `wc` (white collar) and `prof` (professional). `model.matrix()` takes a *one-sided* formula with the terms on the right-hand side. The main effect model looks like this:

```
set.seed(42)
inc <- round(runif(n=9, 20, 40))
type <- rep(c("bc", "wc", "prof"), each =3)

mm <- model.matrix(~ inc + type)
data.frame(type, mm)
#>   type X.Intercept. inc typeprof typewc
#> 1   bc          1  38      0      0
#> 2   bc          1  39      0      0
#> 3   bc          1  26      0      0
#> 4   wc          1  37      0      1
#> 5   wc          1  33      0      1
#> 6   wc          1  30      0      1
#> 7 prof         1  35      1      0
#> 8 prof         1  23      1      0
#> 9 prof         1  33      1      0
```

As you can see, `type`, with 2 degrees of freedom is represented by two **dummy** (0/1) variables, labeled `typeprof` and `typewc` here. Together, these represent *treatment contrasts* (comparisons) between the *baseline* group `type=="bc"`, which is coded (0, 0) and each of the others: `type=="prof"`, coded (1, 0) and `type=="wc"`, codes (0, 1). Different coding schemes are described in the following section.

In a model with the interaction `inc * type`, additional columns are constructed as the **product** of `inc` with each of the columns for `type`. We will see below how this generalizes to an arbitrary number of predictor terms and their possible interactions.

```
model.matrix(~ inc * type)
#>   (Intercept) inc typeprof typewc inc:typeprof inc:typewc
#> 1          1   38      0     0        0        0
#> 2          1   39      0     0        0        0
#> 3          1   26      0     0        0        0
#> 4          1   37      0     1        0       37
#> 5          1   33      0     1        0       33
#> 6          1   30      0     1        0       30
#> 7          1   35      1     0       35        0
#> 8          1   23      1     0       23        0
#> 9          1   33      1     0       33        0
#> attr("assign")
#> [1] 0 1 2 2 3 3
#> attr("contrasts")
#> attr("contrasts")$type
#> [1] "contr.treatment"
```

5.1.3 Coding factors and contrasts

Discrete explanatory variables, such as race, type of occupation or level of education require special attention in linear models because, unlike continuous variables, they cannot be entered into the model equation just as they are. Instead, we need some way to code those variables numerically.

A key insight is that your choice of a coding scheme changes the meaning of the model parameters, and allows you to perform different comparisons (test different statistical hypotheses) about the means of the category levels according to meaningful questions in your research design. For a more general discussion of coding schemes, see Fox & Weisberg (2018a), sec. 4.7 and the vignette [Coding Matrices, Contrast Matrices and Linear Models](#) for the `codingMatrices` package.

Each coding scheme for a factor represents the **same** model in terms of fitted values and overall significance for that term, but they differ in how the coefficients are parameterized and interpreted. This is crucial to understand, because tests of the coefficients can directly answer different research questions depending on the coding scheme used.

In R, categorical variables are called **factors** usually created by `g <- factor(g)` or `g <- as.factor(g)` for a discrete variable `g`. If levels of the variable `g` are ordered, such as `type` of occupation with levels "`bc`" < "`wc`" < "`prof`" or dose of a drug, "`low`" < "`medium`" < "`high`", you can use `g <- ordered(g)` to reflect this.

In any case, a factor with k levels is reflected in an overall test with $k - 1$ degrees of freedom corresponding to the null hypothesis $\mathcal{H}_0 : \mu_1 = \mu_2 = \dots = \mu_k$. This can be represented as $k - 1$ comparisons among the factor level means, or $k - 1$ separate questions asking how the means differ.

Base R provides several coding schemes via assignment to the `contrasts()` function for a factor, as in `contrasts(df$Drug) <- ...` one of:

- `contr.treatment()`: Compares each level to a *reference* level using $k - 1$ dummy (0, 1) variables. This is the default, and the reference level is taken as the first (in alphabetic or numerical order). You can change the reference level using `relevel()` or `reorder()` for the factor, or simply using `factor(A, levels = ...)`.
- `contr.sum()`: Compares each level to the grand mean.

- `contr.helmert()`: Compares each level to the mean of the *previous* levels, which is useful for ordered categories such as type of occupation with levels "bc" < "wc" < "prof" or dose of a drug, "low" < "medium" < "high".
- `contr.poly()`: For ordered factors with *numeric* levels, this creates orthogonal polynomial contrasts, representing the linear, quadratic, cubic ... trends in the factor means, as if these appeared as x, x^2, x^3, \dots terms in a model with x as a numeric variable.

TODO: Move some stuff from 10.3.1 on contrasts here

I take up some of the details of these coding schemes below. But first, it is useful to define exactly what I mean by a **contrast**. For a factor with k groups, a contrast is simply a comparison of the mean of one subset of groups against the mean of another subset. This is specified as a weighted sum, L of the means μ with weights \mathbf{c} that sum to zero,

$$L = \mathbf{c}^\top \boldsymbol{\mu} = \sum_i^k c_i \mu_i \quad \text{such that} \quad \sum c_i = 0 .$$

Two contrasts, \mathbf{c}_1 and \mathbf{c}_2 are *orthogonal* if the sum of products of their weights is zero, i.e., $\mathbf{c}_1^\top \mathbf{c}_2 = \sum c_{1i} \times c_{2i} = 0$. When contrasts are placed as columns of a matrix \mathbf{C} , they are all *mutually orthogonal* if each pair is orthogonal, which means $\mathbf{C}^\top \mathbf{C}$ is a diagonal matrix. If the columns of \mathbf{C} are normalized to have sums of squares = 1, then $\mathbf{C}^\top \mathbf{C} = \mathbf{I}$.

Orthogonal contrasts correspond to statistically independent tests. This is nice because they reflect separate, non-overlapping research questions. Another consequence is that the sums of squares for the overall hypothesis of differences among the groups is *exactly decomposed* as the sum of the sum of squares accounted for by the $k - 1$ contrasts L_i :

$$\text{SS}_{\text{group}} = \sum_i^{k-1} \text{SS}(L_i) .$$

Treatment coding

Let's examine R's default coding scheme, `contr.treatment` (also called dummy coding), for a factor with 4 levels: 'a', 'b', 'c', and 'd', with a view to understanding the relationship between the true population means, μ_a, μ_b, μ_c , and μ_d and the parameters β estimated in a linear model. We get the following:

```
C <- contr.treatment(letters[1:4]) |> print()
#>   b  c  d
#> a  0  0  0
#> b  1  0  0
#> c  0  1  0
#> d  0  0  1
```

Here, the columns of \mathbf{C} correspond to three dummy variables for the levels b, c, d compared to the reference level a. If we denote these columns as x_b, x_c , and x_d , then:

$$x_b = \begin{cases} 1 & \text{if factor=b} \\ 0 & \text{otherwise} \end{cases} ; \quad X_c = \begin{cases} 1 & \text{if factor=c} \\ 0 & \text{otherwise} \end{cases} ; \quad X_d = \begin{cases} 1 & \text{if factor=d} \\ 0 & \text{otherwise} \end{cases}$$

The design matrix $\mathbf{X}_{(4 \times 4)} = [\mathbf{1}, \mathbf{C}] = [\mathbf{1}, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d]$ includes the constant column $\mathbf{1}$ representing the intercept, which averages over the factor levels when there are other terms in the model.

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

With this coding, the parameters β in the model are related to the means μ as,

$$\begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_d \end{pmatrix} = \mathbf{X}\beta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \beta_0 \\ \beta_b \\ \beta_c \\ \beta_d \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_0 + \beta_b \\ \beta_0 + \beta_c \\ \beta_0 + \beta_d \end{pmatrix}$$

Thus we have,

$$\begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_d \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_0 + \beta_b \\ \beta_0 + \beta_c \\ \beta_0 + \beta_d \end{pmatrix}$$

Note that \mathbf{X} is non-singular as long as the comparisons in \mathbf{C} are linearly independent. Its inverse, \mathbf{X}^{-1} determines how the transformed parameters relate to the original class means, that is, it determines the *interpretation* of the parameters in terms of the means.

```
X <- cbind(1, C)
Xinv <- solve(X) |> print()
#>      a b c d
#>      1 0 0 0
#> b -1 1 0 0
#> c -1 0 1 0
#> d -1 0 0 1
```

Thus, you can solve for the parameters in terms of the means symbolically as follows

$$\begin{pmatrix} \beta_0 \\ \beta_b \\ \beta_c \\ \beta_d \end{pmatrix} = \mathbf{X}^{-1}\mu = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_d \end{pmatrix} = \begin{pmatrix} \mu_a \\ \mu_b - \mu_a \\ \mu_c - \mu_a \\ \mu_d - \mu_a \end{pmatrix}$$

Deviation coding

Another common coding scheme, useful for unordered factors, is *deviation coding* given by `contrast.sum()`. This has the property that the parameters are constrained to sum to zero, $\sum \beta_i = 0$.

```
C <- contr.sum(letters[1:4]) |> print()
#>      [,1] [,2] [,3]
#> a      1     0     0
#> b      0     1     0
#> c      0     0     1
#> d     -1    -1    -1
```

The parameters estimated with this coding are: With this coding, the intercept is $\beta_0 = \bar{\mu}$, the grand mean across all levels and the parameters are the deviations from that,

- $\beta_1 = \mu_a - \bar{\mu}$
- $\beta_2 = \mu_b - \bar{\mu}$

- $\beta_3 = \mu_c - \bar{\mu}$

A (redundant) coefficient for the last group is omitted, because with this coding a coefficient for that group would be equal to the negative of the sum of the others, $\beta_d = \mu_d - \bar{\mu} = -(\beta_1 + \beta_2 + \beta_3)$.

Helmert contrasts

For **ordered factors**, it is sensible to take the ordering into account in interpreting the results. *Helmert contrasts* are designed for this purpose. The intercept is again the grand mean across all levels. Each contrast compares the mean of a given level to the *average of all previous ones* in the order; they contrast the second level with the first, the third with the average of the first two, and so on.

```
C <- contr.helmert(letters[1:4]) |> print()
#>   [,1] [,2] [,3]
#> a    -1    -1    -1
#> b     1    -1    -1
#> c     0     2    -1
#> d     0     0     3
```

It is easier to understand these if the columns are normalized so that the largest value in each column is 1.

```
C %*% diag(1/c(1, 2, 3)) |> MASS::fractions()
#>   [,1] [,2] [,3]
#> a    -1 -1/2 -1/3
#> b     1 -1/2 -1/3
#> c     0     1 -1/3
#> d     0     0     1
```

Then we would have the coefficients as:

- $\beta_0 = \bar{\mu}$
- $\beta_1 = \mu_b - \mu_a$
- $\beta_2 = \mu_c - \frac{\mu_a + \mu_b}{2}$
- $\beta_3 = \mu_d - \frac{\mu_a + \mu_b + \mu_c}{3}$

Note that you can easily reverse the ordering of the comparisons to contrast each of the first $k - 1$ means with the average of all those *higher* in the order.

```
C.rev <- C[4:1, 3:1]
row.names(C.rev) <- letters[1:4]
C.rev
#>   [,1] [,2] [,3]
#> a     3     0     0
#> b    -1     2     0
#> c    -1    -1     1
#> d    -1    -1    -1
```

Polynomial contrasts

For ordered factors that are also numeric, like `Age = c(8, 9, 10, 11)` or those that can be considered equally spaced along some continuum, polynomial contrasts, given by `contr.poly()`, provide *orthogonal* (uncorrelated) contrasts that assess the degree to which the factor means vary linearly, quadratically, and so on with the factor levels.

`contr.poly()` scales each column so that its sum of squares is 1. Each pair of columns is orthogonal, $\mathbf{c}_i^\top \mathbf{c}_j = 0$, so that $\mathbf{C}^\top \mathbf{C} = \mathbf{I}$.

```
C <- contr.poly(4) |> print()
#>      .L     .Q     .C
#> [1,] -0.671  0.5 -0.224
#> [2,] -0.224 -0.5  0.671
#> [3,]  0.224 -0.5 -0.671
#> [4,]  0.671  0.5  0.224

# show they are orthonormal
t(C) %*% C |> zapsmall()
#>      .L     .Q     .C
#> [1,] 1 0 0
#> [2,] 0 1 0
#> [3,] 0 0 1
```

We can get a better sense of orthogonal polynomial contrasts by taking a numeric vector x , and raising it to successive powers, 1, 2, 3. Here $x^0 = 1$ is the constant term or intercept.

```
M <- outer(1:8, 0:3, `^^`)
colnames(M) <- c("int", "lin", "quad", "cubic")
rownames(M) <- paste0("x", 1:8)
M
#>      int lin quad cubic
#> x1    1   1    1    1
#> x2    1   2    4    8
#> x3    1   3    9   27
#> x4    1   4   16   64
#> x5    1   5   25  125
#> x6    1   6   36  216
#> x7    1   7   49  343
#> x8    1   8   64  512
```

Then we can make the columns of M orthogonal using the Gram-Schmidt method, where each successive column after the first is made orthogonal to all previous columns by subtracting their projections on that column. Plotting these, as in Figure 5.3 shows that the coefficients for a linear term plot as a line, those for x^2 follow a quadratic, and so forth.

```
op <- par(mar = c(4, 4, 1, 1)+.1)
M1 <- matlib::GramSchmidt(M, normalize = FALSE)
matplot(M1,
  type = "b",
  pch = as.character(0:3),
  cex = 1.5,
  cex.lab = 1.5,
  lty = 1,
  lwd = 3,
  xlab = "X",
  ylab = "Coefficient")
```

Custom contrasts

You don't have to be constrained by the kinds of comparisons available in `contr.*` functions. For a factor with k levels you are free to make up *any* $k - 1$ contrasts that correspond to $k - 1$ different questions or tests

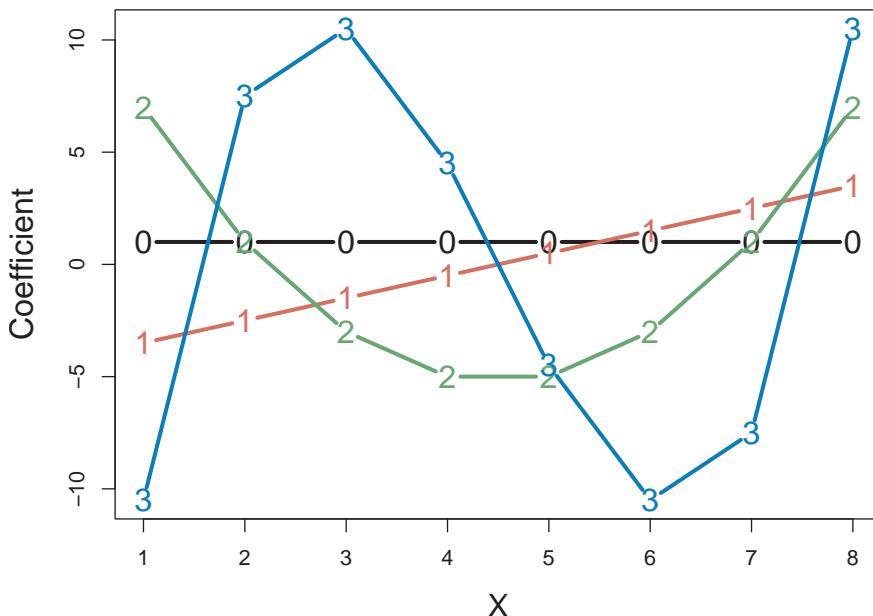


Figure 5.3: The coefficients for orthogonal polynomial contrasts for linear (1), quadratic (2) and cubic (3) terms for a numeric variable X . The intercept or constant term is represented as 0. Orthogonality means that each pair of values is uncorrelated.

of hypotheses about the factor level means. Even better, if your contrasts are orthogonal, their tests are statistically independent.

One useful idea for defining orthogonal comparisons of substantive interest is the idea of **nested dichotomies**. Here you would start with contrasting one meaningful subset of the groups against all the others. Then, successive contrasts are defined by making dichotomies among the groups within each subset.

TODO: Make a figure for the two examples

As one example, suppose we are looking at support on some issue among four political parties: A, B, C and D, where A and B are left-of-center and C and D are to the right of the political spectrum. The following comparisons might of interest:

AB.CD: {A, B} vs. {C, D}
 A.B: {A} vs. {B}
 C.D: {C} vs. {D}

You could set up these comparisons as the following contrasts:

```
# contrasts
AB.CD <- c(1, 1, -1, -1)
A.B   <- c(1, -1, 0, 0)
C.D   <- c(0, 0, 1, -1)

# put them in a matrix
C <- cbind(AB.CD, A.B, C.D)
rownames(C) <- LETTERS[1:4]
C
#>   AB.CD A.B C.D
#> A      1    1    0
```

```
#> B      1   -1    0
#> C     -1    0    1
#> D     -1    0   -1
```

With a data frame like `df` with the factor `party`, you would then use these contrasts in a model by assigning `C` to `contrasts(df$party)`:

```
set.seed(47)
df <- data.frame(
  party = factor(rep(LETTERS[1:4], each = 3)),
  support = c(35, 25, 25, 15) + round(rnorm(12, 0, 1), 1)
)
contrasts(df$party) <- C
```

Then, in a linear model, the coefficients estimate the mean difference between the averages of the subset of parties in each comparison. For example, parties A and B on average are 2.11 higher in support than parties C and D; support for party A is 2.37 greater than party B and so forth.

```
party.mod <- lm(support ~ party, data = df)
coef(party.mod)
#> (Intercept)  partyAB.CD  partyA.B  partyC.D
#>       24.83        2.11        2.37        1.85
```

For another example, say we are examining differences among three psychiatric diagnostic patient groups, “bipolar”, “manic”, “depressed” and also have a matched normal group. One set of meaningful comparisons would be as follows:

```
D1: {Normal} vs. {Bipolar, Depressed, Manic}
D2: {Bipolar} vs. {Depressed, Manic}
D3: {Depressed} vs. {Manic}
```

Weights for these contrasts are assigned by making them positive values for the groups in one subset and negative for the other, and giving numbers that sum to zero for each one:

```
D1 <- c(3, -1, -1, -1)
D2 <- c(0, 2, -1, -1)
D3 <- c(0, 0, 1, -1)

C <- cbind(D1, D2, D3)
rownames(C) <- c("Normal", "Bipolar", "Depressed", "Manic")
C
#>           D1 D2 D3
#> Normal      3  0  0
#> Bipolar     -1  2  0
#> Depressed   -1 -1  1
#> Manic      -1 -1 -1
```

These have the same form as the reversed Helmert contrasts considered earlier.

5.2 What have we learned?

This chapter introduced the fundamental building blocks of linear modeling in R, focusing on the versatile `lm()` function as our primary tool for fitting and understanding linear relationships in data. The goal for the chapter is to help you understand the mechanics of translating between the algebraic formulation of a linear model and your research questions using the tools in the `lm()` family.

Here are the essential takeaways:

- **The `lm()` function is your Swiss Army knife for linear modeling** – Whether you’re fitting simple regression, multiple regression, ANOVA, or ANCOVA models, `lm()` provides a unified interface through R’s elegant formula syntax. The beauty lies in how a model formula like `y ~ x1 + x2 + x1:x2` captures complex relationships with intuitive notation.
- **Model objects contain a wealth of information** – An “`lm`” object isn’t just coefficients; it’s a comprehensive container holding fitted values, residuals, the design matrix, and diagnostic information. Learning to extract and manipulate these components with functions like `coef()`, `fitted()`, `residuals()`, and `model.matrix()` unlocks the full power of linear modeling.
- **Model matrices reveal the algebraic heart of your analysis** – Understanding how R transforms your formula into a design matrix via `model.matrix()` is fundamental to grasping what linear models actually compute. The journey from a formula like `y ~ treatment + block` to a matrix of 0s and 1s illuminates how categorical predictors are used in mathematical operations to calculate fitted values.
- **Contrasts control how factors speak to your research questions** – The choice between treatment, sum, or Helmert contrasts isn’t just technical housekeeping – it determines which comparisons your model coefficients represent. Using `contrasts()` and `C()` strategically means your model output directly answers your scientific questions rather than leaving you to decode cryptic parameter estimates.

However fitting a model is just the first step. While `summary()` and `anova()` provide essential numerical summaries, diagnostic plots created with `plot(lm_object)` expose patterns in residuals. Other plots help to identify influential observations, and validate model assumptions. The interplay between numerical and graphical summaries is where true understanding emerges. This is the topic of Chapter 6.

6

Plots for univariate response models

For a univariate linear model fit using `lm()`, `glm()` and similar functions, the standard `plot()` method gives basic versions of *diagnostic* plots of residuals and other calculated quantities for assessing possible violations of the model assumptions. Some of these can be considerably enhanced using other packages.

Beyond this,

- tables of model coefficients, standard errors and test statistics can often be usefully supplemented or even replaced by suitable *coefficient plots* providing essentially the same information.
- when there are two or more predictors, standard plots of y vs. each x can be misleading because they ignore the other predictors in the model. *Added-variable plots* (also called *partial regression plots*) allow you to visualize *partial* relations between the response and a predictor by adjusting (controlling for) all other predictors. *Component + residual* plots help to diagnose nonlinearity.
- in this same situation, you can more easily understand their separate impact on the response by plotting the marginal *predicted effects* of one or more focal variables, averaging over other variables not shown in a given plot.
- when there are highly correlated predictors, some specialized plots are useful to understand the nature of *multicollinearity*.

The classic reference on regression diagnostics is Belsley et al. (1980). My favorite modern texts are the brief Fox (2020) and the more complete Fox & Weisberg (2018a), both of which are supported by the `car` package (Fox et al., 2023). Some of the examples in this chapter are inspired by Fox & Weisberg (2018a).

Packages

In this chapter I use the following packages. Load them now:

```
library(car)
library(dplyr)
library(easystats)
library(effects)
library(ggeffects)
library(ggplot2)
library(ggstats)
library(marginaleffects)
library(modelsummary)
library(performance)
library(tidyr)
```

6.1 The “regression quartet”

For a fitted model, plotting the model object with `plot(model)` provides for any of six basic plots, of which four are produced by default, giving rise to the term *regression quartet* for this collection. These are:

- **Residuals vs. Fitted:** For well-behaved data, the points should hover around a horizontal line at residual = 0, with no obvious pattern or trend.
- **Normal Q-Q plot:** A plot of sorted standardized residuals e_i (obtained from `rstudent(model)`) against the theoretical values those values would have in a standard normal $\mathcal{N}(0, 1)$ distribution.
- **Scale-Location:** Plots the square-root of the absolute values of the standardized residuals $\sqrt{|e_i|}$ as a measure of “scale” against the fitted values \hat{y}_i as a measure of “location”. This provides an assessment of homogeneity of variance. Violations appear as a tendency for scale (variability) to vary with location.
- **Residuals vs. Leverage:** Plots standardized residuals against leverage to help identify possibly influential observations. Leverage, or “hat” values (given by `hat(model)`) are proportional to the squared Mahalanobis distances of the predictor values \mathbf{x}_i from the means, and measure the potential of an observation to change the fitted coefficients if that observation was deleted. Actual influence is measured by Cooks’s distance (`cooks.distance(model)`) and is proportional to the product of residual times leverage. Contours of constant Cook’s D are added to the plot.

One key feature of these plots is providing **reference** lines or smoothed curves for ease of judging the extent to which a plot conforms to the expected pattern; another is the **labeling** of observations which deviate from an assumption.

The base-R `plot(model)` plots are done much better in a variety of packages. I illustrate some versions from the `car` (Fox et al., 2023) and `performance` (Lüdecke et al., 2021) packages, part of the `easystats` (Lüdecke et al., 2022) suite of packages.

6.1.1 Example: Duncan’s occupational prestige

In a classic study in sociology, Duncan (1961) used data from the U.S. Census in 1950 to study how one could predict the prestige of occupational categories — which is hard to measure — from available information in the census for those occupations. His data is available in `carData:Duncan`, and contains

- `type`: the category of occupation, one of `prof` (professional), `wc` (white collar) or `bc` (blue collar);
- `income`: the percentage of occupational incumbents with a reported income > 3500 (about 40,000 in current dollars);
- `education`: the percentage of occupational incumbents who were high school graduates;
- `prestige`: the percentage of respondents in a social survey who rated the occupation as “good” or better in prestige.

These variables are a bit quirky in they are measured in percents, 0-100, rather dollars for `income` and years for `education`, but this common scale permitted Duncan to ask an interesting sociological question: Assuming that both income and education predict prestige, are they equally important, as might be assessed by testing the hypothesis $H_0 : \beta_{\text{income}} = \beta_{\text{education}}$. If so, this would provide a very simple theory for occupational prestige.

A quick look at the data shows the variables and a selection of the occupational categories, which are the `row.names()` of the dataset.

```
data(Duncan, package = "carData")
set.seed(42)
car::some(Duncan)
```

```
#>           type income education prestige
#> accountant     prof    62      86      82
#> professor      prof    64      93      93
#> engineer       prof    72      86      88
#> factory.owner   prof    60      56      81
#> store.clerk      wc     29      50      16
#> carpenter        bc     21      23      33
#> machine.operator bc     21      20      24
#> barber          bc     16      26      20
#> soda.clerk      bc     12      30       6
#> janitor         bc      7      20       8
```

Let's start by fitting a simple model using just income and education as predictors. The results look very good! Both `income` and `education` are highly significant and the $R^2 = 0.828$ for the model indicates that `prestige` is very well predicted by just these variables.

```
duncan.mod <- lm(prestige ~ income + education, data=Duncan)
summary(duncan.mod)

#>
#> Call:
#> lm(formula = prestige ~ income + education, data = Duncan)
#>
#> Residuals:
#>   Min     1Q Median     3Q    Max
#> -29.54  -6.42   0.65   6.61  34.64
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -6.0647    4.2719  -1.42   0.16
#> income       0.5987    0.1197   5.00  1.1e-05 ***
#> education     0.5458    0.0983   5.56  1.7e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 13.4 on 42 degrees of freedom
#> Multiple R-squared:  0.828,  Adjusted R-squared:  0.82
#> F-statistic: 101 on 2 and 42 DF,  p-value: <2e-16
```

Beyond this, Duncan was interested in the coefficients and whether income and education could be said to have equal impacts on predicting occupational prestige. A nice display of model coefficients with confidence intervals is provided by `parameters::model_parameters()`.

```
parameters::model_parameters(duncan.mod)
#> Parameter | Coefficient | SE |      95% CI | t(42) |      p
#> -----
#> (Intercept) |      -6.06 | 4.27 | [-14.69, 2.56] | -1.42 | 0.163
#> income      |       0.60 | 0.12 | [ 0.36, 0.84] |  5.00 | < .001
#> education    |       0.55 | 0.10 | [ 0.35, 0.74] |  5.56 | < .001
```

We can also test Duncan's hypothesis that income and education have equal effects on prestige with `car::linearHypothesis()`. This is constructed as a test of a restricted model in which the two coefficients are forced to be equal against the unrestricted model. Duncan was very happy with this result.

```

car::linearHypothesis(duncan.mod, "income = education")
#>
#> Linear hypothesis test:
#> income - education = 0
#>
#> Model 1: restricted model
#> Model 2: prestige ~ income + education
#>
#>   Res.Df  RSS Df Sum of Sq    F Pr(>F)
#> 1     43 7519
#> 2     42 7507  1      12.2 0.07    0.8

```

Equivalently, the linear hypothesis that $\beta_{\text{Inc}} = \beta_{\text{Educ}}$ can be tested with a Wald test for difference between these coefficients, expressed as $\mathcal{H}_0 : \mathbf{C}\beta = 0$, using $\mathbf{C} = (0, -1, 1)$. The estimated value, -0.053 has a confidence interval [-0.462, 0.356], consistent with Duncan's hypothesis.

```

wtest <- spida2::wald(duncan.mod, c(0, -1, 1))[[1]]
wtest$estimate
#>
#>           Estimate Std. Error DF t-value p-value Lower 0.95 Upper 0.95
#> Larg -0.0529     0.203 42 -0.261   0.795     -0.462     0.356

```

We can visualize this test and confidence intervals using a joint confidence ellipse for the coefficients for income and education in the model `duncan.mod`. In Figure 6.1 the size of the ellipse is set to $\sqrt{F_{1,\nu}^{0.95}} = t_\nu^{0.95}$, so that its shadows on the horizontal and vertical axes correspond to 1D 95% confidence intervals. In this plot, the line through the origin with slope = 1 corresponds to equal coefficients for income and education and the line with slope = -1 corresponds to their difference, $\beta_{\text{Educ}} - \beta_{\text{Inc}}$. The orthogonal projection of the coefficient vector $(\hat{\beta}_{\text{Inc}}, \hat{\beta}_{\text{Educ}})$ (the center of the ellipse) is the point estimate of $\hat{\beta}_{\text{Educ}} - \hat{\beta}_{\text{Inc}}$ and the shadow of the ellipse along this axis is the 95% confidence interval for the difference in slopes.

```

confidenceEllipse(duncan.mod, col = "blue",
  levels = 0.95, dfn = 1,
  fill = TRUE, fill.alpha = 0.2,
  xlim = c(-.4, 1),
  ylim = c(-.4, 1), asp = 1,
  cex.lab = 1.3,
  grid = FALSE,
  xlab = expression(paste("Income coefficient, ", beta[Inc])),
  ylab = expression(paste("Education coefficient, ", beta[Educ])))

abline(h=0, v=0, lwd = 2)

# confidence intervals for each coefficient
beta <- coef( duncan.mod )[-1]
CI <- confint(duncan.mod)      # confidence intervals
lines( y = c(0,0), x = CI["income",] , lwd = 5, col = 'blue')
lines( x = c(0,0), y = CI["education",] , lwd = 5, col = 'blue')
points(rbind(beta), col = 'black', pch = 16, cex=1.5)
points(diag(beta) , col = 'black', pch = 16, cex=1.4)
arrows(beta[1], beta[2], beta[1], 0, angle=8, len=0.2)
arrows(beta[1], beta[2], 0, beta[2], angle=8, len=0.2)

```

```

# add line for equal slopes
abline(a=0, b = 1, lwd = 2, col = "darkgreen")
text(0.8, 0.8, expression(beta[Educ] == beta[Inc]),
     srt=45, pos=3, cex = 1.5, col = "darkgreen")

# add line for difference in slopes
col <- "darkred"
x <- c(-1.5, .5)
lines(x=x, y=-x)
text(-.15, -.15, expression(~beta["Educ"] - ~beta["Inc"]),
     col=col, cex=1.5, srt=-45)

# confidence interval for b1 - b2
wtest <- spida2::wald(duncan.mod, c(0, -1, 1))[[1]]
lower <- wtest$estimate$Lower / 2
upper <- wtest$estimate$Upper / 2
lines(-c(lower, upper), c(lower,upper), lwd=6, col=col)

# projection of (b1, b2) on b1-b2 axis
beta <- coef( duncan.mod )[-1]
bdiff <- beta %*% c(1, -1)/2
points(bdiff, -bdiff, pch=16, cex=1.3)
arrows(beta[1], beta[2], bdiff, -bdiff,
       angle=8, len=0.2, col=col, lwd = 2)

# calibrate the diff axis
ticks <- seq(-0.3, 0.3, by=0.2)
ticklen <- 0.02
segments(ticks, -ticks, ticks-sqrt(2)*ticklen, -ticks-sqrt(2)*ticklen)
text(ticks-2.4*ticklen, -ticks-2.4*ticklen, ticks, srt=-45)

```

6.1.2 Diagnostic plots

But, should Duncan be so happy? It is unlikely that he ran any model diagnostics or plotted his model; we do so now. Here is the regression quartet (Figure 6.2) for this model. Each plot shows some trend lines, and importantly, labels some observations that stand out and might deserve attention.

```

op <- par(mfrow = c(2,2),
          mar = c(4,4,3,1)+.1)
plot(duncan.mod, lwd=2, pch=16)
par(op)

```

Some points to note:

- A few observations (minister, reporter, conductor, contractor) are flagged in multiple panels. It turns out (Section 6.6.3) that the observations for minister and reporter noted in the residuals vs. leverage plot are highly influential and largely responsible for Duncan's finding that the slopes for income and education could be considered equal.
- The red trend line in the scale-location plot indicates that residual variance is not constant, but rather increases from both ends. This is a consequence of the fact that `prestige` is measured as a percentage, bounded at [0, 100], and the standard deviation of a percentage p is proportional to $\sqrt{p \times (1-p)}$ which is maximal at $p = 0.5$.

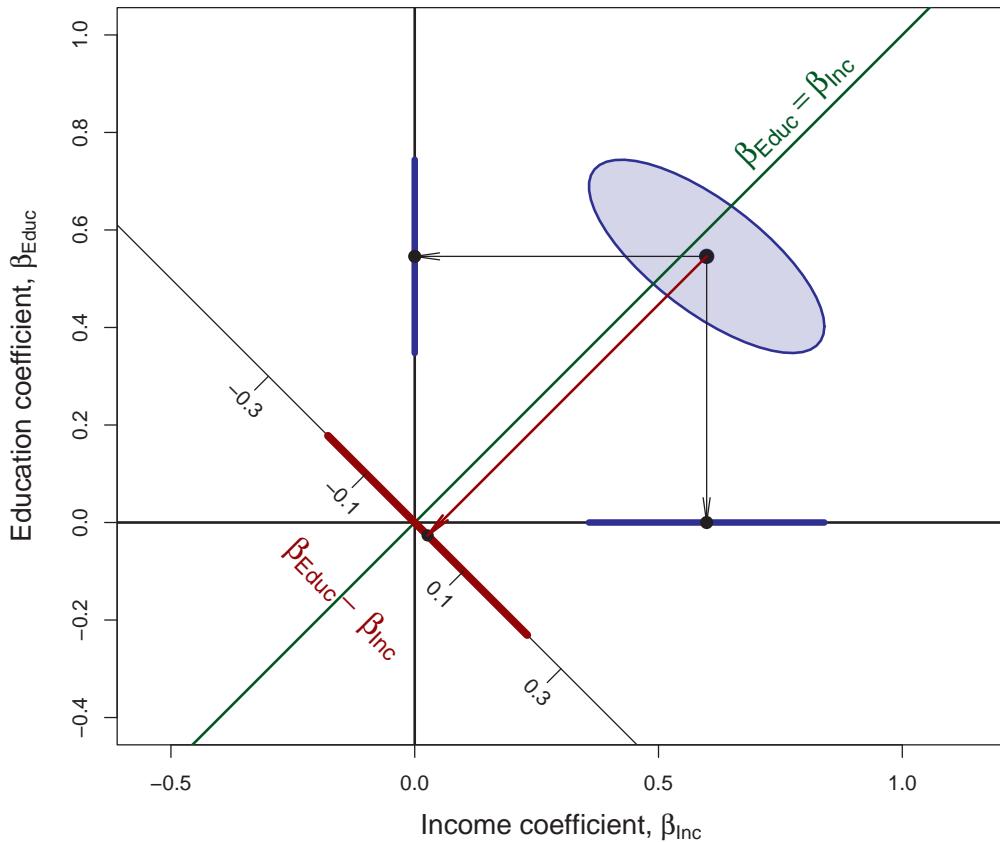


Figure 6.1: Joint 95% confidence ellipse for $(\beta_{\text{Inc}}, \beta_{\text{Educ}})$, together with their 1D shadows, which give 95% confidence intervals for the separate coefficients and the linear hypothesis that the coefficients are equal. Projecting the confidence ellipse along the line with unit slope gives a confidence interval for the difference between coefficients, shown by the dark red line.

Similar, but nicer-looking diagnostic plots are provided by `performance::check_model()` which uses `ggplot2` for graphics. These include helpful captions indicating what should be observed for each for a good-fitting model. However, they don't have as good facilities for labeling unusual observations as the base R `plot.lm()` method or functions in the `car`.

```
check_model(duncan.mod,
            check=c("linearity", "qq",
                   "homogeneity", "outliers"))
```

6.1.3 Example: Canadian occupational prestige

Following Duncan (1961), occupational prestige was studied in a Canadian context by Bernard Blishen and others at York University, giving the dataset `Prestige` which we looked at in Example 3.2. It differs from the `Duncan` dataset primarily in that the main variables—prestige, income and education were revamped to better reflect the underlying constructs in more meaningful units.

- `prestige`: Rather than a simple percentage of “good+” ratings, this uses a wider and more reliable scale from Pineo & Porter (1967) on a scale from 10–90.
- `income` is measured as the average income of incumbents in each occupation, in 1971 dollars, rather than percent exceeding a given threshold (\$3500)

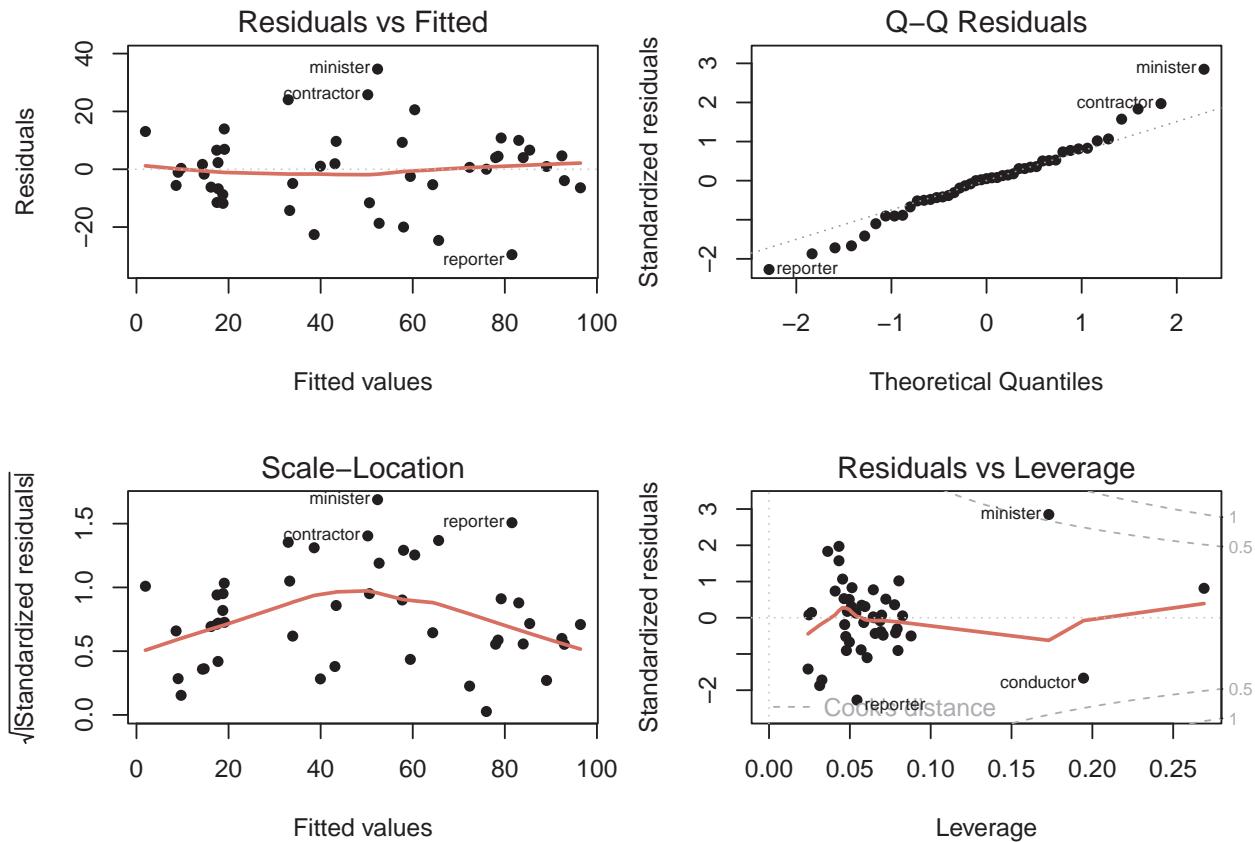


Figure 6.2: Regression quartet of diagnostic plots for the Duncan data. Several possibly unusual observations are labeled.

- education is measured as the average education of occupational incumbents, years.

The dataset again includes type of occupation with the same levels "bc" (blue collar), "wc" (white collar) and "prof" (professional)¹, but in addition includes the percent of women in these occupational categories.

Our interest again is in understanding how prestige is related to census measures of the average education, income, percent women of incumbents in those occupations, but with attention to the scales of measurement and possibly more complex relationships.

```
data(Prestige, package="carData")
# Reorder levels of type
Prestige$type <- factor(Prestige$type,
                         levels=c("bc", "wc", "prof"))
str(Prestige)
#> 'data.frame': 102 obs. of 6 variables:
#> $ education: num 13.1 12.3 12.8 11.4 14.6 ...
#> $ income   : int 12351 25879 9271 8865 8403 11030 8258 14163 11377 11023 ...
#> $ women    : num 11.16 4.02 15.7 9.11 11.68 ...
#> $ prestige : num 68.8 69.1 63.4 56.8 73.5 77.6 72.6 78.1 73.1 68.8 ...
```

¹Note that the factor type in the dataset has its levels ordered alphabetically. For analysis and graphing it is useful to reorder the levels in the natural increasing order. An alternative is to make type an *ordered* factor, but this would represent it using polynomial contrasts for linear and quadratic trends, which seems useful in this context.

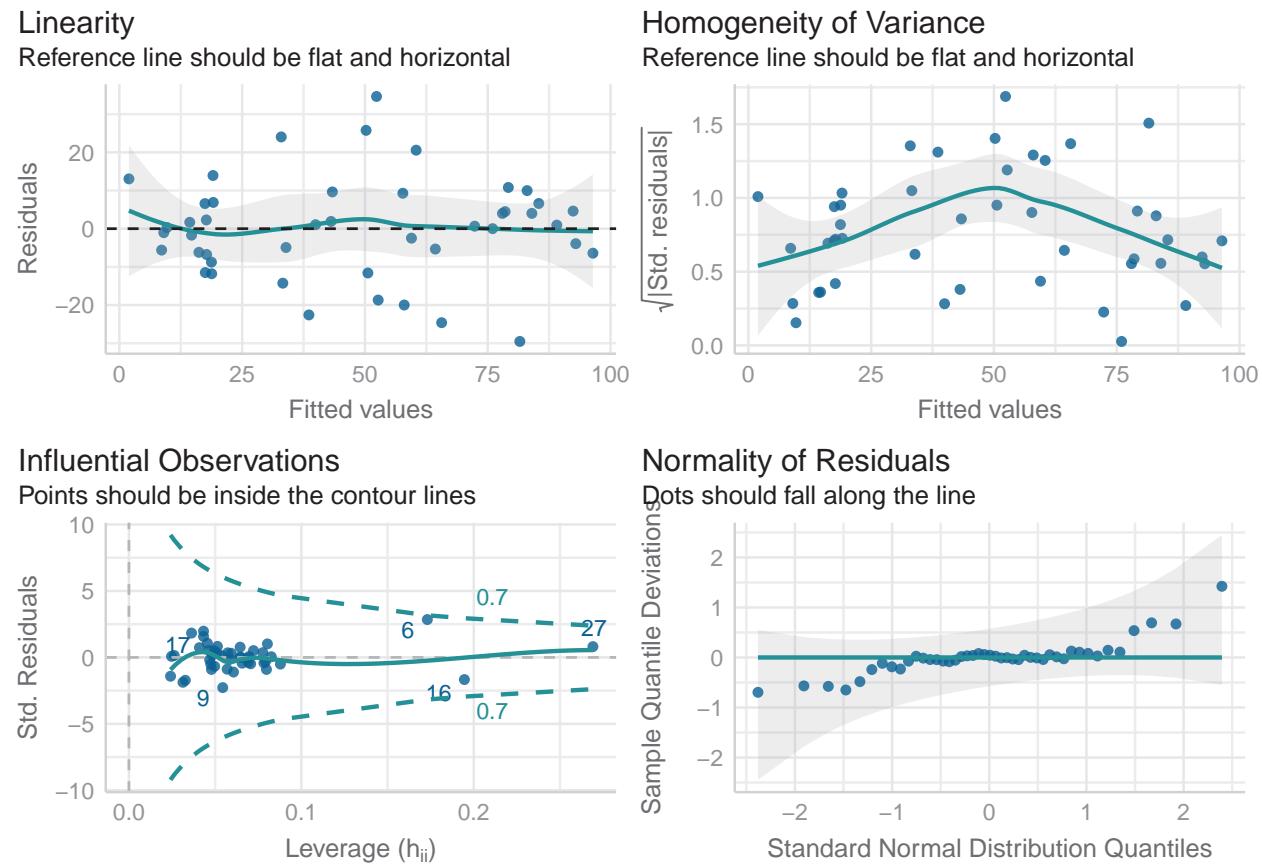


Figure 6.3: Diagnostic plots for the Duncan data, using `performance::check_model()`.

```
#> $ census    : int  1113 1130 1171 1175 2111 2113 2133 2141 2143 2153 ...
#> $ type      : Factor w/ 3 levels "bc","wc","prof": 3 3 3 3 3 3 3 3 3 3 ...
```

We fit a main-effects model using all predictors (ignoring `census`, the Canadian Census occupational code):

```
prestige.mod <- lm(prestige ~ education + income + women + type,
                     data=Prestige)
```

`plot(model)` produces four separate plots. For a quick look, I like to arrange them in a single 2x2 figure.

```
op <- par(mfrow = c(2,2),
           mar=c(4,4,3,1)+.1)
plot(prestige.mod, lwd=2, cex.lab=1.4)
par(op)
```

6.2 Other Model plots

TODO: What goes here?

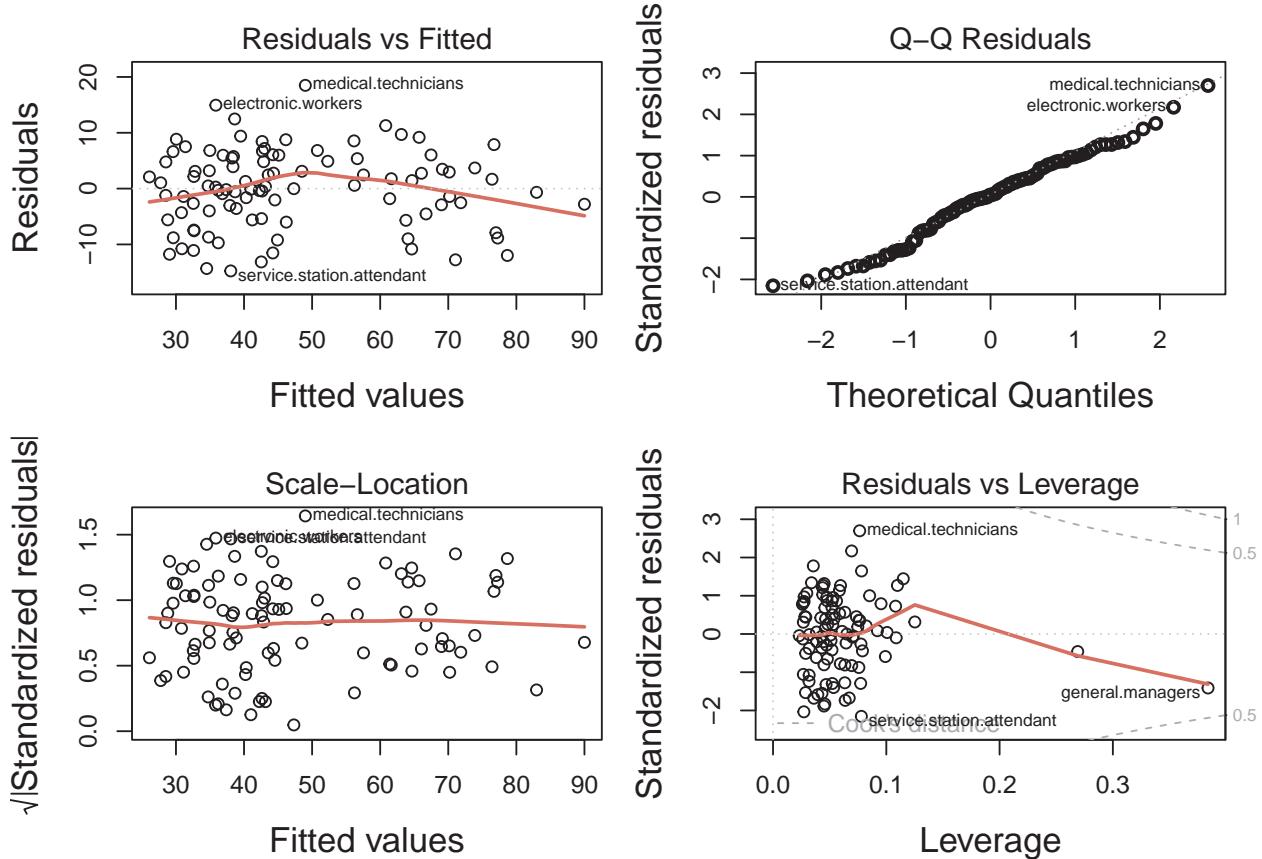


Figure 6.4: Regression quartet of diagnostic plots for the `Prestige` data. Several possibly unusual observations are labeled in each plot.

6.3 Coefficient displays

The results of linear models are most often reported in tables and typically with “significance stars” (*, **, ***) to indicate the outcome of hypothesis tests. These are useful for looking up precise values and you can use this format to compare a small number of competing models side-by-side. However, as illustrated by Kastellec & Leoni (2007), plots of coefficients can increase the clarity of presentation and make it easier to draw correct conclusions. Yet, when you need to present tables, there is a variety of tools in R that can help make them attractive in publications.

For illustration, I'll consider three models for the `Prestige` data of increasing complexity:

- `mod1` fits the main effects of the three quantitative predictors;
- `mod2` adds the categorical variable `type` of occupation;
- `mod3` allows an interaction of `income` with `type`.

```
mod1 <- lm(prestige ~ education + income + women,
            data=Prestige)
mod2 <- lm(prestige ~ education + women + income + type,
            data=Prestige)
```

```
mod3 <- lm(prestige ~ education + women + income * type,
            data=Prestige)
```

From our earlier analyses (`?@sec-prestige`) (Example 3.2) we saw that the marginal relationship between `income` and `prestige` was nonlinear (Figure 3.12), and was better represented in a linear model using `log(income)` (Section 3.2.2.1) shown in Figure 3.15. However, this possibly non-linear relationship could also be explained by stratifying (Section 3.2.2.2) the data by `type` of occupation (Figure 3.16).

6.3.1 Displaying coefficients

`summary()` gives the complete precis of a fitted model, with information about the estimated coefficients, residuals and goodness-of-fit statistics like R^2 . But if you only want to see the coefficients, standard errors, etc. `lmtest::coeftest()` gives these results in the familiar format for console output. `broom::tidy()` places these in a tidy format common to many modeling functions which is useful for further processing (e.g., comparing models).

```
lmtest::coeftest(mod1)
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -6.794334   3.239089  -2.10   0.039 *
#> education    4.186637   0.388701   10.77  < 2e-16 ***
#> income       0.001314   0.000278    4.73  7.6e-06 ***
#> women        -0.008905  0.030407   -0.29   0.770
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

broom::tidy(mod1)
#> # A tibble: 4 x 5
#>   term      estimate std.error statistic p.value
#>   <chr>     <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept) -6.79      3.24     -2.10  3.85e- 2
#> 2 education    4.19      0.389     10.8   2.59e-18
#> 3 income       0.00131   0.000278    4.73  7.58e- 6
#> 4 women        -0.00891  0.0304     -0.293 7.70e- 1
```

The `modelsummary` package (Arel-Bundock, 2025b) is an easy to use, very general package to summarize data and statistical models in R. The main function `modelsummary()` can produce highly customizable tables of coefficients in a wide variety of output formats, including HTML, PDF, LaTeX, Markdown, and MS Word. You can select the statistics displayed for any model term with the `estimate` and `statistic` arguments.

```
modelsummary(list("Model1" = mod1),
            coef.omit = "Intercept",
            shape = term ~ statistic,
            estimate = "{estimate} [{conf.low}, {conf.high}]",
            statistic = c("std.error", "p.value"),
            fmt = fmt_statistic("estimate" = 3, "conf.low" = 3, "conf.high" = 3),
            gof.omit = ".")
```

`gof.omit` allows you to omit or select the goodness-of-fit statistics and other model information available from those listed by `get_gof()`:

Table 6.1: Table of coefficients for the main effects model.

Model1			
	Est.	S.E.	p
education	4.187 [3.415, 4.958]	0.389	0.000
income	0.001 [0.001, 0.002]	0.000	0.000
women	-0.009 [-0.069, 0.051]	0.030	0.770

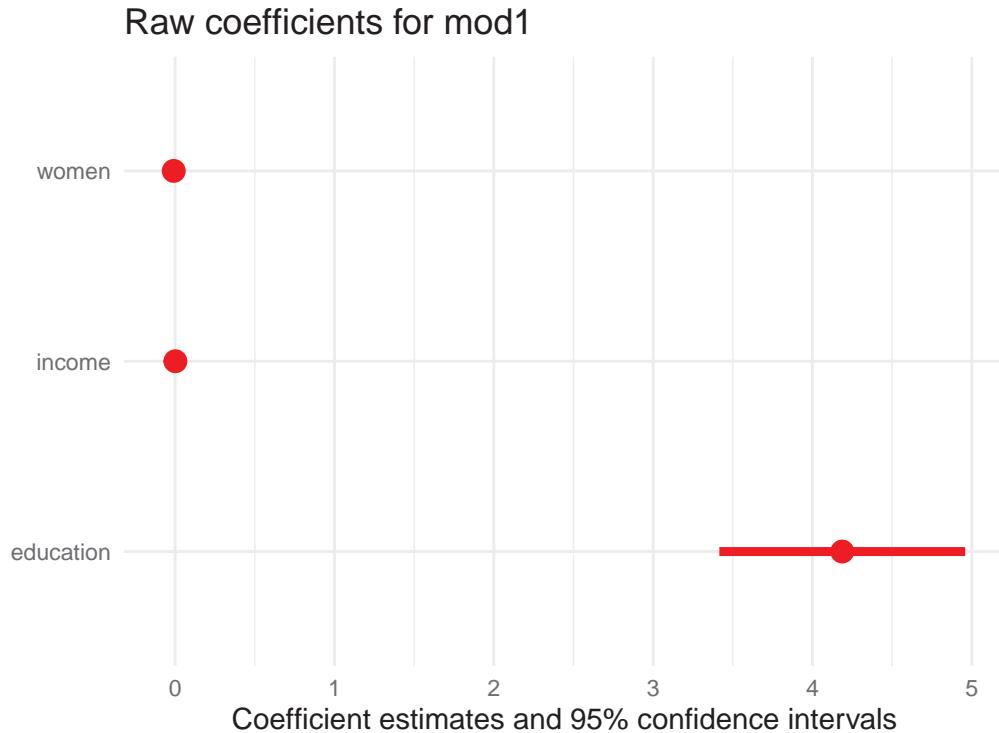
```
get_gof(mod1)
#>   aic bic r.squared adj.r.squared rmse nobs   F logLik
#> 1 716 729      0.798        0.792 7.69 102 129    -353
```

6.3.2 Visualizing coefficients

`modelplot()` is the companion function. It allows you to plot model estimates and confidence intervals. It makes it easy to subset, rename, and customize plots using same mechanics as in `modelsummary()`.

```
theme_set(theme_minimal(base_size = 14))

modelplot(mod1, coef_omit="Intercept",
          color="red", size=1, linewidth=2) +
  labs(title="Raw coefficients for mod1")
```

**Figure 6.5:** Plot of coefficients and their standard error bar for the simple main effects model

But this plot is disappointing and misleading because it shows the **raw** coefficients. From the plot, it looks

Table 6.2: Table of coefficients for three models.

	Model1			Model2			Model3		
	Est.	S.E.	p	Est.	S.E.	p	Est.	S.E.	p
education	4.19***	0.39	<0.01	3.66***	0.65	<0.01	2.80***	0.59	<0.01
income	0.00***	0.00	<0.01	0.00***	0.00	<0.01	0.00***	0.00	<0.01
women	-0.01	0.03	0.77	0.01	0.03	0.83	0.08*	0.03	0.02
typewc				-2.92	2.67	0.28	3.43	5.37	0.52
typeprof				5.91	3.94	0.14	27.55***	5.41	<0.01
income × typewc							-0.00	0.00	0.21
income × typeprof							-0.00***	0.00	<0.01
RMSE	7.69			6.91			6.02		
R2	0.798			0.835			0.875		

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001

like only `education` has a non-zero effect, but the effect of `income` is also highly significant. The problem is that the magnitude of the coefficient $\hat{b}_{\text{education}}$ is more than 40,000 times that of the other coefficients, because education is measured years, while income is measured in dollars. The 95% confidence interval for $\hat{b}_{\text{income}} = [0.0008, 0.0019]$, but this is invisible in the plot.

Before figuring out how to fix this issue, I show the comparable displays from `modelsummary()` and `modelplot()` for all three models. When you give `modelsummary()` a list of models, it displays their coefficients side-by-side as shown in Table 6.2.

```
models <- list("Model1" = mod1, "Model2" = mod2, "Model3" = mod3)
modelsummary(models,
  coef_omit = "Intercept",
  fmt = 2,
  stars = TRUE,
  shape = term ~ statistic,
  statistic = c("std.error", "p.value"),
  gof_map = c("rmse", "r.squared")
)
```

Note that a factor predictor (like `type` here) with d levels is represented by $d - 1$ coefficients in main effects and in interactions with quantitative variables. These levels are coded with *treatment contrasts* by default. Also by default, the first level is set as the reference level in alphabetical order. Here the reference level is blue collar (`bc`), so the coefficient `typeprof = 5.91` indicates that professional occupations on average are rated 5.91 greater on the Prestige scale than blue collar workers.

Note also that unlike the table, the coefficients in Figure 6.5 are ordered from bottom to top, because the Y axis starts at the lower left corner. In Figure 6.6 I use `scale_y_discrete()` to reverse the order. It is also useful to add a vertical reference line at $\beta = 0$.

```
modelplot(models,
  coef_omit="Intercept",
  size=1.3, linewidth=2) +
  ggtitle("Raw coefficients") +
```

```
geom_vline(xintercept = 0, linewidth=1.5) +
scale_y_discrete(limits=rev) +
theme(legend.position = "inside",
legend.position.inside = c(0.85, 0.2))
```

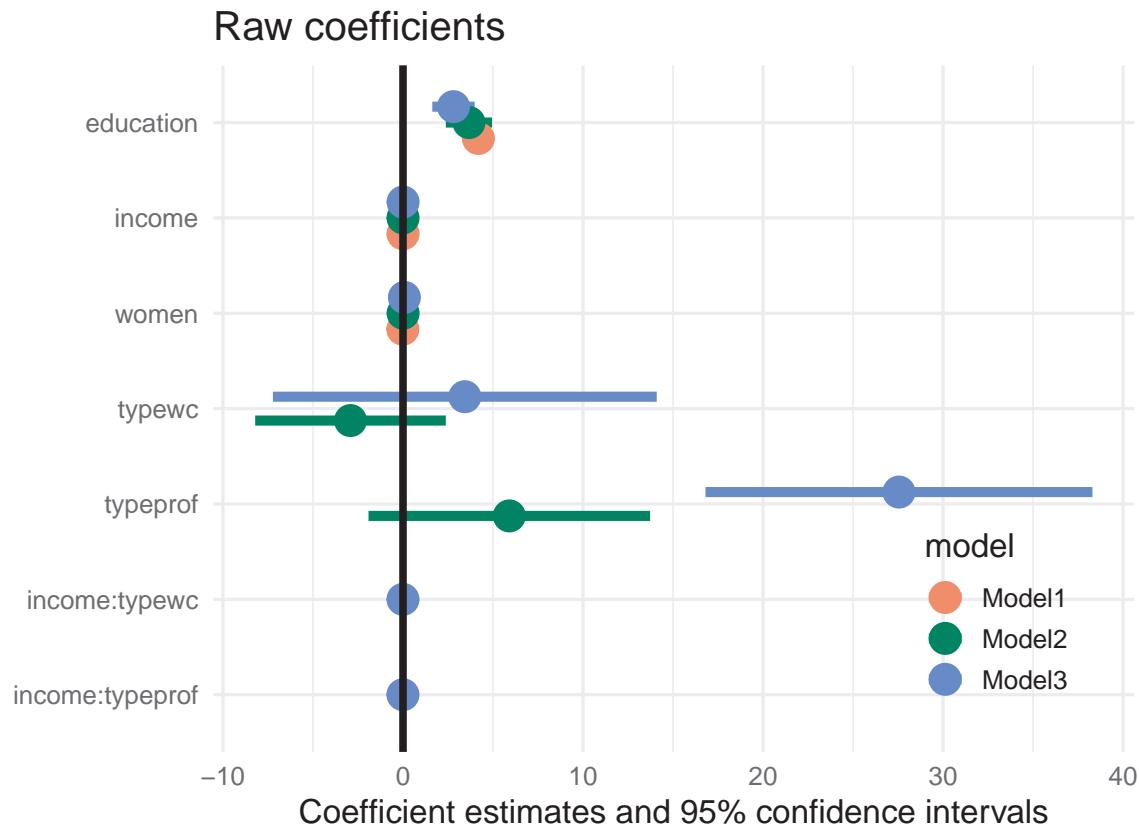


Figure 6.6: Plot of raw coefficients and their confidence intervals for all three models

6.3.3 More useful coefficient plots

The problem with plots of raw coefficients shown in Figure 6.5 and Figure 6.6 is that the coefficients for different predictors are not directly comparable because they are measured in different units.

One alternative is to plot the *standardized coefficients*. Another way is to re-scale the predictors into more comparable and meaningful units. I illustrate these ideas below.

Standardized coefficients

The simplest way to do this is to transform all variables to standardized (z) scores. The coefficients are then interpreted as the standardized change in prestige for a one standard deviation change in the predictors. The syntax below uses `scale` to transform all the numeric variables. Then, we re-fit the models using the standardized data.

```
Prestige_std <- Prestige |>
  as_tibble() |>
  mutate(across(where(is.numeric), scale))
```

```

mod1_std <- lm(prestige ~ education + income + women,
                 data=Prestige_std)
mod2_std <- lm(prestige ~ education + women + income + type,
                 data=Prestige_std)
mod3_std <- lm(prestige ~ education + women + income * type,
                 data=Prestige_std)

```

The plot in Figure 6.7 now shows the significant effect of income in all three models. As well, it offers a more sensitive comparison of the coefficients of other terms across models; for example `women` is not significant in models 1 and 2, but becomes significant in Model 3 when the interaction of `income * type` is included.

```

models <- list("Model1" = mod1_std, "Model2" = mod2_std, "Model3" = mod3_std)
modelplot(models,
          coef_omit="Intercept", size=1.3) +
  ggtitle("Standardized coefficients") +
  geom_vline(xintercept = 0, linewidth = 1.5) +
  scale_y_discrete(limits=rev) +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.2))

```

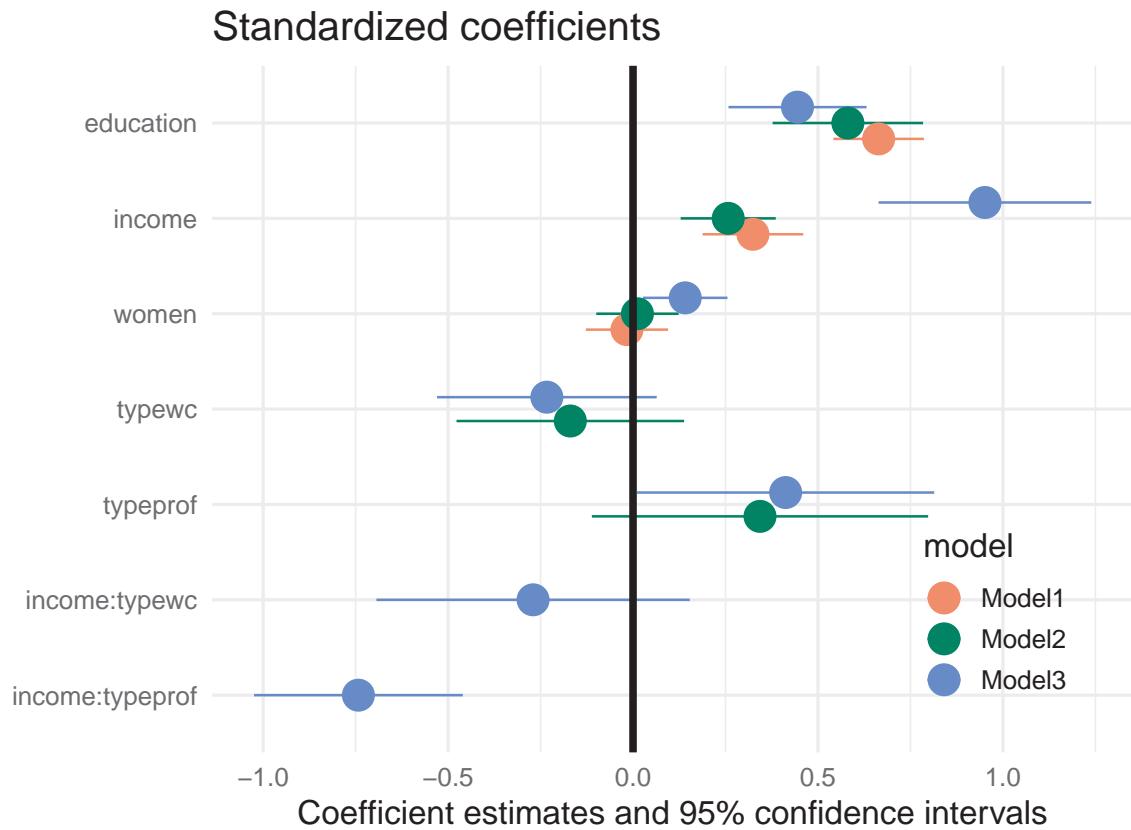


Figure 6.7: Plot of standardized coefficients and their confidence intervals for all three models

It turns out there is an easier way to get plots of standardized coefficients. `modelsummary()` extracts coefficients from model objects using the `parameters` package, and that package offers several options for standardization: See [model parameters documentation](#). We can pass the `standardize="refit"` (or other)

argument directly to `modelsummary()` or `modelplot()`, and that argument will be forwarded to `parameters`. The plot produced by the code below is identical to Figure 6.7 and is not shown.

```
modelplot(list("mod1" = mod1, "mod2" = mod2, "mod3" = mod3),
          standardize = "refit",
          coef OMIT="Intercept", size=1.3) +
  ggtitle("Standardized coefficients") +
  geom_vline(xintercept = 0, linewidth=1.5) +
  scale_y_discrete(limits=rev) +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.2))
```

The `ggstats` package (Larmarange, 2025) provides even nicer versions of coefficient plots that handle factors in a more reasonable way, as levels within the factor. `ggcoef_model()` plots a single model and `ggcoef_compare()` plots a list of models using sensible defaults. A small but nice feature is that it explicitly shows the 0 value for the reference level of a factor (`type = "bc"` here) and uses better labels for factors and their interactions.

```
models <- list(
  "Base model"      = mod1_std,
  "Add type"        = mod2_std,
  "Add interaction" = mod3_std)

ggcoef_compare(models) +
  labs(x = "Standarized Coefficient")
```

More meaningful units

Standardizing the variables makes the coefficients directly comparable, but it may be harder to understand what they mean in terms of the variables. For example, the coefficient of `income` in `mod2_std` is 0.25. A literal interpretation is that occupational prestige is expected to increase 0.25 standard deviations for each standard deviation increase in income, but it may be difficult to appreciate what this means.

A better substantive comparison of the coefficients could use understandable scales for the predictors, e.g., months of education, \$100,000 of income or 10% of women's participation. Note that the effect of this is just to multiply the coefficients and their standard errors by a factor. The statistical conclusions of significance are unchanged.

For simplicity, I do this just for Model 1.

```
Prestige_scaled <- Prestige |>
  mutate(education = 12 * education,
        income = income / 100,
        women = women / 10)

mod1_scaled <- lm(prestige ~ education + income + women,
                  data=Prestige_scaled)
```

When we plot this with `ggcoef_model()`, there are many options to control how variables are labeled and other details.

```
ggcoef_model(mod1_scaled,
             signif_stars = FALSE,
```

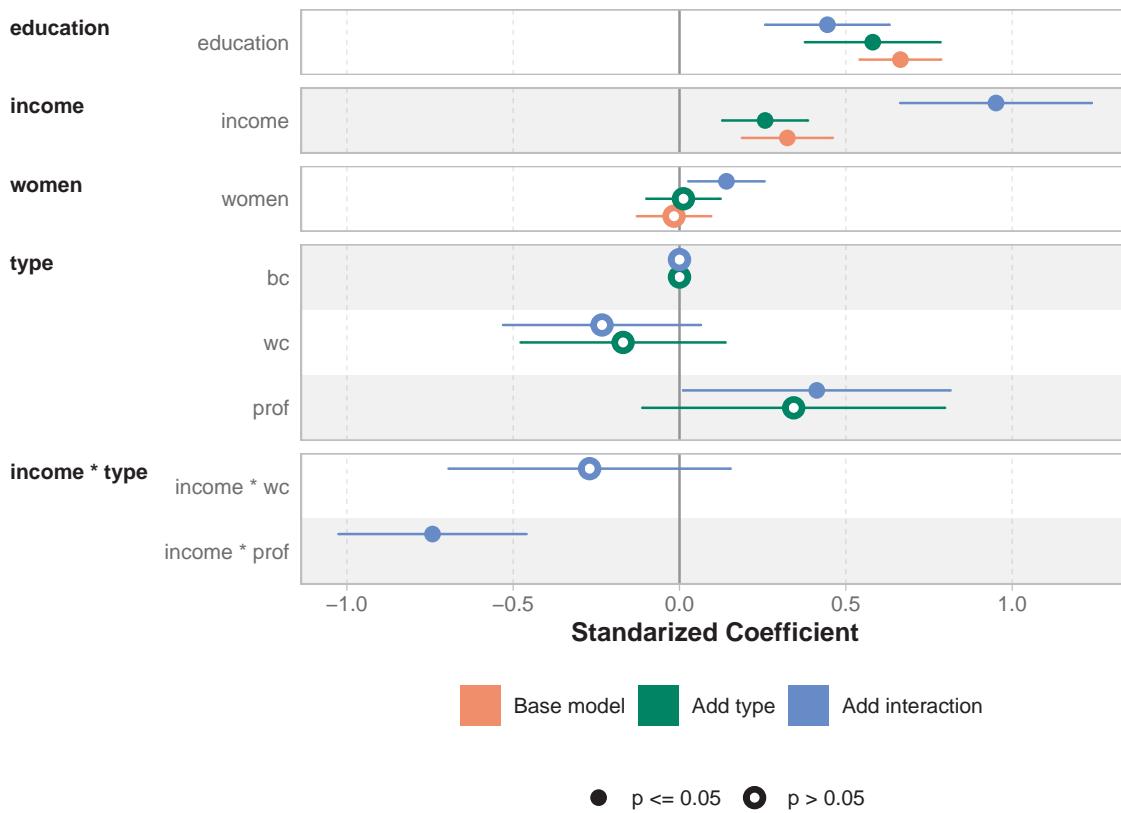


Figure 6.8: Model comparison plot from `ggcoef_compare()`

```
variable_labels = c(education = "education\n(months)",
                  income = "income\n(/$100K)",
                  women = "women\n(/10%)") +
  xlab("Coefficients for prestige with scaled predictors")
```

So, on average, each additional month of education increases the prestige rating by 0.34 units, while an additional \$100,000 of income increases it by 0.13 units. While these are significant effects, they are not large in relation to the scale of `prestige` which ranges 14.8—87.2.

6.4 Added-variable and related plots

In multiple regression problems, it is most often useful to construct a scatterplot matrix and examine the plot of the response vs. each of the predictors as well as those of the predictors against each other. However, the simple, *marginal* scatterplots of a response y against *each* of several predictors x_1, x_2, \dots can be misleading because each one ignores the other predictors.

To see this consider a toy dataset, `coffee`, giving measures of coffee consumption, occupational stress and an index of heart problems in a sample of $n = 20$ graduate students and professors.

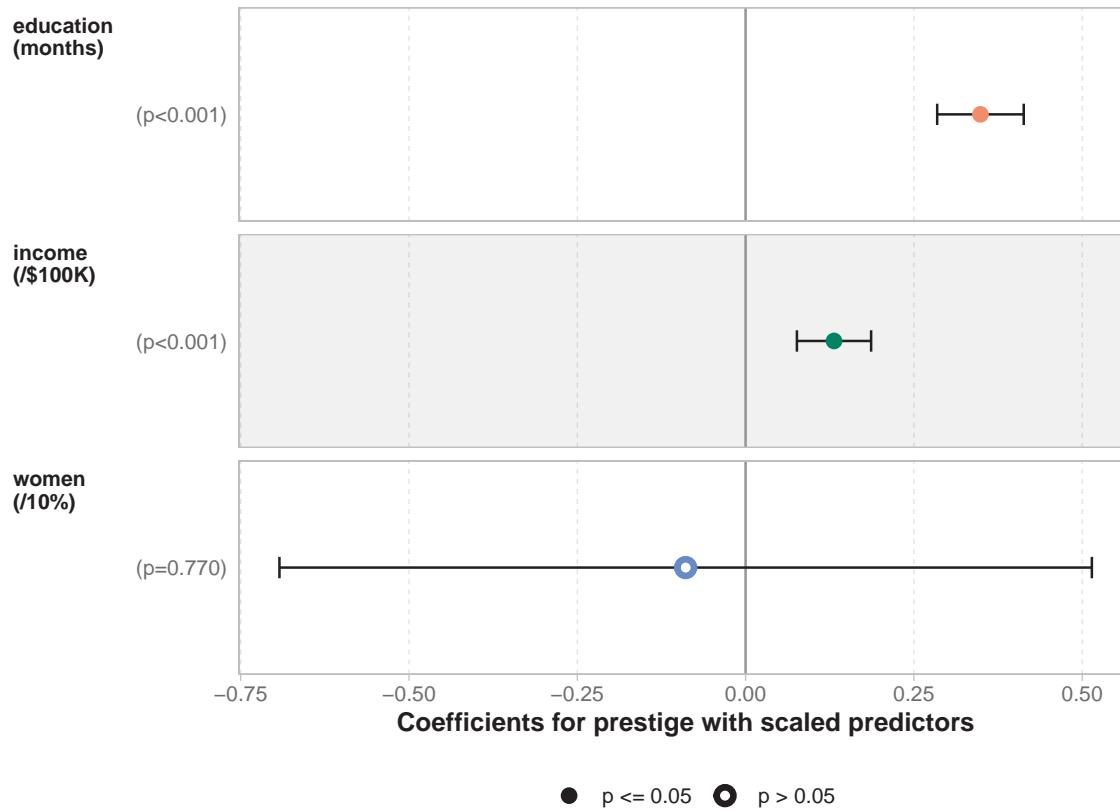


Figure 6.9: Plot of coefficients for prestige with scaled predictors for Model 1.

```
data(coffee, package="matlib")

scatterplotMatrix(~ Heart + Coffee + Stress,
  data=coffee,
  smooth = FALSE,
  ellipse = list(levels=0.68, fill.alpha = 0.1),
  pch = 19, cex.labels = 2.5)
```

The message from these marginal plots in Figure 6.10 seems to be that coffee is bad for your heart, stress is bad for your heart, and stress is also strongly related to coffee consumption. Yet, when we fit a model with both variables together, we get the following results:

```
fit.both <- lm(Heart ~ Coffee + Stress, data=coffee)
lmtest::coeftest(fit.both)
#>
#> t test of coefficients:
#>
#>            Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -7.794      5.793   -1.35    0.20
#> Coffee       -0.409      0.292   -1.40    0.18
#> Stress        1.199      0.224    5.34 5.4e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

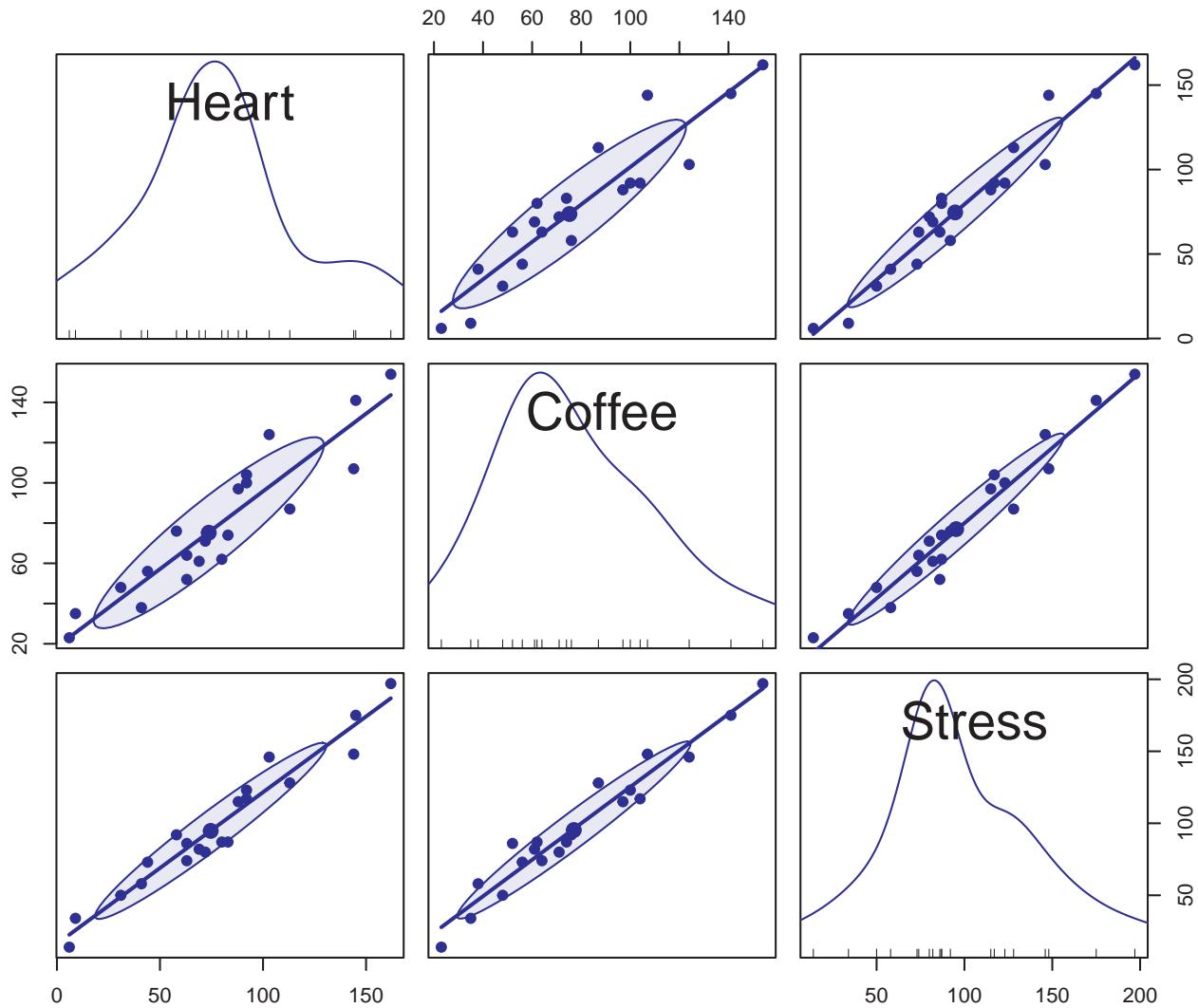


Figure 6.10: Scatterplot matrix showing pairwise relations among Heart (y), Coffee consumption (x_1) and Stress (x_2), with linear regression lines and 68% data ellipses for the bivariate relations

The coefficients suggest that stress is indeed bad for your heart, but the negative (though non-significant) coefficient for coffee suggests that coffee is good for you. How can this be? Does that mean I should drink more coffee, while avoiding stress?

The reason for this apparent paradox is that the general linear model fit by `lm()` estimates all effects together and so the coefficients pertain to the *partial* effect of a given predictor, *adjusting* for the effects of all others. That is, the coefficient for coffee ($\beta_{\text{Coffee}} = -0.41$) estimates the effect of coffee for people with *same* level of stress. In the marginal scatterplot, the positive slope for coffee (1.10) ignores the correlation of coffee and stress.

This is an example of *confounding* in regression when an important predictor is omitted. Stress is positively associated with both coffee consumption and heart damage. When stress is omitted, the coefficient for coffee is biased because it “picks up” the relation with the omitted variable.

A solution to this problem is the *added-variable* plot (“AV plot”, also called *partial regression* plot, MostellerTukey-1977). This is a multivariate analog of a simple marginal scatterplot, designed to visualize directly the partial relation between y and the predictors x_1, x_2, \dots in a multiple regression model.

You can think of this as a magic window that hides the relations of all other variables with each of the y and x_i shown in a given added-variable plot. This gives an unobstructed view of the net relation between y and x_i with the effect of all other variables removed. In effect, it reduces the problem of viewing the complete model in p -dimensional space to a sequence of p 2D plots, each of which tells the story of one predictor, unentangled from the others. This is essentially the same idea as the partial variables plot (Section 3.15.3) used to understand partial correlations.

The construction of an AV plot is conceptually very simple. For variable x_i , imagine that we fit two supplementary regressions:

- Regress \mathbf{y} on $\mathbf{X}_{(-i)}$, the model matrix of all of the regressors except x_i . By definition, the residuals from this regression, $\mathbf{y}^* \equiv \mathbf{y} | \text{others} = \mathbf{y} - \hat{\mathbf{y}} | \mathbf{X}_{(-i)}$, are the part of \mathbf{y} that cannot be explained by all the other regression terms. These residuals are necessarily uncorrelated with the other predictors.
- Regress x_i on the other predictors, $\mathbf{X}_{(-i)}$ and again obtain the residuals. These residuals, $\mathbf{x}_i^* \equiv \mathbf{x}_i | \text{others} = \mathbf{x}_i - \hat{\mathbf{x}}_i | \mathbf{X}_{(-i)}$ give the part of x_i that cannot be explained by the others, and so are uncorrelated with them.

The AV plot is then just a simple scatterplot of these residuals, \mathbf{y}^* on the vertical axis, and \mathbf{x}^* on the horizontal. In practice, it is unnecessary to run the auxilliary regressions this way (Velleman & Welsh, 1981). Both can be calculated using `stats::lsfit()` roughly as follows:

```
AVcalc <- function(model, variable)
  X <- model.matrix(model)
  response <- model.response(model)
  x <- X[, -variable]
  y <- cbind(X[, variable], response)
  fit <- lsfit(x, y, intercept = FALSE)
  resids <- residuals(fit)
  return(resids)
```

Note that \mathbf{y} here contains both the current predictor, \mathbf{x}_i and the response \mathbf{y} , so the residuals `resids` have two columns, one for $x_i | \text{others}$ and one for $y | \text{others}$.

Added-variable plots are produced using `car::avPlot()` for one predictor or `avPlots()` for any number of model terms. The `id` argument controls which points are identified in the plots; `n=2` labels the two points that are furthest from the mean on the horizontal axis *and* the two with the largest absolute residuals. For instance, in Figure 6.11, observations 5 and 13 are flagged because their conditional \mathbf{x}_i^* values are extreme; observation 17 has a large absolute residual, $\mathbf{y}^* = \text{Heart} | \text{others}$.

```
avPlots(fit.both,
  ellipse = list(levels = 0.68, fill=TRUE, fill.alpha = 0.1),
  pch = 19,
  id = list(n = 2),
  cex.lab = 1.5,
  main = "Added-variable plots for Coffee data")
```

The data ellipses for \mathbf{x}_i^* and \mathbf{y}^* summarize the conditional (or partial) relations of the response to each predictor controlling for all other predictors in each plot. The essential idea is that the data ellipse for $(\mathbf{x}_i^*, \mathbf{y}^*)$ has the identical relation to the estimate \hat{b}_i in a multiple regression as the data ellipse of (\mathbf{x}, \mathbf{y}) has to the slope in a simple regression.

6.4.1 Properties of AV plots

AV plots are particularly interesting and useful for the following noteworthy properties:

Added-variable plots for Coffee data

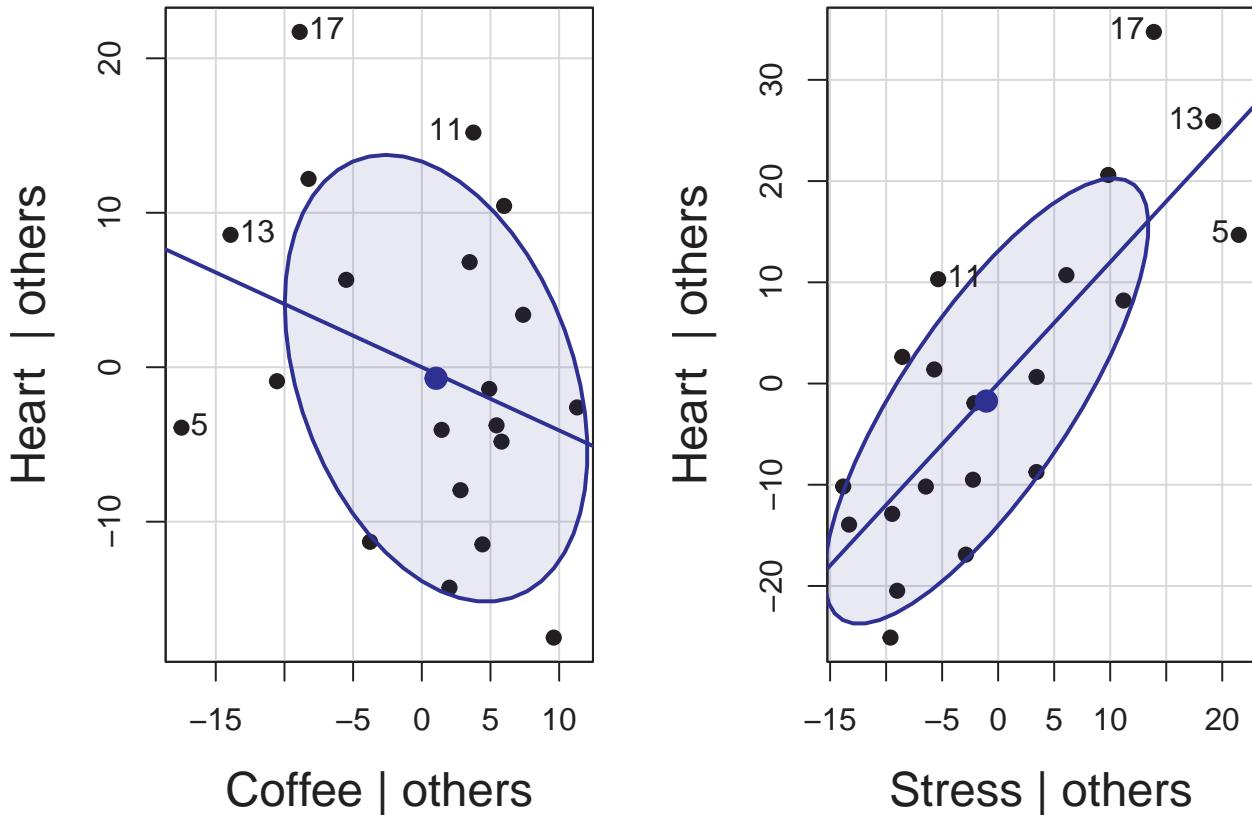


Figure 6.11: Added-variable plots for Coffee and Stress in the multiple regression model

- The slope of the simple regression in the AV plot for variable x_i is identical to the slope b_i for that variable in the full multiple regression model.
- The residuals in this plot are the same as the residuals using all predictors. This means you can see the degree of fit for observations directly in relation to the various predictors, which is not the case for marginal scatterplots.
- Consequentially, the standard deviation of the (vertical) residuals in the AV plot is the same as $s = \sqrt{MSE}$ in the full model and the standard error of a coefficient is $SE(b_i) = s/\sqrt{\sum(\mathbf{x}_i^*)^2}$. This is shown by the size of the shadow of the data ellipses on the vertical axis in Figure 6.11.
- The horizontal positions, \mathbf{x}_i^* , of points adjust for all other predictors, and so we can see points at the extreme left and right as unusual in relation to the others. If these points are also badly fitted (large residuals), we can see their influence on the fitted relation in the full model. AV plots thus provide visual displays of (partial) *leverage* and *influence* on each of the regression coefficients.
- The correlation of \mathbf{x}_i^* and \mathbf{y}^* shown by the shape of the data ellipses is the *partial correlation* between \mathbf{x}_i and \mathbf{y}_i with other predictors partialled out.

6.4.2 Marginal - conditional plots

The relation of the *conditional* data ellipses in AV plots to those in *marginal* plots of the same variables provides further insight into what it means to “control for” other variables. Figure 6.12 shows the same added-variable plots for Heart disease on Stress and Coffee as in Figure 6.11 (with a zoomed-out scaling), but

here we also overlay the marginal data ellipses for $(\mathbf{x}_i, \mathbf{y})$ (centered at the means), and marginal regression lines for Stress and Coffee separately. Drawing arrows connecting the original data points to their positions in the AV plot shows what happens when we condition on or partial out the other variable.

These *marginal - conditional plots* are produced by `car:::mcPlot()` (for one regressor) and `car:::mcPlots()` (for several). The plots for the marginal and conditional relations can be compared separately using the same scales for both, or overlaid as shown here. The points labeled here are only those with large absolute residuals y^* in the vertical direction.

```
mcPlots(fit.both,
  ellipse = list(levels=0.68, fill=TRUE, fill.alpha=0.2),
  id = list(n=2),
  pch = c(16, 16),
  col.marginal = "red", col.conditional = "blue",
  col.arrows = "black",
  cex.lab = 1.5)
```

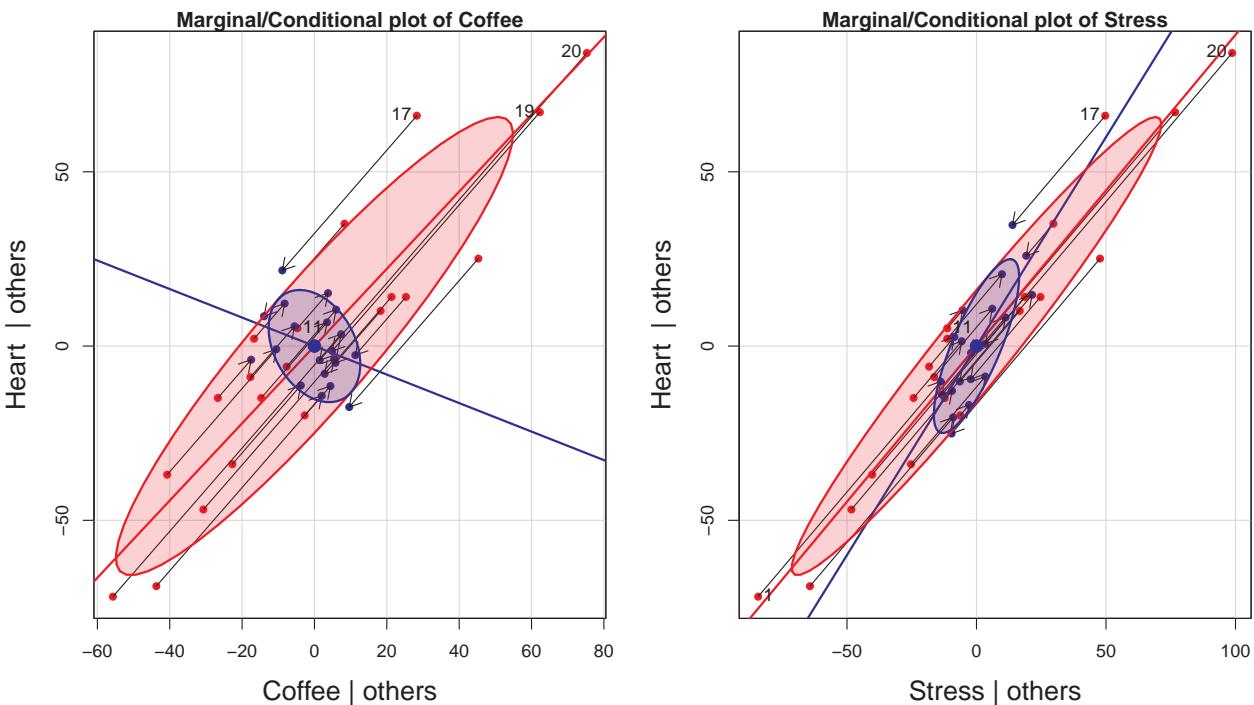


Figure 6.12: Marginal + conditional (added-variable) plots for Coffee and Stress in the multiple regression predicting Heart disease. Each panel shows the 68% conditional data ellipse for x_i^*, y^* residuals (shaded, blue) as well as the marginal 68% data ellipse for the (x_i, y) variables, shifted to the origin. Arrows connect the mean-centered marginal points (red) to the residual points (blue).

The most obvious feature of Figure 6.12 is that Coffee has a negative slope in the conditional AV plot but a positive slope in the marginal plot. This is an example of Simpson's paradox in a regression context: marginal and conditional relations can have opposite signs.

Less obvious is the relation between the marginal and AVP ellipses. In 3D, the marginal data ellipse is the shadow of the ellipsoid for $(\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2)$ on one of the coordinate planes, while the AV plot is a slice through the ellipsoid where either \mathbf{x}_1 or \mathbf{x}_2 is held constant. Thus, the AVP ellipse must be contained in the marginal ellipse, as we can see in Figure 6.12. If there are only two xs , then the AVP ellipse must touch the marginal ellipse at two points.

Finally, Figure 6.12 also shows how conditioning on other predictors works for individual observations, where each point of $(\mathbf{x}_i^*, \mathbf{y}^*)$ is the image of $(\mathbf{x}_i, \mathbf{y})$ along the path of the marginal regression. The variability in the response and in the focal predictor are both reduced, leaving only the uncontaminated relation of \mathbf{y} with \mathbf{x}_i .

These plots are similar in spirit to the ARES plot (“Adding REgressors Smoothly”) proposed by R. D. Cook & Weisberg (1994), but their idea was an interactive animation, displaying a smooth transition between the fit of a marginal model and the fit of a larger model. They used linear interpolation,

$$(\mathbf{x}_i, \mathbf{y}_i)_{\text{interp}} = (\mathbf{x}_i, \mathbf{y}_i) + \lambda[(\mathbf{x}_i^*, \mathbf{y}_i^*) - (\mathbf{x}_i, \mathbf{y}_i)],$$

controlled by a slider whose value, $\lambda \in [0, 1]$, was the weight given to the smaller marginal model. See [this animation](#) for an example using the Duncan data.

6.4.3 Prestige data

For a substantive example, let’s return to the model for income, education and women in the `Prestige` data. The plot in Figure 6.13 shows the strong positive relations of income and education to prestige in the full model, and the negligible relation of percent women. But, in the plot for income, two occupations (physicians and general managers) with high income strongly pull the regression line down from what can be seen in the orientation of the conditional data ellipse.

```
prestige.mod1 <- lm(prestige ~ education + income + women,
                      data=Prestige)

avPlots(prestige.mod1,
        ellipse = list(levels = 0.68),
        id = list(n = 2, cex = 1.2),
        pch = 19,
        cex.lab = 1.5,
        main = "Added-variable plots for prestige")
```

The influential points for physicians and general managers could just be unusual, or suggest that the relation of income to prestige is nonlinear. A rough test of this is to fit a smoothed curve through the points in the AV plot as shown in Figure 6.14.

```
op <- par(mar=c(4, 4, 1, 0) + 0.5)
res <- avPlot(prestige.mod1, "income",
              ellipse = list(levels = 0.68),
              pch = 19,
              cex.lab = 1.5)
smooth <- loess.smooth(res[,1], res[,2])
lines(smooth, col = "red", lwd = 2.5)
```

However, this use of AV plots to diagnose nonlinearity or suggest transformations can be misleading (R. D. Cook, 1996). Curvature in these plots is an indication of *some* model deficiency, but unless the predictors are uncorrelated, they cannot determine the form of a possible transformation of the predictors.

6.4.4 Component + Residual plots

A related method, the *component + residual plot* (“C+R plot”, also called *partial residual plot*, Larsen & McCleary (1972); R. D. Cook (1993)) gives a plot more suited to detecting the need to transform a predictor \mathbf{x}_i to a form $f(\mathbf{x}_i)$ to make it’s relationship with the response \mathbf{y} more nearly linear. This plot displays the partial residual $\mathbf{e} + \hat{b}_i \mathbf{x}_i$ on the vertical axis against \mathbf{x}_i on the horizontal, where \mathbf{e} are the residuals from the full model. A smoothed curve through the points will often suggest the form of the transformation $f()$. The fact that the horizontal axis is \mathbf{x}_i itself rather than \mathbf{x}_i^* makes it easier to see the functional form.

Added-variable plots for prestige

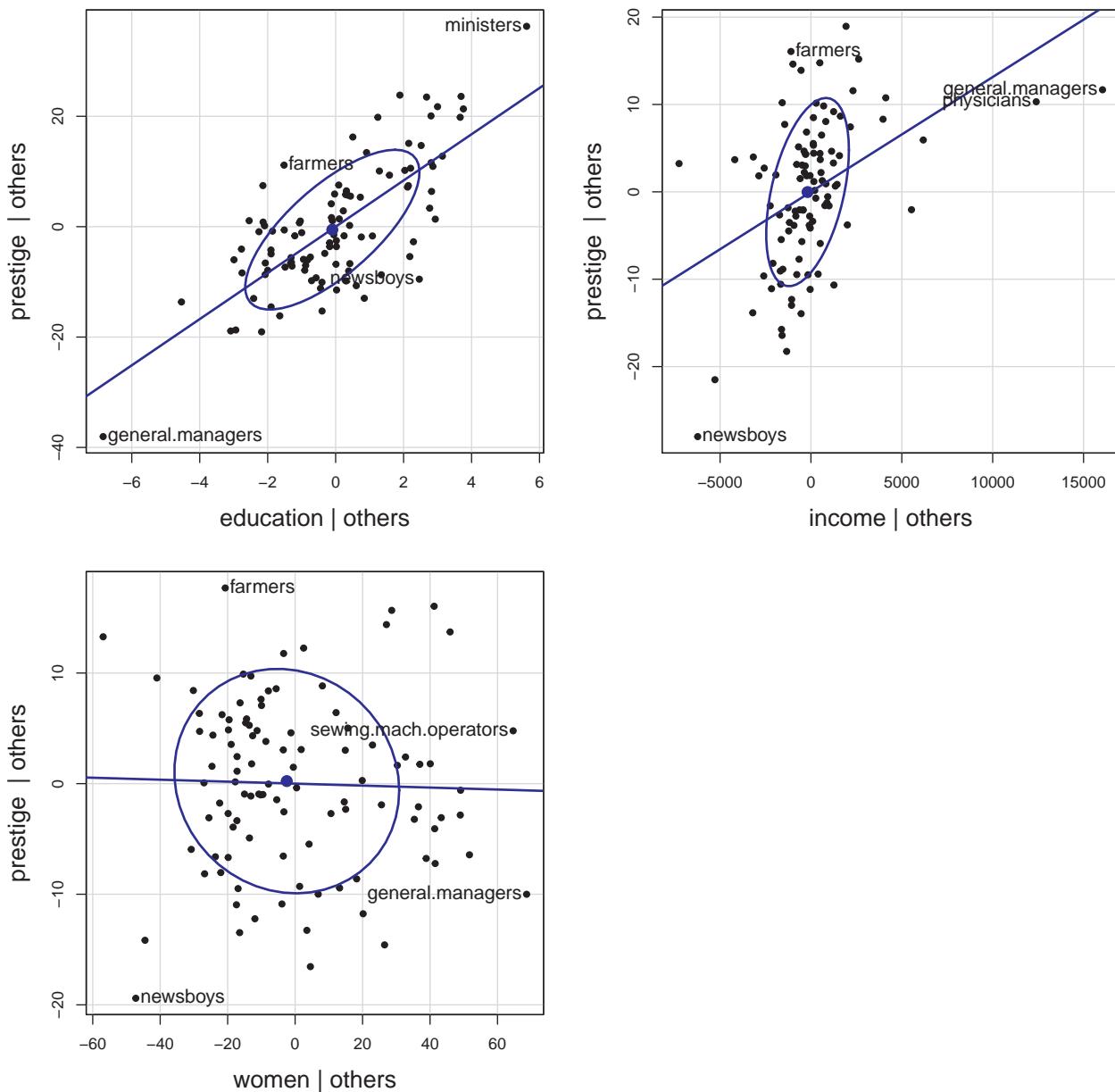


Figure 6.13: Added-variable plot for the quantitative predictors in the `Prestige` data.

The C+R plot has the same desirable properties as the AV plot: The slope \hat{b}_i and residuals e in this plot are the same as those in the full model.

C+R plots are produced by `car::crPlots()` and `car::crPlot()`. Figure 6.15 shows this just for income in the model `prestige.mod1`. (These plots for education and women show no strong evidence of curvilinearity.) The dashed blue line is the linear *partial fit*, $\hat{b}_i \mathbf{x}_i$, whose slope $\hat{b}_2 = 0.0013$ is the same as that for income in `prestige.mod1`. The solid red curve is the loess smooth through the points. The same points are identified as noteworthy as in AV plot in Figure 6.14.

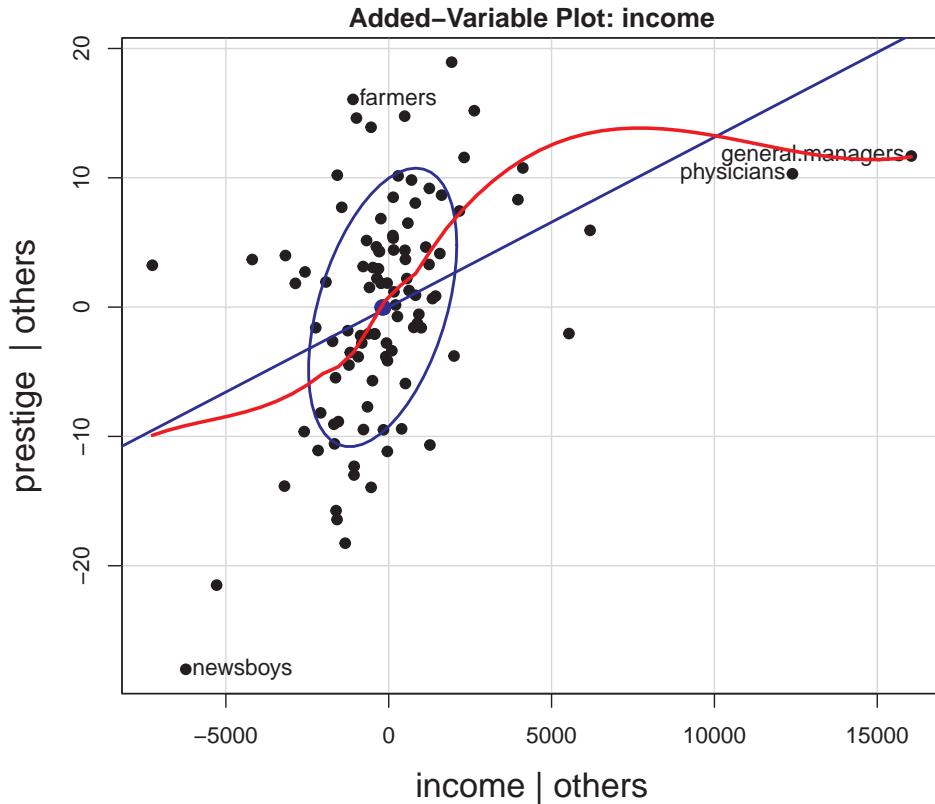


Figure 6.14: Added-variable plot for income, with a loess smooth.

```
crPlot(prestige.mod1, "income",
       smooth = TRUE,
       order = 2,
       pch = 19,
       col.lines = c("blue", "red"),
       id = list(n=2, cex = 1.2),
       cex.lab = 1.5)
```

The partial relation between prestige and income is clearly curved, so it would be appropriate to transform income or to include a polynomial (quadratic) term and refit the model. As suggested earlier (Example 3.2) it makes sense statistically and substantively to model the effect of income on a log scale, so then the slope for `log(income)` would measure the increment in prestige for a constant *percentage increase* in income.

The effect of percent women on prestige seen in Figure 6.13 appears very small and essentially linear. However, if we wished to examine this more closely, we could use the C+R plot in Figure 6.16.

```
crPlot(prestige.mod1, "women",
       pch = 19,
       col.lines = c("blue", "red"),
       id = list(n=2, cex = 1.2),
       cex.lab = 1.5)
```

This shows a slight degree of curvature, with modestly larger values in the extremes. If we wished to test this statistically, we could fit a model with a quadratic effect of women, and compare that to the linear-only effect using `anova()`.

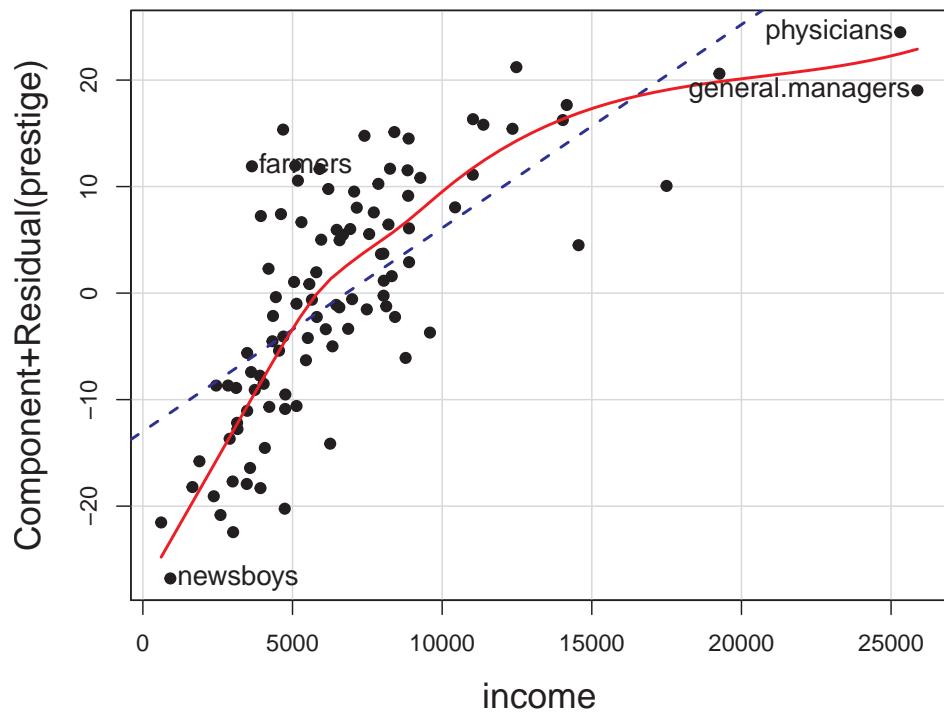


Figure 6.15: Component + residual plot for income in the model for the quantitative predictors of prestige. The dashed blue line is the partial linear fit for income. The solid red curve is the loess smooth.

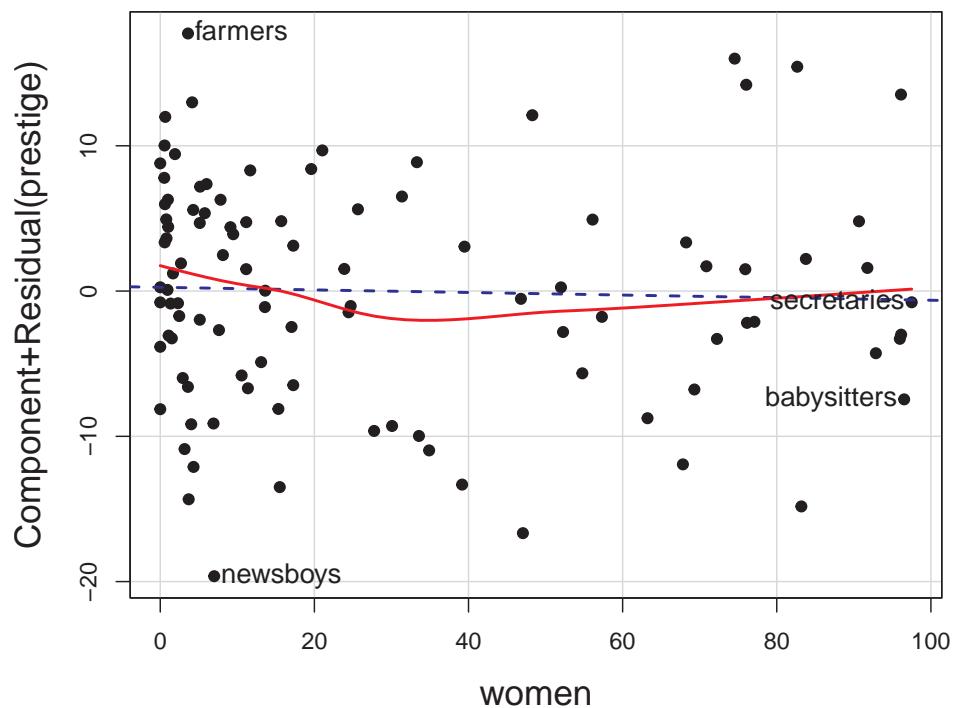


Figure 6.16: Component + residual plot for women in the model for the quantitative predictors of prestige.

```
prestige.mod2 <- lm(prestige ~ education + income + poly(women,2),
```

```

data=Prestige)

anova(prestige.mod1, prestige.mod2)
#> Analysis of Variance Table
#>
#> Model 1: prestige ~ education + income + women
#> Model 2: prestige ~ education + income + poly(women, 2)
#>   Res.Df RSS Df Sum of Sq   F Pr(>F)
#> 1     98 6034
#> 2     97 5907  1      127 2.08  0.15

```

This model ignores the `type` of occupation (“bc”, “wc”, “prof”) as well as any possible interactions of `type` with other predictors. We examine this next, using effect displays.

6.5 Effect displays

For two predictors it is possible, even if awkward, to display the fitted response surface in a 3D plot or faceted 2D views in what I call a *full model plot*. For more than two predictors such displays become cumbersome if not impractical, particularly when there are interactions in the model, when some effects are curvilinear, or when the main substantive interest is focused understanding on one or more main effects or interaction terms in the presence of others. The method of *effect displays*, largely introduced by John Fox (Fox, 1987, 2003; Fox & Weisberg, 2018b) is a generally useful solution to this problem.² These plots are nearly always easier to understand than tables of coefficients.

The idea of effect displays is quite simple, but very general and handles models of arbitrary complexity. Imagine that in a model we have a particular subset of predictors (*focal predictors*) whose effects on the response variable we wish to visualize. The essence of an effect display is that we calculate the predicted values (and standard errors) of the response for the model term(s) involving the focal predictors (and all low-order relatives, e.g, main effects that are marginal to an interaction) as those predictors are allowed to vary over a grid covering their range.

For a given plot, the other, non-focal variables are “controlled” by being fixed at typical values. For example, a quantitative predictor could be fixed at its mean, median or some representative value. A factor could be fixed at equal proportions of its levels or its proportions in the data. The result, when plotted, shows the predicted effects of the focal variables, either with multiple lines or in a faceted display, but with all the other variables controlled, adjusted for or averaged over. For interaction effects all low-order relatives are typically included in the fitted values for the term being graphed.

In practical terms, a scoring matrix \mathbf{X}^* is defined by the focal variables varied over their ranges and the other variables held fixed. The fitted values for a model term are then calculated as $\hat{\mathbf{y}}^* = \mathbf{X}^* \mathbf{b}$ using the equivalent of:

```

predict(model, newdata = X, se.fit = TRUE)

```

which also calculates the standard errors as the square roots of $\text{diag}(\widehat{\text{Var}}(\mathbf{b})\mathbf{X}^{*\top})$ where $\widehat{\text{Var}}(\mathbf{b})$ is the estimated covariance matrix of the coefficients. Consequently, predictor effect values can be obtained for *any* modelling function that has `predict()` and `vcov()` methods. To date, effect displays are available for over 100 different model types in various packages.

²Earlier, but less general expression of these ideas go back to the use of **adjusted means** in analysis of covariance (Fisher, 1925a) or **least squares means** or **population marginal means** in analysis of variance (Searle et al., 1980)

To illustrate the mechanics for the effect of education in the `prestige.mod1` model, construct a data frame varying education, but fixing income and women at their means:

```
X <- expand.grid(
  education = seq(8, 16, 2),
  income = mean(Prestige$income),
  women = mean(Prestige$women)) |>
  print(digits = 3)
#>   education income women
#> 1         8    6798    29
#> 2        10    6798    29
#> 3        12    6798    29
#> 4        14    6798    29
#> 5        16    6798    29
```

`predict()` then gives the fitted values for a simple effect plot of prestige against education. `predict.lm()` returns list, so it is necessary to massage this to a data frame for graphing.

```
pred <- predict(prestige.mod1, newdata=X, se.fit = TRUE)
cbind(X, fit = pred$fit, se = pred$se.fit) |>
  print(digits=3)
#>   education income women  fit      se
#> 1         8    6798    29 35.4 1.318
#> 2        10    6798    29 43.7 0.828
#> 3        12    6798    29 52.1 0.919
#> 4        14    6798    29 60.5 1.487
#> 5        16    6798    29 68.9 2.188
```

As Fox & Weisberg (2018b) note, effect displays can be combined with partial residuals to visualize *both* fit and potential lack of fit simultaneously, by plotting residuals from a model around 2D slices of the fitted response surface. This adds the benefits of C+R plots, in that we can see the impact of unmodeled curvilinearity and interactions in addition to those of predictor effect displays.

There are several implementations of effect displays in R, whose details, terminology and ease of use vary. Among these, `ggeffects` (Lüdecke, 2025) calculates adjusted predicted values under several methods for conditioning. `marginaleffects` (Arel-Bundock, 2025a) is similar and also provides estimation of marginal slopes, contrasts, odds ratios, etc. Both have `plot()` methods based on `ggplot2`. My favorite is the `effects` (Fox et al., 2025) package, which alone provides partial residuals, and is somewhat easier to use, though it uses `lattice` graphics. See the vignette [Predictor Effects Graphics Gallery](#) for details of the computations for effect displays.

The main functions for computing fitted effects are `predictorEffect()` (for one predictor) and `predictorEffects()` (for one or more). For a model `mod` with formula `y ~ x1 + x2 + x3 + x1:x2`, the call to `predictorEffects(mod, ~ x1)` recognizes that an interaction is present and calculates the fitted values for combinations of `x1` and `x2`, holding `x3` fixed at its average value. This returns an object of class "eff" which can be graphed using the `plot.eff()` method.

The effect displays for several predictors can be plotted together, as with `avplots()` (Figure 6.13) by including them in the plot formula, e.g., `predictorEffects(mod, ~ x1 + x3)`. Another function, `allEffects()` calculates the effects for each high-order term in the model, so `allEffects(mod) |> plot()` is handy for getting a visual overview of a fitted model.

6.5.1 Prestige data

To illustrate effect plots, I consider a more complex model, allowing a quadratic effect of women, representing income on a \log_{10} scale, and allowing this to interact with type of occupation. `Anova()` provides the Type II tests of each of the model terms.

```
prestige.mod3 <- lm(prestige ~ education + poly(women, 2) +
                      log10(income)*type, data=Prestige)

# test model terms
Anova(prestige.mod3)
#> Anova Table (Type II tests)
#>
#> Response: prestige
#>             Sum Sq Df F value    Pr(>F)
#> education          994  1 25.87 2.0e-06 ***
#> poly(women, 2)      414  2   5.38 0.00620 **
#> log10(income)     1523  1 39.63 1.1e-08 ***
#> type                589  2   7.66 0.00085 ***
#> log10(income):type  221  2   2.88 0.06133 .
#> Residuals         3420 89
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The fitted coefficients, standard errors and t -tests from `coeftest()` are shown below. The coefficient for education means that an increase of one year of education, holding other predictors fixed, gives an expected increase of 2.96 in prestige. The other coefficients are more difficult to understand. For example, the effect of women is represented by two coefficients for the linear and quadratic components of `poly(women, 2)`. The interpretation of coefficients of terms involving `type` depend on the contrasts used. Here, with the default treatment contrasts, they represent comparisons with `type = "bc"` as the reference level. It is not obvious how to understand the interaction effects like `log10(income):typeprof`.

```
lmtest::coeftest(prestige.mod3)
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -137.500    23.522  -5.85  8.2e-08 ***
#> education      2.959     0.582   5.09  2.0e-06 ***
#> poly(women, 2)1 28.339    10.190   2.78  0.0066 **
#> poly(women, 2)2 12.566     7.095   1.77  0.0800 .
#> log10(income)  40.326     6.714   6.01  4.1e-08 ***
#> typewc          0.969    39.495   0.02  0.9805
#> typeprof        74.276    30.736   2.42  0.0177 *
#> log10(income):typewc -1.073   10.638  -0.10  0.9199
#> log10(income):typeprof -17.725    7.947  -2.23  0.0282 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It is easiest to produce effect displays for all terms in the model using `allEffects()`, accepting all defaults. This gives (Figure 6.17) effect plots for the main effects of education and income and the interaction of income with type, with the non-focal variables held fixed. Each plot shows the fitted regression relation and a default

95% pointwise confidence band using the standard errors. Rug plots at the bottom show the locations of observations for the horizontal focal variable, which is useful when the observations are not otherwise plotted.

```
allEffects(prestige.mod3) |>
  plot()
```

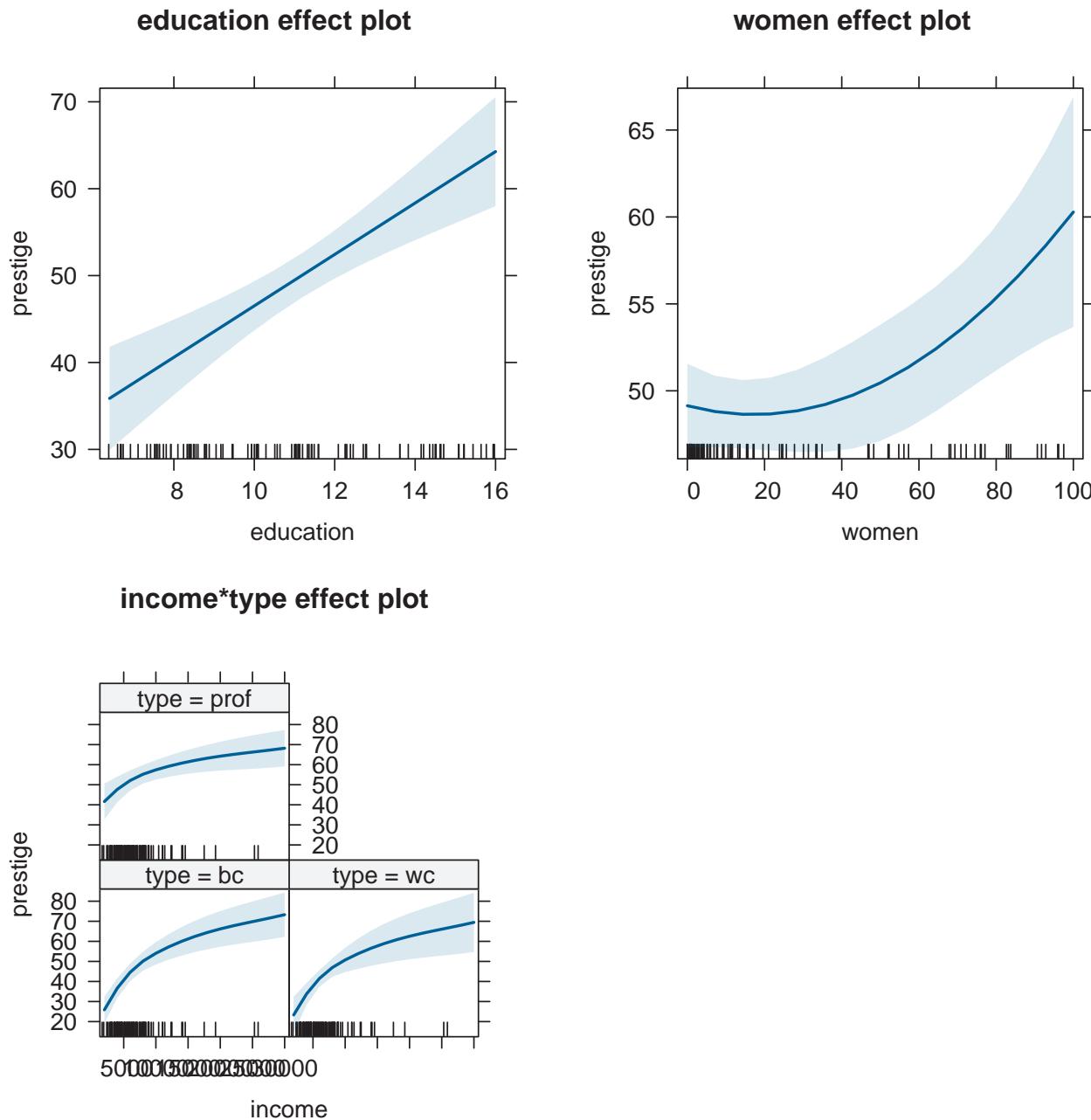


Figure 6.17: Predictor effect plot for all terms in the model with 95% confidence bands.

The effect for women, holding education, income and type constant looks to be quite strong and curved upwards. But note that these plots use different vertical scales for prestige in each plot and the range in the plot for women is much smaller than in the others. The interaction is graphed showing separate curves for the three levels of type.

For a more detailed look, it is useful to make separate plots for the predictors in the model, which allows customizing options for calculation and display. Partial residuals for the observations are computed by using `residuals = TRUE` in the call to `predictorEffects()`. The slope of the fitted line (in blue) is exactly coefficient for education in the full model. As with C+R plots, a smooth loess curve (in red) gives a visual assessment of linearity for a given predictor. A wide variety of graphing options are available in the call to `plot()`. Figure 6.18 shows the effect display for education with partial residuals and point identification of those points with the largest Mahalanobis distances from the centroid.

```
lattice:::trellis.par.set(par.xlab.text=list(cex=1.5),
                          par.ylab.text=list(cex=1.5))

predictorEffects(prestige.mod3, ~ education,
                 residuals = TRUE) |>
  plot(partial.residuals = list(pch = 16, col="blue"),
       id=list(n=4, col="black"))
```

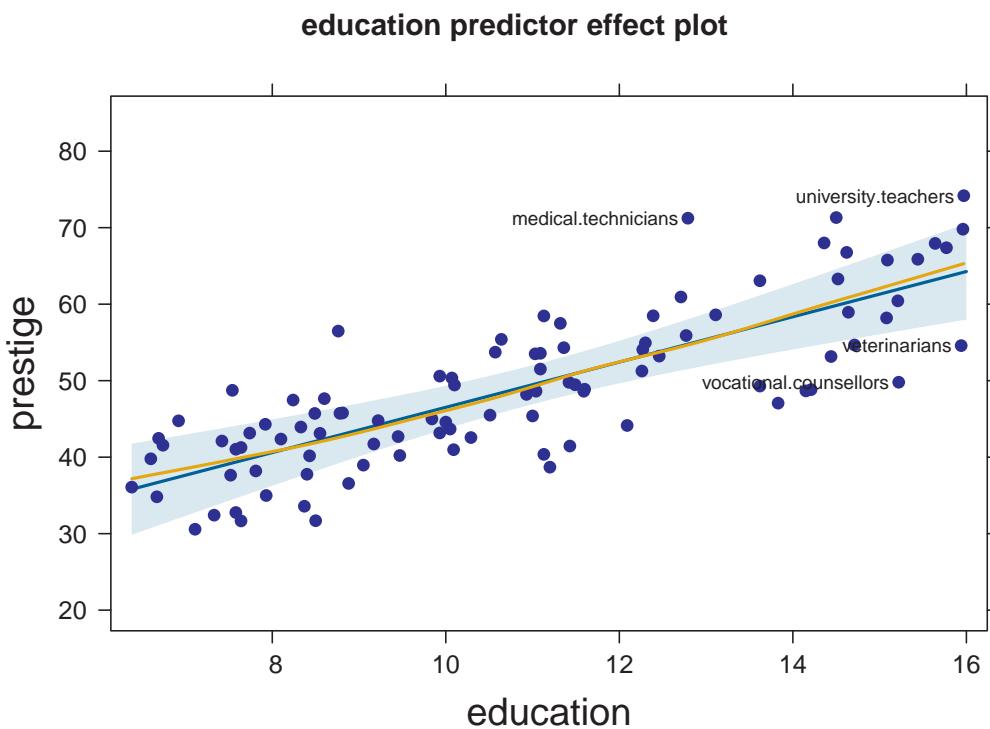


Figure 6.18: Predictor effect plot for education displaying partial residuals. The blue line shows the slice of the fitted regression surface where other variables are held fixed. The red curve shows a loess smooth of the partial residuals.

The effect plot for women in this model is shown in Figure 6.19. This uses the same vertical scale as in Figure 6.18, showing a more modest effect of percent women.

```
predictorEffects(prestige.mod3, ~women,
                 residuals = TRUE) |>
  plot(partial.residuals = list(pch = 16, col="blue", cex=0.8),
       id=list(n=4, col="black"))
```

Because of the interaction with `type`, the fitted effects for `income` are calculated for the three types of

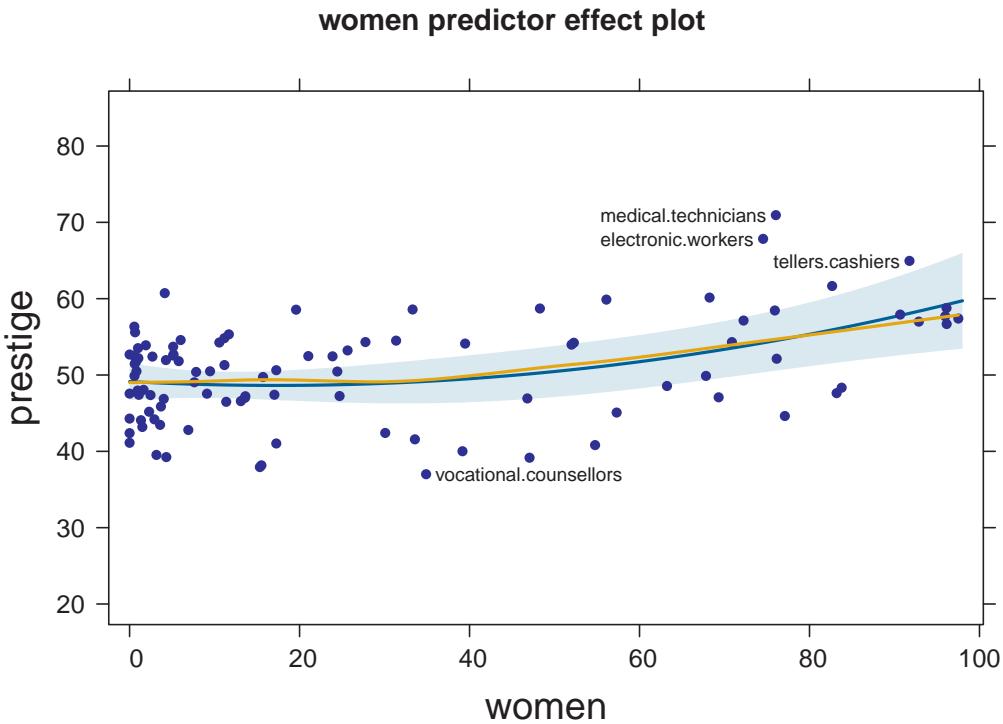


Figure 6.19: Predictor effect plot for women with partial residuals

occupation. It is easiest to compare these in the a single plot (using `multiline = TRUE`), rather than in separate panels as in Figure 6.17. Income is represented as `log10(income)` in the model `prestige.mod3`, and it is also easier to understand the interaction by plotting income on a log scale, using the `axes` argument to specify a transformation of the x axis. I use 68% confidence bands here to make the differences among type more apparent.

```
predictorEffects(prestige.mod3, ~ income,
                 confidence.level = 0.68) |>
  plot(lines=list(multiline=TRUE, lwd=3),
       confint=list(style="bands"),
       axes=list(
         x=list(income=list(transform=list(trans=log, inverse=exp))),
       key.args = list(x=.7, y=.35))
```

Figure 6.20 provides a clear interpretation of the interaction, represented by the coefficients shown above for `log10(income):typewc` and `log10(income):typeprof` in the model. Averaging over three occupation types, prestige increases linearly with log income with a coefficient of 40.33. This means that increasing income by 10% (say) gives an increase of $40.33/10 = 4.033$ in prestige. The slope for professional workers is less steep: the coefficient for `log10(income):typeprof` is -17.725. For these workers compared with blue collar jobs, prestige increases 1.77 less with a 10% increase in income. The difference in slopes for blue collar and white collar jobs is negligible.

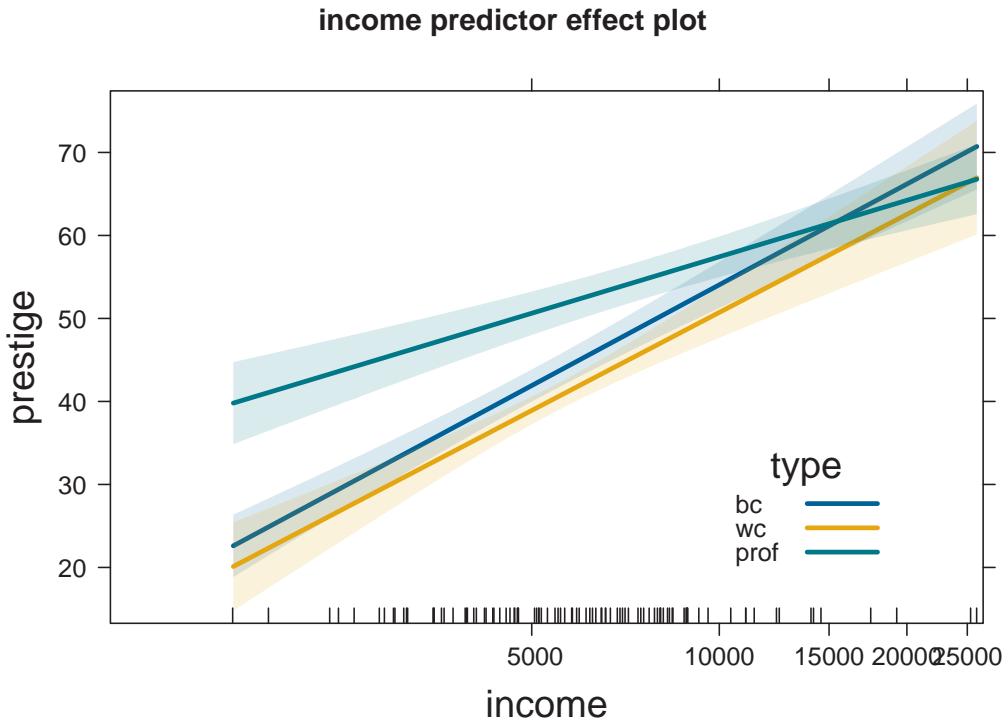


Figure 6.20: Predictor effect plot for income, plotted on a log scale.

6.6 Outliers, leverage and influence

In small to moderate samples, “unusual” observations can have dramatic effects on a fitted regression model, as we saw in the analysis of Davis’s data on reported and measured weight (Section 2.1.2) where one erroneous observations hugely altered the fitted line. As well, it turns out that two observations in Duncan’s data are unusual enough that removing them alters his conclusion that income and education have nearly equal effects on occupational prestige.

An observation can be unusual in three archetypal ways, with different consequences:

- Unusual in the response y , but typical in the predictor(s), \mathbf{x} — a badly fitted case with a large absolute residual, but with x not far from the mean, as in Figure 2.4. This case does not do much harm to the fitted model.
- Unusual in the predictor(s) \mathbf{x} , but typical in y — an otherwise well-fitted point. This case also does little harm, and in fact can be considered to improve precision, a “good leverage” point.
- Unusual in **both** \mathbf{x} and y — This is the case, a “bad leverage” point, revealed in the analysis of Davis’s data, Figure 2.3, where the one erroneous point for women was highly influential, pulling the regression line towards it and affecting the estimated coefficient as well as all the fitted values. In addition, subsets of observations can be *jointly* influential, in that their effects combine, or can mask each other’s influence.

Influential cases are the ones that matter most. As suggested above, to be influential an observation must be unusual in **both** \mathbf{x} and y , and affects the estimated coefficients, thereby also altering the predicted values for all observations. A heuristic formula capturing the relations among leverage, “outlyingness” on y and influence is

$$\text{Influence}_{\text{coefficients}} = X_{\text{leverage}} \times Y_{\text{residual}}$$

As described below, leverage is proportional to the squared distance $(x_i - \bar{x})^2$ of an observation x_i from its mean in simple regression and to the squared Mahalanobis distance in the general case. The Y_{residual} is best measured by a *studentized* residual, obtained by omitting each case i in turn and calculating its residual from the coefficients obtained from the remaining cases.

6.6.1 The leverage-influence quartet

These ideas can be illustrated in the “leverage-influence quartet” by considering a standard simple linear regression for a sample and then adding one additional point reflecting the three situations described above. Below, I generate a sample of $N = 15$ points with x uniformly distributed between (40, 60) and $y \sim 10 + 0.75x + \mathcal{N}(0, 1.25^2)$, duplicated four times.

```
library(tidyverse)
library(car)
set.seed(42)
N <- 15
case_labels <- paste(1:4, c("OK", "Outlier", "Leverage", "Influence"))
levdemo <- tibble(
  case = rep(case_labels,
             each = N),
  x = rep(round(40 + 20 * runif(N), 1), 4),
  y = rep(round(10 + .75 * x + rnorm(N, 0, 1.25), 4)),
  id = " "
)
mod <- lm(y ~ x, data=levdemo)
coef(mod)
#> (Intercept)           x
#>     13.332        0.697
```

The additional points, one for each situation are set to the values below.

- **Outlier:** (52, 60) a low leverage point, but an outlier (0) with a large residual
- **Leverage:** (75, 65) a “good” high leverage point (L) that fits well with the regression line
- **Influence:** (70, 40) a “bad” high leverage point (OL) with a large residual.

```
extra <- tibble(
  case = case_labels,
  x = c(65, 52, 75, 70),
  y = c(NA, 65, 65, 40),
  id = c(" ", "O", "L", "OL")
)

#' Join these to the data
both <- bind_rows(levdemo, extra) |>
  mutate(case = factor(case))
```

We can plot these four situations with `ggplot2` in panels faceted by `case` as shown below. The standard version of this plot shows the regression line for the `original data` and that for the `ammended data` with the additional point. Note that we use the `levdemo` dataset in `geom_smooth()` for the regression line with the original data, but specify `data = both` for that with the additional point.

```
ggplot(levdemo, aes(x = x, y = y)) +
  geom_point(color = "blue", size = 2) +
  geom_smooth(data = both,
              method = "lm", formula = y ~ x, se = FALSE,
              color = "red", linewidth = 1.3, linetype = 1) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE,
              color = "blue", linewidth = 1, linetype = "longdash") +
  stat_ellipse(data = both, level = 0.5, color="blue", type="norm", linewidth = 1.4) +
  geom_point(data=extra, color = "red", size = 4) +
  geom_text(data=extra, aes(label = id), nudge_x = -2, size = 5) +
  facet_wrap(~case, labeller = label_both) +
  theme_bw(base_size = 14)
```

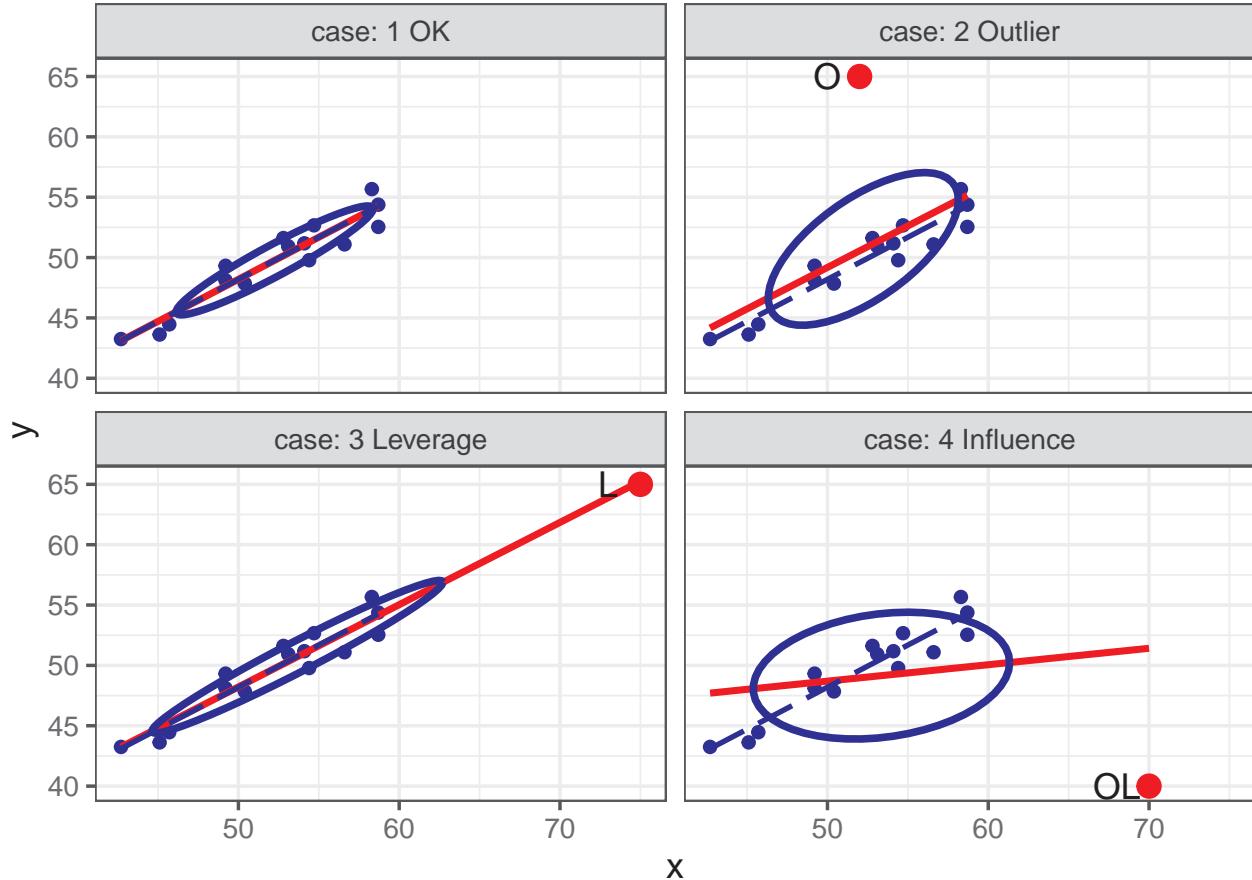


Figure 6.21: Leverage influence quartet with data 50% ellipses. Case (1) original data; (2) adding one low-leverage outlier, “O”; (3) adding one “good” leverage point, “L”; (4) adding one “bad” leverage point, “OL”. The dashed blue line is the fitted line for the original data, while the solid red line reflects the additional point. The data ellipses show the effect of the additional point on precision.

The standard version of this graph shows only the fitted regression lines in each panel. As can be seen, the fitted line doesn’t change very much in panels (2) and (3); only the bad leverage point, “OL” in panel (4) is harmful. Adding data ellipses to each panel immediately makes it clear that there is another part to this story—the effect of the unusual point on *precision* (standard errors) of our estimates of the coefficients.

Now, we see *directly* that there is a big difference in impact between the low-leverage outlier [panel (2)] and the high-leverage, small-residual case [panel (3)], even though their effect on coefficient estimates is negligible.

In panel (2), the single outlier inflates the estimate of residual variance (the size of the vertical slice of the data ellipse at \bar{x}), while in panel (3) this is decreased.

To allow direct comparison and make the added value of the data ellipse more apparent, we overlay the data ellipses from Figure 6.21 in a single graph, shown in Figure 6.22. Here, we can also see why the high-leverage point “L” (added in panel (c) of Figure 6.21) is called a “good leverage” point. By increasing the standard deviation of x , it makes the data ellipse somewhat more elongated, giving increased precision of our estimates of β .

```
colors <- c("black", "blue", "darkgreen", "red")
with(both,
  {dataEllipse(x, y, groups = case,
    levels = 0.68,
    plot.points = FALSE, add = FALSE,
    center.pch = "+",
    col = colors,
    fill = TRUE, fill.alpha = 0.1)
})

case1 <- both |> filter(case == "1 OK")
points(case1[, c("x", "y")], cex=1)

points(extra[, c("x", "y")],
  col = colors,
  pch = 16, cex = 2)

text(extra[, c("x", "y")],
  labels = extra$id,
  col = colors, pos = 2, offset = 0.5)
```

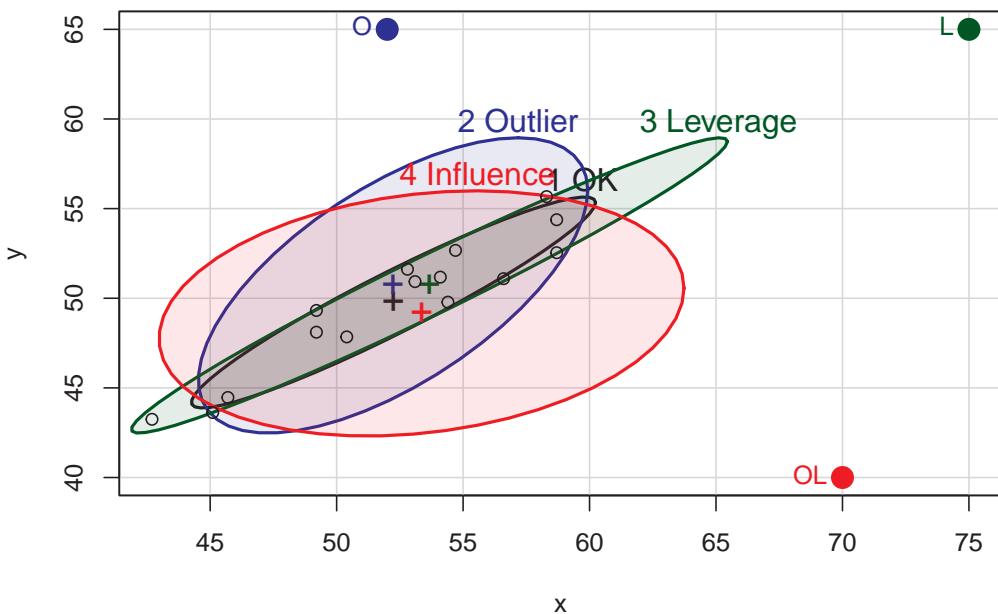


Figure 6.22: Data ellipses in the Leverage-influence quartet. This graph overlays the data ellipses and additional points from the four panels of Figure 6.22. It can be seen that only the OL point affects the slope, while the O and L points affect precision of the estimates in opposite directions.

6.6.1.1 Measuring leverage

Leverage is thus an index of the *potential* impact of an observation on the model due to its' atypical value in the X space of the predictor(s). It is commonly measured by the “hat” value, h_i , so called because it puts the hat ($\widehat{\bullet}$) on \mathbf{y} , i.e., the vector of fitted values can be expressed as

$$\begin{aligned}\hat{\mathbf{y}} &= \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\mathbf{H}} \mathbf{y} \\ &= \mathbf{H} \mathbf{y}.\end{aligned}$$

Here, $h_i \equiv h_{ii}$ are the diagonal elements of the Hat matrix \mathbf{H} . In simple regression, hat values are proportional to the squared distance of the observation x_i from the mean,

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_i (x_i - \bar{x})^2}, \quad (6.1)$$

and range from $1/n$ to 1, with an average value $\bar{h} = 2/n$. Consequently, observations with h_i greater than $2\bar{h}$ or $3\bar{h}$ are commonly considered to be of high leverage.

With $p \geq 2$ predictors, an analogous relationship holds, but the correlations among the predictors must be taken into account. It is demonstrated below that $h_i \propto D^2(\mathbf{x} - \bar{\mathbf{x}})$, the Mahalanobis squared distance of \mathbf{x} from the centroid $\bar{\mathbf{x}}$ ³.

The generalized version of Equation 6.1 is

$$h_i = \frac{1}{n} + \frac{1}{n-1} D^2(\mathbf{x} - \bar{\mathbf{x}}), \quad (6.2)$$

where $D^2(\mathbf{x} - \bar{\mathbf{x}}) = (\mathbf{x} - \bar{\mathbf{x}})^\top \mathbf{S}_X^{-1} (\mathbf{x} - \bar{\mathbf{x}})$. From Section 3.2, it follows that contours of constant leverage correspond to data ellipses or ellipsoids of the predictors in \mathbf{x} , whose boundaries, assuming normality, correspond to quantiles of the χ_p^2 distribution

Example:

To illustrate Equation 6.2, I generate $N = 100$ points from a bivariate normal distribution with means $\mu = (30, 30)$, variances = 10, and a correlation $\rho = 0.7$ and add two noteworthy points that show an apparently paradoxical result.

```
set.seed(421)
N <- 100
r <- 0.7
mu <- c(30, 30)
cov <- matrix(c(10, 10*r,
               10*r, 10), ncol=2)

X <- MASS::mvrnorm(N, mu, cov) |> as.data.frame()
colnames(X) <- c("x1", "x2")

# add 2 points
X <- rbind(X,
            data.frame(x1 = c(28, 38),
                       x2 = c(42, 35)))
```

³See this [Stats StackExchange discussion](#) for a proof.

The Mahalanobis squared distances of these points can be calculated using `heplots::Mahalanobis()`, and their corresponding hatvalues found using `hatvalues()` for any linear model using both `x1` and `x2`.

```
X <- X |>
  mutate(Dsq = heplots::Mahalanobis(X)) |>
  mutate(y = 2*x1 + 3*x2 + rnorm(nrow(X), 0, 5),
         hat = hatvalues(lm(y ~ x1 + x2)))
```

Plotting `x1` and `x2` with data ellipses shows the relation of leverage to squared distance from the mean. The blue point looks to be farther from the mean, but the red point is actually very much further by Mahalanobis squared distance, which takes the correlation into account; it thus has much greater leverage.

```
dataEllipse(X$x1, X$x2,
            levels = c(0.40, 0.68, 0.95),
            fill = TRUE, fill.alpha = 0.05,
            col = "darkgreen",
            xlab = "X1", ylab = "X2")
points(X[1:nrow(X) > N, 1:2], pch = 16,
       col=c("red", "blue"), cex = 2)
X |> slice_tail(n = 2) |>      # last two rows
points(pch = 16, col=c("red", "blue"), cex = 2)
```

The fact that hatvalues are proportional to leverage can be seen by plotting one against the other. I highlight the two noteworthy points in their colors from Figure 6.23 to illustrate how much greater leverage the red point has compared to the blue point.

```
plot(hat ~ Dsq, data = X,
      cex = c(rep(1, N), rep(2, 2)),
      col = c(rep("black", N), "red", "blue"),
      pch = 16,
      ylab = "Hatvalue",
      xlab = "Mahalanobis Dsq")
```

Look back at these two points in Figure 6.23. Can you guess how much further the red point is from the mean than the blue point? You might be surprised that its' D^2 and leverage are about five times as great!

```
X |> slice_tail(n=2)
#>   x1  x2   Dsq    y    hat
#> 1 28  42 25.65 179  0.2638
#> 2 38  35  4.95 175  0.0588
```

6.6.1.2 Outliers: Measuring residuals

From the discussion in Section 6.6, outliers for the response y are those observations for which the residual $e_i = y_i - \hat{y}_i$ are unusually large in magnitude. However, as demonstrated in Figure 6.21, a high-leverage point will pull the fitted line towards it, reducing its' residual and thus making them look less unusual.

The standard approach (R. D. Cook & Weisberg, 1982; Hoaglin & Welsch, 1978) is to consider a *deleted residual* $e_{(-i)}$, conceptually as that obtained by re-fitting the model with observation i omitted and obtaining the fitted value $\hat{y}_{(-i)}$ from the remaining $n - 1$ observations,

$$e_{(-i)} = y_i - \hat{y}_{(-i)}.$$

The (externally) *studentized residual* is then obtained by dividing $e_{(-i)}$ by its estimated standard error,

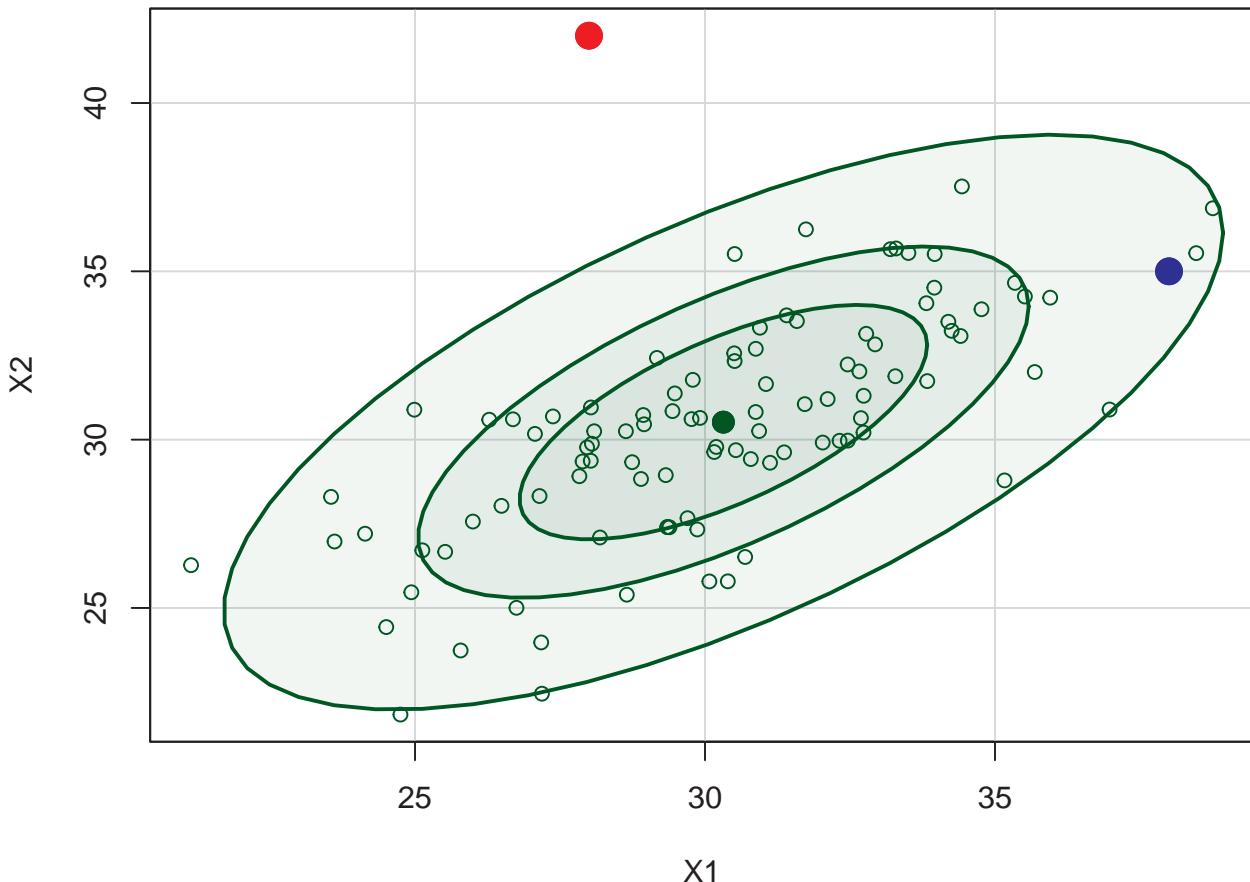


Figure 6.23: Data ellipses for a bivariate normal sample with correlation 0.7, and two additional noteworthy points. The blue point looks to be farther from the mean, but the red point is actually more than 5 times further by Mahalanobis squared distance, and thus has much greater leverage.

giving

$$e_{(-i)}^* = \frac{e_{(-i)}}{\text{sd}(e_{(-i)})} = \frac{e_i}{\sqrt{\text{MSE}_{(-i)} (1 - h_i)}} .$$

This is just the ordinary residual e_i divided by a factor that increases with the residual variance but decreases with leverage. It can be shown that these studentized residuals follow a t distribution with $n - p - 2$ degrees of freedom, so a value $|e_{(-i)}^*| > 2$ can be considered large enough to pay attention to.

In practice for classical linear models, it is unnecessary to actually re-fit the model n times. Velleman & Welsh (1981) show that all these leave-one-out quantities can be calculated from the model fitted to the full data set and the hat (projection) matrix $\mathbf{H} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ from which $\hat{\mathbf{b}} = \mathbf{Hy}$.

6.6.1.3 Measuring influence

As described at the start of this section, the actual influence of a given case depends multiplicatively on its' leverage and residual. But how can we measure it?

The essential idea introduced above, is to delete the observations one at a time, each time refitting the regression model on the remaining $n-1$ observations. Then, for observation i compare the results using all n observations to those with the i^{th} observation deleted to see how much influence the observation has on the analysis.

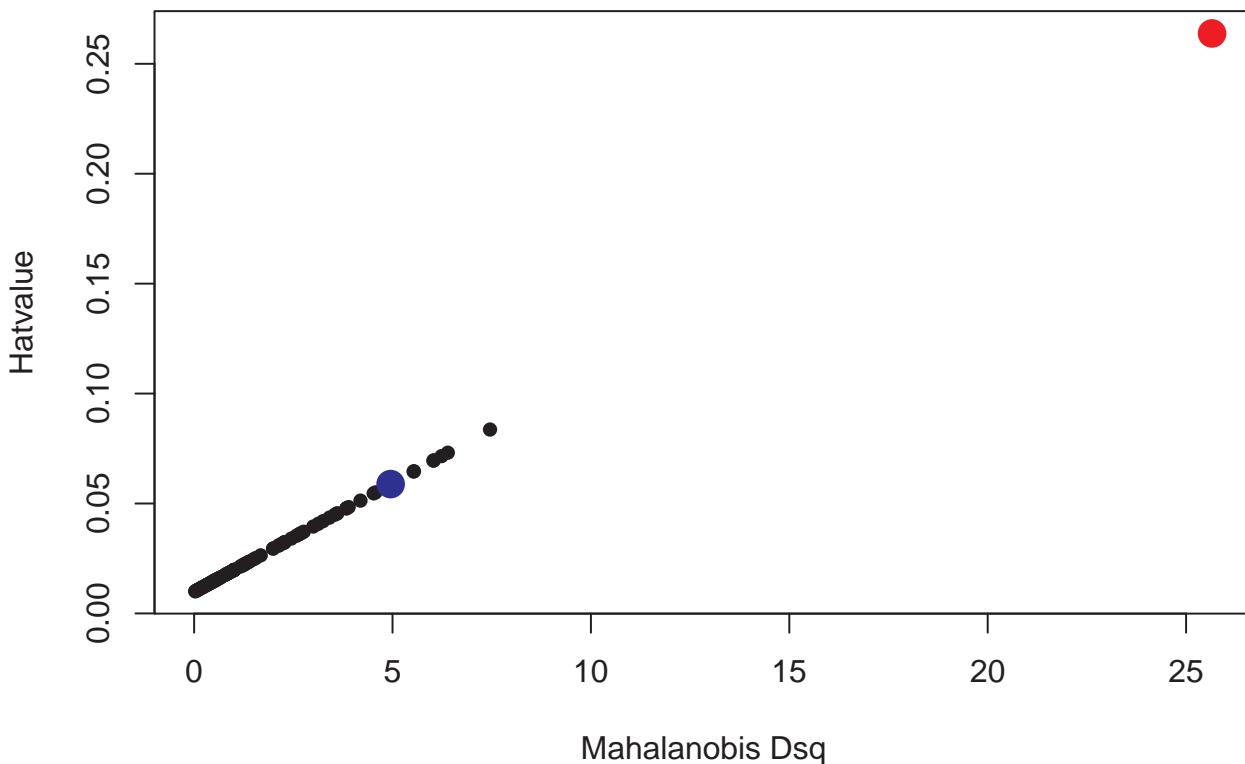


Figure 6.24: Hat values are proportional to squared Mahalanobis distances from the mean.

The simplest such measure, called DFFITS, compares the predicted value for case i with what would be obtained when that observation is excluded.

$$\begin{aligned} \text{DFFITS}_i &= \frac{\hat{y}_i - \hat{y}_{(-i)}}{\sqrt{\text{MSE}_{(-i)} h_i}} \\ &= e_{(-i)}^* \times \sqrt{\frac{h_i}{1 - h_i}} . \end{aligned}$$

The first equation gives the signed difference in fitted values in units of the standard deviation of that difference weighted by leverage; the second version (Belsley et al., 1980) represents that as a product of residual and leverage. A rule of thumb is that an observation is deemed to be influential if $|\text{DFFITS}_i| > 2\sqrt{(p+1)/n}$.

Influence can also be assessed in terms of the change in the estimated coefficients $\mathbf{b} = \hat{\beta}$ versus their values $\mathbf{b}_{(-i)}$ when case i is removed. Cook's distance, D_i , summarizes the size of the difference as a weighted sum of squares of the differences $\mathbf{d} = \mathbf{b} - \mathbf{b}_{(-i)}$ (R. D. Cook, 1977).

$$D_i = \mathbf{d}^\top (\mathbf{X}^\top \mathbf{X}) \mathbf{d} / (p+1)\hat{\sigma}^2$$

This can be re-expressed in terms of the components of residual and leverage

$$D_i = \frac{e_{(-i)}^{*2}}{p+1} \times \frac{h_i}{(1-h_i)} \quad (6.3)$$

Cook's distance is in the metric of an F distribution with p and np degrees of freedom, so values $D_i > 4/n$ are considered large.

6.6.2 Influence plots

The most common plot to detect influence is a bubble plot of the studentized residuals versus hat values, with the size (area) of the plotting symbol proportional to Cook's D . These plots are constructed using `car::influencePlot()` which also fills the bubble symbols with color whose opacity is proportional to Cook's D .

This is shown in Figure 6.25 for the demonstration dataset constructed in Section 6.6.1. In this plot, notable cutoffs for hatvalues at $2\bar{h}$ and $3\bar{h}$ are shown by dashed vertical lines and horizontal cutoffs for studentized residuals are shown at values of ± 2 .

The demonstration data of Section 6.6.1 has four copies of the same (x, y) data, three of which have an unusual observation. The influence plot in Figure 6.25 subsets the data to give the $19 = 15 + 4$ unique observations, including the three unusual cases. As can be seen, the high "Leverage" point has less influence than the point labeled "Influence", which has moderate leverage but a large absolute residual.

```
once <- both[c(1:16, 62, 63, 64),]      # unique observations
once.mod <- lm(y ~ x, data=once)
inf <- influencePlot(once.mod,
                      id = list(cex = 0.01),
                      fill.alpha = 0.5,
                      cex.lab = 1.5)

# custom labels
unusual <- bind_cols(once[17:19,], inf) |>
  print(digits=3)
#> # A tibble: 3 x 7
#>   case       x     y id   StudRes   Hat CookD
#> * <fct>    <dbl> <dbl> <chr>  <dbl> <dbl> <dbl>
#> 1 2 Outlier    52    65  0     3.11  0.0591 0.201
#> 2 3 Leverage    75    65  L     1.52  0.422  0.784
#> 3 4 Influence   70    40  0L    -4.93  0.262  1.82
with(unusual, {
  casetype <- gsub("\d ", "", case)
  text(Hat, StudRes, label = casetype,
       pos = c(4, 2, 3), cex=1.5)
})
```

6.6.3 Duncan data

Let's return to the `Duncan` data used as an example in Section 6.1.1 where a few points stood out as unusual in the basic diagnostic plots (Figure 6.2). The influence plot in Figure 6.26 helps to make sense of these noteworthy observations. The default method for identifying points in `influencePlot()` labels points with any of large studentized residuals, hat-values or Cook's distances.

```
inf <- influencePlot(duncan.mod, id = list(n=3),
                      cex.lab = 1.5)
```

`influencePlot()` returns (invisibly) the influence diagnostics for the cases identified in the plot. It is often useful to look at data values for these cases to understand why each of these was flagged.

```
merge(Duncan, inf, by="row.names", all.x = FALSE) |>
  arrange(desc(CookD)) |>
  print(digits=3)
```

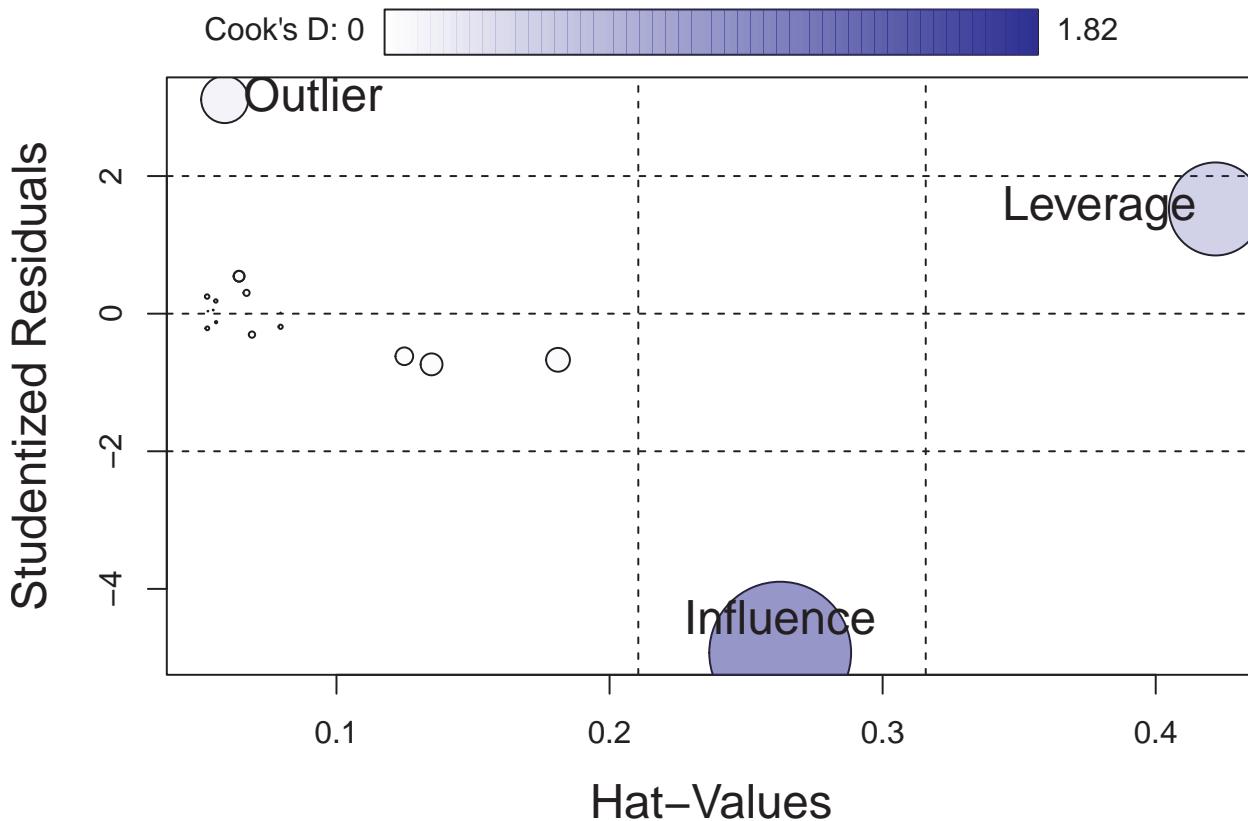


Figure 6.25: Influence plot for the demonstration data. The areas of the bubble symbols are proportional to Cook's D . The impact of the three unusual points on Cook's D is clearly seen.

```
#>      Row.names type income education prestige StudRes     Hat   CookD
#> 1    minister prof    21       84       87  3.135 0.1731 0.5664
#> 2 conductor wc     76       34       38 -1.704 0.1945 0.2236
#> 3 reporter  wc     67       87       52 -2.397 0.0544 0.0990
#> 4 RR.engineer bc     81       28       67  0.809 0.2691 0.0810
#> 5 contractor prof   53       45       76  2.044 0.0433 0.0585
```

- *minister* has by far the largest influence, because it has an extremely positive residual and a large hat value. Looking at the data, we see that ministers have very low income, so their prestige is under-predicted. The large hat value reflects the fact that ministers have low income combined with very high education.
- *conductor* has the next largest Cook's D . It has a large hat value because its combination of relatively high income and low education is unusual in the data.
- Among the others, *reporter* has a relatively large negative residual—its prestige is far less than the model predicts—but its low leverage make it not highly influential. *railroad engineer* has an extremely large hat value because its income is very high in relation to education. But this case is well-predicted and has a small residual, so its leverage is not large.

6.6.4 Influence in added-variable plots

The properties of added-variable plots discussed in Section 6.4 make them also useful for understanding why cases are influential because they control for other predictors in each plot, and therefore show the *partial*

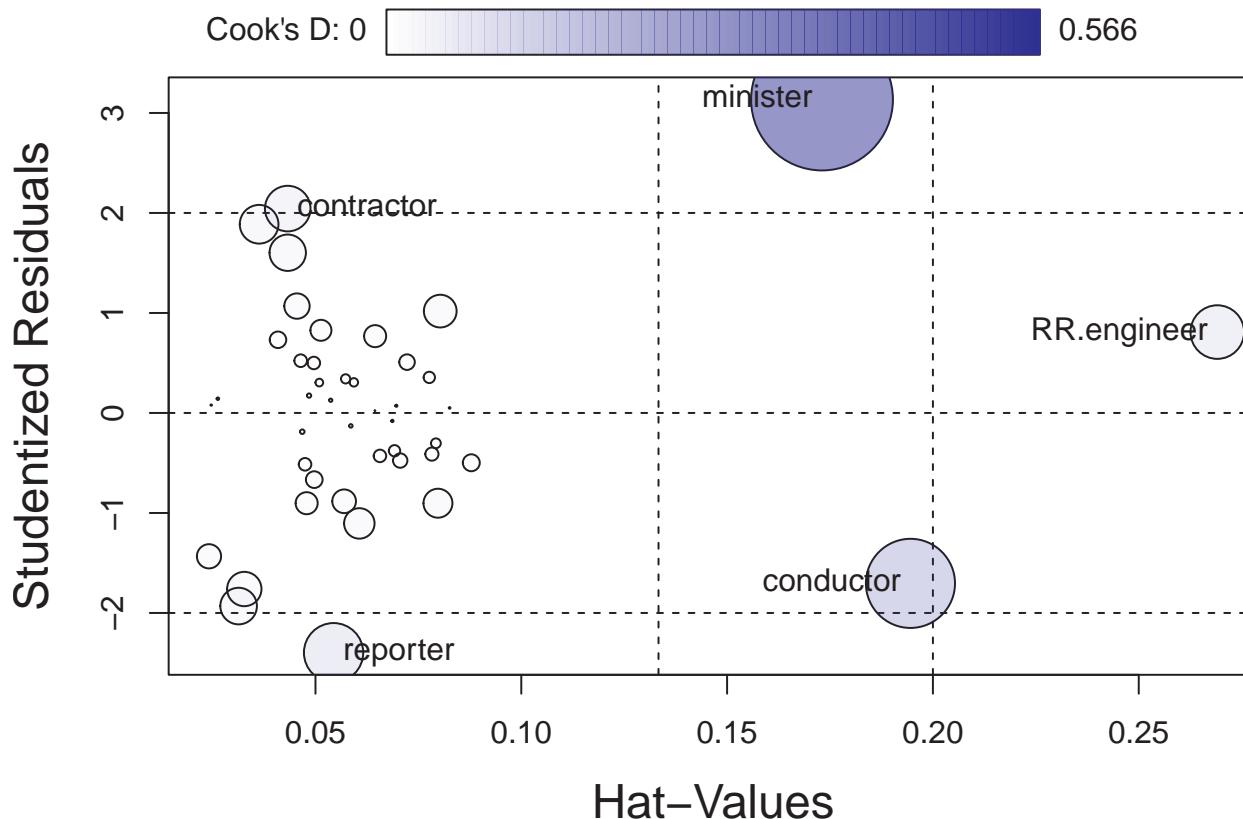


Figure 6.26: Influence plot for the model predicting occupational prestige in Duncan's data. Cases with large studentized residuals, hat-values or Cook's distances are labeled.

contributions of each observation to hat values and residuals. As a consequence, we can see directly the how individual cases become individually or jointly influential.

The Duncan data provides a particularly instructive example of this. Figure 6.27 shows the AV plots for both income and education in the model `duncan.mod`, with some annotations added. I want to focus here on the *joint* influence of the occupations minister and conductor which were seen to be the most influential in Figure 6.26. The green vertical lines show their residuals in each panel and the red lines show the regressions when these two observations are deleted.

The basic AV plots are produced using the call to `avPlots()` below. To avoid clutter, I use the argument `id = list(method = "mahal", n=3)` so that only the three points with the greatest Mahalanobis distances from the centroid in each plot are labeled. These are the cases with the largest leverage seen in Figure 6.26.

```
avPlots(duncan.mod,
  ellipse = list(levels = 0.68, fill = TRUE, fill.alpha = 0.1),
  id = list(method = "mahal", n=3),
  pch = 16, cex = 0.9,
  cex.lab = 1.5)
```

The two cases—minister and conductor—are the most highly influential, but as we can see in Figure 6.27 their influence combines because they are at opposite sides of the horizontal axis and their residuals are of opposite signs. They act together to decrease the slope for income and increase that for education.

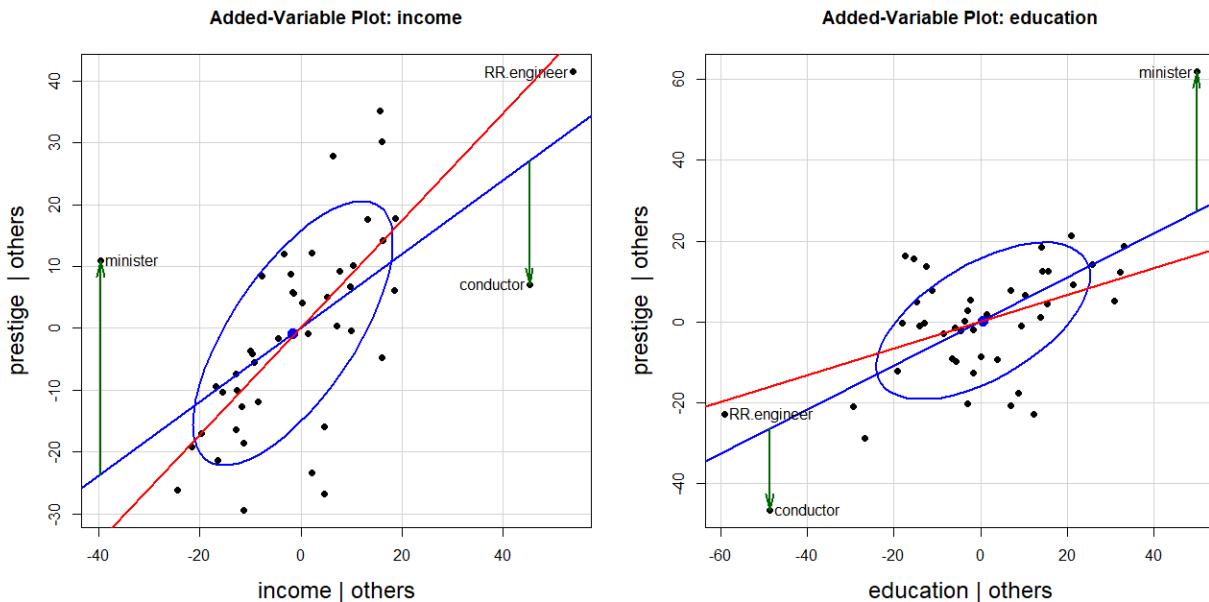


Figure 6.27: Added variable plots for the Duncan model, highlighting the impact of the observations for minister and conductor in each plot. The green lines show the residuals for these observations. The red line in each panel shows the regression line omitting these observations.

```

res <- avPlot(duncan.mod, "income",
               ellipse = list(levels = 0.68),
               id = list(method = "mahal", n=3),
               pch = 16,
               cex.lab = 1.5) |>
  as.data.frame()
fit <- lm(prestige ~ income, data = res)
info <- cbind(res, fitted = fitted(fit),
              resid = residuals(fit),
              hat = hatvalues(fit),
              cookd = cooks.distance(fit))

# noteworthy points in this plot
big <- which(info$cookd > .20)
with(info, {
  arrows(income[big], fitted[big], income[big], prestige[big],
         angle = 12, length = .18, lwd = 2, col = "darkgreen")
})

# line w/o the unusual points
duncan.mod2 <- update(duncan.mod, subset = -c(6, 16))
bs <- coef(duncan.mod2)["income"]
abline(a=0, b=bs, col = "red", lwd=2)

```

Duncan's hypothesis that the slopes for income and education were equal thus fails when these two observations are deleted. The slope for income then becomes 2.6 times that of education.

```
duncan.mod2 <- update(duncan.mod, subset = -c(6, 16))
coef(duncan.mod2)
#> (Intercept)      income   education
#>     -6.409       0.867      0.332
```

6.7 What have we learned?

This chapter unveils the sophisticated visual toolkit that transforms regression models from black boxes into transparent interpretable analyses. Here are the essential insights that will revolutionize how you understand and communicate model results:

- **The regression quartet reveals model reality beyond the summary statistics:** The four diagnostic plots (residuals vs. fitted, Q-Q plot, scale-location, and leverage plots) form a comprehensive health check for your regression models. Like a medical examination, each plot diagnoses different potential problems—linearity violations, non-normality, heteroscedasticity, and influential observations. The enhanced versions in R packages like `car` and `performance` don't just show problems; they label the troublemakers and guide you toward solutions.
- **Coefficient plots make model comparisons effortless and honest:** Raw coefficient tables hide the forest for the trees, especially when predictors have wildly different scales (years vs. dollars vs. percentages). Standardized coefficient plots and meaningful rescaling transform incomprehensible tables into intuitive visual comparisons. When Duncan's income coefficient (0.0013) appears invisible next to education's coefficient (4.19), the plot reveals they're actually comparable forces shaping occupational prestige.
- **Added-variable plots expose the truth behind “controlling for other variables”:** These magical plots perform visual surgery, removing the effects of all other predictors to show the pure relationship between response and focal predictor. They reveal Simpson's paradox in action—how coffee can appear harmful marginally but beneficial conditionally when stress is controlled. The relationship between marginal and conditional ellipses tells the complete story of confounding and adjustment.
- **Effect displays translate complex models into understandable stories:** When interactions, transformations, and multiple predictors make coefficient interpretation impossible, effect displays come to the rescue. They show exactly how prestige changes with income for different occupation types, or how the effects of education varies across meaningful ranges, controlling for other variables. These plots transform “significant at $p < 0.05$ ” into “here's exactly what this means in practice.”
- **Influence diagnostics separate the signal from the statistical noise:** The leverage-influence framework reveals which observations affect your model the most versus which are merely unusual. Through Cook's distance, hat values, and studentized residuals, we learned that `minister` and `conductor` in Duncan's data weren't just outliers—they were the puppet masters controlling his key findings about the equal effects of income and education. Remove them, and the entire conclusion changes.

The chapter's profound message is that regression modeling without visualization is like navigating without a compass. In our R-powered world of complex models and large datasets, sophisticated plotting techniques aren't just helpful—they're essential for understanding what your models actually say about the world.

Topics in Linear Models

The geometric and graphical approach of earlier chapters has already introduced some new ideas for thinking about multivariate data, models for explaining them, and graphical methods for understanding their results. These can be applied to better understand common problems that arise in data analysis.

Packages

In this chapter I use the following packages. Load them now:

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(patchwork)
library(car)
library(matlib)
library(forcats)
library(gganimate)
```

7.1 Ellipsoids in data space and β space

It is most common to look at data and fitted models in “data space,” where axes correspond to variables, points represent observations, and fitted models are plotted as lines (or planes) in this space. As we’ve suggested, data ellipsoids provide informative summaries of relationships in data space. For linear models, particularly regression models with quantitative predictors, there is another space—“ β space”—that provides deeper views of models and the relationships among them. This discussion extends Friendly et al. (2013), Sec. 4.6.

In β space, the axes pertain to coefficients, for example (β_0, β_1) in a simple linear regression. Points in this space are models (true, hypothesized, fitted) whose coordinates represent values of these parameters. For example, one point $\hat{\beta}_{OLS} = (\hat{\beta}_0, \hat{\beta}_1)$ represents the least squares estimate; other points, $\hat{\beta}_{WLS}$ and $\hat{\beta}_{ML}$ would give weighted least squares and maximum likelihood estimates, and the line $\beta_1 = 0$ represents the null hypothesis that the slope is zero.

In the sense described below, data space and β space are each *dual* to the other. In simple linear regression, for example:

- each line, like $\mathbf{y} = \beta_0 + \beta_1 \mathbf{x}$ with intercept β_0 and slope β_1 in data space corresponds to a point (β_0, β_1) in β space, and conversely;
- the set of points on any line $\beta_1 = x + y\beta_0$ in β space corresponds to a set of lines through a given point (x, y) in data space, and conversely;
- the geometric proposition that every pair of points defines a line in one space corresponds to the proposition that every two lines intersect in a point in the other space.

This is illustrated in Figure 7.1. The left panel shows three lines in data space, which can be expressed as

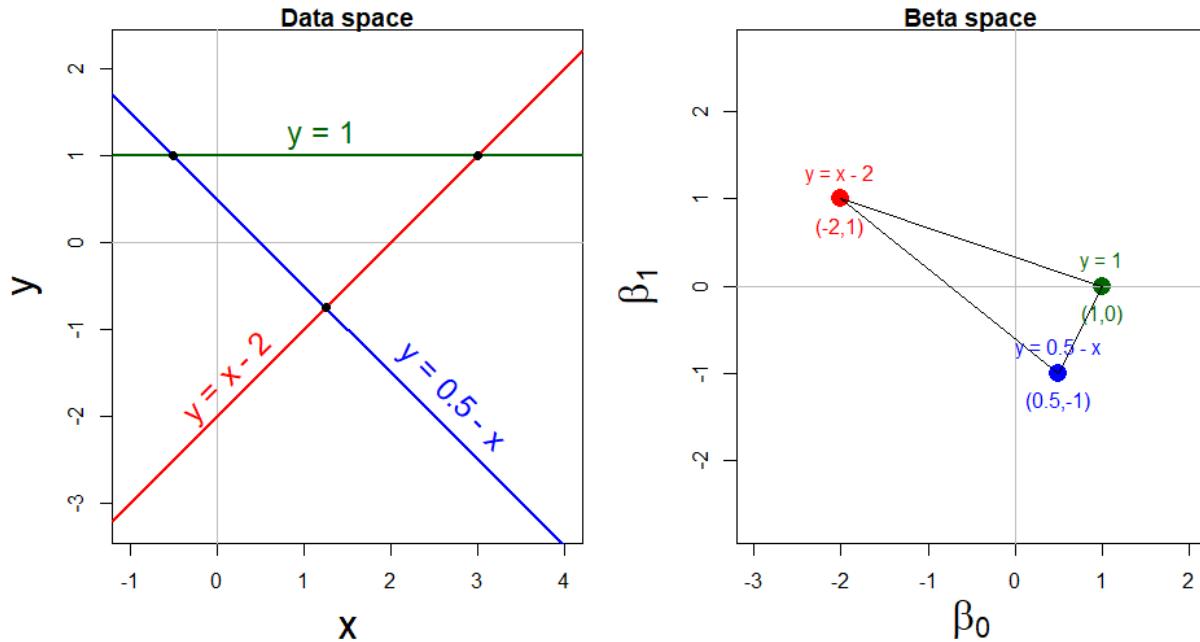


Figure 7.1: Duality of (x, y) lines in data space (left) and points in β -space (right). Each line in data space corresponds to a point, whose intercept and slope are shown in β -space.

linear equations in $\mathbf{z} = (x, y)$ of the form $\mathbf{A}\mathbf{z} = \mathbf{d}$. `matplotlib::showEqn(A, d)` prints these as equations in x and y .

```
A <- matrix(c( 1, 1, 0,
             -1, 1, 1), 3, 2)
d <- c(2, 1/2, 1)
showEqn(A, d, vars = c("x", "y"), simplify = TRUE)
#>   x - 1*y = 2
#>   x + y = 0.5
#> 0*x + y = 1
```

The first equation, $x - y = 2$ can be expressed as the line $y = x - 2$ and corresponds to the point $(\beta_0, \beta_1) = (-2, 1)$ in β space, and similarly for the other two equations. The second equation, $x + y = \frac{1}{2}$, or $y = 0.5 - x$ intersects the first at the point $(x, y) = (1.25, 0.75)$; this corresponds to the line connecting $(-2, 1)$ and $(0.5, -1)$ in β space.

This lovely duality is an example of an [important principle](#) in modern mathematics which translates concepts and structures from one perspective to another and back again. We get two views of the same thing, whose dual nature provides greater insight.

We have seen (Section 3.2) how ellipsoids in data space summarize variance (lack of precision) and correlation of our data. For the purpose of understanding linear models, ellipsoids in β space do the same thing for the estimates of parameters. These ellipsoids are dual and inversely related to each other, a point first made clear by Dempster (1969, Ch. 6):

- In data space, joint confidence intervals for the mean vector or joint prediction regions for the data are given by the ellipsoids $(\bar{x}_1, \bar{x}_2)^T \oplus c\sqrt{\mathbf{S}_{\mathbf{X}}}$, where the covariance matrix $\mathbf{S}_{\mathbf{X}}$ depends on $\mathbf{X}^T \mathbf{X}$ (\oplus here shifts the ellipsoid to one centered at (\bar{x}_1, \bar{x}_2) here, as in Equation 3.2).

- In the dual β space, joint confidence regions for the coefficients of a response variable y on (x_1, x_2) are given by ellipsoids of the form $\hat{\beta} \oplus c\sqrt{\mathbf{S}_\mathbf{X}^{-1}}$, and depend on $(\mathbf{X}^\top \mathbf{X})^{-1}$.

It is useful to understand the underlying geometry here connecting the ellipses for a matrix and its inverse. This can be seen in Figure 7.2, which shows an ellipse for a covariance matrix \mathbf{S} , whose axes, as we saw in Chapter 4 are the eigenvectors \mathbf{v}_i of \mathbf{S} and whose radii are the square roots $\sqrt{\lambda_i}$ of the corresponding eigenvalues. The comparable ellipse for $2\mathbf{S}$ has radii multiplied by $\sqrt{2}$.

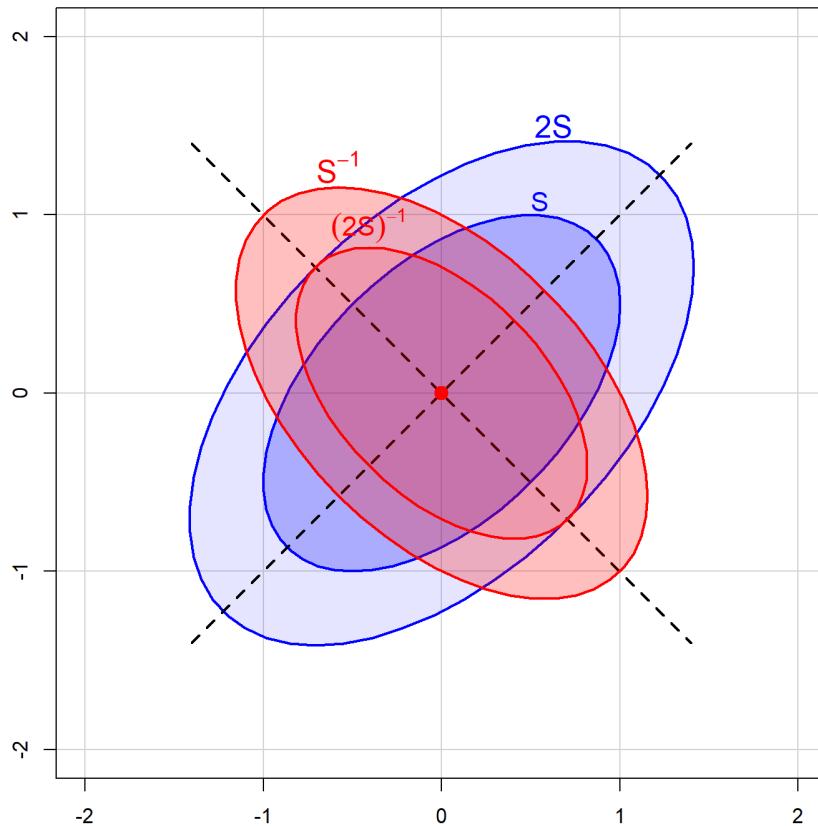


Figure 7.2: Geometric properties of an ellipse \mathbf{S} and its inverse, \mathbf{S}^{-1} . The principal axes (dotted lines) are given by the eigenvectors, which are the same for \mathbf{S} and \mathbf{S}^{-1} . Multiplying \mathbf{S} by 2 makes it's ellipse larger by $\sqrt{2}$, while the same factor makes the ellipse for $(2\mathbf{S})^{-1}$ smaller by the same factor.

As long as \mathbf{S} is of full rank, the eigenvectors of \mathbf{S}^{-1} are identical, while the eigenvalues are $1/\lambda_i$, so the radii are the reciprocals $1/\sqrt{\lambda_i}$. The analogous ellipse for $(2\mathbf{S}^{-1})$ is smaller by a factor of $\sqrt{2}$.

Thus, in two dimensions, the ellipse for \mathbf{S}^{-1} is a 90° rotation of that for \mathbf{S} . It is small in directions where the ellipse for \mathbf{S} is large, and vice-versa. In our statistical applications, this translates as: parameter estimates in β space are more precise (have less variance) in the directions where the data are more widely dispersed, having more information about the relationship.

We illustrate these ideas in the example below.

7.1.1 Coffee, stress and heart disease

Consider the dataset `coffee`, giving measures of `Heart` (y), an index of cardiac damage, `Coffee` (x_1), a measure of daily coffee consumption, and `Stress` (x_2), a measure of occupational stress, in a contrived sample

of $n = 20$ university people.¹ For the sake of the example we assume that the main goal is to determine whether or not coffee is good or bad for your heart, and stress represents one potential confounding variable among others (age, smoking, etc.) that might be useful to control statistically.

```
set.seed(1234)
data(coffee, package="matlib")
coffee |> dplyr::sample_n(6)
#>      Group Coffee Stress Heart
#> 1 Grad_Student    104    117    92
#> 2     Student      52     86    63
#> 3 Grad_Student    76     92    58
#> 4     Student     100    123    92
#> 5     Student      64     74    63
#> 6 Professor      141    175   145
```

Figure 7.3 shows the scatterplot matrix, giving the marginal relations between all pairs of variables. The marginal message seems to be that coffee is bad for your heart, stress is bad for your heart and coffee consumption is also related to occupational stress.

```
scatterplotMatrix(~ Heart + Coffee + Stress, data=coffee,
  smooth = FALSE,
  pch = 16, col = "brown",
  cex.axis = 1.3, cex.labels = 3,
  ellipse = list(levels = 0.68, fill.alpha = 0.1))
```

Yet, when we fit both variables together, we obtain the following results, suggesting that coffee is good for you—the coefficient for coffee is now negative, though non-significant. How can this be?

```
coffee.mod <- lm(Heart ~ Coffee + Stress, data=coffee)
broom::tidy(coffee.mod)
#> # A tibble: 3 x 5
#>   term       estimate std.error statistic  p.value
#>   <chr>     <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept) -7.79      5.79     -1.35  0.196
#> 2 Coffee      -0.409     0.292     -1.40  0.179
#> 3 Stress       1.20      0.224      5.34  0.0000536
```

The answer is that the marginal plots of `Heart` vs. `Coffee` and `Stress` in the first row of Figure 7.3 each ignore the other predictor. In contrast, the coefficients for coffee and stress in the multiple regression model `coffee.mod` are *partial* coefficients, giving the estimated change in heart damage for a unit change in each predictor, but **adjusting for** (controlling for, or holding constant) the other predictor.

We can see these effects directly in **added variable plots** (Section 6.4), but here I consider the relationship of coffee and stress in data space and beta space and how their ellipses relate to each other and to hypothesis tests.

The left panel in Figure 7.4 is the same as that in the (3,2) cell of Figure 7.3 for the relation `Stress ~ Coffee` but with data ellipses of 40% and 60% coverage. The shadows of the 40% ellipse on any axis give univariate intervals of the mean $\bar{x} \pm 1s_x$ (standard deviation) shown by the thick red lines; the shadow of the 68% ellipse corresponds to an interval $\bar{x} \pm 1.5s_x$.

The right panel shows the joint 95% confidence region for the coefficients $(\beta_{\text{Coffee}}, \beta_{\text{Stress}})$ and individual confidence intervals in β space. These are determined as

¹This example was developed by Georges Monette.

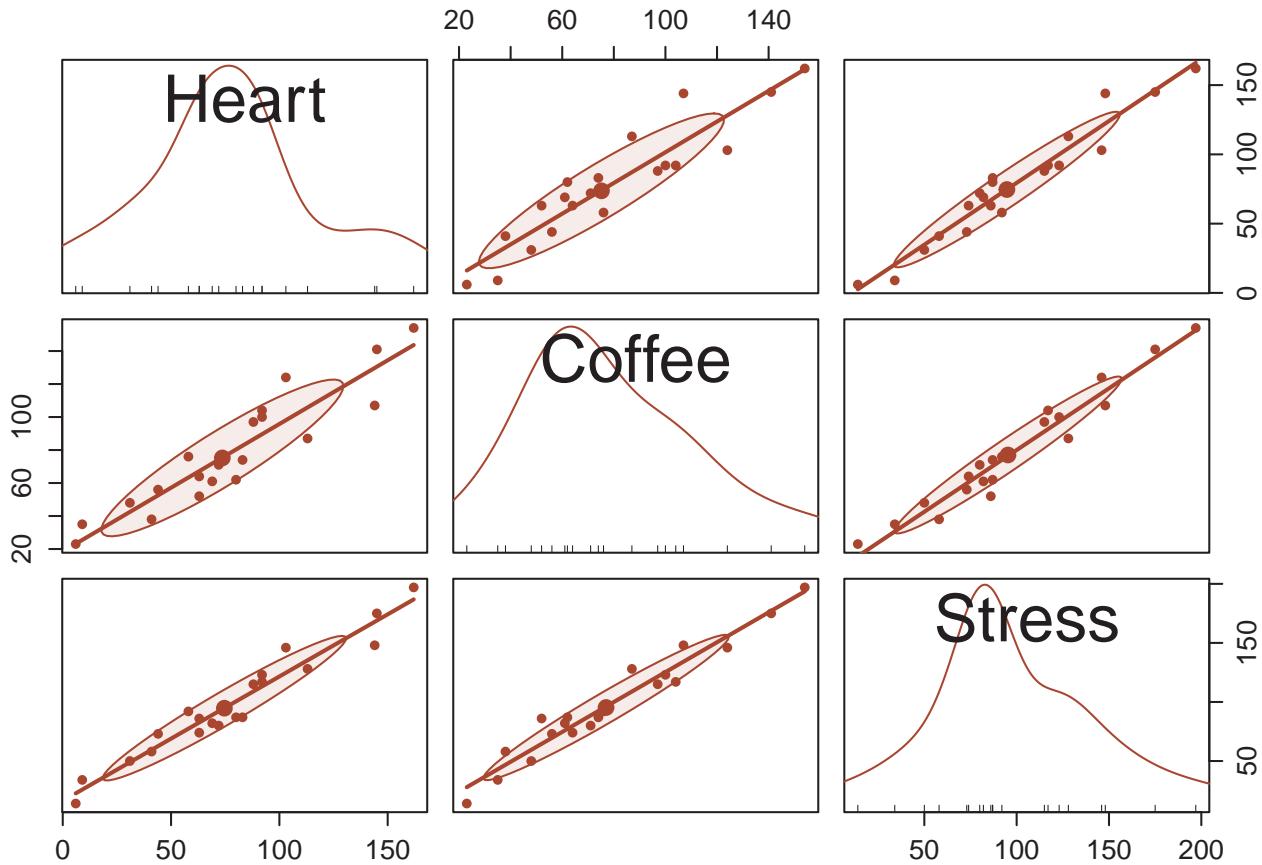


Figure 7.3: Scatterplot matrix for the `coffee` data showing the pairwise relationships among Heart damage (y), Coffee consumption (x_1), and Stress (x_2), with linear regression lines and 68% data ellipses.

$$\hat{\beta} \oplus \sqrt{dF_{d,\nu}^{.95}} \times s_e \times \mathbf{S}_X^{-1/2} .$$

where d is the number of dimensions for which we want coverage, ν is the residual degrees of freedom for s_e , and \mathbf{S}_X is the covariance matrix of the predictors.

Thus, the blue ellipse in Figure 7.4 (right) is the ellipse of **joint** 95% coverage, using the factor $\sqrt{2F_{2,\nu}^{.95}}$, which covers the true values of $(\beta_{\text{Stress}}, \beta_{\text{Coffee}})$ in 95% of samples. Moreover:

- Any *joint* hypothesis (e.g., $\mathcal{H}_0 : \beta_{\text{Stress}} = 0, \beta_{\text{Coffee}} = 0$) can be tested visually, simply by observing whether the hypothesized point, $(0, 0)$ here, lies inside or outside the joint confidence ellipse. That hypothesis is rejected
- The shadows of this ellipse on the horizontal and vertical axes give Scheff'e joint 95% confidence intervals for the parameters, with protection for simultaneous inference ("fishing") in a 2-dimensional space.
- Similarly, using the factor $\sqrt{F_{1,\nu}^{1-\alpha/d}} = t_{\nu}^{1-\alpha/2d}$ would give an ellipse whose 1D shadows are $1 - \alpha$ Bonferroni confidence intervals for d posterior hypotheses.

Visual hypothesis tests and $d = 1$ confidence intervals for the parameters *separately* are obtained from the red ellipse in Figure 7.4, which is scaled by $\sqrt{F_{1,\nu}^{.95}} = t_{\nu}^{.975}$. We call this the *confidence-interval generating ellipse* (or, more compactly, the "confidence-interval ellipse"). The shadows of the confidence-interval ellipse on the axes (thick red lines) give the corresponding individual 95% confidence intervals, which are equivalent to the (partial, Type III) t -tests for each coefficient given in the standard multiple regression output shown above.

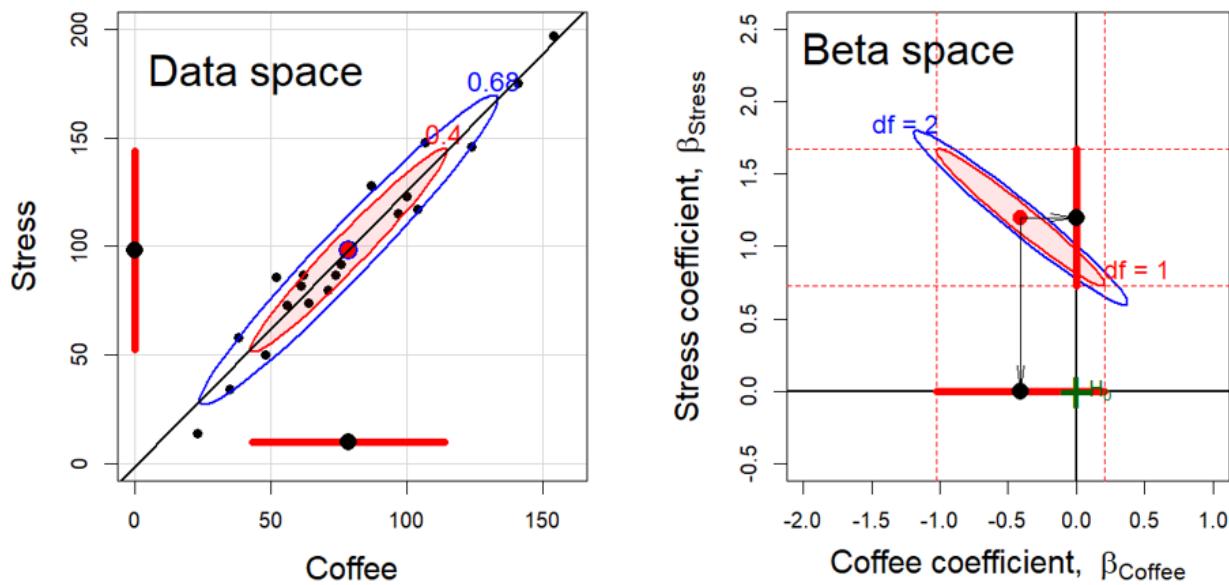


Figure 7.4: Data space and β space representations of Coffee and Stress. Left: 40% and 68% data ellipses. Right: Joint 95% confidence ellipse (blue) for $(\beta_{\text{Coffee}}, \beta_{\text{Stress}})$, confidence interval generating ellipse (red) with 95% univariate shadows. H_0 marks the joint hypothesis that both coefficients equal zero.

Thus, controlling for Stress, the confidence interval for the slope for Coffee includes 0, so we cannot reject the hypothesis that $\beta_{\text{Coffee}} = 0$ in the multiple regression model, as we saw above in the numerical output. On the other hand, the interval for the slope for Stress excludes the origin, so we reject the null hypothesis that $\beta_{\text{Stress}} = 0$, controlling for Coffee consumption.

Finally, consider the relationship between the data ellipse and the confidence ellipse. These have exactly the same shape, but (with equal coordinate scaling of the axes), the confidence ellipse is exactly a 90° rotation and rescaling of the data ellipse. In directions in data space where the slice of the data ellipse is wide—where we have more information about the relationship between Coffee and Stress—the projection of the confidence ellipse is narrow, reflecting greater precision of the estimates of coefficients. Conversely, where slice of the data ellipse is narrow (less information), the projection of the confidence ellipse is wide (less precision).

Confidence ellipses are drawn using `car::confidenceEllipse()`. Click the button to show the code.

```
confidenceEllipse(coffee.mod,
  grid = FALSE,
  xlim = c(-2, 1), ylim = c(-0.5, 2.5),
  xlab = expression(paste("Coffee coefficient, ", beta["Coffee"])),
  ylab = expression(paste("Stress coefficient, ", beta["Stress"])),
  cex.lab = 1.5)
confidenceEllipse(coffee.mod, add=TRUE, draw = TRUE,
  col = "red", fill = TRUE, fill.alpha = 0.1,
  dfn = 1)
abline(h = 0, v = 0, lwd = 2)

# confidence intervals
beta <- coef( coffee.mod )[-1]
CI <- confint(coffee.mod)
lines( y = c(0,0), x = CI["Coffee",] , lwd = 6, col = 'red')
```

```

lines( x = c(0,0), y = CI["Stress"], lwd = 6, col = 'red')
points( diag( beta ), col = 'black', pch = 16, cex=1.8)

abline(v = CI["Coffee"], col = "red", lty = 2)
abline(h = CI["Stress"], col = "red", lty = 2)

text(-2.1, 2.35, "Beta space", cex=2, pos = 4)
arrows(beta[1], beta[2], beta[1], 0, angle=8, len=0.2)
arrows(beta[1], beta[2], 0, beta[2], angle=8, len=0.2)

text( -1.5, 1.85, "df = 2", col = 'blue', adj = 0, cex=1.2)
text( 0.2, .85, "df = 1", col = 'red', adj = 0, cex=1.2)

heplots::mark.H0(col = "darkgreen", pch = "+", lty = 0, pos = 4, cex = 3)

```

7.2 Measurement error

7.2.1 OLS is BLUE

In classical linear models, the predictors are often considered to be fixed variables, or, if random, to be measured without error and independent of the regression errors. Either condition, along with the assumption of linearity, guarantees that the standard OLS estimators are *unbiased*. That is, in a simple linear regression, $y = \beta_0 + \beta_1 x + \epsilon$, the estimated slope $\hat{\beta}_1$ will have an average, expected value $\mathcal{E}(\hat{\beta}_1)$ equal to the true population value β_1 over repeated samples.

Not only this, but the Gauss-Markov theorem guarantees that the OLS estimator is also the most *efficient* because it has the least variance among all linear and unbiased estimators. The classical OLS estimator is said to be **BLUE**: **B**est (lowest variance), **L**inear (among linear estimators), **U**nbiased, **E**stimator.

7.2.2 Errors in predictors

Errors in the response y are accounted for in the model and measured by the mean squared error, $MSE = \hat{\sigma}_\epsilon^2$. But in practice, of course, predictor variables are often also observed indicators, subject to their own error. Indeed, in the behavioral sciences it is rare that predictors are perfectly reliable and measured exactly. This fact that is recognized in errors-in-variables regression models (Fuller, 2006) and in more general structural equation models, but often ignored otherwise. Ellipsoids in data space and β space are well suited to showing the effect of measurement error in predictors on OLS estimates.

The statistical facts are well known, though perhaps counter-intuitive in certain details: measurement error in a predictor biases regression coefficients (towards 0), while error in the measurement in y increases the MSE and thus standard errors of the regression coefficients but does not introduce bias in the coefficients.

7.2.2.1 Example

An illuminating example can be constructed by starting with the simple linear regression

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i ,$$

where x_i is the *true*, fully reliable predictor and y is the response, with error variance σ_ϵ^2 . Now consider that we don't measure x_i exactly, but instead observe x_i^* .

$$x_i^* = x_i + \eta_i ,$$

where the measurement error η_i is independent of the true x_i with variance σ_η^2 . We can extend this example to also consider the effect of adding additional, independent error variance to y , so that instead of y_i we observe

$$y_i^* = y_i + \nu_i$$

with variance σ_ν^2 .

Let's simulate an example where the true relation is $y = 0.2 + 0.3x$ with error standard deviation $\sigma = 0.5$. I'll take x to be uniformly distributed in $[0, 10]$ and calculate y as normally distributed around that linear relation.

```
set.seed(123)
n <- 300

a <- 0.2    # true intercept
b <- 0.3    # true slope
sigma <- 0.5 # baseline error standard deviation

x <- runif(n, 0, 10)
y <- rnorm(n, a + b*x, sigma)
demo <- data.frame(x,y)
```

Then, generate alternative values x^* and y^* with additional error standard deviations around x given by $\sigma_\eta = 4$ and around y given by $\sigma_\nu = 1$.

```
err_y <- 1    # additional error stdev for y
err_x <- 4    # additional error stdev for x
demo <- demo |>
  mutate(y_star = rnorm(n, y, err_y),
        x_star = rnorm(n, x, err_x))
```

There are four possible models we could fit and compare, using the combinations of (x, x^*) and (y, y^*)

```
fit_1 <- lm(y ~ x,           data = demo)    # no additional error
fit_2 <- lm(y_star ~ x,      data = demo)    # error in y
fit_3 <- lm(y ~ x_star,     data = demo)    # error in x
fit_4 <- lm(y_star ~ x_star, data = demo)    # error in x and y
```

However, to show the differences visually, we can simply plot the data for each pair and show the regression lines (with confidence bands) and the data ellipses. To do this efficiently with `ggplot2`, it is necessary to transform the `demo` data to long format with columns `x` and `y`, distinguished by `name` for the four combinations.

```
# make the demo dataset long, with names for the four conditions
df <- bind_rows(
  data.frame(x=demo$x,       y=demo$y,       name="No measurement error"),
  data.frame(x=demo$x,       y=demo$y_star,   name="Measurement error on y"),
  data.frame(x=demo$x_star,  y=demo$y,       name="Measurement error on x"),
  data.frame(x=demo$x_star,  y=demo$y_star,   name="Measurement error on x and y")) |>
  mutate(name = fct_inorder(name))
```

Then, we can plot the data in `df` with points, regression lines and a data ellipse, facetting by `name` to give the **measurement error quartet**.

```
ggplot(df, aes(x, y)) +
  geom_point(alpha = 0.2) +
  stat_ellipse(geom = "polygon",
               color = "blue", fill= "blue",
               alpha=0.05, linewidth = 1.1) +
  geom_smooth(method="lm", formula = y~x, fullrange=TRUE, level=0.995,
              color = "red", fill = "red", alpha = 0.2) +
  facet_wrap(~name)
```

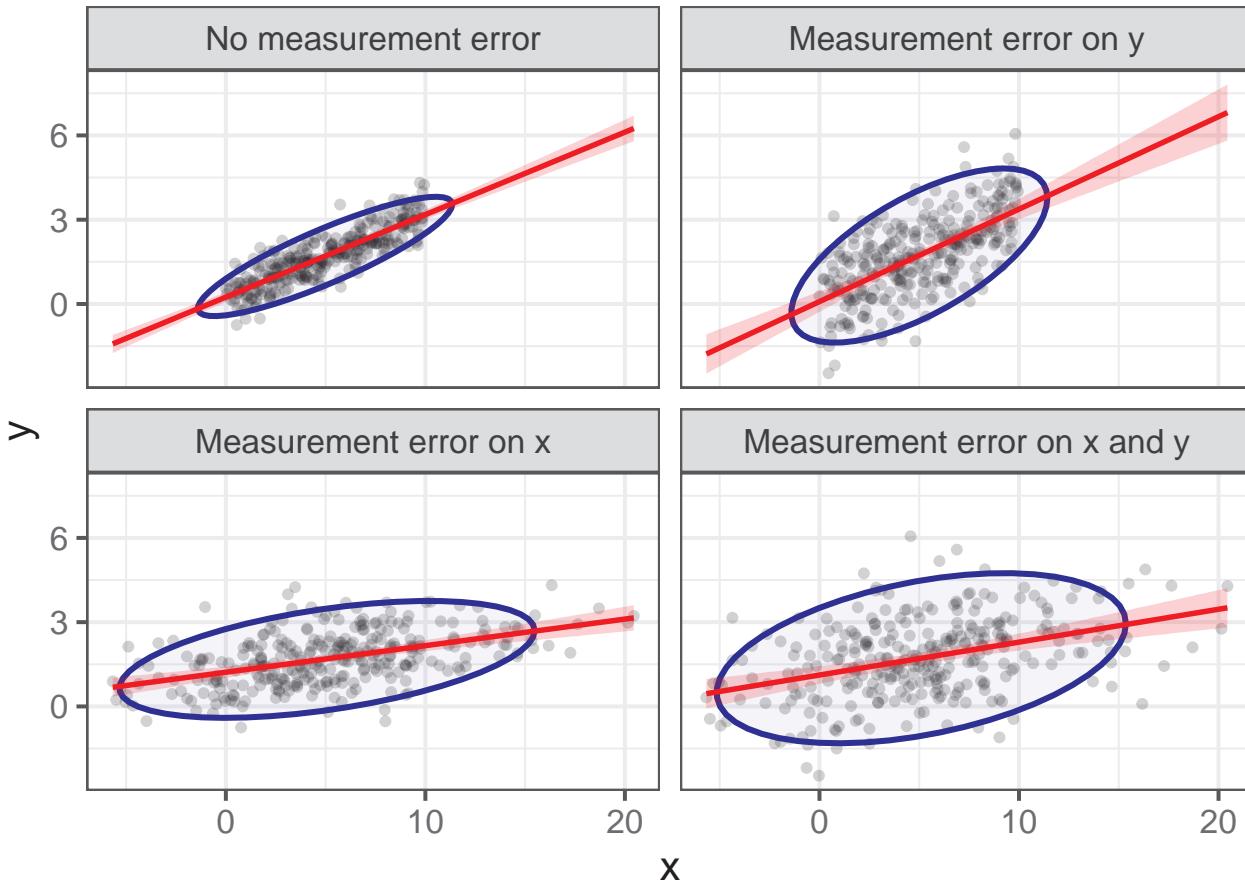


Figure 7.5: The measurement error quartet: Each plot shows the linear regression of y on x , but where additional error variance has been added to y or x or both. The widths of the confidence bands and the vertical extent of the data ellipses show the effect on precision.

Comparing the plots in the first row, you can see that when additional error is added to y , the regression slope remains essentially unchanged, illustrating that the estimate is unbiased. However, the confidence bounds on the regression line become wider, and the data ellipse becomes fatter in the y direction, illustrating the loss of precision.

The effect of error in x is less kind. Comparing the first row of plots with the second row, you can see that the estimated slope decreases when errors are added to x . This is called *attenuation bias*, and it can be shown that

$$\hat{\beta}_{x^*} \longrightarrow \frac{\beta}{1 + \sigma_\eta^2/\sigma_x^2},$$

where β here refers to the regression slope and \longrightarrow means “converges to”, as the sample size gets large. Thus, as σ_η^2 increases, $\hat{\beta}_{x^*}$ becomes less than β .

Beyond plots like Figure 7.5, we can see the effects of error in x or y on the model summary statistics such as the correlation r_{xy} or MSE by extracting these from the fitted models. This is easily done using `dplyr::nest_by(name)` and fitting the regression model to each subset, from which we can obtain the model statistics using `sigma()`, `coef()` and so forth. A bit of `dplyr::mutate()` magic is used to construct indicators `errX` and `errY` giving whether or not error was added to x and/or y .

```
model_stats <- df |>
  dplyr::nest_by(name) |>
  mutate(model = list(lm(y ~ x, data = data)),
         sigma = sigma(model),
         intercept = coef(model)[1],
         slope = coef(model)[2],
         r = sqrt(summary(model)$r.squared)) |>
  mutate(errX = stringr::str_detect(name, " x"),
         errY = stringr::str_detect(name, " y")) |>
  mutate(errX = factor(errX, levels = c("TRUE", "FALSE")),
         errY = factor(errY, levels = c("TRUE", "FALSE"))) |>
  relocate(errX, errY, r, .after = name) |>
  select(-data) |>
  print()
#> # A tibble: 4 x 8
#> # Rowwise:  name
#>   name          errX  errY      r model sigma intercept slope
#>   <fct>        <fct> <fct> <dbl> <lis> <dbl>     <dbl> <dbl>
#> 1 No measurement err~ FALSE FALSE 0.858 <lm>  0.495    0.244  0.294
#> 2 Measurement error ~ FALSE TRUE  0.648 <lm>  1.09     0.0838 0.329
#> 3 Measurement error ~ TRUE  FALSE 0.481 <lm>  0.844    1.22   0.0946
#> 4 Measurement error ~ TRUE  TRUE  0.401 <lm>  1.31     1.12   0.117
```

We plot the model $R = r_{xy}$ and the estimated residual standard error in Figure 7.6 below. The lines connecting the points are approximately parallel, indicating that errors of measurement in x and y have nearly additive effects on model summaries.

```
p1 <- ggplot(data=model_stats,
              aes(x = errX, y = r,
                  group = errY, color = errY,
                  shape = errY, linetype = errY)) +
  geom_point(size = 4) +
  geom_line(linewidth = 1.2) +
  labs(x = "Error on X?",
       y = "Model R",
       color = "Error on Y?",
       shape = "Error on Y?",
       linetype = "Error on Y?")
  legend_inside(c(0.25, 0.8))

p2 <- ggplot(data=model_stats,
              aes(x = errX, y = sigma,
                  group = errY, color = errY,
                  shape = errY, linetype = errY)) +
  geom_point(size = 4) +
  geom_line(linewidth = 1.2) +
```

```

  labs(x = "Error on X?",
       y = "Model residual standard error",
       color = "Error on Y?",
       shape = "Error on Y?",
       linetype = "Error on Y?")

p1 + p2

```

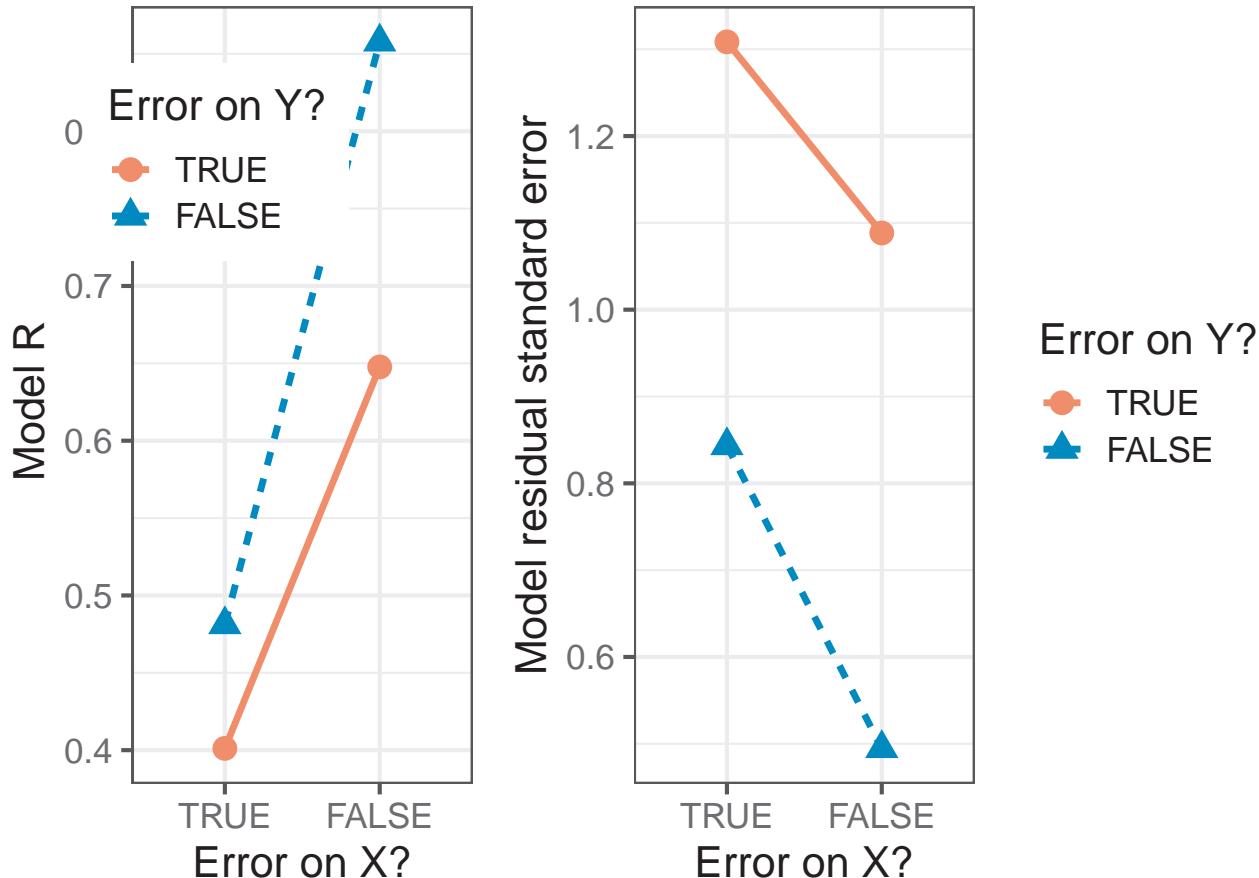


Figure 7.6: Model statistics for the combinations of additional error variance in x or y or both. Left: model R; right: Residual standard error.

7.2.3 Coffee data: β space

In multiple regression the effects of measurement error in a predictor become more complex, because error variance in one predictor, x_1 , say, can affect the coefficients of other terms in the model.

Consider the marginal relation between Heart disease and Stress in the `coffee` data. Figure 7.7 shows this with data ellipses in data space and the corresponding confidence ellipses in β space. Each panel starts with the observed data (the darkest ellipse, marked 0), then adds random normal error, $\mathcal{N}(0, \delta \times \text{SD}_{\text{Stress}})$, with $\delta = \{0.75, 1.0, 1.5\}$, to the value of Stress, while keeping the mean of Stress the same. All of the data ellipses have the same vertical shadows (SD_{Heart}), while the horizontal shadows increase with δ , driving the slope for Stress toward 0.

In β space, it can be seen that the estimated coefficients, $(\beta_0, \beta_{\text{Stress}})$ vary along a line and approach $\beta_{\text{Stress}} = 0$

as δ gets sufficiently large. The shadows of ellipses for $(\beta_0, \beta_{\text{Stress}})$ along the β_{Stress} axis also demonstrate the effects of measurement error on the standard error of β_{Stress} .

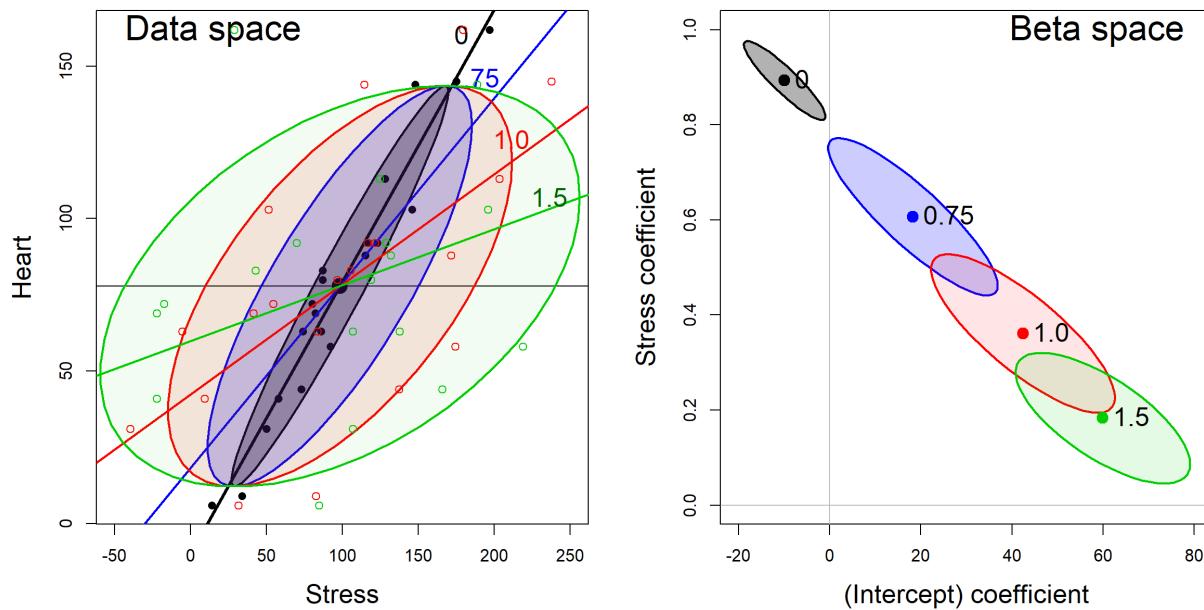


Figure 7.7: Effects of measurement error in Stress on the marginal relationship between Heart disease and Stress. Each panel starts with the observed data ($\delta = 0$), then adds random normal error, $\mathcal{N}(0, \delta \times \text{SD}_{\text{Stress}})$ with standard deviations multiplied by $\delta = 0.75, 1.0, 1.5$, to the value of Stress. Increasing measurement error biases the slope for Stress toward 0. Left: 50% data ellipses; right: 50% confidence ellipses.

Perhaps less well-known, but both more surprising and interesting, is the effect that measurement error in one variable, x_1 , has on the estimate of the coefficient for an *other* variable, x_2 , in a multiple regression model. Figure 7.8 shows the confidence ellipses for $(\beta_{\text{Coffee}}, \beta_{\text{Stress}})$ in the multiple regression predicting Heart disease, adding random normal error $\mathcal{N}(0, \delta \times \text{SD}_{\text{Stress}})$, with $\delta = \{0, 0.2, 0.4, 0.8\}$, to the value of Stress alone. As can be plainly seen, while this measurement error in Stress attenuates its coefficient, it also has the effect of biasing the coefficient for Coffee toward that in the *marginal* regression of Heart disease on Coffee alone.

7.3 What have we learned?

- **Data space and β space are dualities of each other** - While we typically visualize regression models in data space (where points are observations), there's a parallel β space where points represent models and their coefficients. These spaces mirror each other in elegant ways: lines in one space become points in the other, and confidence ellipses in β space are 90° rotations of data ellipses. This duality reveals that we gain precision in estimating coefficients precisely where our data spread the most.
- **Confidence ellipses make hypothesis testing visual and intuitive** - Instead of squinting at p-values in regression output, we can literally see whether hypotheses are supported by checking whether null hypothesis points fall inside or outside confidence ellipses. The shadows of these ellipses automatically give us individual confidence intervals, while the full ellipse captures joint uncertainty about multiple coefficients.
- **Measurement error in predictors is far more dangerous than measurement error in responses** - While errors in your response variable (y) simply inflate standard errors without biasing coefficients, errors in predictors create attenuation bias that systematically pulls slope estimates toward zero. This

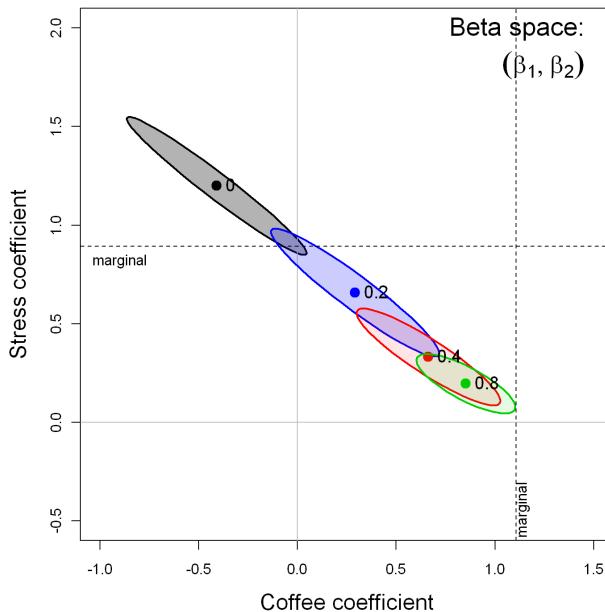


Figure 7.8: Biasing effect of measurement error in one variable (Stress) on the coefficient of another variable (Coffee) in a multiple regression. The coefficient for Coffee is driven towards its value in the marginal model using Coffee alone, as measurement error in Stress makes it less informative in the joint model.

“errors-in-variables” problem means that unreliable measurements of your predictors can make real effects appear weaker than they actually are.

- **In multiple regression, measurement error in one predictor contaminates estimates of other predictors** - Perhaps most surprisingly, when one predictor in your model suffers from measurement error, it doesn’t just bias its own coefficient—it also distorts the coefficients of other variables in unpredictable ways. This distortion will cascade, meaning that measurement quality affects your entire model, not just individual variables.
- **Ellipses reveal the hidden geometry behind familiar statistical concepts** - Data ellipses, confidence ellipses, and their mathematical relationships provide a geometric foundation for understanding correlation, regression coefficients, confidence intervals, and hypothesis tests. This visual approach transforms abstract statistical concepts into concrete geometric relationships that you can literally see and manipulate.

Collinearity & Ridge Regression

In univariate multiple regression models, we usually hope to have high correlations between the outcome y and each of the predictors, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots]$, but high correlations *among* the predictors can cause problems in estimating and testing their effects. Exactly the same problems can exist in multivariate response models, because they involve only the relations among the predictor variables.

The problem of high correlations among the predictors in a model is called **collinearity** (or multicollinearity), referring to the situation when two or more predictors are very nearly linearly related to each other (collinear). This chapter illustrates the nature of collinearity geometrically, using data and confidence ellipsoids. It describes diagnostic measures to assess these effects, and presents some novel visual tools for these purposes using the `VisCollin` package.

One class of solutions for collinearity involves *regularization methods* such as ridge regression. Another collection of graphical methods, generalized ridge trace plots, implemented in the `genridge` package, sheds further light on what is accomplished by this technique. More generally, the methods of this chapter are further examples of how data and confidence ellipsoids can be used to visualize bias and precision of regression estimates.

Packages

In this chapter I use the following packages. Load them now.

```
library(car)
library(VisCollin)
library(genridge)
library(MASS)
library(dplyr)
library(factoextra)
library(ggrepel)
library(patchwork)
```

8.1 What is collinearity?

Researchers who have studied standard treatments of linear models (e.g, Graybill (1961); Hocking (2013)) are often less than clear about what collinearity is, how to find its sources and how to take steps to resolve them. There are a number of important diagnostic measures that can help, but these are usually presented in a tabular display like Figure 8.1, which prompted this query on an online forum:

Some of my collinearity diagnostics have large values, or small values, or whatever they are *not* supposed to be

- What is bad?
- If bad, what can I do about it?

The trouble with displays like Figure 8.1 is that the important information is hidden in a sea of numbers,

Model	Dimension	Eigenvalue	Condition Index	(Constant)	Variance Proportions					
					x1	x2	x3	x4	x5	x6
1	1	6,257	1,000	,00	,00	,00	,00	,00	,00	,00
2		,265	4,857	,00	,00	,00	,00	,00	,24	,21
3		,232	5,196	,00	,00	,00	,00	,00	,39	,39
4		,187	5,791	,00	,00	,00	,00	,01	,00	,33
5		,058	10,384	,03	,01	,00	,00	,01	,30	,04
6		,001	80,295	,13	,96	,95	,04	,05	,04	,00
7		,001	109,115	,85	,02	,05	,96	,94	,03	,02

a. Dependent Variable: y

Figure 8.1: Collinearity diagnostics for a multiple regression model from SPSS. Source: Arndt Regorzh, How to interpret a Collinearity Diagnostics table in SPSS, <https://bit.ly/3YRB82b>

some of which are bad when *large*, others bad when they are *small* and a large bunch which are irrelevant to interpretation.

In Friendly & Kwan (2009), we liken this problem to that of the reader of Martin Hansford's successful series of books, *Where's Waldo*. These consist of a series of full-page illustrations of hundreds of people and things and a few Waldos—a character wearing a red and white striped shirt and hat, glasses, and carrying a walking stick or other paraphernalia. Waldo was never disguised, yet the complex arrangement of misleading visual cues in the pictures made him very hard to find. Collinearity diagnostics often provide a similar puzzle: where should you look in traditional tabular displays?¹

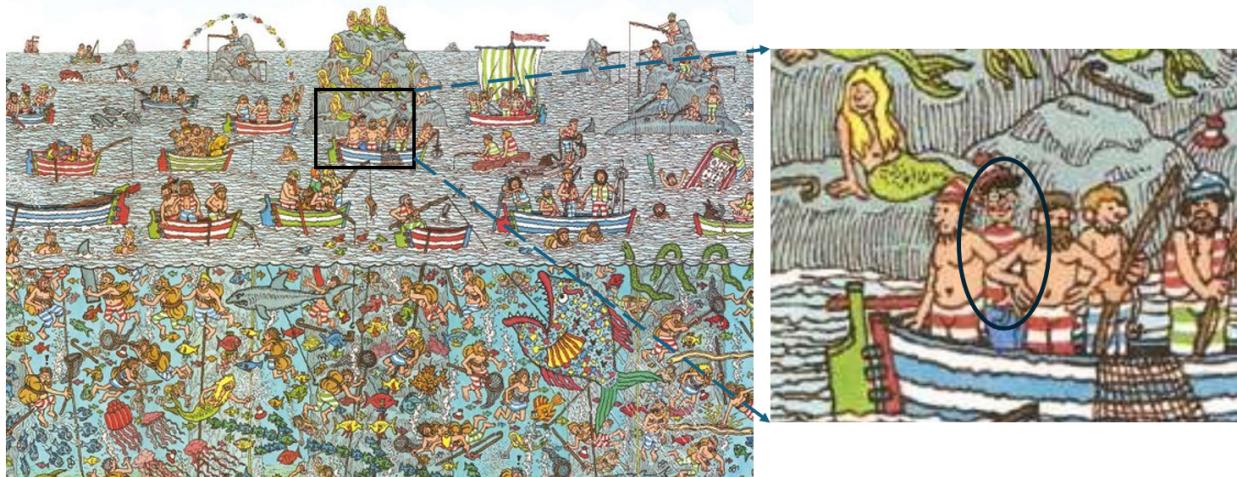


Figure 8.2: A scene from one of the *Where's Waldo* books. Waldo wears a red-striped shirt, but far too many of the other figures in the scene have horizontal red stripes, making it very difficult to find him among all the distractors. This is often the problem with collinearity diagnostics. Source: Modified from <https://bit.ly/48KPcOo>

Recall the standard classical linear model for a response variable y with a collection of predictors in $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p)$

$$\begin{aligned}\mathbf{y} &= \beta_0 + \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \cdots + \beta_p \mathbf{x}_p + \epsilon \\ &= \mathbf{X}\beta + \epsilon,\end{aligned}$$

¹The “Where’s Waldo” problem has attracted attention in machine learning, AI and computational image analysis circles. One approach uses convolutional neural networks. [FindWaldo](#) is one example implemented in Python.

for which the ordinary least squares solution is:

$$\hat{\mathbf{b}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

The sampling variances and covariances of the estimated coefficients is $\text{Var}(\hat{\mathbf{b}}) = \sigma_\epsilon^2 \times (\mathbf{X}^T \mathbf{X})^{-1}$ and σ_ϵ^2 is the variance of the residuals ϵ , estimated by the mean squared error (MSE).

In the limiting case, collinearity becomes particularly problematic when one x_i is *perfectly* predictable from the other xs , i.e., $R^2(x_i | \text{other } x) = 1$. This is problematic because:

- there is no *unique* solution for the regression coefficients $\mathbf{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X} \mathbf{y}$;
- the standard errors $s(b_i)$ of the estimated coefficients are infinite and t statistics $t_i = b_i / s(b_i)$ are 0.

This extreme case reflects a situation when one or more predictors are effectively redundant, for example when you include two variables x and y and their sum $z = x + y$ in a model. For instance, a dataset may include variables for income, expenses, and savings. But income is the sum of expenses and savings, so not all three should be used as predictors.

A more subtle case is the use *ipsatized*, defined as scores that sum to a constant, such as proportions of a total. You might have scores on tests of reading, math, spelling and geography. With ipsatized scores, any one of these is necessarily 1 – sum of the others, i.e., if reading is 0.5, math and geography are both 0.15, then geography must be 0.2. Once three of the four scores are known, the last provides no new information.

More generally, collinearity refers to the case when there are very high **multiple correlations** among the predictors, such as $R^2(x_i | \text{other } x) \geq 0.9$. Note that you can't tell simply by looking at the simple correlations. A large correlation r_{ij} is *sufficient* for collinearity, but not *necessary*—you can have variables x_1, x_2, x_3 for which the pairwise correlation are low, but the multiple correlation is high.

The consequences are:

- The estimated coefficients have large standard errors, $s(\hat{b}_j)$. They are multiplied by the square root of the variance inflation factor, $\sqrt{\text{VIF}}$, discussed below.
- The large standard errors deflate the t -statistics, $t = \hat{b}_j / s(\hat{b}_j)$, by the same factor, so a coefficient that would significant if the predictors were uncorrelated becomes insignificant when collinearity is present.
- Thus you may find a situation where an overall model is highly significant (large F -statistic), while no (or few) of the individual predictors are. This is a puzzlement!
- Beyond this, the least squares solution may have poor numerical accuracy ([Longley, 1967](#)), because the solution depends inversely on the determinant $|\mathbf{X}^T \mathbf{X}|$, which approaches 0 as multiple correlations increase.
- There is an interpretive problem as well. Recall that the coefficients \hat{b} are *partial coefficients*, meaning that they estimate change Δy in y when x changes by one unit Δx , but holding **all other variables constant**. Then, the model may be trying to estimate something that does not occur in the data. (For example: predicting strength from the highly correlated height and weight)

8.1.1 Visualizing collinearity

Collinearity can be illustrated in data space for two predictors in terms of the stability of the regression plane for a linear model $Y = X_1 + X_2$. Figure 8.3 (adapted from [Fox \(2016\)](#), Fig. 13.2) shows three cases as 3D plots of (X_1, X_2, Y) , where the correlation of predictors can be observed in the (X_1, X_2) plane.

- shows a case where X_1 and X_2 are uncorrelated as can be seen in their scatter in the horizontal plane (+ symbols). The gray regression plane is well-supported; a small change in Y for one observation won't make much difference.
- In panel (b), X_1 and X_2 have a perfect correlation, $r(x_1, x_2) = 1.0$. The regression plane is not unique; in fact there are an infinite number of planes that fit the data equally well. Note that, if all we care about is prediction (not the coefficients), we could use X_1 or X_2 , or both, or any weighted sum of them in a model and get the same predicted values.

- (c) Shows a typical case where there is a strong correlation between X_1 and X_2 . The regression plane here is unique, but is not well determined. A small change in Y **can** make quite a difference in the fitted value or coefficients, depending on the values of X_1 and X_2 . Where X_1 and X_2 are far from their near linear relation in the bottom plane, you can imagine that it is easy to tilt the plane substantially by a small change in Y.

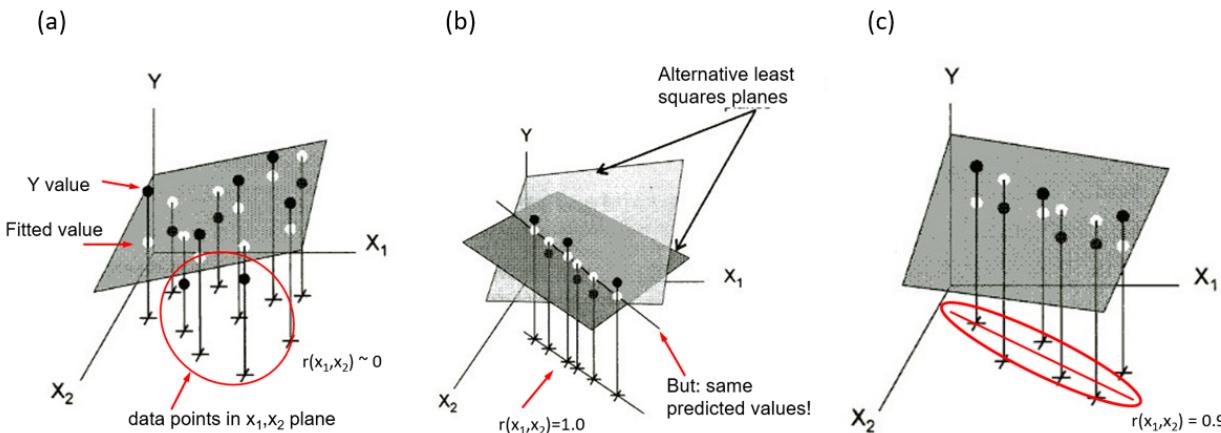


Figure 8.3: Effect of collinearity on the least squares regression plane. (a) Small correlation between predictors; (b) Perfect correlation ; (c) Very strong correlation. The black points show the data Y values, white points are the fitted values in the regression plane, and + signs represent the values of X₁ and X₂. Source: Adapted from Fox (2016), Fig. 13.2

8.1.2 Data space and β space

It is also useful to visualize collinearity by comparing the representation in **data space** with the analogous view of the confidence ellipses for coefficients in **beta space**. To do so in this example, I generate data from a known model $y = 3x_1 + 3x_2 + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 100)$ and various true correlations between x_1 and x_2 , $\rho_{12} = (0, 0.8, 0.97)$ ².

First, I use `MASS:mvrnorm()` to construct a list of three data frames `XY` with the same means and standard deviations, but with different correlations. In each case, the variable y is generated with true coefficients

R file: `R/collin-data-beta.R` β = (3, 3), and the fitted model for that value of `rho` is added to a corresponding list of models, `mods`.

```
library(MASS)
library(car)

set.seed(421)                      # reproducibility
N <- 200                            # sample size
mu <- c(0, 0)                        # means
s <- c(1, 1)                         # standard deviations
rho <- c(0, 0.8, 0.97)               # correlations
beta <- c(3, 3)                      # true coefficients

# Specify a covariance matrix, with standard deviations
# s[1], s[2] and correlation r
Cov <- function(s, r){
  matrix(c(s[1],           r * s[1]*s[2],
           r * s[1]*s[2], s[2]), 2, 2)
}
```

²This example is adapted from one by John Fox (2022), [Collinearity Diagnostics](#)

```

        r * s[1]*s[2], s[2]), nrow = 2, ncol = 2)
}

# Generate a data frame of X, y for each rho
# Fit the model for each
XY <- vector(mode ="list", length = length(rho))
mods <- vector(mode ="list", length = length(rho))
for (i in seq_along(rho)) {
  r <- rho[i]
  X <- mvrnorm(N, mu, Sigma = Cov(s, r))
  colnames(X) <- c("x1", "x2")
  y <- beta[1] * X[,1] + beta[2] * X[,2] + rnorm(N, 0, 10)

  XY[[i]] <- data.frame(X, y=y)
  mods[[i]] <- lm(y ~ x1 + x2, data=XY[[i]])
}

```

The estimated coefficients can then be extracted using `coef()` applied to each model:

```

coefs <- sapply(mods, coef)
colnames(coefs) <- paste0("mod", 1:3, " (rho=", rho, ")")
coefs
#>           mod1 (rho=0) mod2 (rho=0.8) mod3 (rho=0.97)
#> (Intercept)      1.01       -0.0535      0.141
#> x1              3.18        3.4719      3.053
#> x2              1.68        2.9734      2.059

```

Then, I define a function to plot the data ellipse (`car::dataEllipse()`) for each data frame and confidence ellipse (`car::confidenceEllipse()`) for the coefficients in the corresponding fitted model. In the plots in Figure 8.4, I specify the x, y limits for each plot so that the relative sizes of these ellipses are comparable, so that variance inflation can be assessed visually.

```

do_plots <- function(XY, mod, r) {
  X <- as.matrix(XY[, 1:2])
  dataEllipse(X,
    levels= 0.95,
    col = "darkgreen",
    fill = TRUE, fill.alpha = 0.05,
    xlim = c(-3, 3),
    ylim = c(-3, 3), asp = 1)
  text(0, 3, bquote(rho == .(r)), cex = 2, pos = NULL)

  confidenceEllipse(mod,
    col = "red",
    fill = TRUE, fill.alpha = 0.1,
    xlab = expression(paste("x1 coefficient, ", beta[1])),
    ylab = expression(paste("x2 coefficient, ", beta[2])),
    xlim = c(-5, 10),
    ylim = c(-5, 10),
    asp = 1)
  points(beta[1], beta[2], pch = "+", cex=2)
  abline(v=0, h=0, lwd=2)
}

```

```

}

op <- par(mar = c(4,4,1,1)+0.1,
          mfc = c(2, 3),
          cex.lab = 1.5)
for (i in seq_along(rho)) {
  do_plots(XY[[i]], mods[[i]], rho[i])
}
par(op)

```

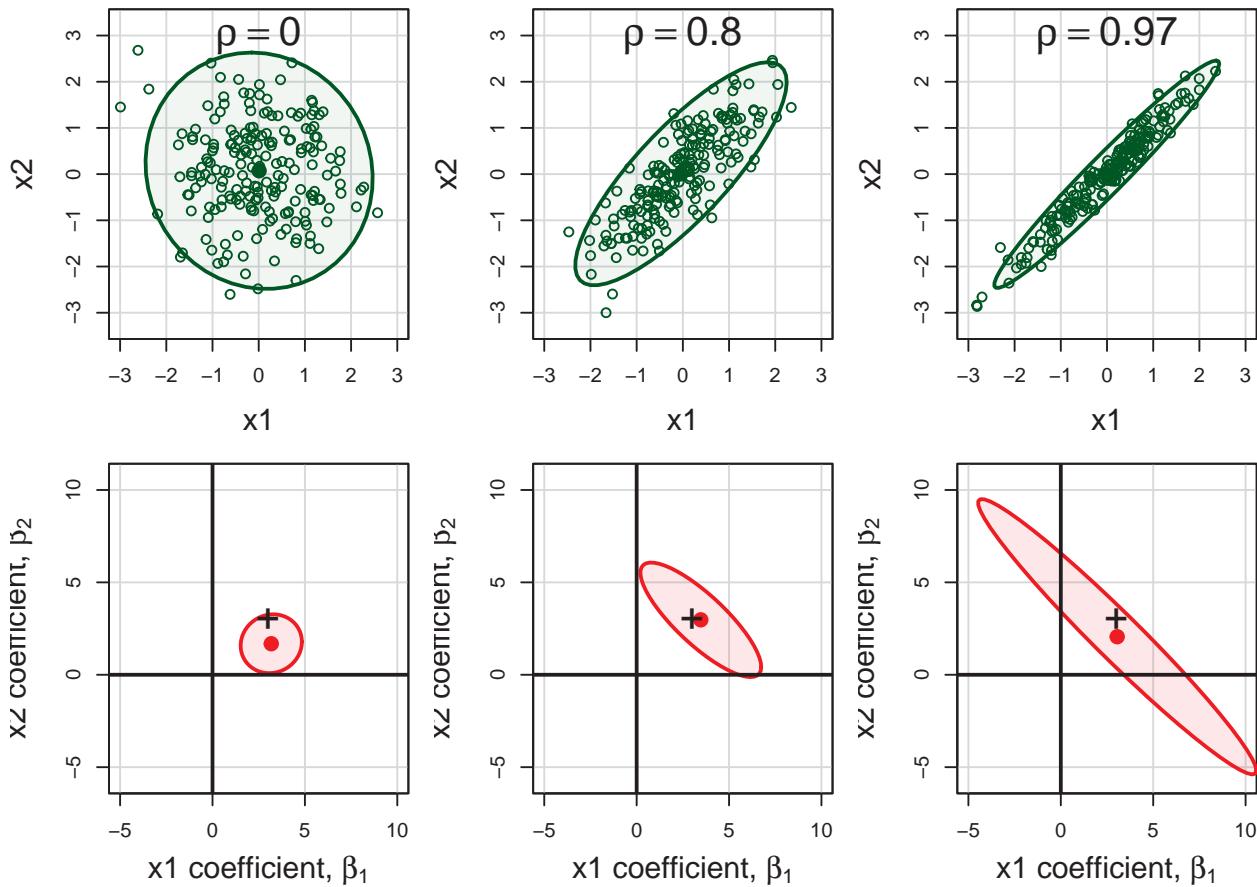


Figure 8.4: 95% Data ellipses for x_1 , x_2 and the corresponding 95% confidence ellipses for their coefficients in the model predicting y . In the confidence ellipse plots, reference lines show the value $(0,0)$ for the null hypothesis and “+” marks the true values for the coefficients. This figure adapts an example by John Fox (2022).

Recall (Section 7.1) that the confidence ellipse for (β_1, β_2) is just a 90 degree rotation (and rescaling) of the data ellipse for (x_1, x_2) : it is wide (more variance) in any direction where the data ellipse is narrow.

The shadows of the confidence ellipses on the coordinate axes in Figure 8.4 represent the standard errors of the coefficients, and get larger with increasing ρ . This is the effect of variance inflation, described in the following section.

8.2 Measuring collinearity

This section first describes the *variance inflation factor* (VIF) used to measure the effect of possible collinearity on each predictor and a collection of diagnostic measures designed to help interpret these. Then I describe some novel graphical methods to make these effects more readily understandable, to answer the “Where’s Waldo” question posed at the outset.

8.2.1 Variance inflation factors

How can we measure the effect of collinearity? The essential idea is to compare, for each predictor the variance $s^2(\hat{b}_j)$ that the coefficient that x_j would have if it was totally unrelated to the other predictors to the actual variance it has in the given model.

For two predictors such as shown in Figure 8.4 the sampling variance of x_1 can be expressed as

$$s^2(\hat{b}_1) = \frac{MSE}{(n - 1) s^2(x_1)} \times \left[\frac{1}{1 - r_{12}^2} \right]$$

The first term here is the variance of b_1 when the two predictors are uncorrelated. The term in brackets represents the **variance inflation factor** (Marquardt, 1970), the amount by which the variance of the coefficient is multiplied as a consequence of the correlation r_{12} of the predictors. As $r_{12} \rightarrow 1$, the variances approaches infinity.

More generally, with any number of predictors, this relation has a similar form, replacing the simple correlation r_{12} with the multiple correlation predicting x_j from all others,

$$s^2(\hat{b}_j) = \frac{MSE}{(n - 1) s^2(x_j)} \times \left[\frac{1}{1 - R_{j|\text{others}}^2} \right]$$

So, we have that the variance inflation factors are:

$$\text{VIF}_j = \frac{1}{1 - R_{j|\text{others}}^2}$$

In practice, it is often easier to think in terms of the square root, $\sqrt{\text{VIF}_j}$ as the multiplier of the standard errors. The denominator, $1 - R_{j|\text{others}}^2$ is sometimes called **tolerance**, a term I don’t find particularly useful, but it is just the proportion of the variance of x_j that is *not* explainable from the others.

For the cases shown in Figure 8.4 the VIFs and their square roots are:

```
vifs <- sapply(mods, car::vif)
colnames(vifs) <- paste("rho:", rho)
vifs
#>   rho: 0 rho: 0.8 rho: 0.97
#> x1      1     3.09     18.6
#> x2      1     3.09     18.6

sqrt(vifs)
#>   rho: 0 rho: 0.8 rho: 0.97
#> x1      1     1.76     4.31
#> x2      1     1.76     4.31
```

Note that when there are terms in the model with more than one degree of freedom, such as education with four levels (and hence 3 df) or a polynomial term specified as `poly(age, 3)`, that variable, education or age

is represented by three separate *xs* in the model matrix, and the standard VIF calculation gives results that vary with how those terms are coded in the model.

To allow for these cases, Fox & Monette (1992) define *generalized*, GVIFs as the inflation in the squared area of the confidence ellipse for the coefficients of such terms, relative to what would be obtained with uncorrelated data. Visually, this can be seen by comparing the areas of the ellipses in the bottom row of Figure 8.4. Because the magnitude of the GVIF increases with the number of degrees of freedom for the set of parameters, Fox & Monette suggest the analog $\sqrt{\text{GVIF}^{1/2\text{df}}}$ as the measure of impact on standard errors. This is what `car::vif()` calculates for a factor or other term with more than 1 df.

Example 8.1. Cars data

This example uses the `cars` dataset in the `VisCollin` package containing various measures of size and performance on 406 models of automobiles from 1982. Interest is focused on predicting gas mileage, `mpg`.

```
data(cars, package = "VisCollin")
str(cars)
#> 'data.frame': 406 obs. of 10 variables:
#> $ make     : Factor w/ 30 levels "amc","audi","bmw",...: 6 4 22 1 12 12 6 22 23 1 ...
#> $ model    : chr "chevelle" "skylark" "satellite" "rebel" ...
#> $ mpg      : num 18 15 18 16 17 15 14 14 14 15 ...
#> $ cylinder: int 8 8 8 8 8 8 8 8 8 ...
#> $ engine   : num 307 350 318 304 302 429 454 440 455 390 ...
#> $ horse    : int 130 165 150 150 140 198 220 215 225 190 ...
#> $ weight   : int 3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
#> $ accel    : num 12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
#> $ year     : int 70 70 70 70 70 70 70 70 70 ...
#> $ origin   : Factor w/ 3 levels "Amer","Eur","Japan": 1 1 1 1 1 1 1 1 1 1 ...
```

We fit a model predicting gas mileage (`mpg`) from the number of cylinders, engine displacement, horsepower, weight, time to accelerate from 0 – 60 mph and model year (1970–1982). Perhaps surprisingly, only `weight` and `year` appear to significantly predict gas mileage. What's going on here?

```
cars.mod <- lm (mpg ~ cylinder + engine + horse +
                  weight + accel + year,
                  data=cars)
Anova(cars.mod)
#> Anova Table (Type II tests)
#>
#> Response: mpg
#>           Sum Sq Df F value Pr(>F)
#> cylinder     12   1  0.99  0.32
#> engine        13   1  1.09  0.30
#> horse         0   1  0.00  0.98
#> weight       1214   1 102.84 <2e-16 ***
#> accel         8   1  0.70  0.40
#> year         2419   1 204.99 <2e-16 ***
#> Residuals   4543 385
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We check the variance inflation factors, using `car::vif()`. We see that most predictors have very high VIFs, indicating moderately severe multicollinearity.

```
vif(cars.mod)
#> cylinder engine horse weight accel year
#> 10.63    19.64   9.40  10.73   2.63  1.24

sqrt(vif(cars.mod))
#> cylinder engine horse weight accel year
#> 3.26     4.43   3.07   3.28   1.62  1.12
```

According to \sqrt{VIF} , the standard error of `cylinder` has been multiplied by $\sqrt{10.63} = 3.26$ and it's t -value is divided by this number, compared with the case when all predictors are uncorrelated. `engine`, `horse` and `weight` suffer a similar fate.

If we also included the factor `origin` in the models, we would get the generalized GVIF:

```
cars.mod2 <- lm (mpg ~ cylinder + engine + horse +
                  weight + accel + year + origin,
                  data=cars)
vif(cars.mod2)
#>           GVIF Df GVIF^(1/(2*Df))
#> cylinder 10.74  1      3.28
#> engine   22.94  1      4.79
#> horse    9.96  1      3.16
#> weight   11.07  1      3.33
#> accel    2.63  1      1.62
#> year     1.30  1      1.14
#> origin   2.10  2      1.20
```

💡 Connection with inverse of correlation matrix

In the linear regression model with standardized predictors, the covariance matrix of the estimated intercept-excluding parameter vector \mathbf{b}^* has the simpler form,

$$\mathcal{V}(\mathbf{b}^*) = \frac{\sigma^2}{n-1} \mathbf{R}_X^{-1}.$$

where \mathbf{R}_X is the correlation matrix among the predictors. It can then be seen that the VIF_j are just the diagonal entries of \mathbf{R}_X^{-1} .

More generally, the matrix $\mathbf{R}_X^{-1} = (r^{ij})$, when standardized to a correlation matrix as $-r^{ij}/\sqrt{r^{ii} r^{jj}}$ gives the matrix of all partial correlations, $r_{ij} | \text{others}$. }

8.2.2 Collinearity diagnostics

OK, we now know that large VIF_j indicate predictor coefficients whose estimation is degraded due to large $R_{j|\text{others}}^2$. But for this to be useful, we need to determine:

- how many dimensions in the space of the predictors are associated with nearly collinear relations?
- which predictors are most strongly implicated in each of these?

Answers to these questions are provided using measures developed by Belsley and colleagues (Belsley et al., 1980; Belsley, 1991). These measures are based on the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_p$ of the correlation matrix R_X of the predictors (preferably centered and scaled, and not including the constant term for the intercept), and the corresponding eigenvectors in the columns of $\mathbf{V}_{p \times p}$, given by the the eigen decomposition

$$\mathbf{R}_X = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^\top$$

By elementary matrix algebra, the eigen decomposition of \mathbf{R}_{XX}^{-1} is then

$$\mathbf{R}_X^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^T, \quad (8.1)$$

so, \mathbf{R}_X and \mathbf{R}_{XX}^{-1} have the same eigenvectors, and the eigenvalues of \mathbf{R}_X^{-1} are just λ_i^{-1} . Using Equation 8.1, the variance inflation factors may be expressed as

$$\text{VIF}_j = \sum_{k=1}^p \frac{V_{jk}^2}{\lambda_k},$$

which shows that only the *small* eigenvalues contribute to variance inflation, but only for those predictors that have large eigenvector coefficients on those small components. These facts lead to the following diagnostic statistics for collinearity:

- **Condition indices:** The smallest of the eigenvalues, those for which $\lambda_j \approx 0$, indicate collinearity and the number of small values indicates the number of near collinear relations. Because the sum of the eigenvalues, $\Sigma \lambda_i = p$ increases with the number of predictors p , it is useful to scale them all in relation to the largest. This leads to *condition indices*, defined as $\kappa_j = \sqrt{\lambda_1/\lambda_j}$. These have the property that the resulting numbers have common interpretations regardless of the number of predictors.
 - For completely uncorrelated predictors, all $\kappa_j = 1$.
 - $\kappa_j \rightarrow \infty$ as any $\lambda_k \rightarrow 0$.
- **Variance decomposition proportions:** Large VIFs indicate variables that are involved in *some* nearly collinear relations, but they don't indicate *which* other variable(s) each is involved with. For this purpose, Belsley et al. (1980) and Belsley (1991) proposed calculation of the proportions of variance of each variable associated with each principal component as a decomposition of the coefficient variance for each dimension.

These measures can be calculated using `VisCollin::colldiag()`. For the current model, the usual display contains both the condition indices and variance proportions. However, even for a small example, it is often difficult to know what numbers to pay attention to.

```
(cd <- colldiag(cars.mod, center=TRUE))
#> Condition
#> Index      Variance Decomposition Proportions
#>          cylinder engine horse weight accel year
#> 1    1.000  0.005   0.003  0.005  0.004   0.009  0.010
#> 2    2.252  0.004   0.002  0.000  0.007   0.022  0.787
#> 3    2.515  0.004   0.001  0.002  0.010   0.423  0.142
#> 4    5.660  0.309   0.014  0.306  0.087   0.063  0.005
#> 5    8.342  0.115   0.000  0.654  0.715   0.469  0.052
#> 6   10.818  0.563   0.981  0.032  0.176   0.013  0.004
```

Belsley (1991) recommends that the sources of collinearity be diagnosed (a) only for those components with large κ_j , and (b) for those components for which the variance proportion is large (say, ≥ 0.5) on *two* or more predictors. The print method for "colldiag" objects has a `fuzz` argument controlling this.

```
print(cd, fuzz = 0.5)
#> Condition
#> Index      Variance Decomposition Proportions
#>          cylinder engine horse weight accel year
#> 1    1.000   .     .     .     .     .
#> 2    2.252   .     .     .     .     .     0.787
#> 3    2.515   .     .     .     .     .
#> 4    5.660   .     .     .     .     .     .
```

```
#> 5   8.342  .       .      0.654 0.715  .       .
#> 6 10.818 0.563  0.981  .       .      .       .
```

The mystery is solved, if you can read that table with these recommendations in mind. There are two nearly collinear relations among the predictors, corresponding to the two smallest dimensions.

- Dimension 5 reflects the high correlation between horsepower and weight,
- Dimension 6 reflects the high correlation between number of cylinders and engine displacement.

Note that the high variance proportion for `year` (0.787) on the second component creates no problem and should be ignored because (a) the condition index is low and (b) it shares nothing with other predictors.

8.3 Tableplots

The default tabular display of condition indices and variance proportions from `colldiag()` is what triggered the comparison to “Where’s Waldo”. It suffers from the fact that the important information — (a) how many Waldos? (b) where are they hiding — is disguised by being embedded in a sea of mostly irrelevant numbers. The simple option of using a principled `fuzz` factor helps considerably, but not entirely.

The simplified tabular display above can be improved to make the patterns of collinearity more visually apparent and to signify warnings directly to the eyes. A **tableplot** (Kwan et al., 2009) is a semi-graphic display that presents numerical information in a table using shapes proportional to the value in a cell and other visual attributes (shape type, color fill, and so forth) to encode other information.

For collinearity diagnostics, these show:

- the condition indices, using *squares* whose background color is **red** for condition indices > 10 , **brown** for values > 5 and **green** otherwise, reflecting danger, warning and OK respectively. The value of the condition index is encoded within this using a white square whose side is proportional to the value (up to some maximum value, `cond.max` that fills the cell).
- Variance decomposition proportions are shown by filled *circles* whose radius is proportional to those values and are filled (by default) with shades ranging from white through pink to red. Rounded values of those diagnostics are printed in the cells.

The tableplot below (Figure 8.5) encodes all the information from the values of `colldiag()` printed above. To aid perception, it uses `prop.col` color breaks such that variance proportions < 0.3 are shaded white. The visual message is that one should attend to collinearities with large condition indices **and** large variance proportions implicating two or more predictors.

```
tableplot(cd, title = "Tableplot of cars data",
          cond.max = 30 )
```

R file:
R/cars-colldiag.R

8.4 Collinearity biplots

As we have seen, the collinearity diagnostics are all functions of the eigenvalues and eigenvectors of the correlation matrix of the predictors in the regression model, or alternatively, the SVD of the \mathbf{X} matrix in the linear model (excluding the constant). The standard biplot (Gabriel, 1971; Gower & Hand, 1996) (see: Section 4.3) can be regarded as a multivariate analog of a scatterplot, obtained by projecting a multivariate

Tableplot of cars data

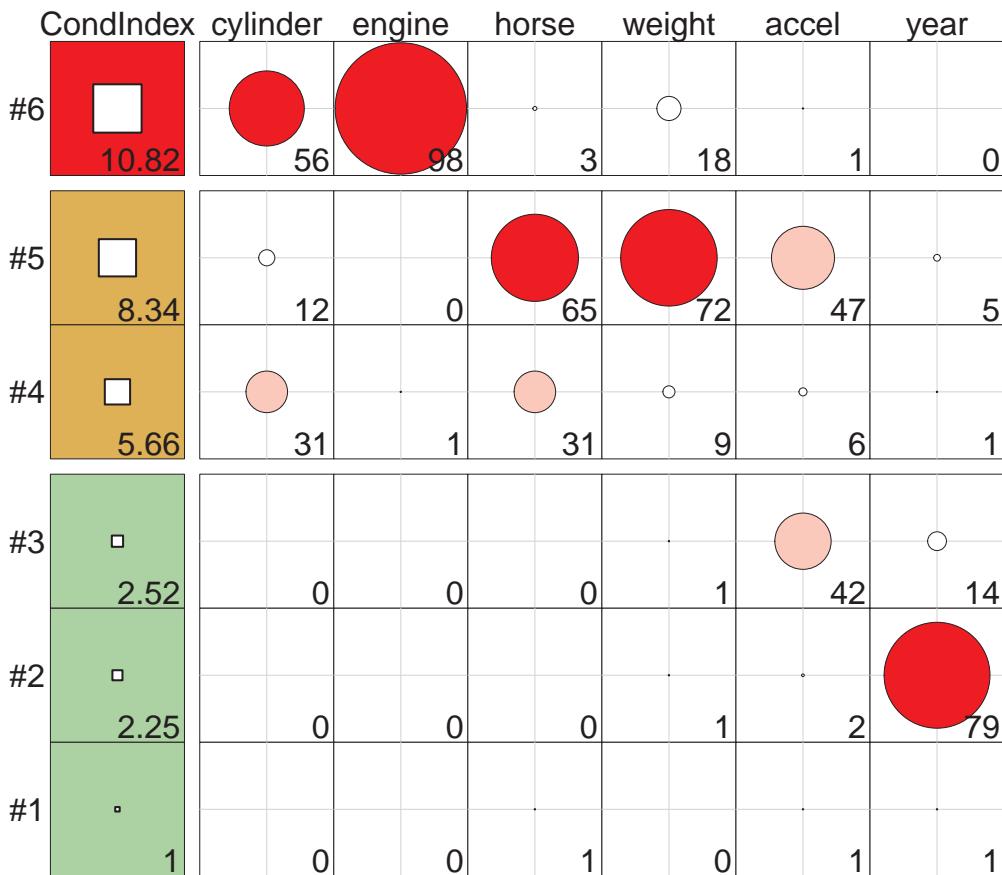


Figure 8.5: Tableplot of condition indices and variance proportions for the `cars` data. In column 1, the square symbols are scaled relative to a maximum condition index of 30. In the remaining columns, variance proportions (times 100) are shown as circles scaled relative to a maximum of 100.

sample into a low-dimensional space (typically of 2 or 3 dimensions) accounting for the greatest variance in the data.

However the standard biplot is less useful for visualizing the relations among the predictors that lead to nearly collinear relations. Instead, biplots of the **smallest dimensions** show these relations directly, and can show other features of the data as well, such as outliers and leverage points. We use `prcomp(X, scale.=TRUE)` to obtain the PCA of the correlation matrix of the predictors:

```

cars.X <- cars |>
  select(where(is.numeric)) |>
  select(-mpg) |>
  tidyverse::drop_na()
cars.pca <- prcomp(cars.X, scale. = TRUE)
cars.pca
#> Standard deviations (1, ..., p=6):
#> [1] 2.070 0.911 0.809 0.367 0.245 0.189
#>
#> Rotation (n x k) = (6 x 6):
#>          PC1     PC2     PC3     PC4     PC5     PC6

```

```
#> cylinder -0.454 -0.1869  0.168 -0.659 -0.2711 -0.4725
#> engine    -0.467 -0.1628  0.134 -0.193 -0.0109  0.8364
#> horse     -0.462 -0.0177 -0.123  0.620 -0.6123 -0.1067
#> weight    -0.444 -0.2598  0.278  0.350  0.6860 -0.2539
#> accel     0.330 -0.2098  0.865  0.143 -0.2774  0.0337
#> year      0.237 -0.9092 -0.335  0.025 -0.0624  0.0142
```

The standard deviations above are the square roots $\sqrt{\lambda_j}$ of the eigenvalues of the correlation matrix, and are returned in the `sdev` component of the "prcomp" object. The eigenvectors are returned in the `rotation` component, whose directions are arbitrary. Because we are interested in seeing the relative magnitude of variable vectors, we are free to multiply them by any constant to make them more visible in relation to the scores for the cars.

```
cars.pca$rotation <- -2.5 * cars.pca$rotation      # reflect & scale the variable vectors

ggp <- fviz_pca_biplot(
  cars.pca,
  axes = 6:5,
  geom = "point",
  col.var = "blue",
  labelsize = 5,
  pointsize = 1.5,
  arrowsize = 1.5,
  addEllipses = TRUE,
  ggtheme = ggplot2::theme_bw(base_size = 14),
  title = "Collinearity biplot for cars data")

# add point labels for outlying points
dsq <- heplots::Mahalanobis(cars.pca$x[, 6:5])
scores <- as.data.frame(cars.pca$x[, 6:5])
scores$name <- rownames(scores)

ggp + geom_text_repel(data = scores[dsq > qchisq(0.95, df = 6), ],
  aes(x = PC6,
      y = PC5,
      label = name),
  vjust = -0.5,
  size = 5)
```

As with the tabular display of variance proportions, Waldo is hiding in the dimensions associated with the smallest eigenvalues (largest condition indices). As well, it turns out that outliers in the predictor space (also high leverage observations) can often be seen as observations far from the centroid in the space of the smallest principal components.

The projections of the variable vectors in Figure 8.6 on the Dimension 5 and Dimension 6 axes are proportional to their variance proportions shown above. The relative lengths of these variable vectors can be considered to indicate the extent to which each variable contributes to collinearity for these two near-singular dimensions.

Thus, we see again that Dimension 6 is largely determined by `engine` size, with a substantial (negative) relation to `cylinder`. Dimension 5 has its' strongest relations to `weight` and `horse`.

Moreover, there is one observation, #20, that stands out as an outlier in predictor space, far from the centroid. It turns out that this vehicle, a Buick Estate wagon, is an early-year (1970) American behemoth, with an

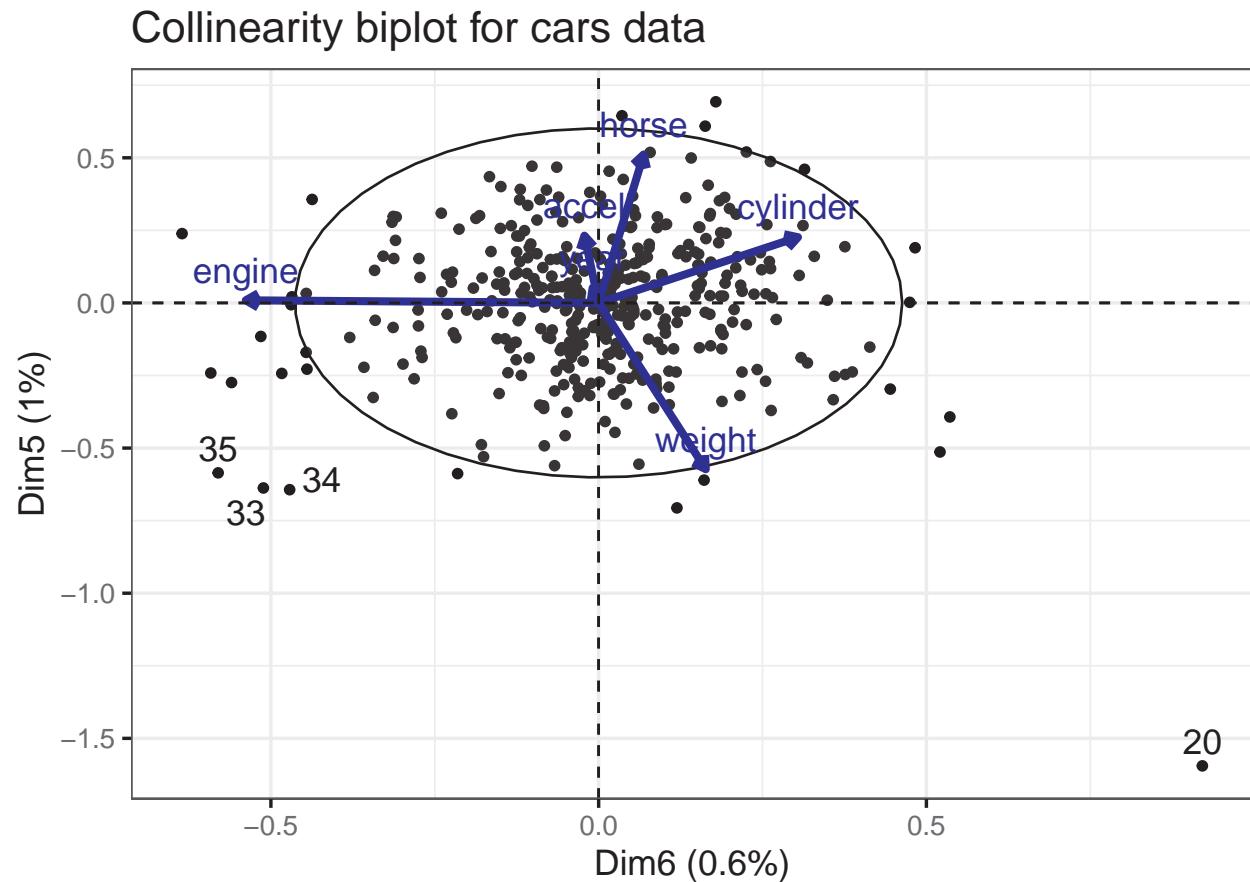


Figure 8.6: Collinearity biplot of the Cars data, showing the last two dimensions. The projections of the variable vectors on the coordinate axes are proportional to their variance proportions. To reduce graphic clutter, only the most outlying observations in predictor space are identified by case labels. An extreme outlier (case 20) appears in the lower right corner.

8-cylinder, 455 cu. in, 225 horse-power engine, and able to go from 0 to 60 mph in 10 sec. (Its MPG is only slightly under-predicted from the regression model, however.)

With PCA and the biplot, we are used to looking at the dimensions that account for the most variation, but the answer to *Where's Waldo?* is that he is hiding in the *smallest* data dimensions, just as he does in Figure 8.2 where the weak signals of his stripped shirt, hat and glasses are embedded in a visual field of noise. As we just saw, outliers hide there also, hoping to escape detection. These small dimensions are also implicated in ridge regression as we will see shortly (Section 8.6).

8.5 Remedies for collinearity: What can I do?

Collinearity is often a **data** problem, for which there is no magic cure. Nevertheless there are some general guidelines and useful techniques to address this problem.

- **Pure prediction:** If we are only interested in predicting / explaining an outcome, and not the model coefficients or which are “significant”, collinearity can be largely ignored. The fitted values are unaffected by collinearity, even in the case of perfect collinearity as shown in Figure 8.3 (b).

- **Structural collinearity:** Sometimes collinearity results from structural relations among the variables that relate to how they have been defined.

- For example, polynomial terms, like x, x^2, x^3 or interaction terms like $x_1, x_2, x_1 * x_2$ are necessarily correlated. A simple cure is to *center* the predictors at their means, using $x - \bar{x}, (x - \bar{x})^2, (x - \bar{x})^3$ or $(x_1 - \bar{x}_1), (x_2 - \bar{x}_2), (x_1 - \bar{x}_1) * (x_2 - \bar{x}_2)$. Centering removes the spurious ill-conditioning, thus reducing the VIFs. Note that in polynomial models, using `y ~ poly(x, 3)` to specify a cubic model generates *orthogonal* (uncorrelated) regressors, whereas in `y ~ x + I(x^2) + I(x^3)` the terms have built-in correlations.
- When some predictors share a common cause, as in GNP or population in time-series or cross-national data, you can reduce collinearity by re-defining predictors to reflect *per capita measures*. In a related example with sports data, when you have cumulative totals (e.g., runs, hits, homeruns in baseball) for players over years, expressing these measures as *per year* will reduce the common effect of longevity on these measures.

- **Model re-specification:**

- Drop one or more regressors that have a high VIF, if they are not deemed to be essential to understanding the model. Care must be taken here to not omit variables which should be controlled or accounted for in interpretation.
- Replace highly correlated regressors with less correlated linear combination(s) of them. For example, two related variables, x_1 and x_2 can be replaced without any loss of information by replacing them with their sum and difference, $z_1 = x_1 + x_2$ and $z_2 = x_1 - x_2$. For instance, in a dataset on fitness, we may have correlated predictors of resting pulse rate and pulse rate while running. Transforming these to average pulse rate and their difference gives new variables which are interpretable and less correlated.

- **Statistical remedies:**

- Transform the predictors \mathbf{X} to uncorrelated principal component scores $\mathbf{Z} = \mathbf{X}\mathbf{V}$, and regress \mathbf{y} on \mathbf{Z} . These will have the identical overall model fit without loss of information. A related technique is *incomplete* principal components regression, where some of the smallest dimensions (those causing collinearity) are omitted from the model. The trade-off is that it may be more difficult to interpret what the model means, but this can be countered with a biplot, showing the projections of the original variables into the reduced space of the principal components.
- Use **regularization methods** such as ridge regression and lasso, which correct for collinearity by introducing shrinking coefficients towards 0, introducing a small amount of bias, . See the [genridge](#) package and its [pkgdown documentation](#) for visualization methods.
- use Bayesian regression; if multicollinearity prevents a regression coefficient from being estimated precisely, then a prior on that coefficient will help to reduce its posterior variance.

Example: Centering

To illustrate the effect of centering a predictor in a polynomial model, we generate a perfect quadratic relationship, $y = x^2$ and consider the correlations of y with x and with $(x - \bar{x})^2$. The correlation of y with x is 0.97, while the correlation of y with $(x - \bar{x})^2$ is zero.

```

x <- 1:20
y1 <- x^2
y2 <- (x - mean(x))^2
XY <- data.frame(x, y1, y2)

(R <- cor(XY))
#>      x     y1     y2

```

```
#> x 1.000 0.971 0.000
#> y1 0.971 1.000 0.238
#> y2 0.000 0.238 1.000
```

The effect of centering here is remove the linear association in what is a purely quadratic relationship, as can be seen by plotting y1 and y2 against x.

```
r1 <- R[1, 2]
r2 <- R[1, 3]

gg1 <-
  ggplot(XY, aes(x = x, y = y1)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", formula = y~x, linewidth = 2, se = FALSE) +
  labs(x = "X", y = "Y") +
  theme_bw(base_size = 16) +
  annotate("text", x = 5, y = 350, size = 6,
          label = paste("X Uncentered\nnr =", round(r1, 3)))

gg2 <-
  ggplot(XY, aes(x = x, y = y2)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", formula = y~x, linewidth = 2, se = FALSE) +
  labs(x = "X", y = "Y") +
  theme_bw(base_size = 16) +
  annotate("text", x = 5, y = 80, size = 6,
          label = paste("X Centered\nnr =", round(r2, 3)))

gg1 + gg2      # show plots side-by-side
```

Example: Interactions

manufacturing process to produce acetylene in relation to reactor temperature (`temp`), the ratio of two components and the contact `time` in the reactor. A naive response surface model might suggest that yield is quadratic in time and there are potential interactions among all pairs of predictors.

```
data(Acetylene, package = "genridge")
acetyl.mod0 <- lm(
  yield ~ temp + ratio + time + I(time^2) +
  temp:time + temp:ratio + time:ratio,
  data=Acetylene)

(acetyl.vif0 <- vif(acetyl.mod0))
#>      temp        ratio        time  I(time^2)  temp:time temp:ratio
#>     383       10555     18080       564       9719      9693
#> ratio:time
#>     225
```

These results are horrible! How much does centering help? I first center all three predictors and then use `update()` to re-fit the same model using the centered data.

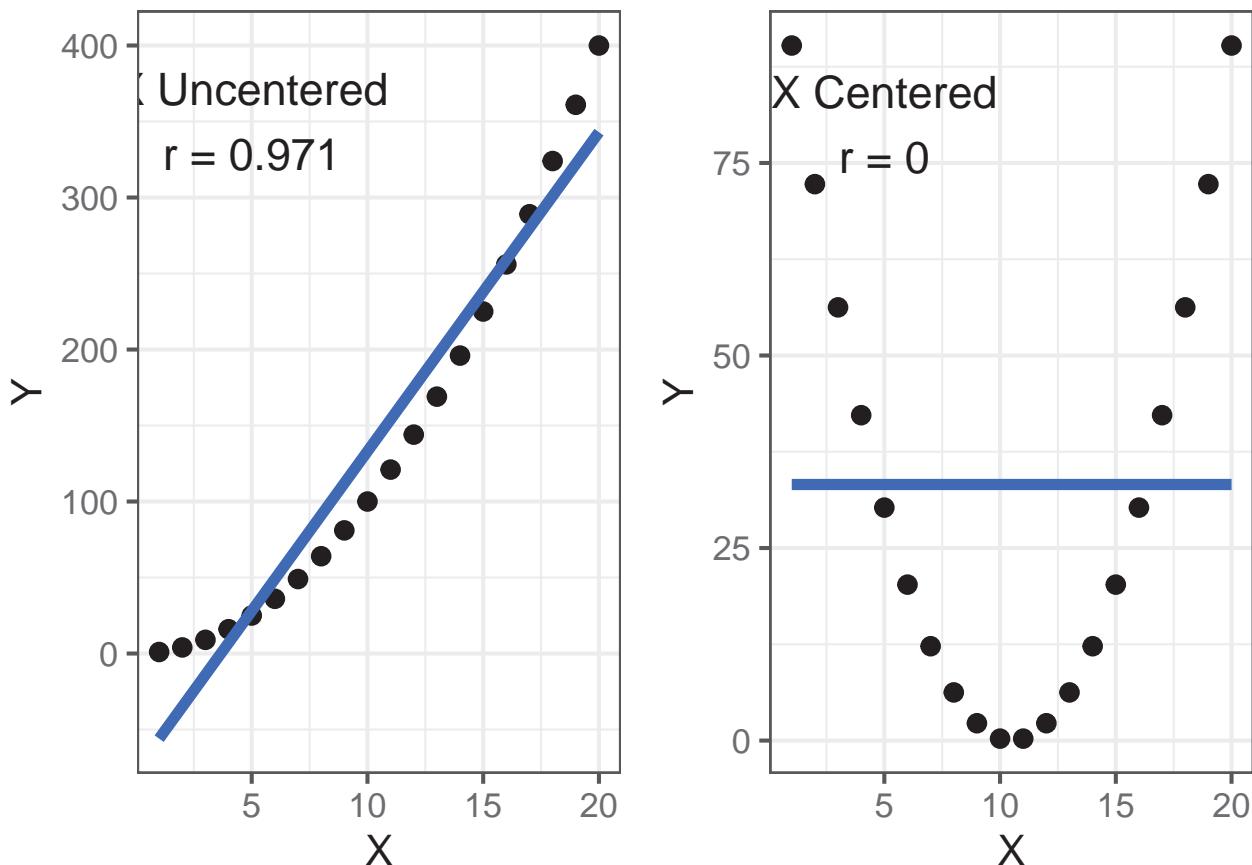


Figure 8.7: Centering a predictor removes the necessary correlation in a quadratic regression

```
Acetylene.centered <-  
  Acetylene |>  
  mutate(temp = temp - mean(temp),  
         time = time - mean(time),  
         ratio = ratio - mean(ratio))  
  
acetyl.mod1 <- update(acetyl.mod0, data=Acetylene.centered)  
(acetyl.vif1 <- vif(acetyl.mod1))  
#>      temp          ratio          time    I(time^2)  temp:time temp:ratio  
#>  57.09        1.09     81.57      51.49      44.67     30.69  
#> ratio:time  
#>      33.33
```

This is far better, although still not great in terms of VIF. But, how much have we improved the situation by the simple act of centering the predictors? The square roots of the ratios of VIFs tell us the impact of centering on the standard errors.

```
sqrt(acetyl.vif0 / acetyl.vif1)  
#>      temp          ratio          time    I(time^2)  temp:time temp:ratio  
#>  2.59        98.24     14.89      3.31      14.75     17.77  
#> ratio:time  
#>      2.60
```

Finally, we use `poly(time, 2)` in the model for the centered data. Because there are multiple degree of freedom terms in the model, `car::vif()` calculates GVIFs here. The final column gives $\sqrt{\text{GVIF}^{1/2\text{df}}}$, the remaining effect of collinearity on the standard errors of terms in this model.

```
acetyl.mod2 <- lm(yield ~ temp + ratio + poly(time, 2) +
                     temp:time + temp:ratio + time:ratio,
                     data=Acetylene.centered)

vif(acetyl.mod2, type = "term")
#>           GVIF Df GVIF^(1/(2*Df))
#> temp       57.09  1     7.56
#> ratio      1.09  1     1.05
#> poly(time, 2) 1733.56  2     6.45
#> temp:time   44.67  1     6.68
#> temp:ratio   30.69  1     5.54
#> ratio:time   33.33  1     5.77
```

8.6 Ridge regression

Ridge regression is an instance of a class of techniques designed to obtain more favorable predictions at the expense of some increase in bias, compared to ordinary least squares (OLS) estimation. These methods began as a way of solving collinearity problems in OLS regression with highly correlated predictors (Hoerl & Kennard, 1970). More recently ridge regression developed to a larger class of model selection methods, of which the LASSO method of Tibshirani (1996) and LAR method of Efron et al. (2004) are well-known instances. See, for example, the reviews in Vinod (1978) and McDonald (2009) for details and context omitted here. The case of ridge regression has also been extended to the case of two or more response variables (P. J. Brown & Zidek, 1980; Haitovsky, 1987).

An essential idea behind these methods is that the OLS estimates are constrained in some way, shrinking them, on average, toward zero, to achieve increased predictive accuracy at the expense of some increase in bias. Another common characteristic is that they involve some tuning parameter (k) or criterion to quantify the tradeoff between bias and variance. In many cases, analytical or computationally intensive methods have been developed to choose an optimal value of the tuning parameter, for example using generalized cross validation, bootstrap methods.

A common means to visualize the effects of shrinkage in these problems is to make what are called *univariate ridge trace plots* (Section 8.7) showing how the estimated coefficients $\hat{\beta}_k$ change as the shrinkage criterion k increases. (An example is shown in Fig XX below.) But this only provides a view of bias. It is the wrong graphic form for a multivariate problem where we want to visualize bias in the coefficients $\hat{\beta}_k$ vs. their precision, as reflected in their estimated variances, $\widehat{\text{Var}}(\hat{\beta}_k)$. A more useful graphic plots the confidence ellipses for the coefficients, showing both bias and precision (Section 8.8). Some of the material below borrows from Friendly (2011) and Friendly (2013).

8.6.1 Properties of ridge regression

To provide some context, I summarize the properties of ridge regression below, comparing the OLS estimates with their ridge counterparts. To avoid unnecessary details related to the intercept, assume the predictors have been centered at their means and the unit vector is omitted from \mathbf{X} . Further, to avoid scaling issues, we standardize the columns of \mathbf{X} to unit length, so that $\mathbf{X}^\top \mathbf{X}$ is a also correlation matrix.

The ordinary least squares estimates of coefficients and their estimated variance covariance matrix take the (hopefully now) familiar form

$$\begin{aligned}\hat{\beta}^{\text{OLS}} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \\ \widehat{\text{Var}}(\hat{\beta}^{\text{OLS}}) &= \hat{\sigma}_\epsilon^2 (\mathbf{X}^\top \mathbf{X})^{-1}.\end{aligned}$$

{#eq-OLS-beta-var}

As we saw earlier, one signal of the problem of collinearity is that the determinant $\det(\mathbf{X}^\top \mathbf{X})$ approaches zero as the predictors become more collinear. The inverse $(\mathbf{X}^\top \mathbf{X})^{-1}$ becomes numerically unstable, or does not exist if the determinant becomes zero in the case of exact dependency of one variable on the others.

Ridge regression uses a trick to avoid this. It adds a constant, k to the diagonal elements, replacing $\mathbf{X}^\top \mathbf{X}$ with $\mathbf{X}^\top \mathbf{X} + k\mathbf{I}$ in ?@eq-OLS-beta-var. This drives the determinant away from zero as k increases. The ridge regression estimates then become,

$$\begin{aligned}\hat{\beta}_k^{\text{RR}} &= (\mathbf{X}^\top \mathbf{X} + k\mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \\ &= \mathbf{G}_k \hat{\beta}^{\text{OLS}}, \\ \widehat{\text{Var}}(\hat{\beta}_k^{\text{RR}}) &= \hat{\sigma}^2 \mathbf{G}_k (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{G}_k^\top,\end{aligned}$$

{#eq-ridge-beta-var}

where $\mathbf{G}_k = [\mathbf{I} + k(\mathbf{X}^\top \mathbf{X})^{-1}]^{-1}$ is the $(p \times p)$ *shrinkage* matrix. Thus, as k increases, \mathbf{G}_k decreases, and drives $\hat{\beta}_k^{\text{RR}}$ toward $\mathbf{0}$ (Hoerl & Kennard, 1970).

Another insight, from the shrinkage literature, is that ridge regression can be formulated as least squares regression, minimizing a residual sum of squares, $\text{RSS}(k)$, which adds a penalty for large coefficients,

$$\text{RSS}(k) = (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta) + k\beta^\top \beta \quad (k \geq 0), \quad (8.2)$$

where the penalty restrict the coefficients to some squared length $\beta^\top \beta = \Sigma \beta_i^2 \leq t(k)$.

The geometry of ridge regression is illustrated in Figure 8.8 for two coefficients $\beta = (\beta_1, \beta_2)$. The blue circles at the origin, having radii $\sqrt{t_k}$, show the constraint that the sum of squares of coefficients, $\beta^\top \beta = \beta_1^2 + \beta_2^2$ be less than k . The red ellipses show contours of the covariance ellipse of $\hat{\beta}^{\text{OLS}}$. As the shrinkage constant k increases, the center of these ellipses travel along the path illustrated toward $\beta = \mathbf{0}$. This path is called the *locus of osculation*, the path along which circles or ellipses first kiss as they expand, like the pattern of ripples from rocks dropped into a pond (Friendly et al., 2013).

?@eq-ridge-beta-var is computationally expensive, potentially numerically unstable for small k , and it is conceptually opaque, in that it sheds little light on the underlying geometry of the data in the column space of \mathbf{X} . An alternative formulation can be given in terms of the singular value decomposition (SVD) of \mathbf{X} ,

$$\mathbf{X} = \mathbf{UDV}^\top$$

where \mathbf{U} and \mathbf{V} are respectively $n \times p$ and $p \times p$ orthonormal matrices, so that $\mathbf{U}^\top \mathbf{U} = \mathbf{V}^\top \mathbf{V} = \mathbf{I}$, and $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_p)$ is the diagonal matrix of ordered singular values, with entries $d_1 \geq d_2 \geq \dots \geq d_p \geq 0$.

Because $\mathbf{X}^\top \mathbf{X} = \mathbf{V} \mathbf{D}^2 \mathbf{V}^\top$, the eigenvalues of $\mathbf{X}^\top \mathbf{X}$ are given by \mathbf{D}^2 and therefore the eigenvalues of \mathbf{G}_k can be shown (Hoerl & Kennard, 1970) to be the diagonal elements of

$$\mathbf{D}(\mathbf{D}^2 + k\mathbf{I})^{-1} \mathbf{D} = \text{diag} \left(\frac{d_i^2}{d_i^2 + k} \right).$$

Noting that the eigenvectors, \mathbf{V} are the principal component vectors, and that $\mathbf{X}\mathbf{V} = \mathbf{UD}$, the ridge estimates can be calculated more simply in terms of \mathbf{U} and \mathbf{D} as

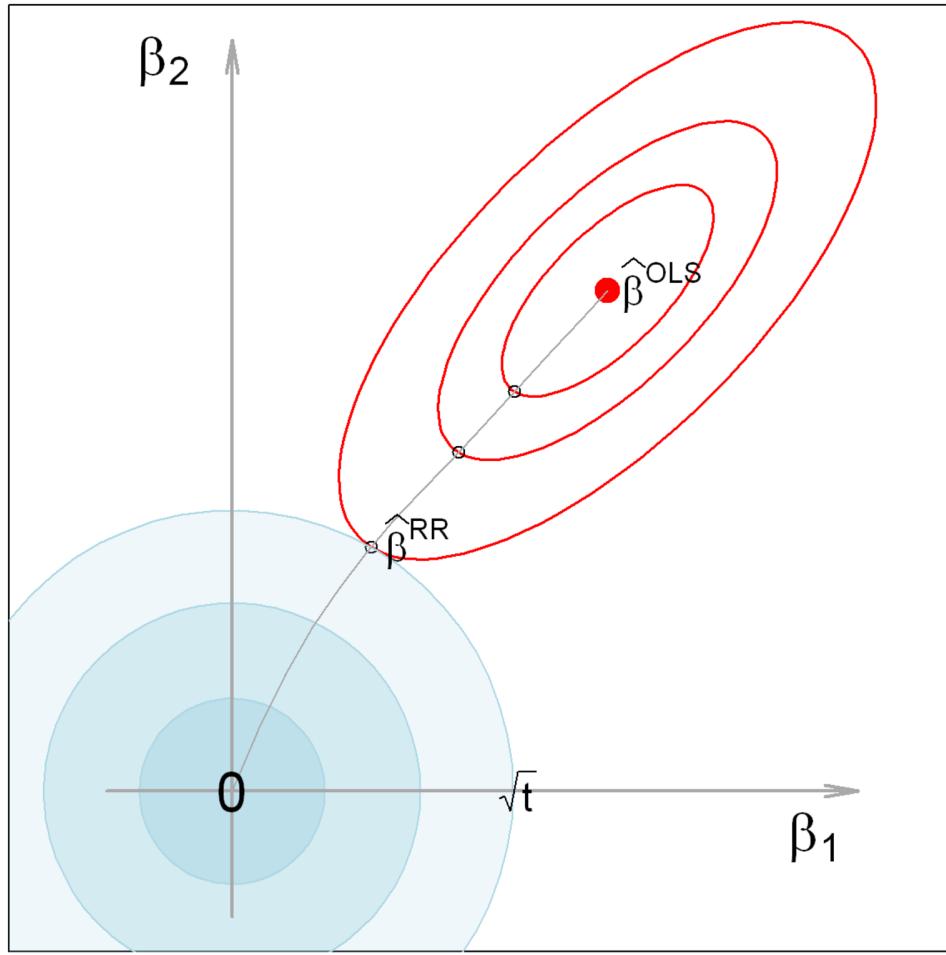


Figure 8.8: Geometric interpretation of ridge regression, using elliptical contours of the $\text{RSS}(k)$ function. The blue circles at the origin show the constraint that the sum of squares of coefficients, $\beta^\top \beta$ be less than k . The red ellipses show the covariance ellipse of two coefficients β . Ridge regression finds the point $\hat{\beta}_k^{\text{RR}}$ where the OLS contours just kiss the constraint region. *Source:* Friendly et al. (2013).

$$\hat{\beta}_k^{\text{RR}} = (\mathbf{D}^2 + k\mathbf{I})^{-1} \mathbf{D} \mathbf{U}^\top \mathbf{y} = \left(\frac{d_i}{d_i^2 + k} \right) \mathbf{u}_i^\top \mathbf{y}, \quad i = 1, \dots, p.$$

The terms $d_i^2/(d_i^2 + k) \leq 1$ are thus the factors by which the coordinates of $\mathbf{u}_i^\top \mathbf{y}$ are shrunk with respect to the orthonormal basis for the column space of \mathbf{X} . The small singular values d_i correspond to the directions which ridge regression shrinks the most. These are the directions which contribute most to collinearity, discussed earlier.

This analysis also provides an alternative and more intuitive characterization of the ridge tuning constant. By analogy with OLS, where the hat matrix, $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ reflects degrees of freedom $\text{df} = \text{tr}(\mathbf{H}) = p$ corresponding to the p parameters, the effective degrees of freedom for ridge regression (Hastie et al., 2009) is

$$\begin{aligned} \text{df}_k &= \text{tr}[\mathbf{X}(\mathbf{X}^T \mathbf{X} + k\mathbf{I})^{-1} \mathbf{X}^T] \\ &= \sum_i^p \text{df}_k(i) = \sum_i^p \left(\frac{d_i^2}{d_i^2 + k} \right) . \end{aligned}$$

{#eq-dfk}

df_k is a monotone decreasing function of k , and hence any set of ridge constants can be specified in terms of equivalent df_k . Greater shrinkage corresponds to fewer coefficients being estimated.

There is a close connection with principal components regression mentioned in Section 8.5. Ridge regression shrinks *all* dimensions in proportion to $\text{df}_k(i)$, so the low variance dimensions are shrunk more. Principal components regression discards the low variance dimensions and leaves the high variance dimensions unchanged.

8.6.2 The genridge package

Ridge regression and other shrinkage methods are available in several packages including **MASS** (the `lm.ridge()` function), **glmnet** (Friedman et al., 2025), and **penalized** (Goeman et al., 2022), but none of these provides insightful graphical displays. `glmnet::glmnet()` also implements a method for multivariate responses with a ‘family=“mgaussian”’.

Here, I focus in the **genridge** package (Friendly, 2024), where the `ridge()` function is the workhorse and `pca.ridge()` transforms these results to PCA/SVD space. `vif.ridge()` calculates VIFs for class “ridge” objects and `precision()` calculates precision and shrinkage measures.

A variety of plotting functions is available for univariate, bivariate and 3D plots:

- `traceplot()` Traditional univariate ridge trace plots
- `plot.ridge()` Bivariate 2D ridge trace plots, showing the covariance ellipse of the estimated coefficients
- `pairs.ridge()` All pairwise bivariate ridge trace plots
- `plot3d.ridge()` 3D ridge trace plots with ellipsoids
- `biplot.ridge()` ridge trace plots in PCA/SVD space

In addition, the `pca()` method for “ridge” objects transforms the coefficients and covariance matrices of a ridge object from predictor space to the equivalent, but more interesting space of the PCA of $\mathbf{X}^T \mathbf{X}$ or the SVD of \mathbf{X} . `biplot.pcaridge()` adds variable vectors to the bivariate plots of coefficients in PCA space

8.7 Univariate ridge trace plots

A classic example for ridge regression is Longley’s (1967) data, consisting of 7 economic variables, observed yearly from 1947 to 1962 ($n=16$), in the dataset `longley`. The goal is to predict Employed from GNP, Unemployed, Armed.Forces, Population, Year, and GNP.deflator.

```
data(longley, package="datasets")
str(longley)
#> 'data.frame':   16 obs. of  7 variables:
#> $ GNP.deflator: num  83 88.5 88.2 89.5 96.2 ...
#> $ GNP          : num  234 259 258 285 329 ...
#> $ Unemployed   : num  236 232 368 335 210 ...
#> $ Armed.Forces: num  159 146 162 165 310 ...
#> $ Population   : num  108 109 110 111 112 ...
```

```
#> $ Year      : int  1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 ...
#> $ Employed  : num  60.3 61.1 60.2 61.2 63.2 ...
```

These data were constructed to illustrate numerical problems in least squares software at the time, and they are (purposely) perverse, in that:

- Each variable is a time series so that there is clearly a lack of independence among predictors. `Year` is at least implicitly correlated with most of the others.
- Worse, there is also some structural collinearity among the variables `GNP`, `Year`, `GNP.deflator`, and `Population`; for example, `GNP.deflator` is a multiplicative factor to account for inflation.

We fit the regression model, and sure enough, there are some extremely large VIFs. The largest, for `GNP` represents a multiplier of $\sqrt{1788.5} = 42.3$ on the standard errors.

```
longley.lm <- lm(Employed ~ GNP + Unemployed + Armed.Forces +
                    Population + Year + GNP.deflator,
                    data=longley)
vif(longley.lm)
#>          GNP    Unemployed Armed.Forces    Population       Year
#> 1788.51        33.62       3.59     399.15     758.98
#> GNP.deflator
#>      135.53
```

Shrinkage values can be specified using k (where $k = 0$ corresponds to OLS) or the equivalent degrees of freedom `$ df_k$` ([?@eq-dfk](#)). (The function uses the notation $\lambda \equiv k$, so the argument is `lambda`.) Among other quantities, `ridge()` returns a matrix containing the coefficients for each predictor for each shrinkage value and other quantities.

```
lambda <- c(0, 0.002, 0.005, 0.01, 0.02, 0.04, 0.08)
lridge <- ridge(Employed ~ GNP + Unemployed + Armed.Forces +
                  Population + Year + GNP.deflator,
                  data=longley, lambda=lambda)
print(lridge, digits = 2)
#> Ridge Coefficients:
#>          GNP    Unemployed Armed.Forces    Population       Year
#> 0.000 -3.447   -1.828    -0.696    -0.344     8.432
#> 0.002 -2.114   -1.644    -0.658    -0.713     7.466
#> 0.005 -1.042   -1.491    -0.623    -0.936     6.567
#> 0.010 -0.180   -1.361    -0.588    -1.003     5.656
#> 0.020  0.499   -1.245    -0.548    -0.868     4.626
#> 0.040  0.906   -1.155    -0.504    -0.523     3.577
#> 0.080  1.091   -1.086    -0.458    -0.086     2.642
#>          GNP.deflator
#> 0.000   0.157
#> 0.002   0.022
#> 0.005  -0.042
#> 0.010  -0.026
#> 0.020   0.098
#> 0.040   0.321
#> 0.080   0.570
```

The standard univariate plot, given by `traceplot()`, simply plots the estimated coefficients for each predictor against the shrinkage factor k .

```
traceplot(lridge,
  X = "lambda",
  xlab = "Ridge constant (k)",
  xlim = c(-0.02, 0.08), cex.lab=1.25)
```

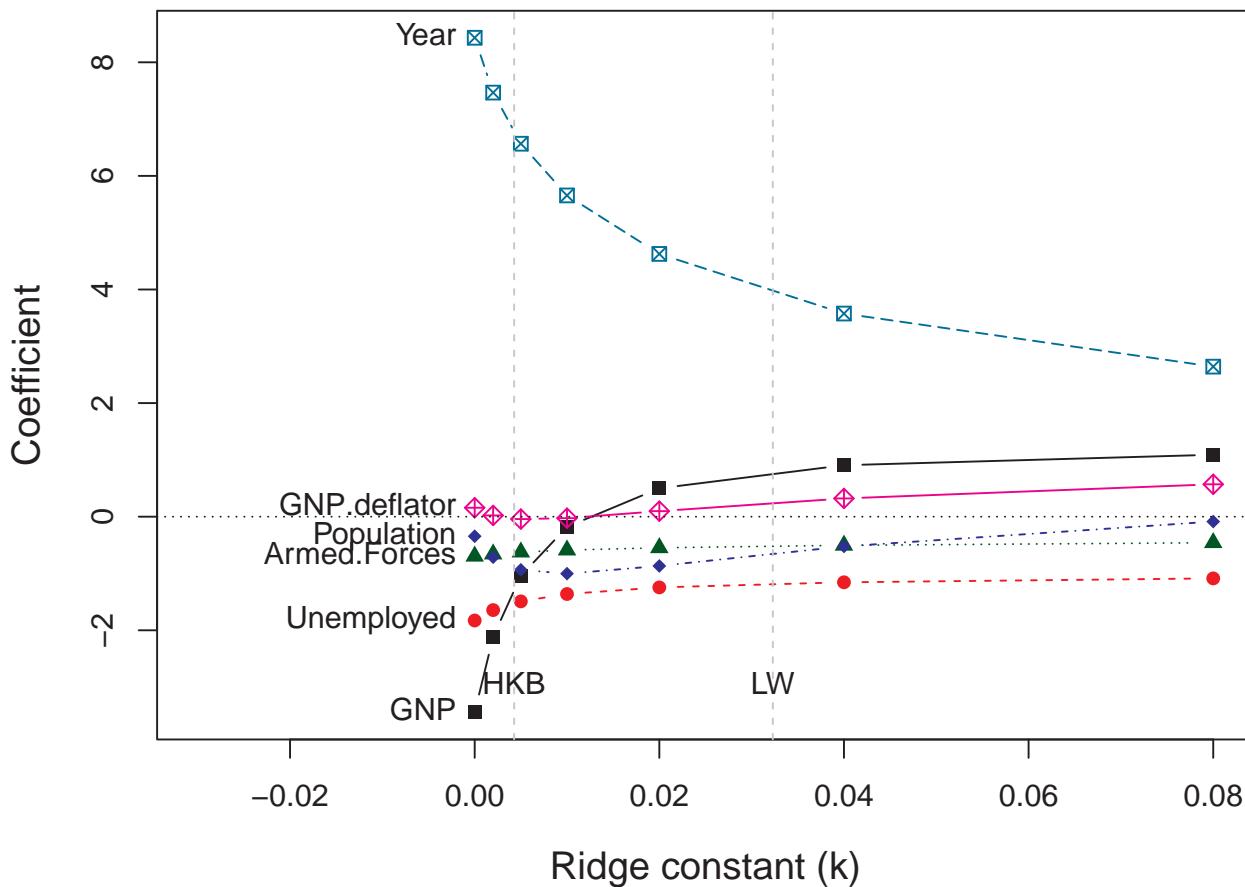


Figure 8.9: Univariate ridge trace plot for the coefficients of predictors of Employment in Longley's data via ridge regression, with ridge constants $k = (0, 0.002, 0.005, 0.01, 0.02, 0.04, 0.08)$. The dotted lines show optimal values for shrinkage by two criteria (HKB, LW).

You can see that the coefficients for Year and GNP are shrunk considerably. Differences from the β value at $k = 0$ represent the bias (smaller $|\beta|$) needed to achieve more stable estimates.

The dotted lines in Figure 8.9 show choices for the ridge constant by two commonly used criteria to balance bias against precision due to Hoerl et al. (1975) (HKB) and Lawless & Wang (1976) (LW). These values (along with a generalized cross-validation value GCV) are also stored in the “ridge” object as a vector `criteria`.

```
lridge$criteria
#>   kHKB      kW      kGCV
#> 0.00428  0.03230  0.00200
```

The shrinkage constant k doesn't have much intrinsic meaning, so it is often easier to interpret the plot when coefficients are plotted against the equivalent degrees of freedom, df_k . OLS corresponds to $df_k = 6$ degrees of freedom in the space of six parameters, and the effect of shrinkage is to decrease the degrees of freedom, as if estimating fewer parameters. This more natural scale also makes the changes in coefficient with shrinkage more nearly linear.

```
traceplot(lridge,
          X = "df",
          xlim = c(4, 6.2), cex.lab=1.25)
```

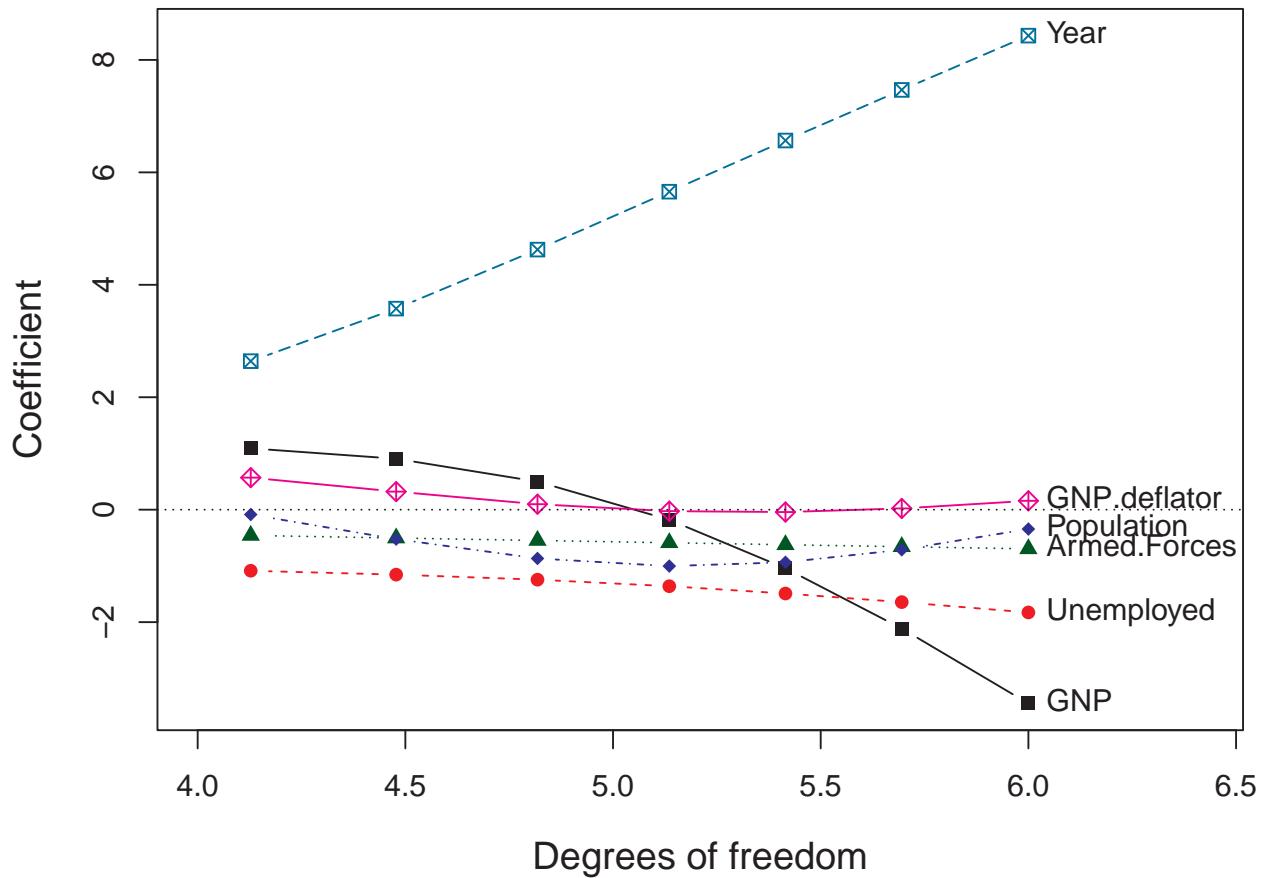


Figure 8.10: Univariate ridge trace plot using equivalent degrees of freedom, df_k to specify shrinkage. This scale is easier to understand and makes the traces of parameters more nearly linear.

But a bigger problem is that these univariate plots are the **wrong kind** of plot! They show the trends in increased bias (toward smaller $|\beta|$) associated with larger k , but they do not show the accompanying increase in *precision* (decrease in variance) achieved by allowing a bit of bias.

For that, we need to consider the variances and covariances of the estimated coefficients. The univariate trace plot is the wrong graphic form for what is essentially a multivariate problem, where we would like to visualize how *both* coefficients and their variances change with k .

8.8 Bivariate ridge trace plots

The bivariate analog of the trace plot suggested by Friendly (2013) plots bivariate confidence ellipses for *pairs* of coefficients. Their centers, $(\hat{\beta}_i, \hat{\beta}_j)$ compared to the OLS values show the bias induced for each coefficient, and also how the change in the ridge estimate for one parameter is related to changes for other parameters.

The size and shapes of the covariance ellipses show directly the effect on precision of the estimates as a

function of the ridge tuning constant. and their size and shape indicate sampling variance, $\widehat{\text{Var}}(\widehat{\beta}_{ij})$. Here, I plot those for GNP against four of the other predictors. The `plot()` method for "ridge" objects plots these ellipses for a pair of variables.

```
clr <- c("black", "red", "brown", "darkgreen", "blue", "cyan4", "magenta")
pch <- c(15:18, 7, 9, 12)
lambdaaf <- c(expression(~widehat(beta)^OLS), as.character(lambda[-1]))

for (i in 2:5) {
  plot(lridge, variables=c(1,i),
    radius=0.5, cex.lab=1.5, col=clr,
    labels=NULL, fill=TRUE, fill.alpha=0.2)
  text(lridge$coef[1,1], lridge$coef[1,i],
    expression(~widehat(beta)^OLS), cex=1.5, pos=4, offset=.1)
  text(lridge$coef[-1,c(1,i)], lambdaaf[-1], pos=3, cex=1.3)
}
```

As can be seen, the coefficients for each pair of predictors trace a path generally in toward the origin (0,0), and the covariance ellipses get smaller, indicating increased precision. Sometimes, these paths are rather direct, but it takes a peculiar curvilinear route in the case of population and GNP.

The `pairs()` method for "ridge" objects shows all pairwise views in scatterplot matrix form. `radius` sets the base size of the ellipse-generating circle for the covariance ellipses.

```
pairs(lridge, radius=0.5, diag.cex = 2,
      fill = TRUE, fill.alpha = 0.1)
```

8.8.1 Visualizing the bias-variance tradeoff

The function `precision()` calculates a number of measures of the effect of shrinkage of the coefficients in relation to the “size” of the covariance matrix $\mathcal{V}_k \equiv \widehat{\text{Var}}(\widehat{\beta}_k^{\text{RR}})$. Larger shrinkage k should lead to a smaller ellipsoid for \mathcal{V}_k , indicating increased precision.

```
pdat <- precision(lridge) |> print()
#>   lambda  df  det  trace max.eig norm.beta norm.diff
#> 0.000  0.000 6.00 -12.9 18.119  15.419     1.000     0.000
#> 0.002  0.002 5.70 -13.6 11.179   8.693     0.857     0.695
#> 0.005  0.005 5.42 -14.4  6.821   4.606     0.741     1.276
#> 0.010  0.010 5.14 -15.4  4.042   2.181     0.637     1.783
#> 0.020  0.020 4.82 -16.8  2.218   1.025     0.528     2.262
#> 0.040  0.040 4.48 -18.7  1.165   0.581     0.423     2.679
#> 0.080  0.080 4.13 -21.1  0.587   0.260     0.337     3.027
```

Here, the first three terms described below are (inverse) measures of precision; the last two quantify shrinkage:

- `det` = $\log |\mathcal{V}_k|$ is an overall measure of variance of the coefficients. It is the (linearized) volume of the covariance ellipsoid and corresponds conceptually to Wilks' Lambda criterion.
- `trace` = $\text{trace}(\mathcal{V}_k)$ is the sum of the variances and also the sum of the eigenvalues of \mathcal{V}_k , conceptually similar to Pillai's trace criterion.
- `max.eig` is the largest eigenvalue measure of size, an analog of Roy's maximum root test.
- `norm.beta` = $\|\beta\| / \max \|\beta\|$ is a summary measure of shrinkage, the normalized root mean square of the estimated coefficients. It starts at 1.0 for $k = 0$ and decreases with the penalty for large coefficients.

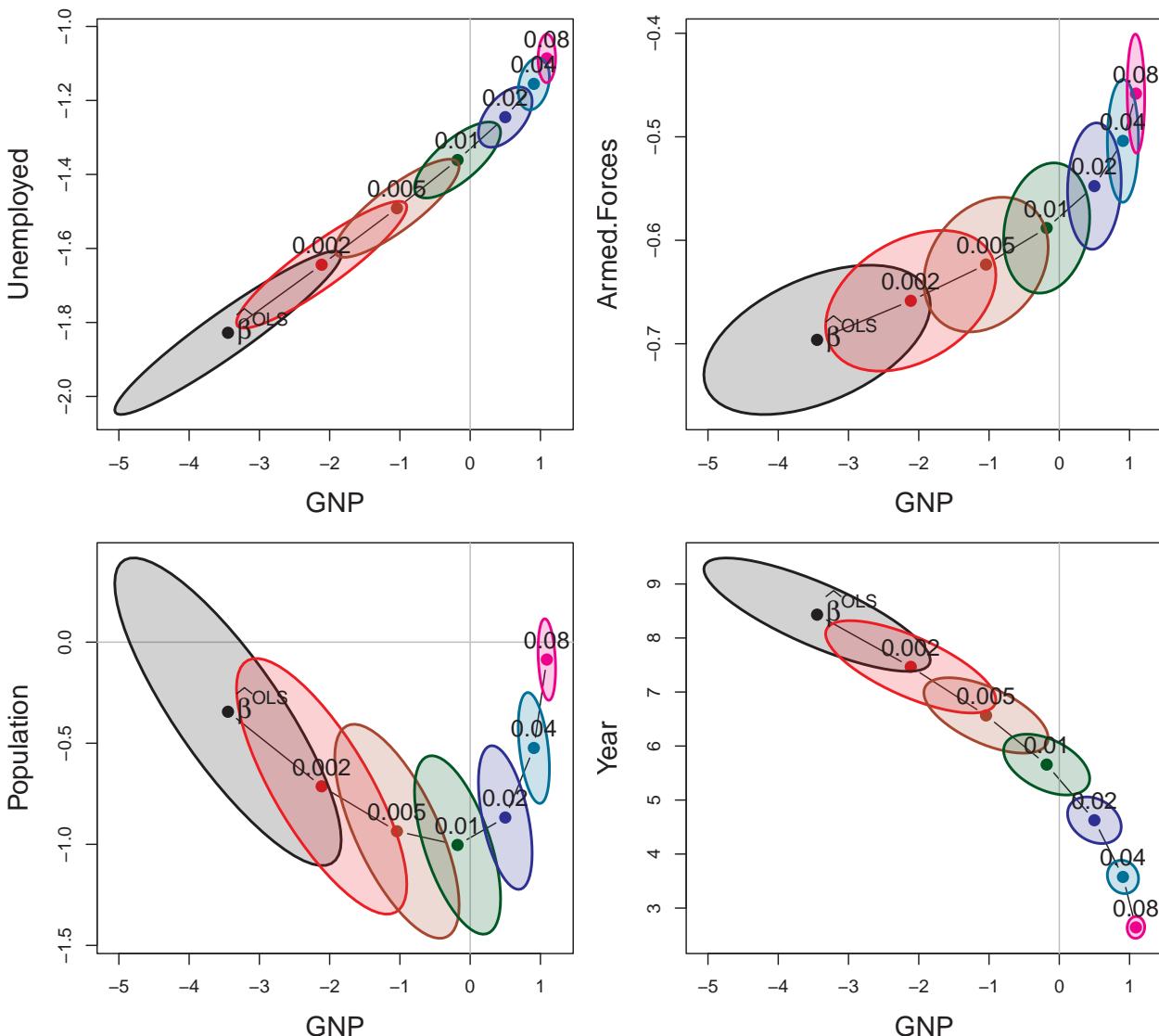


Figure 8.11: Bivariate ridge trace plots for the coefficients of four predictors against the coefficient for GNP in Longley's data, with $k = 0, 0.002, 0.005, 0.01, 0.02, 0.04, 0.08$. In most cases, the coefficients are driven toward zero, but the bivariate plot also makes clear the reduction in variance, as well as the bivariate path of shrinkage.

- `diff.beta` is the root mean square of the difference from the OLS estimate $\|\beta_{OLS} - \beta_k\|$. This measure is inversely related to `norm.beta`.

Plotting shrinkage against a measure of variance gives a direct view of the tradeoff between bias and precision. Here I plot `norm.beta` against `det`, and join the points with a curve. You can see that in this example the HKB criterion prefers a smaller degree of shrinkage, but achieves only a modest decrease in variance. But variance decreases more sharply thereafter and the LW choice achieves greater precision.

```
library(splines)
with(pdat, {
  plot(norm.beta, det, type="b",
    xlab="det", ylab="norm.beta",
    main="Bivariate Ridge Trace Plot for GNP Shrinkage")})
```

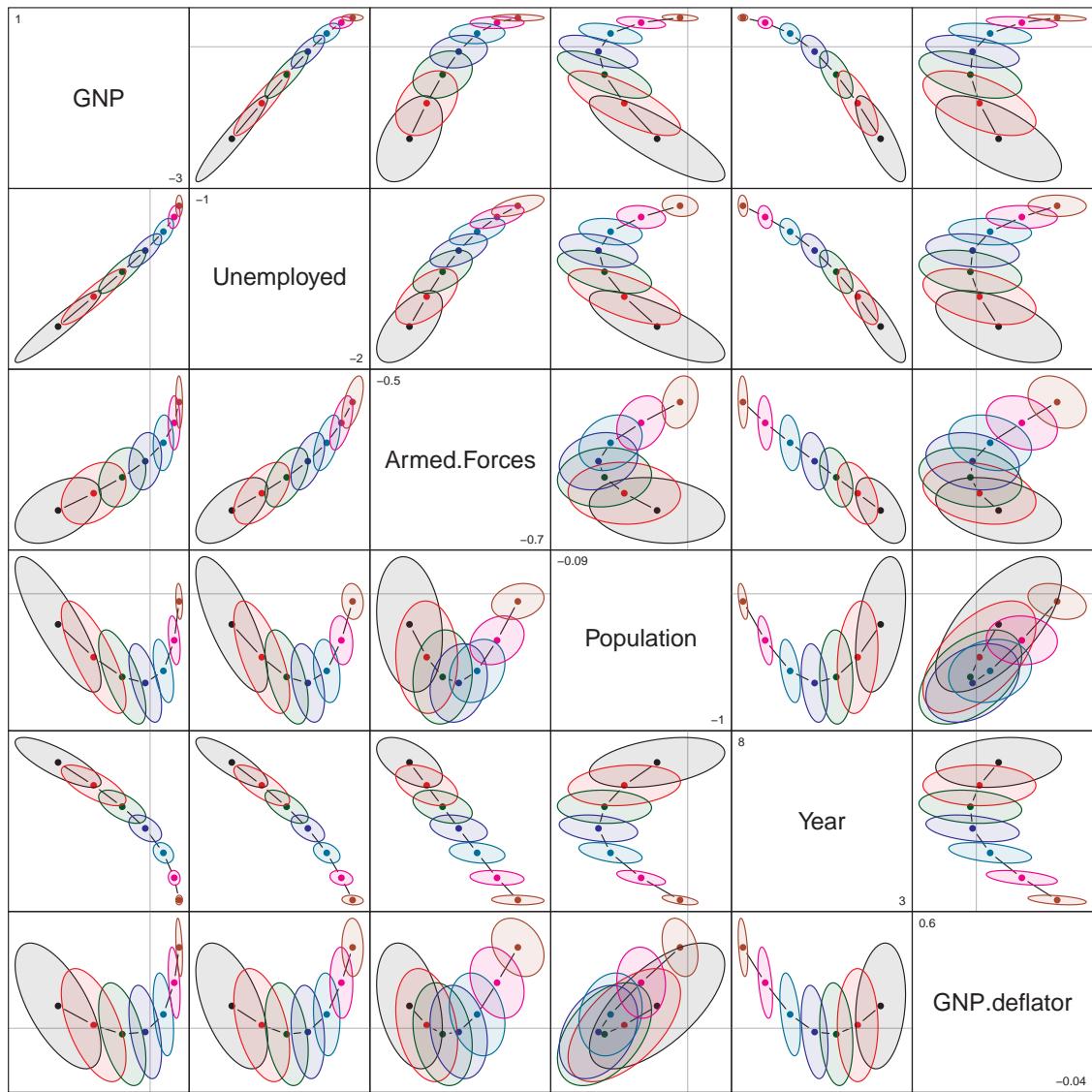


Figure 8.12: Scatterplot matrix of bivariate ridge trace plots. Each panel shows the effect of shrinkage on the covariance ellipse for a pair of predictors.

```

cex.lab=1.25, pch=16,
cex=1.5, col=clr, lwd=2,
xlab='shrinkage: ||b|| / max(||b||)',
ylab='variance: log |Var(b)|')
text(norm.beta, det,
      labels = lambdaf,
      cex = 1.25,
      pos = c(rep(2,length(lambda)-1),4))
text(min(norm.beta), max(det),
      labels = "log |Variance| vs. Shrinkage",
      cex=1.5, pos=4)
})

```

```
# find locations for optimal shrinkage criteria
mod <- lm(cbind(det, norm.beta) ~ bs(lambda, df=5),
           data=pdat)
x <- data.frame(lambda=c(lridge$kHKB,
                         lridge$kLW))
fit <- predict(mod, x)
points(fit[,2:1], pch=15,
       col=gray(.50), cex=1.6)
text(fit[,2:1], c("HKB", "LW"),
     pos=3, cex=1.5, col=gray(.50))
```

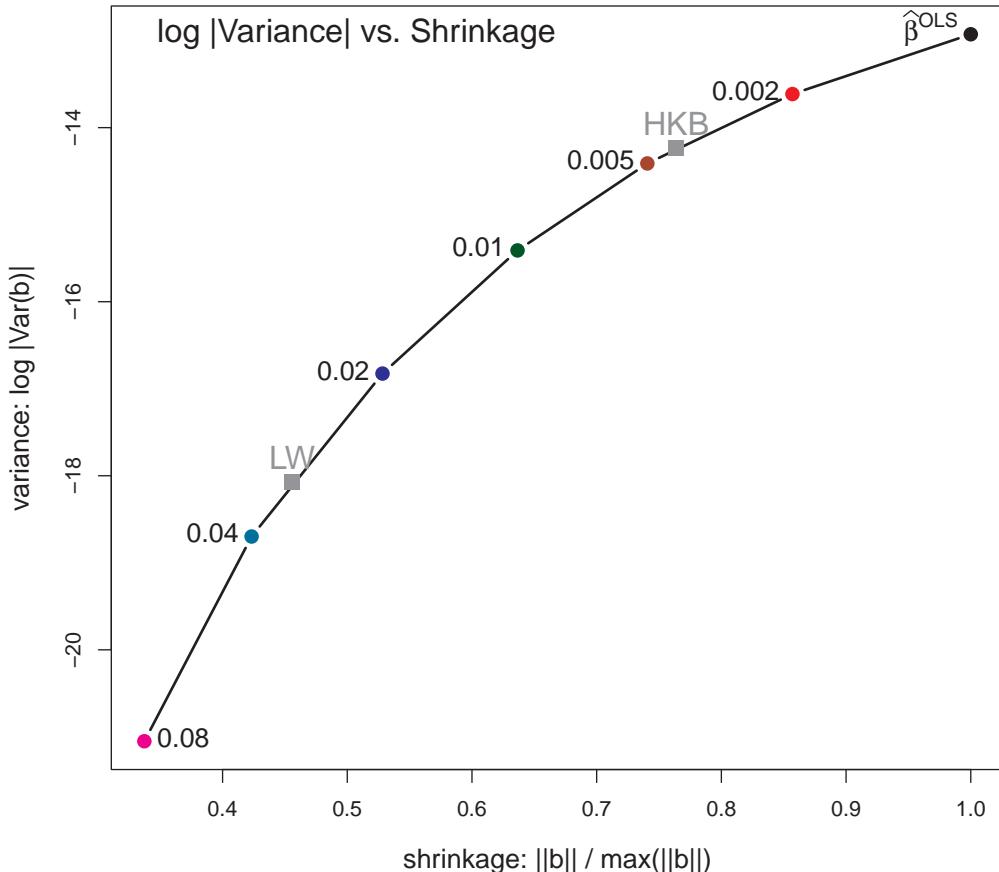


Figure 8.13: The tradeoff between bias and precision. Bias increases as we move away from the OLS solution, but precision increases.

8.9 Low-rank views

Just as principal components analysis gives low-dimensional views of a data set, PCA can be useful to understand ridge regression, just as it did for the problem of collinearity.

The `pca` method transforms a "ridge" object from parameter space, where the estimated coefficients are β_k with covariance matrices \mathbf{V}_k , to the principal component space defined by the right singular vectors, \mathbf{V} , of

the singular value decomposition \mathbf{UDV}^T of the scaled predictor matrix, \mathbf{X} . In PCA space the total variance of the predictors remains the same, but it is distributed among the linear combinations that account for successively greatest variance.

```
plridge <- pca(lridge)
plridge
#> Ridge Coefficients:
#>      dim1     dim2     dim3     dim4     dim5     dim6
#> 0.000  1.51541  0.37939  1.80131  0.34595  5.97391  6.74225
#> 0.002  1.51537  0.37935  1.80021  0.34308  5.69497  5.06243
#> 0.005  1.51531  0.37928  1.79855  0.33886  5.32221  3.68519
#> 0.010  1.51521  0.37918  1.79579  0.33205  4.79871  2.53553
#> 0.020  1.51500  0.37898  1.79031  0.31922  4.00988  1.56135
#> 0.040  1.51459  0.37858  1.77944  0.29633  3.01774  0.88291
#> 0.080  1.51377  0.37778  1.75810  0.25915  2.01876  0.47238
```

Then, a `traceplot()` of the resulting "pcaridge" object shows how the dimensions are affected by shrinkage, shown on the scale of degrees of freedom in Figure 8.14.

```
traceplot(plridge, X="df",
          cex.lab = 1.2, lwd=2)
```

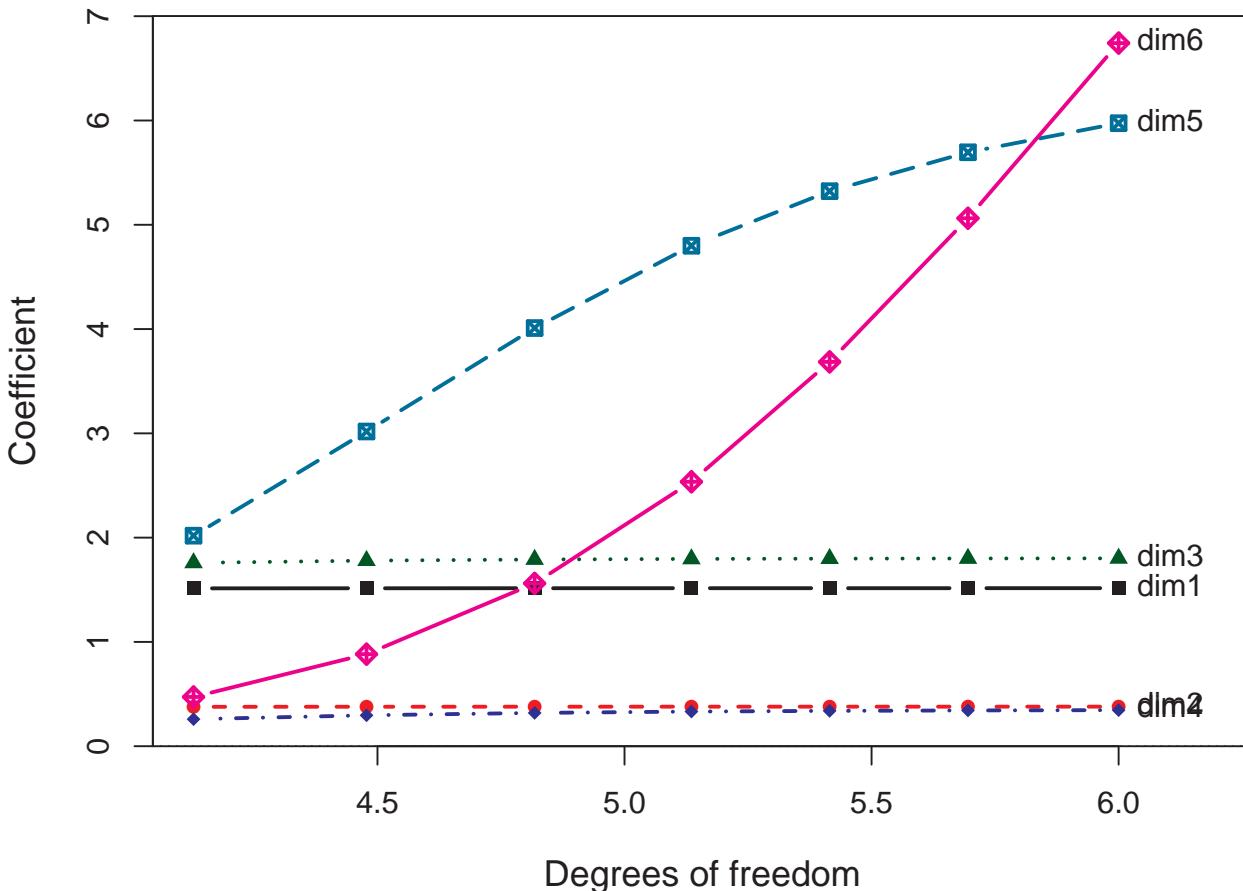


Figure 8.14: Ridge traceplot for the longley regression viewed in PCA space. The dimensions are the linear combinations of the predictors which account for greatest variance.

What may be surprising at first is that the coefficients for the first 4 components are not shrunk at all. These large dimensions are immune to ridge tuning. Rather, the effect of shrinkage is seen only on the *last two dimensions*. But those also are the directions that contribute most to collinearity as we saw earlier.

A `pairs()` plot gives a dramatic representation bivariate effects of shrinkage in PCA space: the principal components of X are uncorrelated, so the ellipses are all aligned with the coordinate axes and the ellipses largely coincide for dimensions 1 to 4. You can see them shrink in one direction in the last two columns and rows.

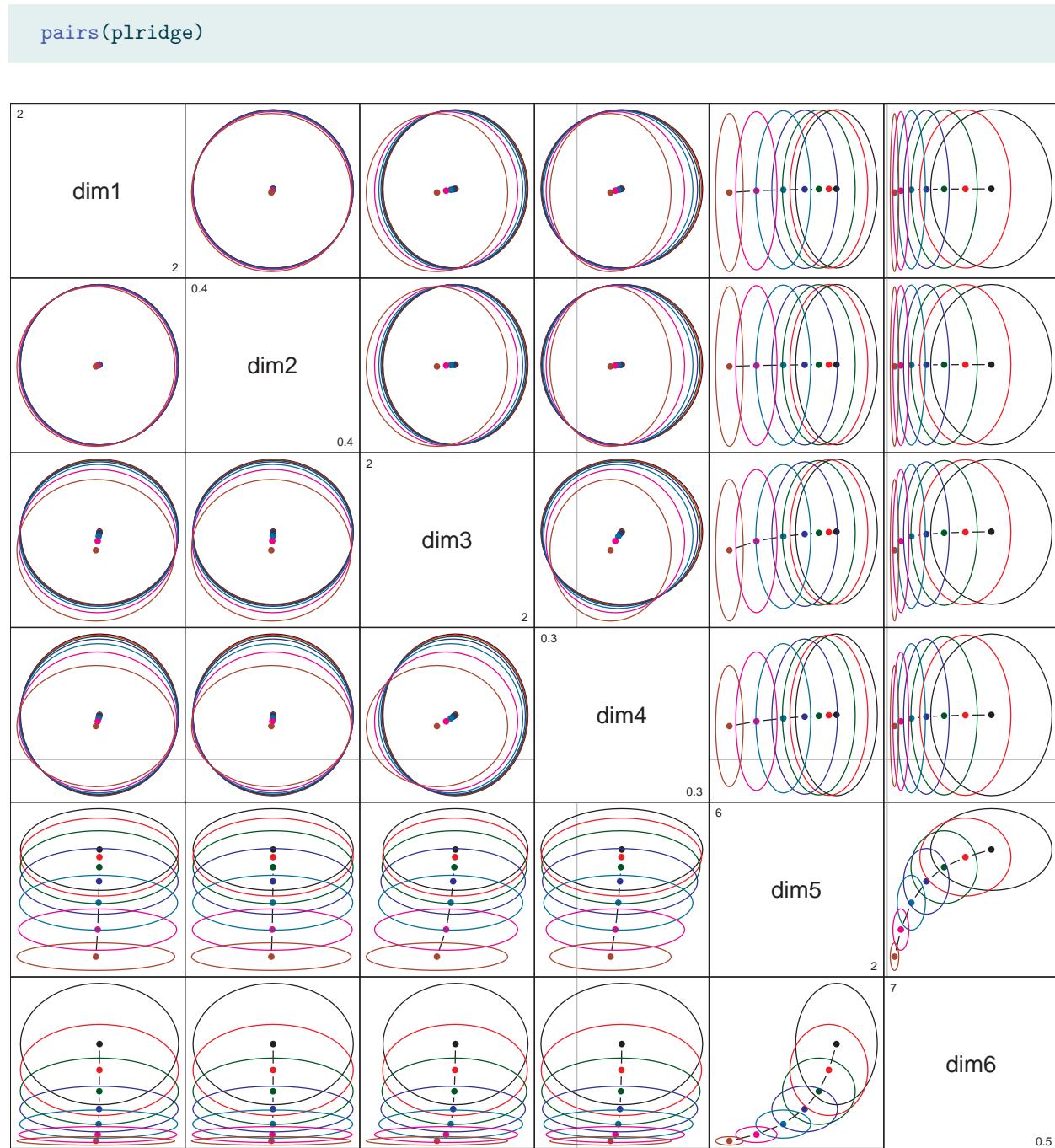


Figure 8.15: All pairwise bivariate ridge plots shown in PCA space.

If we focus on the plot of dimensions 5:6, we can see where all the shrinkage action is in this representation. Generally, the predictors that are related to the smallest dimension (6) are shrunk quickly at first.

```
plot(plridge, variables=5:6,
      fill = TRUE, fill.alpha=0.15, cex.lab = 1.5)
text(plridge$coef[, 5:6],
     label = lambdaef,
     cex=1.5, pos=4, offset=.1)
```

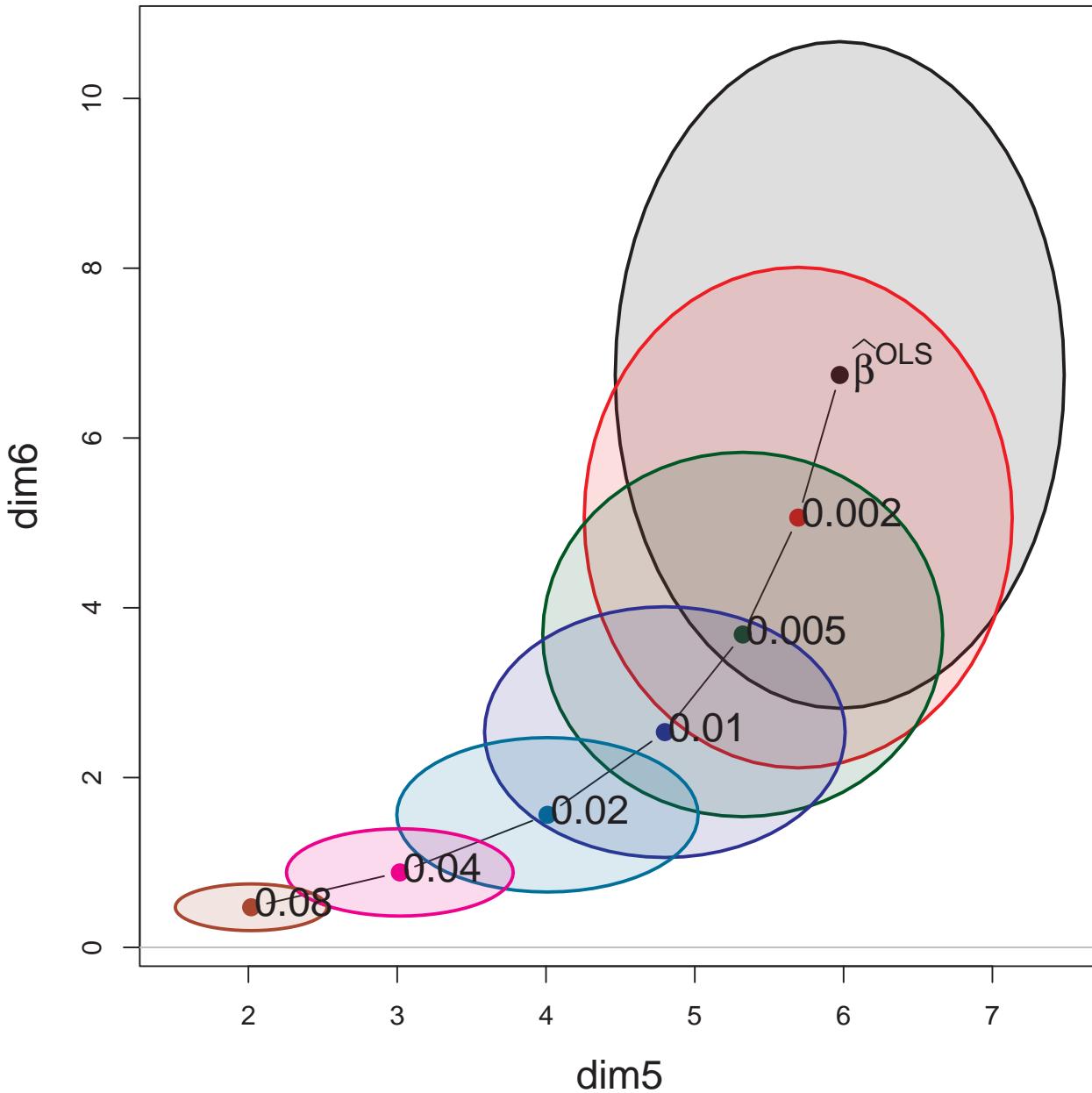


Figure 8.16: Bivariate ridge trace plot for the smallest two dimensions ...

8.9.1 Biplot view

The question arises how to relate this view of shrinkage in PCA space to the original predictors. The biplot is again your friend. You can project variable vectors for the predictor variables into the PCA space of the smallest dimensions, where the shrinkage action mostly occurs to see how the predictor variables relate to these dimensions.

`biplot.pcaridge()` supplements the standard display of the covariance ellipsoids for a ridge regression problem in PCA/SVD space with labeled arrows showing the contributions of the original variables to the dimensions plotted. Recall from Section 4.3 that these reflect the correlations of the variables with the PCA dimensions. The lengths of the arrows reflect the proportion of variance that each predictors shares with the components.

```
biplot(plridge, radius=0.5,
       ref=FALSE, asp=1,
       var.cex=1.15, cex.lab=1.3, col=clr,
       fill=TRUE, fill.alpha=0.15,
       prefix="Dimension ")
#> Vector scale factor set to 5.25
text(plridge$coef[,5:6], lambdaef, pos=2, cex=1.3)
```

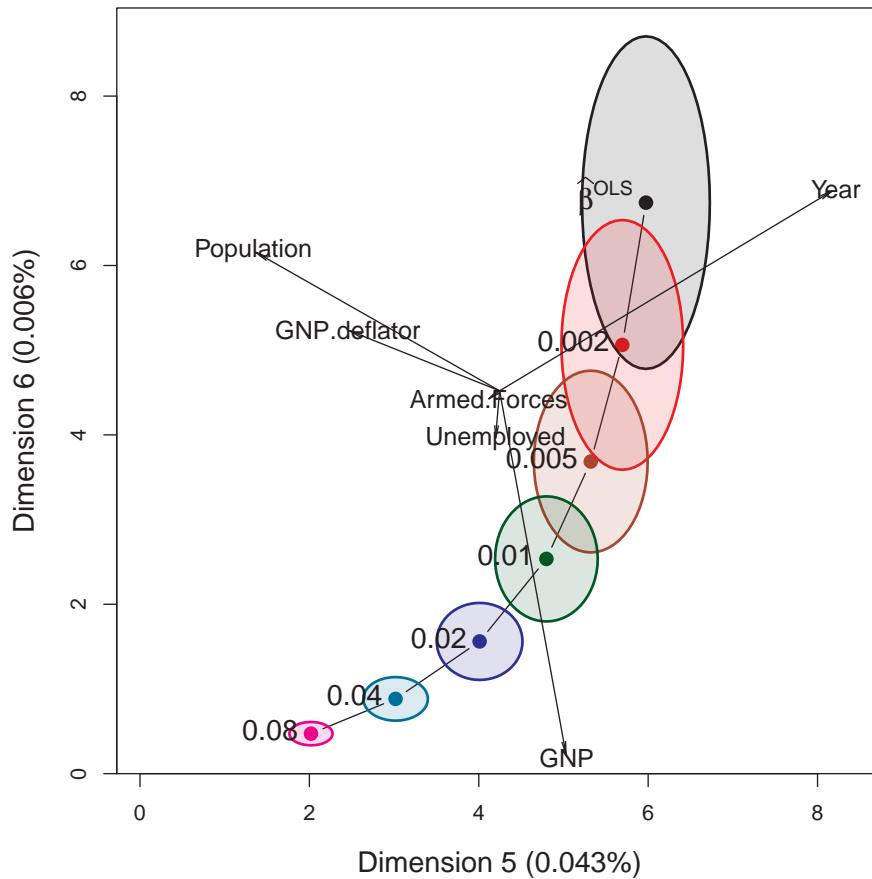


Figure 8.17: Biplot view of the ridge trace plot for the smallest two dimensions, where the effects of shrinkage are most apparent.

The biplot view in Figure 8.17 showing the two smallest dimensions is particularly useful for understanding

how the predictors contribute to shrinkage in ridge regression. Here, Year and Population largely contribute to dimension 5; a contrast between (Year, Population) and GNP contributes to dimension 6.

8.10 What have we learned?

This chapter has considered the problems in regression models which stem from high correlations among the predictors. We saw that collinearity results in unstable estimates of coefficients with larger uncertainty, often dramatically more so than would be the case if the predictors were uncorrelated. Collinearity can be seen as merely a “data problem” which can safely be ignored if we are only interested in prediction. When we want to understand a model, ridge regression can tame the collinearity beast by shrinking the coefficients slightly to gain greater precision in the estimates.

Beyond these statistical considerations, the methods of this chapter highlight the roles of multivariate thinking and visualization in understanding these phenomena and the methods developed for solving them. Data ellipses and confidence ellipses for coefficients again provide tools for visualizing what is concealed in numerical summaries. A perhaps surprising feature of both collinearity and ridge regression is that the important information usually resides in the smallest PCA dimensions and biplots help again to understand these dimensions.

Part IV

Multivariate Linear Models

Hotelling's T^2

Just as the one- and two- sample univariate t -test is the gateway drug for understanding analysis of variance, so too Hotelling's T^2 test provides an entry point to multivariate analysis of variance. This simple case provides an entry point to understanding the collection of methods I call the **HE plot framework** for visualizing effects in multivariate linear models, which are a main focus of this book.

The essential idea is that Hotelling's T^2 provides a test of the difference in means between two groups on a *collection* of variables, $\mathbf{x} = x_1, x_2, \dots, x_p$ *simultaneously*, rather than one by one. This has the advantages that it:

- does not require p -value corrections for multiple tests (e.g., Bonferroni);
- combines the evidence from the multiple response variables, and *pools strength*, accumulating support across measures;
- clarifies how the multiple response are *jointly* related to the group effect along a single dimension, the *discriminant axis*;
- in so doing, it reduces the problem of testing differences for two (and potentially more) response variables to testing the difference on a single variable that best captures the multivariable relations.

After describing it's features, I use an example of a two-group T^2 test to illustrate the basic ideas behind multivariate tests and hypothesis error plots. Then, we'll dip our toes into the visual ideas for representing the statistical quantities involved in such tests.

Packages

In this chapter I use the following packages. Load them now.

```
library(car)
library(heplots)
library(Hotelling)
library(ggplot2)
library(dplyr)
library(tidyr)
library(ggbiplot)
library(broom)
```

9.1 T^2 as a generalized t -test

Hotelling's T^2 ([Hotelling, 1931](#)) is an analog of the square of a univariate t statistic, extended to the case of two or more response variables tested together. Consider the basic one-sample t -test, where we wish to test the hypothesis that the mean \bar{x} of a set of N measures on a test of basic math, with standard deviation s does not differ from an assumed mean $\mu_0 = 150$ for a population. The t statistic for testing $H_0 : \mu = \mu_0$ against the two-sided alternative, $H_0 : \mu \neq \mu_0$ is

$$t = \frac{(\bar{x} - \mu_0)}{s/\sqrt{N}} = \frac{(\bar{x} - \mu_0)\sqrt{N}}{s}$$

Squaring this gives

$$t^2 = \frac{N(\bar{x} - \mu_0)^2}{s^2} = N(\bar{x} - \mu_0)(s^2)^{-1}(\bar{x} - \mu_0)$$

Now consider we also have measures on a test of solving word problems for the same sample. Then, a hypothesis test for the means on basic math (BM) and word problems (WP) is the test of the means of these two variables jointly equal some specified values, say, ($\mu_{0,BM} = 150$, $\mu_{0,WP} = 100$):

$$\mathcal{H}_0 : \mu = \mu_0 = \begin{pmatrix} \mu_{0,BM} \\ \mu_{0,WP} \end{pmatrix} = \begin{pmatrix} 150 \\ 100 \end{pmatrix}$$

Hotelling's T^2 is then the analog of t^2 , with the variance-covariance matrix \mathbf{S} of the scores on (BM, WP) replacing the variance of a single score. This is nothing more than the squared Mahalanobis D_M^2 distance between the sample mean vector $(\bar{x}_{BM}, \bar{x}_{WP})^\top$ and the hypothesized means μ_0 , in the metric of \mathbf{S} , as shown in Figure 9.1.

$$\begin{aligned} T^2 &= N(\bar{\mathbf{x}} - \mu_0)^\top \mathbf{S}^{-1} (\bar{\mathbf{x}} - \mu_0) \\ &= ND_M^2(\bar{\mathbf{x}}, \mu_0) \end{aligned}$$

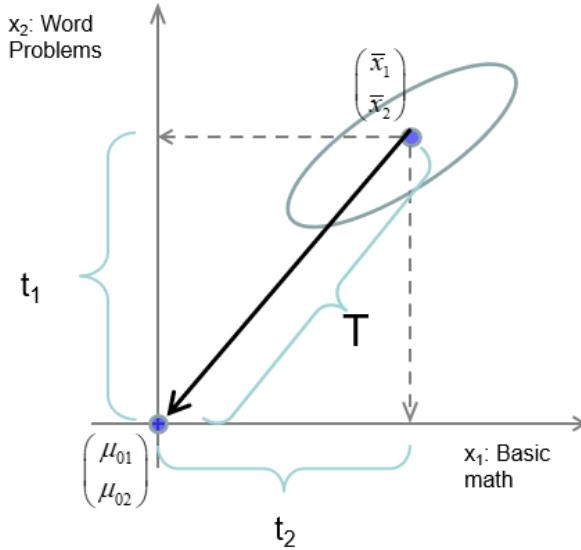


Figure 9.1: Hotelling's T^2 statistic as the squared distance between the sample means and hypothesized means relative to the variance-covariance matrix. *Source:* Author

9.2 T^2 properties

Aside from its elegant geometric interpretation Hotelling's T^2 has simple properties that aid in understanding the extension to more complex multivariate tests.

- **Maximum t^2** : Consider constructing a new variable w as a linear combination of the scores in a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p]$ with weights \mathbf{a} ,

$$w = a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \dots + a_p\mathbf{x}_p = \mathbf{X}\mathbf{a}$$

Hotelling's T^2 is then the maximum value of a univariate $t^2(\mathbf{a})$ over all possible choices of the weights in \mathbf{a} . In this way, Hotellings test reduces a multivariate problem to a univariate one.

- **Eigenvalue** : Hotelling showed that T^2 is the one non-zero eigenvalue (latent root) λ of the matrix $\mathbf{Q}_H = N(\bar{\mathbf{x}} - \mu_0)^T(\bar{\mathbf{x}} - \mu_0)$ relative to $\mathbf{Q}_E = \mathbf{S}$ that solves the equation

$$(\mathbf{Q}_H - \lambda\mathbf{Q}_E)\mathbf{a} = 0 \quad (9.1)$$

In more complex MANOVA problems, there are more than one non-zero latent roots, $\lambda_1, \lambda_2, \dots, \lambda_s$, and test statistics (Wilks' Λ , Pillai and Hotelling-Lawley trace criteria, Roy's maximum root test) are functions of these.

- **Eigenvector** : The corresponding eigenvector is $\mathbf{a} = \mathbf{S}^{-1}(\bar{\mathbf{x}} - \mu_0)$. These are the (raw) *discriminant coefficients*, giving the relative contribution of each variable to T^2 .
- **Critical values** : For a single response, the square of a t statistic with $N - 1$ degrees of freedom is an $F(1, N - 1)$ statistic. But we chose \mathbf{a} to give the *maximum $t^2(\mathbf{a})$* ; this can be taken into account with a transformation of T^2 to give an **exact F** test with the correct sampling distribution:

$$F^* = \frac{N - p}{p(N - 1)} T^2 \sim F(p, N - p) . \quad (9.2)$$

- **Invariance under linear transformation** : Just as a univariate t -test is unchanged if we apply a linear transformation to the variable, $x \rightarrow ax + b$, T^2 is invariant under all linear (*affine*) transformations,

$$\mathbf{x}_{p \times 1} \rightarrow \mathbf{C}_{p \times p}\mathbf{x} + \mathbf{b}$$

So, you get the same results if you convert penguins flipper lengths from millimeters to centimeters or inches. The same is true for all MANOVA tests.

- **Two-sample tests** : With minor variations in notation, everything above applies to the more usual test of equality of multivariate means in a two sample test of $\mathcal{H}_0 : \mu_1 = \mu_2$.

$$T^2 = N(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \mathbf{S}_p^{-1} (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$$

where \mathbf{S}_p is the pooled within-sample variance covariance matrix.

Example: Maths score data

The data set `mathscore` gives (fictitious) scores on a test of basic math skills (BM) and solving word problems (WP) for two groups of $N = 6$ students in an algebra course, each taught by different instructors. The null hypothesis is that the means are equal for both variables, $\mathcal{H}_0 : \mu_{\text{BM}} = \mu_{\text{WP}}$.

```
data(mathscore, package = "heplots")
str(mathscore)
#> 'data.frame': 12 obs. of 3 variables:
#> $ group: Factor w/ 2 levels "1","2": 1 1 1 1 1 2 2 2 2 ...
#> $ BM   : int 190 170 180 200 150 180 160 190 150 160 ...
#> $ WP   : int 90 80 80 120 60 70 120 150 90 130 ...
```

You can carry out the test that the means for both variables are jointly equal across groups using either

`Hotelling::hotelling.test()` ([Curran & Hersh, 2021](#)) or `car::Anova()`, but the latter is more generally useful

```
hotelling.test(cbind(BM, WP) ~ group, data=mathscore) |> print()
#> Test stat: 64.174
#> Numerator df: 2
#> Denominator df: 9
#> P-value: 0.0001213

math.mod <- lm(cbind(BM, WP) ~ group, data=mathscore)
Anova(math.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> group    1     0.865     28.9      2     9 0.00012 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What's wrong with just doing the two t -tests (or equivalent F -test with `lm()`)?

```
Anova(mod1 <- lm(BM ~ group, data=mathscore))
#> Anova Table (Type II tests)
#>
#> Response: BM
#>          Sum Sq Df F value Pr(>F)
#> group      1302  1   4.24  0.066 .
#> Residuals  3071 10
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Anova(mod2 <- lm(WP ~ group, data=mathscore))
#> Anova Table (Type II tests)
#>
#> Response: WP
#>          Sum Sq Df F value Pr(>F)
#> group      4408  1   10.4  0.009 **
#> Residuals  4217 10
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From this, we might conclude that the two groups do *not* differ significantly on Basic Math but strongly differ on Word problems. But the two univariate tests do not take the correlation among the mean differences into account.

If you want to just extract the t -tests, here's a handy trick using `broom::tidy.mlm()`, which summarizes the test statistics for each response and each term in a MLM. The mean difference shown below is that for group 2 - group 1.

```
tidy(math.mod) |>
  filter(term != "(Intercept)") |>
  select(-term) |>
  rename(Mean_diff = estimate,
        t = statistic) |>
```

```

  mutate(signif = noquote(gtools::stars.pval(p.value)))
#> # A tibble: 2 x 6
#>   response Mean_diff std.error      t p.value signif
#>   <chr>     <dbl>     <dbl> <dbl>    <dbl> <noquote>
#> 1 BM        -20.8     10.1  -2.06  0.0665  .
#> 2 WP         38.3     11.9   3.23  0.00897 **
```

To see the differences between the groups on both variables together, we draw their data (68%) ellipses, using `heplots::covEllipses()`. (Setting `pooled=FALSE` here omits drawing the ellipse for the pooled covariance matrix \mathbf{S}_p .)

```

colors <- c("darkgreen", "blue")
covEllipses(mathscore[,c("BM", "WP")], mathscore$group,
            pooled = FALSE,
            col = colors,
            fill = TRUE,
            fill.alpha = 0.05,
            cex = 2, cex.lab = 1.5,
            asp = 1,
            xlab="Basic math", ylab="Word problems")
# plot points
pch <- ifelse(mathscore$group==1, 15, 16)
col <- ifelse(mathscore$group==1, colors[1], colors[2])
points(mathscore[,2:3], pch=pch, col=col, cex=1.25)
```

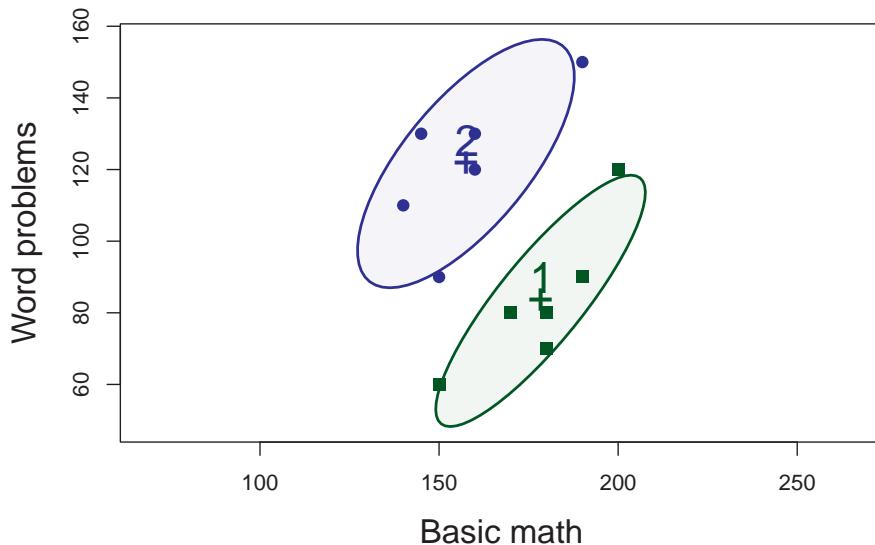


Figure 9.2: Data ellipses for the `mathscore` data, enclosing approximately 68% of the observations in each group

We can see that:

- Group 1 > Group 2 on Basic Math, but worse on Word Problems
- Group 2 > Group 1 on Word Problems, but worse on Basic Math
- Within each group, those who do better on Basic Math also do better on Word Problems

We can also see why the univariate test, at least for Basic math is non-significant: the scores for the two groups overlap considerably on the horizontal axis. They are slightly better separated along the vertical axis

for word problems. The plot also reveals why Hotelling's T^2 reveals such a strongly significant result: the two groups are very widely separated along an approximately 45° line between them.

A relatively simple interpretation is that the groups don't really differ in overall math ability, but perhaps the instructor in Group 1 put more focus on basic math skills, while the instructor for Group 2 placed greater emphasis on solving word problems.

In Hotelling's T^2 , the “size” of the difference between the means (labeled “1” and “2”) is assessed relative to the pooled within-group covariance matrix \mathbf{S}_p , which is just a size-weighted average of the two within-sample matrices, \mathbf{S}_1 and \mathbf{S}_2 ,

$$\mathbf{S}_p = [(n_1 - 1)\mathbf{S}_1 + (n_2 - 1)\mathbf{S}_2]/(n_1 + n_2 - 2) .$$

Visually, imagine sliding the the separate data ellipses to the grand mean, $(\bar{x}_{BM}, \bar{x}_{WP})$ and finding their combined data ellipse. This is just the data ellipse of the sample of deviations of the scores from their group means, or that of the residuals from the model `lm(cbind(BM, WP) ~ group, data=mathscore)`

To see this, we plot \mathbf{S}_1 , \mathbf{S}_2 and \mathbf{S}_p together,

```
covEllipses(mathscore[,c("BM", "WP")], mathscore$group,
            col = c(colors, "red"),
            fill = c(FALSE, FALSE, TRUE),
            fill.alpha = 0.3,
            cex = 2, cex.lab = 1.5,
            asp = 1,
            xlab="Basic math", ylab="Word problems")
```

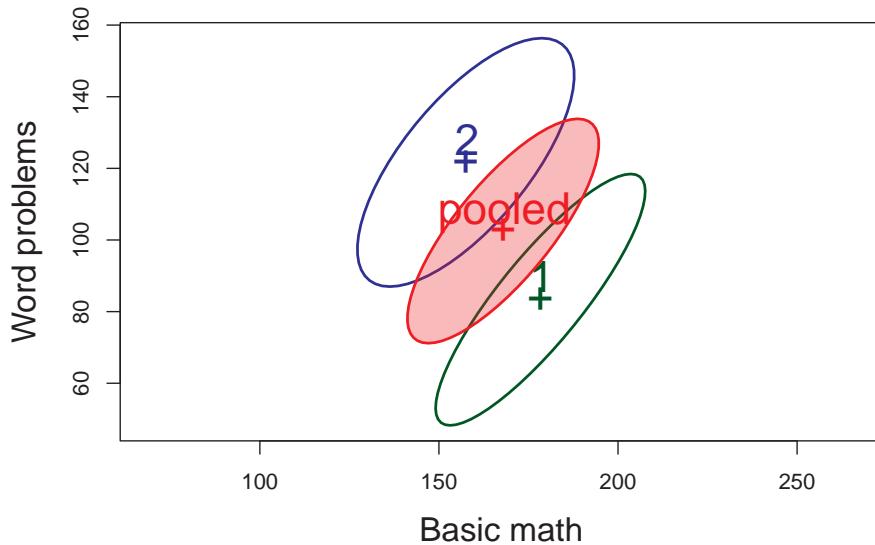


Figure 9.3: Data ellipses and the pooled covariance matrix `mathscore` data.

One of the assumptions of the T^2 test (and of MANOVA) is that the within-group variance covariance matrices, \mathbf{S}_1 and \mathbf{S}_2 , are the same. In Figure 9.3, you can see how the shapes of \mathbf{S}_1 and \mathbf{S}_2 are very similar, differing in that the variance of word Problems is slightly greater for group 2. In Chapter XX we take of the topic of visualizing tests of this assumption, based on Box's M -test.

9.3 HE plot and discriminant axis

As we describe in detail in Chapter 11, all the information relevant to the T^2 test and MANOVA can be captured in the remarkably simple *Hypothesis Error* plot, which shows the relative size of two data ellipses,

- **H:** the data ellipse of the *fitted* values, which are just the group means on the two variables, $\bar{\mathbf{x}}$, corresponding to \mathbf{Q}_H in Equation 9.1. In case of T^2 , the **H** matrix is of rank 1, so the “ellipse” plots as a line.

```
# calculate H directly
fit <- fitted(math.mod)
xbar <- colMeans(mathscore[,2:3])
N <- nrow(mathscore)
crossprod(fit) - N * outer(xbar, xbar)
#>      BM     WP
#> BM  1302 -2396
#> WP -2396  4408

# same as: SSP for group effect from Anova
math.aov <- Anova(math.mod)
(H <- math.aov$SSP)
#> $group
#>      BM     WP
#> BM  1302 -2396
#> WP -2396  4408
```

- **E:** the data ellipse of the *residuals*, the deviations of the scores from the group means, $\mathbf{x} - \bar{\mathbf{x}}$, corresponding to \mathbf{Q}_E .

```
# calculate E directly
resids <- residuals(math.mod)
crossprod(resids)
#>      BM     WP
#> BM 3071 2808
#> WP 2808 4217

# same as: SSPE from Anova
(E <- math.aov$SSPE)
#>      BM     WP
#> BM 3071 2808
#> WP 2808 4217
```

9.3.1 heplot()

`heplots::heplot()` takes the model object, extracts the **H** and **E** matrices (from `summary(Anova(math.mod))`) and plots them. There are many options to control the details.

```
heplot(math.mod,
       fill=TRUE, lwd = 3,
       asp = 1,
```

```
cex=2, cex.lab=1.8,
xlab="Basic math", ylab="Word problems")
```

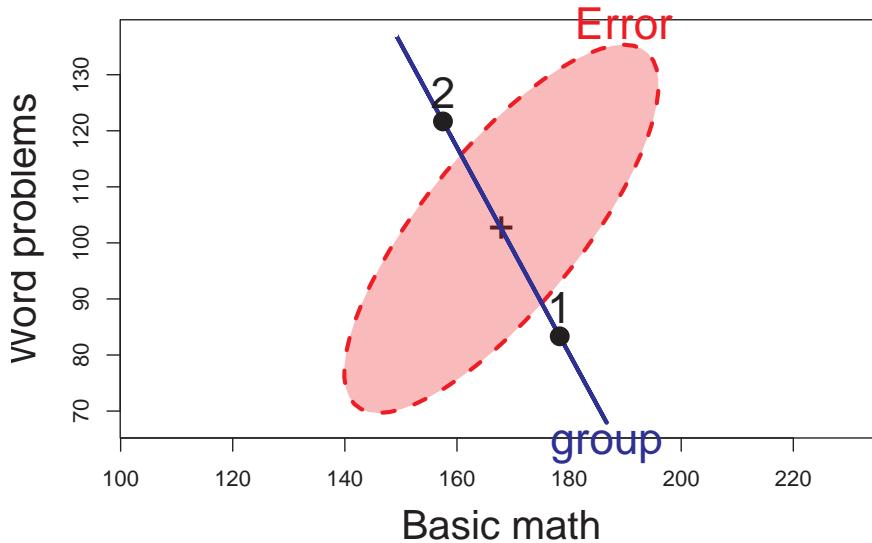


Figure 9.4: Hypothesis error plot of the `mathscore` data. The line through the group means is the **H** ellipse, which plots as a line here. The red ellipse labeled ‘Error’ represents the pooled within-group covariance matrix.

But the HE plot offers more:

- A visual test of significance: the **H** ellipse is scaled so that it projects *anywhere* outside the **E** ellipse, if and only if the test is significant at a given α level ($\alpha = 0.05$ by default)
- The **H** ellipse, which appears as a line, goes through the means of the two groups. This is also the *discriminant axis*, the direction in the space of the variables which maximally discriminates between the groups. That is, if we project the data points onto this line, we get the linear combination w which has the maximum possible univariate t^2 .

You can see how the HE plot relates to the plots of the separate data ellipses by overlaying them in a single figure. We also plot the scores on the discriminant axis, by using this small function to find the orthogonal projection of a point **a** on the line joining two points, **p₁** and **p₂**, which in math is $\mathbf{p}_1 + \frac{\mathbf{d}^\top(\mathbf{a}-\mathbf{p}_1)}{\mathbf{d}^\top\mathbf{d}} \mathbf{d}$, letting $\mathbf{d} = \mathbf{p}_1 - \mathbf{p}_2$.

```
dot <- function(x, y) sum(x*y)      # dot product of two vectors
project_on <- function(a, p1, p2) {
  a <- as.numeric(a)
  p1 <- as.numeric(p1)
  p2 <- as.numeric(p2)
  t <- dot(p2-p1, a-p1) / dot(p2-p1, p2-p1)
  C <- p1 + t*(p2-p1)
  C
}
```

Then, we run the same code as before to plot the data ellipses, and follow this with a call to `heplot()` using the option `add=TRUE` which adds to an existing plot. Following this, we find the group means and draw lines projecting the points on the line between them.

```

covEllipses(mathscore[,c("BM", "WP")], mathscore$group,
            pooled=FALSE,
            col = colors,
            cex=2, cex.lab=1.5,
            asp=1,
            xlab="Basic math", ylab="Word problems"
            )
pch <- ifelse(mathscore$group==1, 15, 16)
col <- ifelse(mathscore$group==1, "red", "blue")
points(mathscore[,2:3], pch=pch, col=col, cex=1.25)

# overlay with HEplot (add = TRUE)
heplot(math.mod,
       fill=TRUE,
       cex=2, cex.lab=1.8,
       fill.alpha=0.2, lwd=c(1,3),
       add = TRUE,
       error.ellipse=TRUE)

# find group means
means <- mathsore |>
  group_by(group) |>
  summarize(BM = mean(BM), WP = mean(WP))

for(i in 1:nrow(mathsore)) {
  gp <- mathsore$group[i]
  pt <- project_on( mathsore[i, 2:3], means[1, 2:3], means[2, 2:3])
  segments(mathsore[i, "BM"], mathsore[i, "WP"], pt[1], pt[2], lwd = 1.2)
}

```

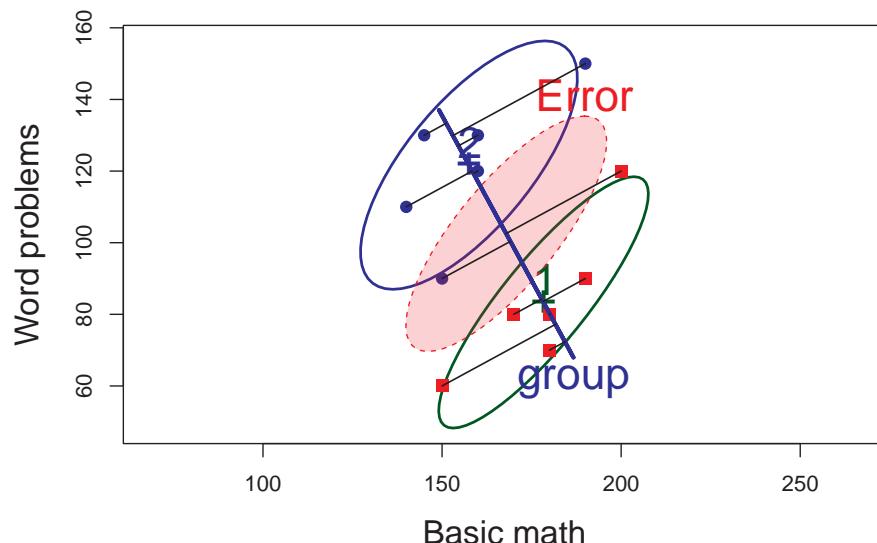


Figure 9.5: HE plot overlaid on top of the within-group data ellipses, with lines showing the projection of each point on the discriminant axis.

9.4 Discriminant analysis

Discriminant analysis for two-group designs or for one-way MANOVA essentially turns the problem around: Instead of asking whether the mean vectors for two or more groups are equal, discriminant analysis tries to find the linear combination w of the response variables that has the greatest separation among the groups, allowing cases to be best classified.

For the maths score data, you can perform the discriminant analysis as follows, using the **MASS** function **lda()**:

```
(math.lda <- MASS::lda(group ~ ., data=mathscore))
#> Call:
#> lda(group ~ ., data = mathscore)
#>
#> Prior probabilities of groups:
#>   1   2
#> 0.5 0.5
#>
#> Group means:
#>   BM    WP
#> 1 178  83.3
#> 2 158 121.7
#>
#> Coefficients of linear discriminants:
#>   LD1
#> BM -0.0835
#> WP  0.0753
```

The coefficients give $w = -0.084 \text{ BM} + 0.075 \text{ WP}$. This is exactly the direction given by the line for the **H** ellipse in Figure 9.5.

To round this out, we can calculate the discriminant scores by multiplying the matrix \mathbf{X} by the vector $\mathbf{a} = \mathbf{S}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$ of the discriminant weights. These were shown in Figure 9.5 as the projections of the data points on the line joining the group means,

```
math.lda$scaling
#>       LD1
#> BM -0.0835
#> WP  0.0753

scores <- cbind(group = mathscore$group,
                 as.matrix(mathscore[, 2:3]) %*% math.lda$scaling) |>
  as.data.frame()
scores |>
  group_by(group) |>
  slice(1:3)
#> # A tibble: 6 x 2
#> # Groups:   group [2]
#>   group   LD1
#>   <dbl> <dbl>
#> 1     1 -9.09
#> 2     1 -8.17
```

```
#> 3     1 -9.01
#> 4     2 -4.33
#> 5     2 -4.58
#> 6     2 -5.75
```

Then a t -test on these scores gives the same value as Hotelling's T ; it is accessed via the `statistic` component of `t.test()`

```
t <- t.test(LD1 ~ group, data=scores)$statistic
c(t, T2 = t^2)
#>      t    T2.t
#> -8.01 64.17
```

Finally, it is instructive to compare violin plots for the three measures, BM, WP and LD1. To do this with `ggplot2` requires reshaping the data from wide to long format so the plots can be faceted.

```
scores <- mathscore |>
  bind_cols(LD1 = scores[, "LD1"])

scores |>
  tidyrr::gather(key = "measure", value ="Score", BM:LD1) |>
  mutate(measure = factor(measure, levels = c("BM", "WP", "LD1"))) |>
  ggplot(aes(x = group, y = Score, color = group, fill = group)) +
  geom_violin(alpha = 0.2) +
  geom_jitter(width = .2, size = 2) +
  facet_wrap(~ measure, scales = "free", labeller = label_both) +
  scale_fill_manual(values = c("darkgreen", "blue")) +
  scale_color_manual(values = c("darkgreen", "blue")) +
  theme_bw(base_size = 14) +
  theme(legend.position = "none")
```

You can readily see how well the groups are separated on the discriminant axes, relative to the two individual variables.

9.5 More variables

The `mathscore` data gave a simple example with two outcomes to explain the essential ideas behind Hotelling's T^2 and multivariate tests. Multivariate methods become increasingly useful as the number of response variables increases because it is harder to show them all together and see how they relate to differences between groups.

A classic example is the dataset `banknote`, containing six size measures made on 100 genuine and 100 counterfeit old-Swiss 1000-franc bank notes (Flury & Riedwyl, 1988). The goal is to see how well the real and fake banknotes can be distinguished. The measures are the `Length` and `Diagonal` lengths of a banknote and the `Left`, `Right`, `Top` and `Bottom` edge margins in mm.

Before considering hypothesis tests, let's look at some exploratory graphics. Figure 9.7 shows univariate violin and boxplots of each of the measures. To make this plot, faceted by measure, I first reshape the data from wide to long and make `measure` a factor with levels in the order of the variables in the data set.

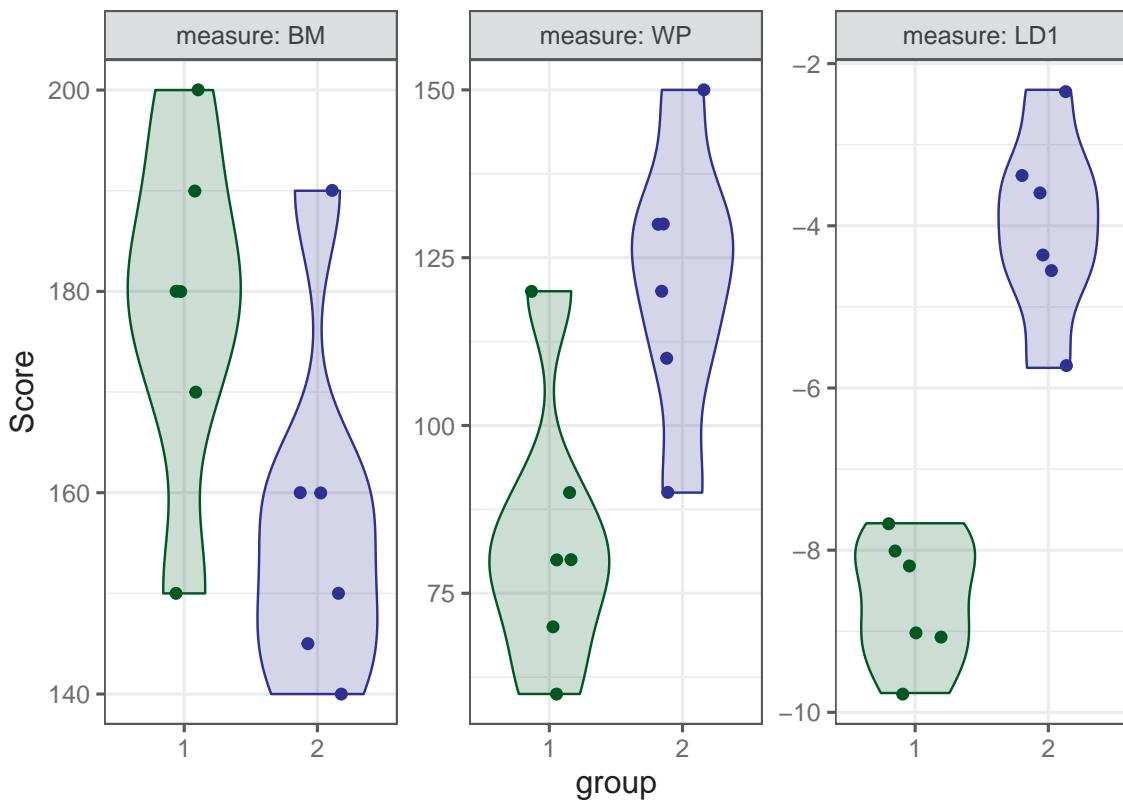


Figure 9.6: Violin plots comparing group 1 and 2 for the two observed measures and the linear discriminant score.

```
data(banknote, package= "mclust")
banknote |>
  tidyverse::gather(key = "measure",
                    value = "Size",
                    Length:Diagonal) |>
  mutate(measure = factor(measure,
                          levels = c(names(banknote)[-1]))) |>

  ggplot(aes(x = Status, y = Size, color = Status)) +
  geom_violin(aes(fill = Status), alpha = 0.2) +          # (1)
  geom_jitter(width = .2, size = 1.2) +                  # (2)
  geom_boxplot(width = 0.25,                             # (3)
                linewidth = 1.1,
                color = "black",
                alpha = 0.5) +
  labs(y = "Size (mm)") +
  facet_wrap(~ measure, scales = "free", labeller = label_both) +
  theme_bw(base_size = 14) +
  theme(legend.position = "top")
```

A quick glance at Figure 9.7 shows that the counterfeit and genuine bills differ in their means on most of the measures, with the counterfeit ones slightly larger on Left, Right, Bottom and Top margins. But univariate plots don't give an overall sense of how these variables are related to one another.

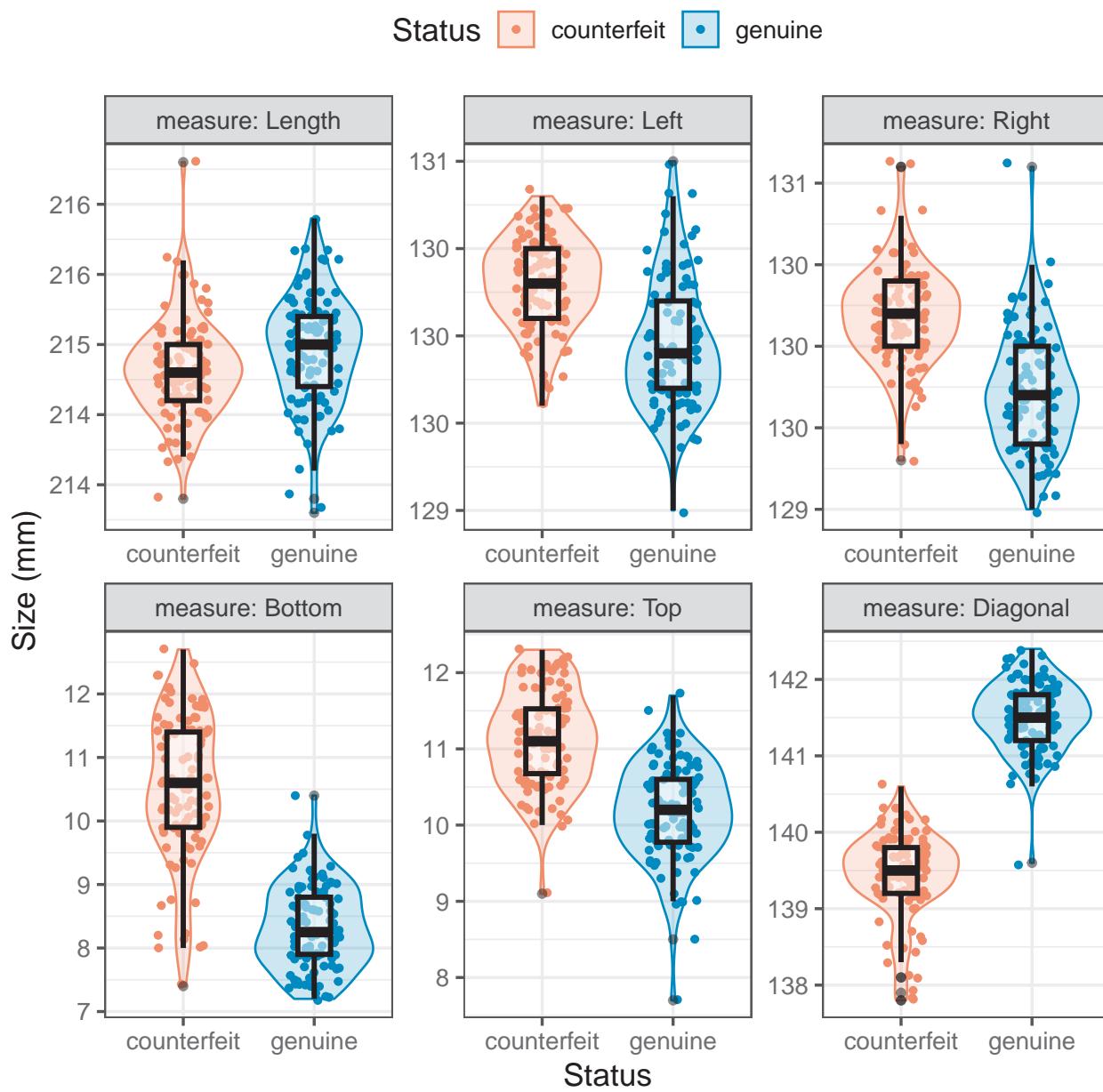


Figure 9.7: Overlaid violin and boxplots of the banknote variables. The violin plots give a sense of the shapes of the distributions, while the boxplots highlight the center and spread.

i Graph craft: Layers and transparency

Figure 9.7 is somewhat complex, so it is useful to understand the steps needed to make this figure show what I wanted. The plot in each panel contains three layers:

- (1) the violin plot based on a density estimate, showing the shape of each distribution;
- (2) the data points, but they are jittered horizontally using `geom_jitter()` because otherwise they would all overlap on the X axis;
- (3) the boxplot, showing the center (median) and spread (IQR) of each distribution.

In composing graphs with layers, order matters, and also does the `alpha` transparency, because each layer adds data ink on top of earlier ones. I plotted these in the order shown because I wanted the violin plot to provide the background, and the boxplot to show a simple univariate summary, not obscured by the other layers. The `alpha` values allow the data ink to be blended for each layer, and in this case, `alpha = 0.5` for the boxplot let the earlier layers show through.

9.5.1 Biplots

Multivariate relations among these six variables could be explored in data space using scatterplots or other methods, but I turn to my trusty multivariate juicer, a biplot, to give a 2D summary. Two dimensions account for 70% of the total variance of all the banknotes, while three would give 85%.

```
banknote.pca <- prcomp(banknote[, -1], scale = TRUE)
summary(banknote.pca)
#> Importance of components:
#>                               PC1    PC2    PC3    PC4    PC5    PC6
#> Standard deviation     1.716 1.131 0.932 0.671 0.5183 0.4346
#> Proportion of Variance 0.491 0.213 0.145 0.075 0.0448 0.0315
#> Cumulative Proportion  0.491 0.704 0.849 0.924 0.9685 1.0000
```

The biplot in Figure 9.8 gives a nicely coherent overview, at least in two dimensions. The first component shows the positive correlations among the measures of the margins, where the counterfeit bills are larger than the real ones and a negative correlation of the Diagonal with the other measures. The length of bills only distinguishes the types of banknotes on the second dimension.

```
banknote.pca <- ggbiplots::reflect(banknote.pca)
ggbiplots(banknote.pca,
          obs.scale = 1, var.scale = 1,
          groups = banknote>Status,
          ellipse = TRUE,
          ellipse.level = 0.5,
          ellipse.alpha = 0.1,
          ellipse.linewidth = 0,
          varname.size = 4,
          varname.color = "black") +
  labs(fill = "Status",
       color = "Status") +
  theme_minimal(base_size = 14) +
  theme(legend.position = 'top')
```

9.5.2 Testing mean differences

As noted above, Hotelling's T^2 is equivalent to a one-way MANOVA, fitting the size measures to the `Status` of the banknotes. `Anova()` reports only the F -statistic based on Pillai's trace criterion.

```
banknote.mlm <- lm(cbind(Length, Left, Right, Bottom, Top, Diagonal) ~ Status,
                     data = banknote)
Anova(banknote.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
```

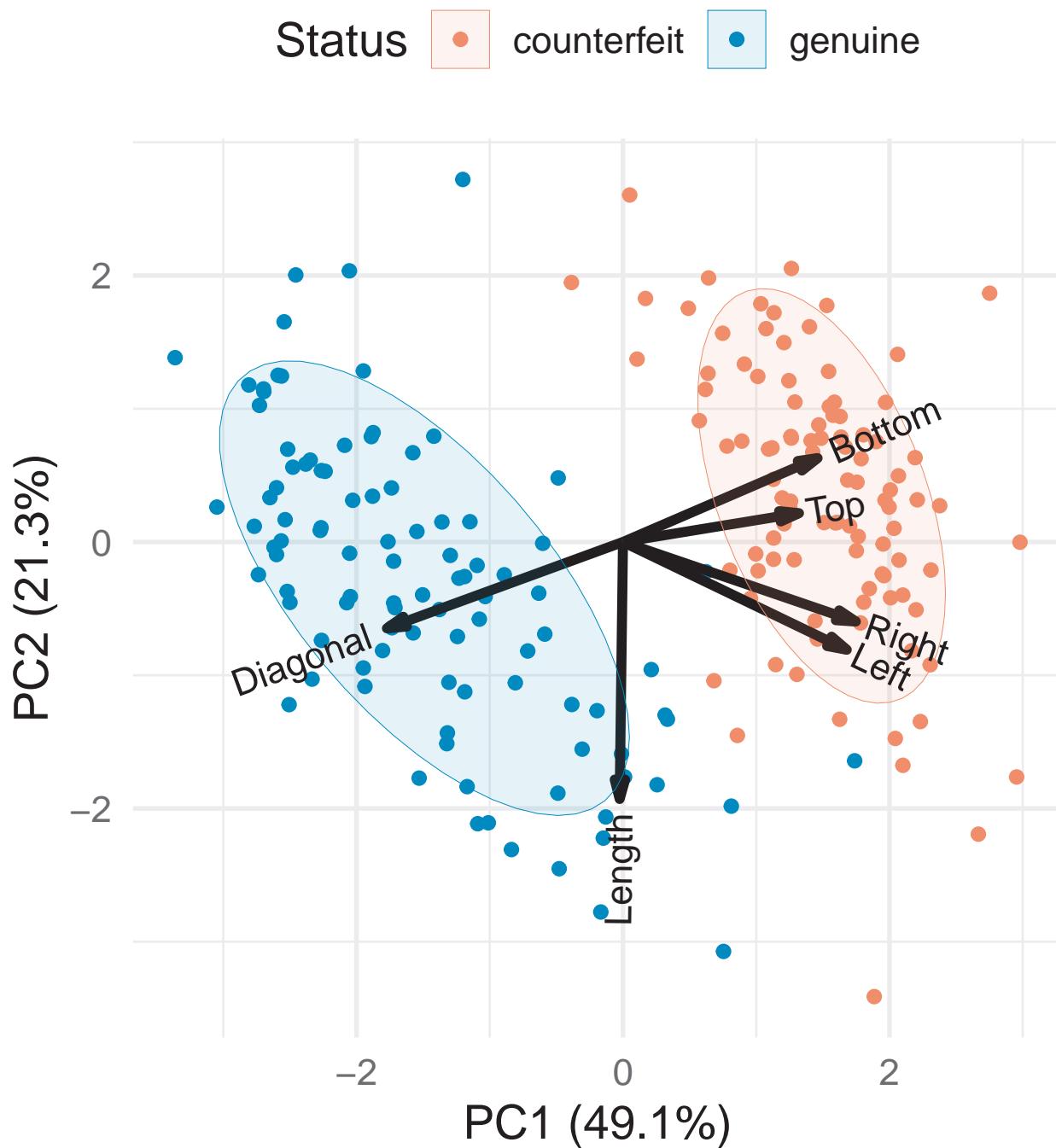


Figure 9.8: Biplot of the banknote variables, showing how the size measurements are related to each other. The points and data ellipses for the component scores are colored by Status, showing how the counterfeit and genuine bills are distinguished by these measures.

```
#>      Df test stat approx F num Df den Df Pr(>F)
#> Status  1    0.924      392     6    193 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You can see all the multivariate test statistics with the `summary()` method for "Anova.mlm" objects. With two groups, and hence a 1 df test, these all translate into identical F -statistics.

```
summary(Anova(banknote.mlm)) |> print(SSP = FALSE)
#>
#> Type II MANOVA Tests:
#>
#> -----
#>
#> Term: Status
#>
#> Multivariate Tests: Status
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1    0.92     392      6    193 <2e-16 ***
#> Wilks       1    0.08     392      6    193 <2e-16 ***
#> Hotelling-Lawley 1   12.18     392      6    193 <2e-16 ***
#> Roy         1   12.18     392      6    193 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If you wish, you can extract the univariate t -tests or equivalent $F = t^2$ statistics from the "mlm" object using `broom::tidy.mlm()`. What is given as the `estimate` is the difference in the mean for the genuine banknotes relative to the counterfeit ones.

```
broom::tidy(banknote.mlm) |>
  filter(term != "(Intercept)") |>
  dplyr::select(-term) |>
  rename(t = statistic) |>
  mutate(F = t^2) |>
  relocate(F, .after = t)
#> # A tibble: 6 x 6
#>   response estimate std.error     t     F p.value
#>   <chr>     <dbl>     <dbl> <dbl> <dbl>   <dbl>
#> 1 Length     0.146    0.0524   2.79   7.77 5.82e- 3
#> 2 Left       -0.357   0.0445  -8.03   64.5 8.50e-14
#> 3 Right      -0.473   0.0464 -10.2    104.  6.84e-20
#> 4 Bottom     -2.22    0.130   -17.1   292.  7.78e-41
#> 5 Top        -0.965   0.0909 -10.6    113.  3.85e-21
#> 6 Diagonal    2.07    0.0715  28.9    836.  5.35e-73
```

The individual $F_{(1,198)}$ statistics can be compared to the $F_{(6,193)} = 392$ value for the overall multivariate test. While all of the individual tests are highly significant, the average of the univariate F s is only 236. The multivariate test gains power by taking the correlations of the size measures into account.

9.6 Variance accounted for: Eta square (η^2)

In a univariate multiple regression model, the coefficient of determination $R^2 = \text{SS}_H/\text{SS}_{\text{Total}}$ gives the proportion of variance accounted for by hypothesized terms in H relative to the total variance. An analog for ANOVA-type models with categorical, group factors as predictors is η^2 (Pearson, 1903), defined as

$$\eta^2 = \frac{SS_{\text{Between groups}}}{SS_{\text{Total}}} .$$

For multivariate response models, the generalization of η^2 uses multivariate analogs of these sums of squares, \mathbf{Q}_H and $\mathbf{Q}_T = \mathbf{Q}_H + \mathbf{Q}_E$, and there are different calculations for a single measure corresponding to the various test statistics (Wilks' Λ , etc.), as described in Chapter 10.

Let's calculate the η^2 for the multivariate model `banknote.mlm` with `Status` as the only predictor, giving $\eta^2 = 0.92$, or 92% of the total variance.

```
heplots::etasq(banknote.mlm)
#>      eta^2
#> Status 0.924
```

This can be compared to the principal components analysis and the biplot in Figure 9.8, where two components (less favorably) accounted for 70% of total variance and it took four PCA dimensions to account for over 90%. The goals of PCA and MANOVA are different, of course, but they are both concerned with accounting for variance of multivariate data. We will meet another multivariate juicer, **canonical discriminant analysis** in Chapter 11.

9.7 What we've learned

TODO: Combine with 'summary/Ch09-summary.qmd

This chapter was designed to illustrate the main ideas for visualizing differences between means on multiple response variables in a two-group design. Hotelling's T^2 is the generalization of a simple univariate t -test and works by combining the responses into a weighted sum that has the maximum possible univariate t for all choices of weights.

Figure 9.9 summarizes what was shown in Section 9.3 and Section 9.4. The data ellipses for the two groups in the `mathscore` data summarize the information about means and within-group variances. In the HE plot, the difference between the means is itself summarized by the line through them, which represents the $\mathbf{H} = \mathbf{Q}_H$ matrix and within-group variation is represented by the "Error" ellipse which is the $\mathbf{E} = \mathbf{S}_p = \mathbf{Q}_E$ matrix.

As we will see later (Chapter 11), the \mathbf{H} ellipse is scaled so that it provides a visual test of significance: it projects somewhere outside the \mathbf{E} ellipse if and only if the means differ significantly. The direction of the line between the means is also the discriminant axis and scores on this axis are weighted sum of the responses that have the greatest possible mean difference.

9.8 Exercises

Exercise 9.1. The value of Hotelling's T^2 found by `hotelling.test()` is 64.17. The value of the equivalent F statistic found by `Anova()` is 28.9. Verify that Equation 9.2 gives this result.

Packages used here:

11 packages used here: broom, car, carData, corpcor, dplyr, ggbio, ggplot2, heplots, Hotelling, knitr, tidyverse

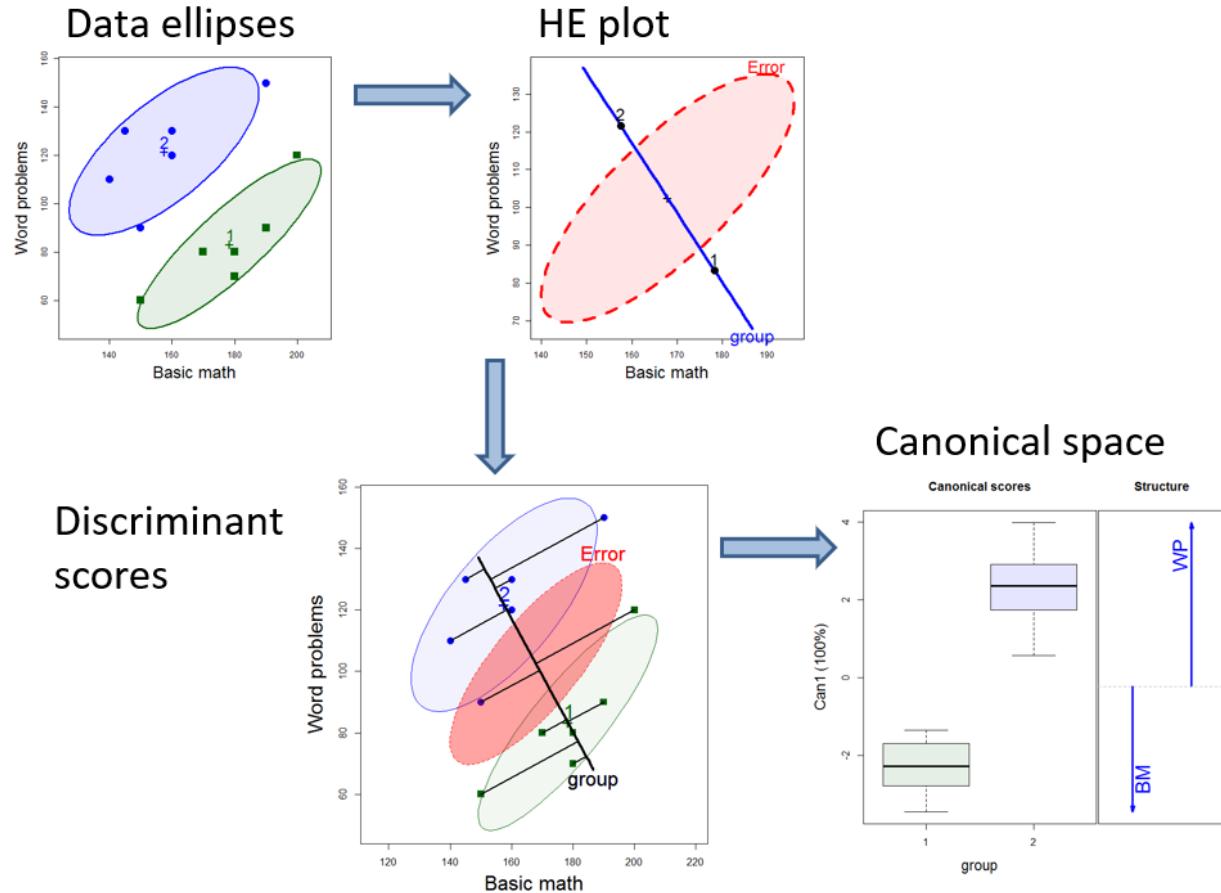


Figure 9.9: The Hypothesis Error plot framework for a two-group design. Above: Data ellipses can be summarized in an HE plot showing the pooled within-group error (\mathbf{E}) ellipse and the \mathbf{H} ‘ellipse’ for the group means. Below: Observations projected on the line joining the means give discriminant scores which correspond to a one-dimensional canonical space, represented by a boxplot of their scores and arrows reflecting the variable weights.

10

Multivariate Linear Models

Chapter 9 introduced the essential ideas of multivariate analysis in the context of a two-group design using Hotelling's T^2 . Through the magical power of multivariate thinking, I extend this here to a "Holy of Holies", inner sanctuary of the Tabernacle, the awed general **Multivariate Linear Model** (MLM).¹

This can be understood as a simple extension of the univariate linear model, with the main difference being that there are multiple response variables considered **together**, instead of just one, analysed alone. (Or, from my perspective, the univariate version is a restricted form of the MLM.) These outcomes might reflect several different *ways* or scales for measuring an underlying theoretical construct, or they might represent different *aspects* of some phenomenon that we hope to better understand when they are studied jointly.

For example, in the case of different measures, there are numerous psychological scales used to assess depression or anxiety and it may be important to include more than one measure to ensure that the construct has been measured adequately. It would add considerably to our understanding to know if the different outcome measures all had essentially the *same* relations to the predictor variables, or if they differ across measures.

In the second case of various aspects, student "aptitude" or "achievement" reflects competency in different various subjects (reading, math, history, science, ...) that are better studied together. We get a better understanding of the factors that influence each of aspects by testing them *jointly*.

Just as in univariate analysis there are variously named techniques (ANOVA, regression) that can be applied to several outcomes, depending on the structure of the predictors at hand. For instance, with one or more continuous predictors and multiple response variables, you could use **multivariate** multiple regression (MMRA) to obtain estimates useful for prediction.

Instead, if the predictors are categorical factors, multivariate analysis of variance (MANOVA) can be applied to test for differences between groups. Again, this is akin to ANOVA in the univariate context—the same underlying model is utilized, but the tests for terms in the model are multivariate ones for the collection of all response variables, rather than univariate ones for a single response.

The main goal of this chapter is to describe the details of the extension of univariate linear models to the case of multiple outcome measures. But the larger goal is to set the stage for the visualization methods using HE plots and low-D views discussed separately in Chapter 11. Some of the example datasets used here will re-appear there, and also in Chapter 12 which concerns some model-diagnostic graphical methods.

However, before considering the details and examples that apply separately to MANOVA and MMRA, it is useful to consider the general features of the multivariate linear model of which these cases are examples.

TODO: Offer defense against Huang (2019) and others here; cite Huberty & Morris (1989). Or, maybe not!

Packages

In this chapter I use the following packages. Load them now:

¹There's a bit of a puzzle here, and therefore a gap in methods and therefore an opportunity. The classical linear models fit by `lm()` extend naturally to non-gaussian data via `glm()` which provides for other families (binary: Bernoulli, count data: Poisson). The standard `lm()` extends quite naturally to a multivariate responses. Yet the combination of these ideas—non-gaussian multivariate models—remains elusive. The **VGAM** (Yee (2015), Yee (2025)) handles the bivariate cases of logistic (and probit) regression, but not much more. See Friendly & Meyer (2016), Sec. 10.4 for an example and graphs of odds ratios in these models. There is a lot more to do on this topic.

```

library(broom)
library(car)
library(dplyr)
library(ggplot2)
library(heplots)
library(patchwork)
library(tidyr)
library(matlib)
library(ggrepel)
library(mvinfluence)
library(MVN)
# set ggplot theme
#ggplot2::theme_set(theme_bw(base_size = 14))

```

10.1 Structure of the MLM

With p response variables, the multivariate linear model is most easily appreciated as the collection of p linear models, one for each response. We have p outcomes, so why not just consider a separate model for each?

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{X}\boldsymbol{\beta}_1 + \boldsymbol{\epsilon}_1 \\ \mathbf{y}_2 &= \mathbf{X}\boldsymbol{\beta}_2 + \boldsymbol{\epsilon}_2 \\ &\vdots \\ \mathbf{y}_p &= \mathbf{X}\boldsymbol{\beta}_p + \boldsymbol{\epsilon}_p \end{aligned}$$

{#eq-mlm-models}

But the problems with fitting separate univariate models are that:

- They don't give *simultaneous* tests for all regressions. The situation is similar to that in a one-way ANOVA, where an *overall* test for group differences is usually applied before testing individual comparisons to avoid problems of multiple testing: g groups gives $g \times (g - 1)/2$ pairwise tests.
- More importantly, fitting separate univariate models does not take *correlations* among the \mathbf{y} s into account.
 - It might be the case that the response variables are all essentially measuring the same thing, but perhaps weakly. If so, the multivariate approach can pool strength across the outcomes to detect their common relations to the predictors, giving greater power.
 - On the other hand, perhaps the responses are related in *different ways* to the predictors. A multivariate approach can help you understand *how many* different ways there are, and characterize each.

The model matrix \mathbf{X} in ?@eq-mlm-models is the same for all responses, but each one gets its own vector $\boldsymbol{\beta}_j$ of coefficients for how the predictors in \mathbf{X} fit a given response \mathbf{y}_j .

Among the beauties of multivariate thinking is that we can put these separate equations together in single equation by joining the responses \mathbf{y}_j as columns in a matrix \mathbf{Y} and similarly arranging the vectors of coefficients $\boldsymbol{\beta}_j$ as columns in a matrix \mathbf{B} .²

TODO Revise notation here, to be explicit about inclusion of $\boldsymbol{\beta}_0$

²A slight hiccup in notation is that the *uppercase* for the Greek Beta (β) is the same as the uppercase Roman \mathbf{B} , so I use $\mathbf{b}_1, \mathbf{b}_2, \dots$ below to refer to its' columns.

The MLM then becomes:

$$\underset{n \times p}{\mathbf{Y}} = \underset{n \times (q+1)}{\mathbf{X}} \underset{(q+1) \times p}{\mathbf{B}} + \underset{n \times p}{\boldsymbol{\varepsilon}}, \quad (10.1)$$

where:

- $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_p)$ is the matrix of n observations on p responses, with typical column \mathbf{y}_j ;
- \mathbf{X} is the model matrix with columns \mathbf{x}_i for q regressors, which typically includes an initial column \mathbf{x}_0 of 1s for the intercept;
- $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_p)$ is a matrix of regression coefficients, one column \mathbf{b}_j for each response variable;
- $\boldsymbol{\varepsilon}$ is a matrix of errors in predicting \mathbf{Y} .

Writing Equation 10.1 in terms of its elements, we have

$$\begin{matrix} \mathbf{Y} \\ \left[\begin{array}{cccc} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \end{array} \right] \end{matrix} = \begin{matrix} \mathbf{X} \\ \left[\begin{array}{cccc} 1 & x_{11} & \cdots & x_{1q} \\ 1 & x_{21} & \cdots & x_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nq} \end{array} \right] \end{matrix} \begin{matrix} \mathbf{B} \\ \left[\begin{array}{cccc} \beta_{01} & \beta_{02} & \cdots & \beta_{0p} \\ \beta_{11} & \beta_{12} & \cdots & \beta_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{q1} & \beta_{q2} & \cdots & \beta_{qp} \end{array} \right] \end{matrix} \\ + \begin{matrix} \boldsymbol{\varepsilon} \\ \left[\begin{array}{cccc} \epsilon_{11} & \epsilon_{12} & \cdots & \epsilon_{1p} \\ \epsilon_{21} & \epsilon_{22} & \cdots & \epsilon_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_{n1} & \epsilon_{n2} & \cdots & \epsilon_{np} \end{array} \right] \end{matrix}$$

The structure of the model matrix \mathbf{X} is exactly the same as the univariate linear model, and may therefore contain,

- **quantitative predictors**, such as `age`, `income`, years of `education`
- **transformed predictors** like $\sqrt{\text{age}}$ or $\log(\text{income})$
- **polynomial terms**: age^2 , age^3, \dots (using `poly(age, k)` in R)
- **categorical predictors** (“factors”), such as treatment (Control, Drug A, drug B), or sex; internally a factor with k levels is transformed to $k-1$ dummy (0, 1) variables, representing comparisons with a reference level, typically the first.
- **interaction terms**, involving either quantitative or categorical predictors, e.g., `age * sex`, `treatment * sex`.

10.1.1 Assumptions

Just as in univariate models, the assumptions of the multivariate linear model almost entirely concern the behavior of the errors (residuals). Let $\boldsymbol{\epsilon}'_i$ represent the i th row of $\boldsymbol{\varepsilon}$. Then it is assumed that:

- **Normality**: The residuals, $\boldsymbol{\epsilon}'_i$ are distributed as multivariate normal, $\mathcal{N}_p(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a non-singular error-covariance matrix.
 - Statistical tests of multivariate normality of the residuals include the Shapiro-Wilk ([Shapiro & Wilk, 1965](#)) and Mardia ([1970, 1974](#)) tests (and others, in the `MVN` package).
 - As in univariate models, the MLM is relatively robust, however this is often better assessed visually using a χ^2 QQ plot of Mahalanobis squared distance against their corresponding χ^2_p values for p degrees of freedom using `heplots::cplot()`.
- **Homoscedasticity**: The error-covariance matrix $\boldsymbol{\Sigma}$ is constant across all observations and grouping factors. Graphical methods to show if this assumption is met are illustrated in Chapter 12.

- **Independence:** ϵ'_i and ϵ'_j are independent for $i \neq j$, so knowing the data for case i gives no information about case j (as would be true if the data consisted of pairs of husbands and wives);
- The predictors, \mathbf{X} , are fixed and measured without error or at least they are independent of the errors, $\boldsymbol{\varepsilon}$.

These statements are simply the multivariate analogs of the assumptions of normality, constant variance and independence of the errors in univariate models. Note that it is unnecessary to assume that the predictors (regressors, columns of \mathbf{X}) are normally distributed.

Implicit in the above is perhaps the most important assumption—that the model has been *correctly specified*. This means:

- **Linearity:** The form of the relations between each \mathbf{y} and the \mathbf{x} s is correct. Typically this means that the relations are *linear*, but if not, we have specified a correct transformation of \mathbf{y} and/or \mathbf{x} .
 - **Completeness:** No relevant predictors have been omitted from the model. For example in the coffee, stress example (Section 7.1.1), omitting stress from the model biases the effect of coffee on heart disease.
 - **Additive effects:** The combined effect of different predictors is the sum of their individual effects.
-

10.2 Fitting the model

The least squares (and also maximum likelihood) solution for the coefficients \mathbf{B} is given by

$$\hat{\mathbf{B}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} .$$

This is precisely the same as fitting the separate responses $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_p$, and placing the estimated coefficients $\hat{\mathbf{b}}_i$ as columns in $\hat{\mathbf{B}}$

$$\hat{\mathbf{B}} = [\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \dots, \hat{\mathbf{b}}_p] .$$

In R, we fit the multivariate linear model with `lm()` simply by giving a collection of response variables `y1`, `y2`, ... on the left-hand side of the model formula, wrapped in `cbind()` which combines them to form a matrix response.

```
lm(cbind(y1, y2, y3) ~ x1 + x2 + ..., data=)
```

In the presence of possible outliers, robust methods are available for univariate linear models (e.g., `MASS::rlm()`). So too, `heplots::robmlm()` provides robust estimation in the multivariate case as illustrated in Section 13.5.

10.2.1 Example: Dog food data

As a toy example to make these ideas concrete, consider the dataset `dogfood`. Here, a dogfood manufacturer wanted to study preference for different dogfood formulas, two of their own (“Old”, “New”) and two from other manufacturers (“Major”, “Alps”).

In a between-dog design, each of $n = 4$ dogs were presented with a bowl of *one* formula and the time to `start` eating and `amount` eaten were recorded. Greater preference would be seen in a shorter delay to start eating and a greater amount, so these responses are expected to be negatively correlated.

```
data(dogfood, package = "heplots")
str(dogfood)
```

```
#> 'data.frame': 16 obs. of 3 variables:
#> $ formula: Factor w/ 4 levels "Old","New","Major",...: 1 1 1 1 2 2 2 2 3 3 ...
#> $ start : int 0 1 1 0 0 1 2 3 1 5 ...
#> $ amount : int 100 97 88 92 95 85 82 89 77 84 ...
```

For this data, boxplots for the two responses provide an initial look, shown in Figure 10.1. Putting these side-by-side makes it easy to see the inverse relation between the medians on the two response variables.

```
dog_long <- dogfood |>
  pivot_longer(c(start, amount),
               names_to = "variable")
ggplot(data = dog_long,
       aes(x=formula, y = value, fill = formula)) +
  geom_boxplot(alpha = 0.2) +
  geom_point(size = 2.5) +
  facet_wrap(~ variable, scales = "free") +
  theme_bw(base_size = 14) +
  theme(legend.position="none")
```

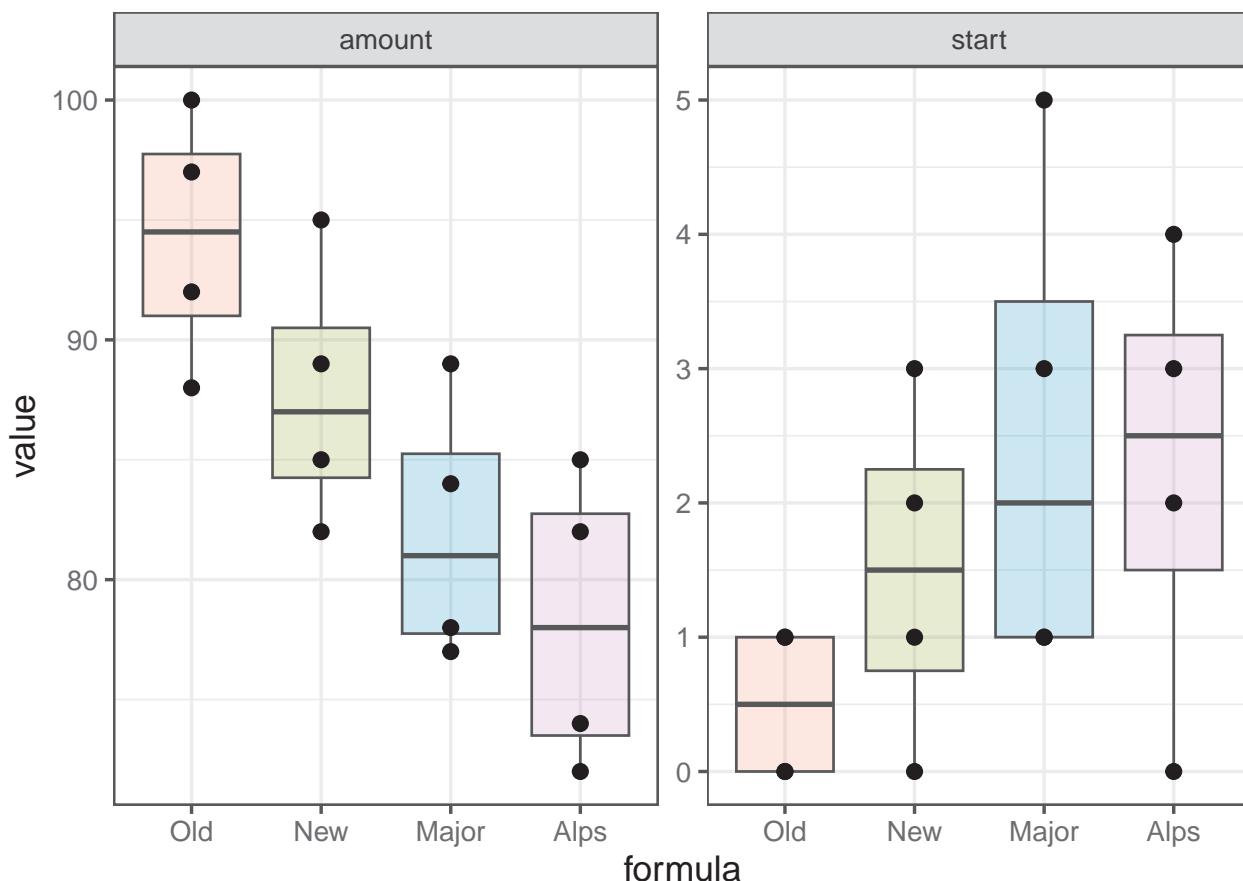


Figure 10.1: Boxplots for time to start eating and amount eaten by dogs given one of four dogfood formulas.

As suggested above, the multivariate model for testing mean differences due to the dogfood `formula` is fit using `lm()` on the matrix `Y` constructed with `cbind(start, amount)`.

```

dogfood.mod <- lm(cbind(start, amount) ~ formula,
                   data=dogfood) |>
  print()
#>
#> Call:
#> lm(formula = cbind(start, amount) ~ formula, data = dogfood)
#>
#> Coefficients:
#>             start     amount
#> (Intercept) 0.50    94.25
#> formulaNew   1.00   -6.50
#> formulaMajor 2.00  -12.25
#> formulaAlps  1.75  -16.00

```

By default, the factor `formula` is represented by three columns in the \mathbf{X} matrix that correspond to treatment contrasts, which are comparisons of the Old formula (a baseline level) with each of the others. The coefficients, for example `formulaNEW`, are the difference in means from those for Old.

Then, the overall multivariate test that means on both variables do not differ is carried out using `car::Anova()`.

```

dogfood.aov <- Anova(dogfood.mod) |>
  print()
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> formula  3    0.702      2.16     6     24   0.083 .
#> ---
#> Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The details of these analysis steps are explained below.

10.2.2 Sums of squares

In univariate response models, statistical tests and model summaries (like R^2) are based on the familiar decomposition of the total sum of squares SS_T into regression or hypothesis (SS_H) and error (SS_E) sums of squares. In the multivariate linear model each of these becomes a $p \times p$ matrix SSP containing sums of squares for the p responses on the diagonal and sums of cross products in the off-diagonal elements. For the MLM this is expressed as:

$$\begin{aligned}
SSP_T &= \mathbf{Y}^\top \mathbf{Y} - n \bar{\mathbf{y}} \bar{\mathbf{y}}^\top \\
&= (\hat{\mathbf{Y}}^\top \hat{\mathbf{Y}} - n \bar{\mathbf{y}} \bar{\mathbf{y}}^\top) + \hat{\varepsilon}^\top \hat{\varepsilon} \\
&= SSP_H + SSP_E \\
&\equiv \mathbf{H} + \mathbf{E} ,
\end{aligned}$$

{#eq-SSP}

where,

- $\bar{\mathbf{y}}$ is the $(p \times 1)$ vector of means for the response variables;
- $\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}}$ is the matrix of fitted values; and
- $\hat{\varepsilon} = \mathbf{Y} - \hat{\mathbf{Y}}$ is the matrix of residuals.

We can visualize this decomposition in the simple case of a two-group design (for the `mathscore` data in

Section 9.2) as shown in Figure 10.2. Let \mathbf{y}_{ij} be the vector of p responses for subject j in group i , $i = 1, \dots, g$ for $j = 1, \dots, n_i$. Then, using \cdot to represent a subscript averaged over, $\mathbf{?@eq-SSP}$ comes from the identity

$$\underbrace{(\mathbf{y}_{ij} - \bar{\mathbf{y}}_{..})}_{T} = \underbrace{(\bar{\mathbf{y}}_{i\cdot} - \bar{\mathbf{y}}_{..})}_{H} + \underbrace{(\mathbf{y}_{ij} - \bar{\mathbf{y}}_{i\cdot})}_{E} \quad (10.2)$$

where each side of Equation 10.2 is squared and summed over observations to give $\mathbf{?@eq-SSP}$. In Figure 10.2,

- The total variance \mathbf{SSP}_T reflects the deviations of the observations \mathbf{y}_{ij} from the grand mean $\bar{\mathbf{y}}_{..}$ and has the data ellipse shown in gray.
- In the middle \mathbf{SSP}_H panel, all the observations are represented at their group means, $\bar{\mathbf{y}}_{i\cdot}$, the fitted values. Their variance and covariance is then reflected by deviations of the group means (weighted for the number of observations per group) around the grand mean.
- The right \mathbf{SSP}_E panel then shows the residual variance, which is the variation of the observations \mathbf{y}_{ij} around their group means, $\bar{\mathbf{y}}_{i\cdot}$. Centering the two data ellipses at the centroid $\bar{\mathbf{y}}_{..}$ then gives the ellipse for the \mathbf{SSP}_E , also called the pooled within-group covariance matrix.

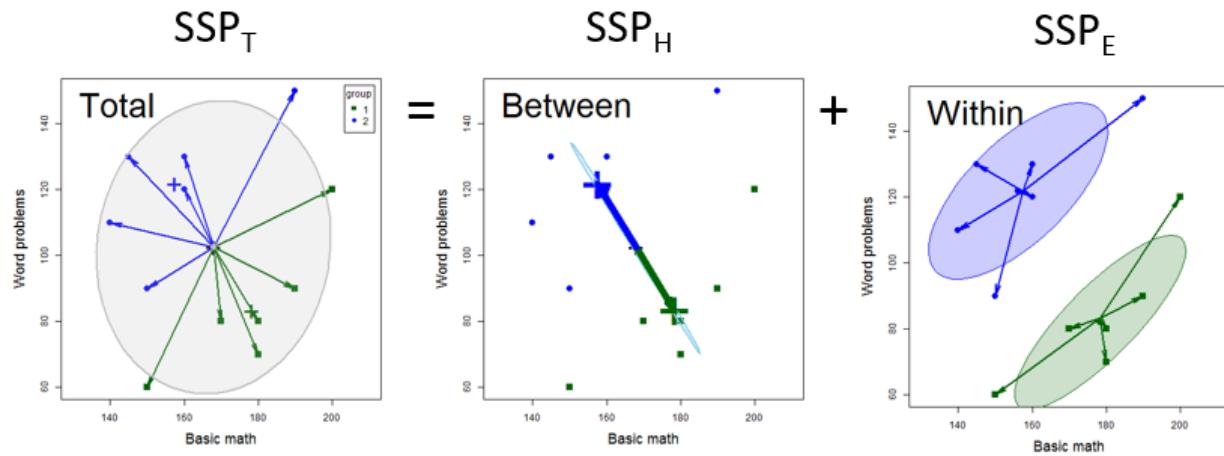


Figure 10.2: Breakdown of the total \mathbf{SSP}_T into sums of squares and products for between-group hypothesis variance (\mathbf{SSP}_H) and within-group, error variance (\mathbf{SSP}_E).

The formulas for these sum of squares and products matrices can be shown explicitly as follows, where the notation \mathbf{zz}^\top generates the $p \times p$ outer product of a vector \mathbf{z} , giving $z_k \times z_\ell$ for all pairs of elements.

$$\mathbf{SSP}_T = \sum_{i=1}^g \sum_{j=1}^{n_i} (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{..}) (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{..})^\top \quad (10.3)$$

$$\mathbf{SSP}_H = \sum_{i=1}^g \mathbf{n}_i (\bar{\mathbf{y}}_{i\cdot} - \bar{\mathbf{y}}_{..}) (\bar{\mathbf{y}}_{i\cdot} - \bar{\mathbf{y}}_{..})^\top \quad (10.4)$$

$$\mathbf{SSP}_E = \sum_{i=1}^g \sum_{j=1}^{n_i} (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{i\cdot}) (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{i\cdot})^\top \quad (10.5)$$

This is the decomposition that we visualize in HE plots, where the size and direction of \mathbf{H} and \mathbf{E} can be represented as ellipsoids.

But first, let's find these results for the example. The easy way is to get them from the result returned by `car:::Anova()`, where the hypothesis \mathbf{SSP}_H for each term in the model is returned as an element in a named list `SSP` and the error \mathbf{SSP}_E is returned as the matrix `SSPE`.

```

SSP_H <- dogfood.aov$SSP |> print()
#> $formula
#>      start amount
#> start    9.69  -70.9
#> amount -70.94  585.7

SSP_E <- dogfood.aov$SSPE |> print()
#>      start amount
#> start   25.8   11.8
#> amount   11.8   390.3

```

You can calculate these directly as shown below. `sweep()` is used to subtract the `colMeans()` from \mathbf{Y} and $\hat{\mathbf{Y}}$ and `crossprod()` premultiplies a matrix by its' transpose.

```

Y <- dogfood[, c("start", "amount")]
Ydev <- sweep(Y, 2, colMeans(Y)) |> as.matrix()
SSP_T <- crossprod(as.matrix(Ydev)) |> print()
#>      start amount
#> start   35.4  -59.2
#> amount -59.2  975.9

fitted <- fitted(dogfood.mod)
Yfit <- sweep(fitted, 2, colMeans(fitted)) |> as.matrix()
SSP_H <- crossprod(Yfit) |> print()
#>      start amount
#> start    9.69  -70.9
#> amount -70.94  585.7

residuals <- residuals(dogfood.mod)
SSP_E <- crossprod(residuals) |> print()
#>      start amount
#> start   25.8   11.8
#> amount   11.8   390.3

```

The decomposition of the total sum of squares and products in ?@eq-**SSP** can be shown as:

$$\begin{pmatrix} \text{SSP}_T \\ 35.4 & -59.2 \\ -59.2 & 975.9 \end{pmatrix} = \begin{pmatrix} \text{SSP}_H \\ 9.69 & -70.94 \\ -70.94 & 585.69 \end{pmatrix} + \begin{pmatrix} \text{SSP}_E \\ 25.8 & 11.8 \\ 11.8 & 390.3 \end{pmatrix}$$

These numbers are the variances in the diagonal and covariance between start and amount in the off-diagonal, where the sign is important: In SSP_H the negative covariance reflects the fact that for these brands larger time to start eating is negatively related to the amount eaten. In SSP_E the covariance is slightly positive. This might reflect a mild hunger factor: for a given brand, dogs who lunge for their bowls sooner also eat more. But, they're all good boys, right?

10.2.3 How big is SS_H compared to SS_E ?

In a univariate response model, SS_H and SS_E are both scalar numbers and the univariate F test statistic,

$$F = \frac{SS_H/\text{df}_h}{SS_E/\text{df}_e} = \frac{\text{Var}(H)}{\text{Var}(E)} , \quad (10.6)$$

assesses “how big” SS_H is, relative to SS_E , the variance accounted for by a hypothesized model or model

terms relative to error variance. The measure $R^2 = SS_H/(SS_H + SS_E) = SS_H/SS_T$ gives the proportion of total variance accounted for by the model terms.

In the multivariate analog \mathbf{H} and \mathbf{E} are both $p \times p$ matrices, and \mathbf{H} “divided by” \mathbf{E} becomes \mathbf{HE}^{-1} . The answer, “how big” SS_H is compared to SS_E is expressed in terms of the p eigenvalues $\lambda_i, i = 1, 2, \dots, p$ of \mathbf{HE}^{-1} . These are the p values λ which solve the determinant equation

$$\det(\mathbf{HE}^{-1} - \lambda \mathbf{I}) = 0 .$$

The solution also gives the λ_i as the eigenvalues, with vectors \mathbf{v}_i as the corresponding eigenvectors,

$$\mathbf{HE}^{-1} \lambda_i = \lambda_i \mathbf{v}_i . \quad (10.7)$$

This can also be expressed in terms of the size of \mathbf{H} relative to total variation ($\mathbf{H} + \mathbf{E}$) as

$$\mathbf{H}(\mathbf{H} + \mathbf{E})^{-1} \rho_i = \rho_i \mathbf{v}_i , \quad (10.8)$$

which has the same eigenvectors as Equation 10.7 and the eigenvalues are $\rho_i = \lambda_i/(1 + \lambda_i)$.

However, when the hypothesized model terms have df_h degrees of freedom (columns of the \mathbf{X} matrix for that term), \mathbf{H} is of rank df_h , so only $s = \min(p, df_h)$ eigenvalues can be non-zero. For example, a test for a hypothesis about a single quantitative predictor \mathbf{x} , has $df_h = 1$ degree of freedom and $\text{rank}(\mathbf{H}) = 1$; for a factor with g groups, $df_h = \text{rank}(\mathbf{H}) = g - 1$.

For the `dogfood` data, we get the following results:

```
HEinv <- SSP_H %*% solve(SSP_E) |> print()
#>           start amount
#> start    0.466 -0.196
#> amount   -3.488  1.606
eig <- eigen(HEinv)
eig$values
#> [1] 2.0396 0.0317

# as proportions
eig$values / sum(eig$values)
#> [1] 0.9847 0.0153
```

The factor `formula` has four levels and therefore $df_h = 3$ degrees of freedom. But there are only $p = 2$ responses, so there are $s = \min(p, df_h) = 2$ eigenvalues (and corresponding eigenvectors). The eigenvalues tell us that 98.5% of the hypothesis variance due to `formula` can be accounted for by a single dimension.

The overall multivariate test for the model in Equation 10.1 is essentially a test of the hypothesis $\mathcal{H}_0 : \mathbf{B} = 0$ (excluding the row for the intercept). Equivalently, this is a test based on the *incremental* \mathbf{SSP}_H for the hypothesized terms in the model—that is, the difference between the \mathbf{SSP}_H for the full model and the null, intercept-only model. The same idea can be applied to test the difference between any pair of *nested* models—the added contribution of terms in a larger model relative to a smaller model containing a subset of terms.

The eigenvectors \mathbf{v}_i in Equation 10.7 are also important. These are the weights for the variables in a linear combination $v_{i1}\mathbf{y}_1 + v_{i2}\mathbf{y}_2 + \dots + v_{ip}\mathbf{y}_p$ which produces the largest univariate F statistic for the i -th dimension. We exploit this in canonical discriminant analysis and the corresponding canonical HE plots (Section 11.7).

The eigenvectors of \mathbf{HE}^{-1} for the `dogfood` model are shown below:

Table 10.1: Test statistics for multivariate tests combine the size of dimensions of $\mathbf{H}\mathbf{E}^{-1}$ into a single measure.

Criterion	Formula	Partial η^2
Wilks's Λ	$\Lambda = \prod_i^s \frac{1}{1+\lambda_i}$	$\eta^2 = 1 - \Lambda^{1/s}$
Pillai trace	$V = \sum_i^s \frac{\lambda_i}{1+\lambda_i}$	$\eta^2 = \frac{V}{s}$
Hotelling-Lawley trace	$H = \sum_i^s \lambda_i$	$\eta^2 = \frac{H}{H+s}$
Roy maximum root	$R = \lambda_1$	$\eta^2 = \frac{\lambda_1}{1+\lambda_1}$

```
rownames(eig$vectors) <- rownames(HEinv)
colnames(eig$vectors) <- paste("Dim", 1:2)
eig$vectors
#>           Dim 1  Dim 2
#> start    0.123 -0.411
#> amount   -0.992 -0.911
```

The first column corresponds to the weighted sum $0.12 \times \text{start} - 0.99 \times \text{amount}$, which as we saw above accounts for 95.5% of the differences in the group means.

10.3 Multivariate test statistics

In the univariate case, the overall F -test of $\mathcal{H}_0 : \boldsymbol{\beta} = \mathbf{0}$ is the uniformly most powerful invariant test when the assumptions are met. There is nothing better. This is not the case in the MLM.

The reason is that when there are $p > 1$ response variables, and we are testing a hypothesis comprising $\text{df}_h > 1$ coefficients or degrees of freedom, there are $s > 1$ possible dimensions in which \mathbf{H} can be large relative to \mathbf{E} , each measured by the eigenvalue λ_i . There are several test statistics that combine these into a single measure, shown in Table 10.1.

These correspond to different kinds of “means” of the λ_i : geometric (Wilks), arithmetic (Pillai), harmonic (Hotelling-Lawley) and supremum (Roy). See Friendly et al. (2013) for the geometry behind these measures.

Each of these statistics have different sampling distributions under the null hypothesis. But conveniently they can all be converted to F statistics. These conversions are exact when the hypothesis has $s \leq 2$ degrees of freedom, and approximations otherwise.

As well, each has an analog of the R^2 -like partial η^2 measure, giving the partial association accounted for by each term in the MLM. These reflect the proportion of total variation attributable to a given model term, partialling out (excluding) other factors from the total non-error variation. They can be used as a measure of effect size in a MLM.

10.3.1 Testing contrasts and linear hypotheses

Even more generally, these multivariate tests apply to *every* linear hypothesis concerning the coefficients in \mathbf{B} . Suppose we want to test the hypothesis that a subset of rows (predictors) and/or columns (responses) simultaneously have null effects. This can be expressed in the general linear test,

$$\mathcal{H}_0 : \mathbf{C}_{h \times q} \mathbf{B}_{q \times p} = \mathbf{0}_{h \times p},$$

where \mathbf{C} is a full rank $h \leq q$ hypothesis matrix of constants, that selects subsets or linear combinations (contrasts) of the coefficients in \mathbf{B} to be tested in a h degree-of-freedom hypothesis.

In this case, the SSP matrix for the hypothesis has the form

$$\mathbf{H} = (\mathbf{C}\widehat{\mathbf{B}})^T [\mathbf{C}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{C}^T]^{-1} (\mathbf{C}\widehat{\mathbf{B}}) , \quad (10.9)$$

where there are $s = \min(h, p)$ non-zero eigenvalues of $\mathbf{H}\mathbf{E}^{-1}$. In Equation 10.9, \mathbf{H} measures the (Mahalanobis) squared distances (and cross products) among the linear combinations $\mathbf{C}\widehat{\mathbf{B}}$ from the origin under the null hypothesis.

For example, with three responses y_1, y_2, y_3 and three predictors x_1, x_2, x_3 , we can test the hypothesis that neither x_1 nor x_2 contribute at all to predicting the y ’s in terms of the hypothesis that the coefficients for the corresponding rows of \mathbf{B} are zero using a 1-row \mathbf{C} matrix that simply selects those rows:

$$\begin{aligned} \mathcal{H}_0 : \mathbf{CB} &= [0 \ 1 \ 1 \ 0] \begin{pmatrix} \beta_{0,y_1} & \beta_{0,y_2} & \beta_{0,y_3} \\ \beta_{1,y_1} & \beta_{1,y_2} & \beta_{1,y_3} \\ \beta_{2,y_1} & \beta_{2,y_2} & \beta_{2,y_3} \\ \beta_{3,y_1} & \beta_{3,y_2} & \beta_{3,y_3} \end{pmatrix} \\ &= \begin{bmatrix} \beta_{1,y_1} & \beta_{1,y_2} & \beta_{1,y_3} \\ \beta_{2,y_1} & \beta_{2,y_2} & \beta_{2,y_3} \end{bmatrix} = \mathbf{0}_{(2 \times 3)} \end{aligned}$$

In MANOVA designs, it is often desirable to follow up a significant effect for a factor with subsequent tests to determine which groups differ. While you can simply test for all pairwise differences among groups (using Bonferroni or other corrections for multiplicity), a more substantively-driven approach uses planned comparisons or *contrasts* among the factor levels as described in Section 5.1.3.

For a factor with g groups, a contrast is simply a comparison of the mean of one subset of groups against the mean of another subset. This is specified as a weighted sum, L of the means with weights \mathbf{c} that sum to zero,

$$L = \mathbf{c}^T \boldsymbol{\mu} = \sum_i c_i \mu_i \quad \text{such that} \quad \sum c_i = 0$$

Two contrasts, \mathbf{c}_1 and \mathbf{c}_2 are *orthogonal* if the sum of products of their weights is zero, i.e., $\mathbf{c}_1^T \mathbf{c}_2 = \sum c_{1i} c_{2i} = 0$. When contrasts are placed as columns of a matrix \mathbf{C} , they are all *mutually orthogonal* if each pair is orthogonal, which means $\mathbf{C}^T \mathbf{C}$ is a diagonal matrix. Orthogonal contrasts correspond to statistically independent tests. This is nice because they reflect separate, non-overlapping research questions. When these questions are posed *a priori*, in advance of analysis there is no need to correct for multiple testing, on the grounds that you shouldn’t be penalized for having more ideas!

For example, with the $g = 4$ groups for the `dogfood` data, the company might want to test the following comparisons among the formulas Old, New, Major and Alps: (a) Ours vs. Theirs: The average of (Old, New) compared to (Major, Alps); (b) Old vs. New; (c) Major vs. Alps. The contrasts that do this are:

$$\begin{aligned} L_1 &= \frac{1}{2}(\mu_O + \mu_N) - \frac{1}{2}(\mu_M + \mu_A) & \rightarrow \mathbf{c}_1 &= \frac{1}{2}(1 \ 1 \ -1 \ -1) \\ L_2 &= \mu_O - \mu_N & \rightarrow \mathbf{c}_2 &= (1 \ -1 \ 0 \ 0) \\ L_3 &= \mu_M - \mu_A & \rightarrow \mathbf{c}_3 &= (0 \ 0 \ 1 \ -1) \end{aligned}$$

Note that these correspond to *nested dichotomies* among the four groups: first we compare groups (Old and New) against groups (Major and Alps), then subsequently within each of these sets. Nested dicontomy contrasts are always *orthogonal*, and therefore correspond to statistically independent tests. We are effectively taking a three degree-of-freedom question, “do the means differ?” and breaking it down into three separate 1 df tests that answer specific parts of that overall question.

In R, contrasts for a factor are specified as columns of matrix, each of which sums to zero. For this example, we can set this up by creating each as a vector and joining them as columns using `cbind()`:

```

c1 <- c(1, 1, -1, -1)/2    # Old,New vs. Major,Alps
c2 <- c(1, -1, 0, 0)       # Old vs. New
c3 <- c(0, 0, 1, -1)       # Major vs. Alps
C <- cbind(c1,c2,c3)
rownames(C) <- levels(dogfood$formula)

C
#>      c1 c2 c3
#> Old   0.5 1 0
#> New   0.5 -1 0
#> Major -0.5 0 1
#> Alps  -0.5 0 -1

# show they are mutually orthogonal
t(C) %*% C
#>      c1 c2 c3
#> c1  1 0 0
#> c2  0 2 0
#> c3  0 0 2

```

For the `dogfood` data, with `formula` as the group factor, you can set up the analyses to use these contrasts by assigning the matrix `C` to `contrasts()` for that factor in the dataset itself. When the contrasts are changed, it is necessary to refit the model. The estimated coefficients then become the estimated mean differences for the contrasts.

```

contrasts(dogfood$formula) <- C
dogfood.mod <- lm(cbind(start, amount) ~ formula,
                   data=dogfood)
coef(dogfood.mod)
#>           start amount
#> (Intercept) 1.688 85.56
#> formulac1   -1.375 10.88
#> formulac2   -0.500  3.25
#> formulac3    0.125  1.88

```

For example, Ours vs. Theirs estimated by `formulac1` takes 0.69 less time to start eating and eats 5.44 more on average.

For multivariate tests, when all contrasts are pairwise orthogonal, the overall test of a factor with $df_h = g - 1$ degrees of freedom can be broken down into $g - 1$ separate 1 df tests. This gives rise to a set of df_h rank 1 \mathbf{H} matrices that additively decompose the overall hypothesis SSCP matrix,

$$\mathbf{H} = \mathbf{H}_1 + \mathbf{H}_2 + \cdots + \mathbf{H}_{df_h}, \quad (10.10)$$

exactly as the univariate SS_H can be decomposed using orthogonal contrasts in an ANOVA (Section 5.1.3)

You can test such contrasts or any other hypotheses involving linear combinations of the coefficients using `car::linearHypothesis()`. Here, "`formulac1`" refers to the contrast `c1` for the difference between Ours and Theirs. Note that because this is a 1 df test, all four test statistics yield the same F values.

```

hyp <- rownames(coef(dogfood.mod))[-1] |> print()
#> [1] "formulac1" "formulac2" "formulac3"
H1 <- linearHypothesis(dogfood.mod, hyp[1],

```

```

      title="Ours vs. Theirs") |>
print()
#>
#> Sum of squares and products for the hypothesis:
#>          start amount
#> start     7.56 -59.8
#> amount -59.81 473.1
#>
#> Sum of squares and products for error:
#>          start amount
#> start    25.8   11.7
#> amount   11.7   390.3
#>
#> Multivariate Tests: Ours vs. Theirs
#>              Df test stat approx F num Df den Df Pr(>F)
#> Pillai        1   0.625    9.18      2     11 0.0045 ***
#> Wilks         1   0.375    9.18      2     11 0.0045 ***
#> Hotelling-Lawley 1   1.669    9.18      2     11 0.0045 ***
#> Roy           1   1.669    9.18      2     11 0.0045 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Similarly we can test the other two contrasts within these each of these two subsets, but I don't print the results

```

H2 <- linearHypothesis(dogfood.mod, hyp[2],
                        title="Old vs. New")
H3 <- linearHypothesis(dogfood.mod, hyp[3],
                        title="Alps vs. Major")

```

Then, we can illustrate Equation 10.10 by extracting the 1 df **H** matrices (SSPH) from the results of `linearHypothesis`.

$$\begin{pmatrix} 9.7 & \mathbf{H} \\ -70.9 & -70.9 \\ -70.9 & 585.7 \end{pmatrix} = \begin{pmatrix} 7.6 & \mathbf{H}_1 \\ -59.8 & -59.8 \\ -59.8 & 473.1 \end{pmatrix} + \begin{pmatrix} 0.13 & \mathbf{H}_2 \\ 1.88 & 1.88 \\ 1.88 & 28.12 \end{pmatrix} + \begin{pmatrix} 2 & \mathbf{H}_3 \\ -13 & -13 \\ -13 & 84 \end{pmatrix}$$

10.4 ANOVA → MANOVA

Multivariate analysis of variance (MANOVA) generalizes the familiar ANOVA model to situations where there are two or more response variables. Unlike ANOVA, which focuses on discerning statistical differences in one continuous dependent variable influenced by an independent variable (or grouping variable), MANOVA considers several dependent variables at once. It integrates these variables into a single, composite variable through a weighted linear combination, allowing for a comprehensive analysis of how these dependent variables collectively vary with respect to the levels of the independent variable. Essentially, MANOVA investigates whether the grouping variable explains significant variations in the combined dependent variables.

The situation is illustrated in Figure 10.3 where there are two response measures, Y_1 and Y_2 with data collected for three groups. For concreteness, Y_1 might be a score on a math test and Y_2 might be a reading score. Let's also say that group 1 has been studying Shakespeare, while group 2 has concentrated on physics, but group 3 has done nothing beyond the normal curriculum.

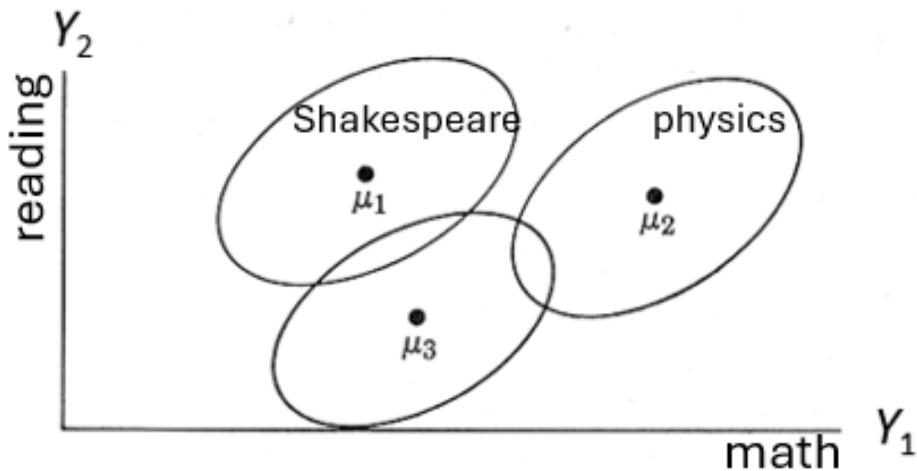


Figure 10.3: Data from simple MANOVA design involving three groups and two response measures, Y_1 and Y_2 , summarized by their data ellipses.

As shown in the figure, the centroids, (μ_{1g}, μ_{2g}) , clearly differ—the data ellipses barely overlap. A multivariate analysis would show a highly difference among groups. From a rough visual inspection, it seems that means differ on the math test Y_1 , with the physics group out-performing the other two. On the reading test Y_2 however it might turn out that the three group means don't differ significantly in an ANOVA, but the Shakespeare and physics groups appear to outperform the normal curriculum group. Doing separate ANOVAs on these variables would miss what is so obvious from Figure 10.3: there is wide separation among the groups in the two tests considered *jointly*.

Figure 10.4 illustrates a second important advantage of performing a multivariate analysis over separate ANOVAS: that of determining the *number of dimensions* or aspects along which groups differ. In the panel on the left, the means of the three groups increase nearly linearly on the combination of Y_1 and Y_2 , so their differences can be ascribed to a single dimension, which simplifies the interpretation: *both* memory and attention scores decrease together with degree of schizophrenia.

For example, the groups here might be patients diagnosed as normal, mild schizophrenia and profound schizophrenia, and the measures could be tests of memory and attention. The obvious multivariate interpretation from the figure is that of increasing impairment of cognitive functioning across the groups, comprised by memory and attention. Note also the positive association *within* each group: those who perform better on the memory task also do better on attention.

In contrast, the right panel of Figure 10.4 shows a situation where the group means have a low correlation. Data like this might arise in a study of parental competency, where there are measures of both the degree of caring (Y_1) and time spent in play (Y_2) by fathers and groups consisting of fathers of children with no disability, or a physical disability or a mental ability.

As can be seen in Figure 10.4 fathers of the disabled children differ from those of the not disabled group in two different directions corresponding to being higher on either Y_1 or Y_2 . The red arrows suggest that the differences among groups could be interpreted in terms of two uncorrelated dimensions, perhaps labeled overall competency and emphasis on physical activity. (The pattern in Figure 10.4 (right) is contrived for the sake of illustration; it does not reflect the data analyzed in the example below.)

10.4.1 Example: Father parenting data

I use a simple example of a three-group multivariate design to illustrate the basic ideas of fitting MLMs in R and testing hypotheses. Visualization methods using HE plots are discussed in Chapter 11.

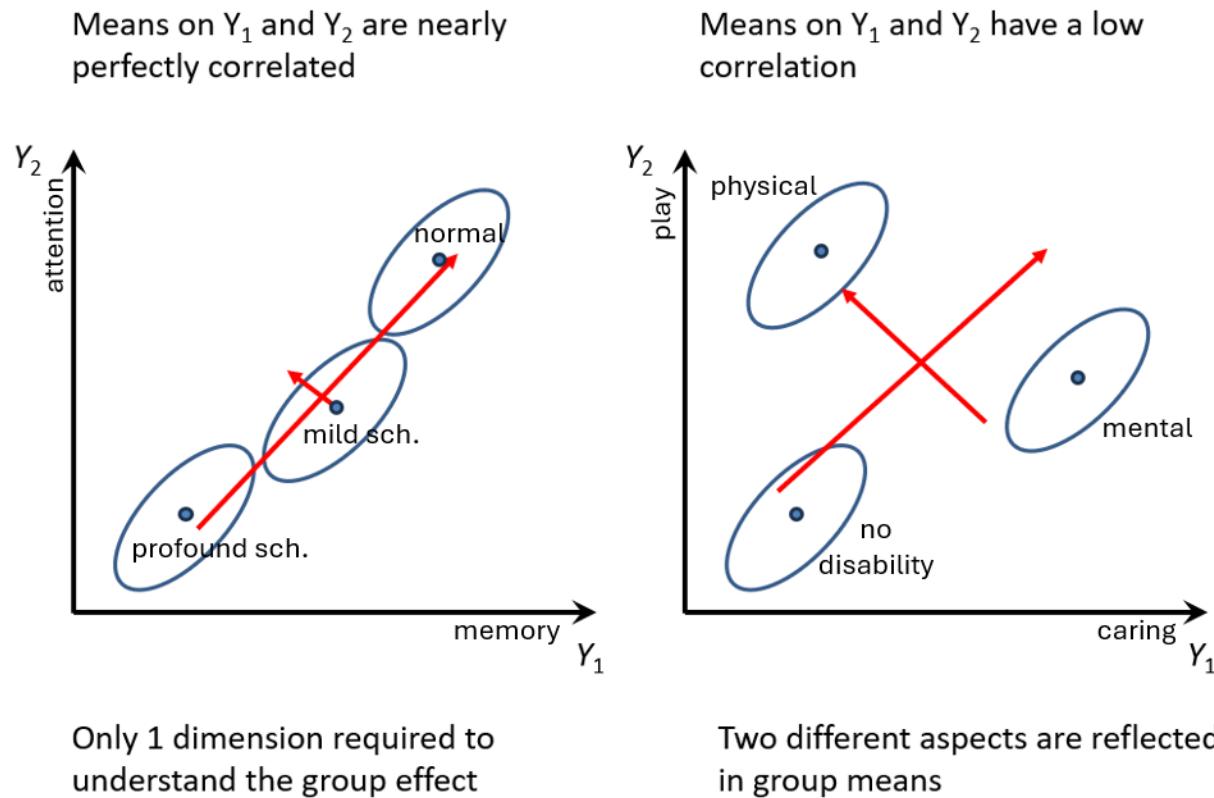


Figure 10.4: A simple MANOVA design involving three groups and two response measures, Y_1 and Y_2 , but with different patterns of the differences among the group means. The red arrows suggest interpretations in terms of dimensions or aspects of the response variables.

The dataset `Parenting` come from an exercise (10B) in Meyers et al. (2006) and are probably contrived, but are modeled on a real study in which fathers were assessed on three subscales of a *Perceived Parenting Competence Scale*,

- `caring`, caretaking responsibilities;
- `emotion`, emotional support provided to the child; and
- `play`, recreational time spent with the child.

The dataset `Parenting` comprises 60 fathers selected from three groups of $n = 20$ each: (a) fathers of a child with *no disabilities* ("Normal"); (b) fathers with a *physically* disabled child; (c) fathers with a *mentally* disabled child. The design is thus a three-group MANOVA, with three response variables.

The main questions concern whether the group means differ on these scales, and the nature of these differences. That is, do the means differ significantly on all three measures? Is there a consistent order of groups across these three aspects of parenting?

More specific questions are: (a) Do the fathers of typical children differ from the other two groups on average? (b) Do the physical and mental groups differ? These questions can be tested using contrasts, and are specified by assigning a matrix to `contrasts(Parenting$group)`; each column is a contrast whose values sum to zero. They are given labels "group1" (normal vs. other) and "group2" (physical vs. mental) in some output.

```
data(Parenting, package="heplots")
C <- matrix(c(1, -.5, -.5,
              0,  1, -1),
```

```

nrow = 3, ncol = 2) |> print()
#>      [,1] [,2]
#> [1,]  1.0   0
#> [2,] -0.5   1
#> [3,] -0.5  -1
contrasts(Parenting$group) <- C

```

Exploratory plots

Before setting up a model and testing, it is well-advised to examine the data graphically. The simplest plots are side-by-side boxplots (or violin plots) for the three responses. With `ggplot2`, this is easily done by reshaping the data to long format and using faceting. In Figure 10.5, I've also plotted the group means with white dots.

```

parenting_long <- Parenting |>
  tidyr::pivot_longer(cols=caring:play,
                      names_to = "variable")

ggplot(parenting_long,
       aes(x=group, y=value, fill=group)) +
  geom_boxplot(outlier.size=2.5,
               alpha=.5,
               outlier.alpha = 0.9) +
  stat_summary(fun=mean,
               color="white",
               geom="point",
               size=2) +
  scale_fill_hue(direction = -1) +      # reverse default colors
  labs(y = "Scale value", x = "Group") +
  facet_wrap(~ variable) +
  theme_bw(base_size = 14) +
  theme(legend.position="top") +
  theme(axis.text.x = element_text(angle = 15,
                                    hjust = 1))

```

In this figure, differences among the groups on `play` are most apparent, with fathers of non-disabled children scoring highest. Differences among the groups on `emotion` are very small, but one high outlier for the fathers of mentally disabled children is apparent. On `caring`, fathers of children with a physical disability stand out as highest.

For exploratory purposes, you might also make a scatterplot matrix. Here, because the MLM assumes homogeneity of the variances and covariance matrices \mathbf{S}_i , I show only the data ellipses in scatterplot matrix format, using `heplots:covEllipses()` (with 50% coverage, for clarity):

```

colors <- scales::hue_pal()(3) |> rev() # match color use in ggplot
covEllipses(cbind(caring, play, emotion) ~ group,
            data=Parenting,
            variables = 1:3,
            fill = TRUE, fill.alpha = 0.2,
            pooled = FALSE,
            level = 0.50,
            col = colors)

```

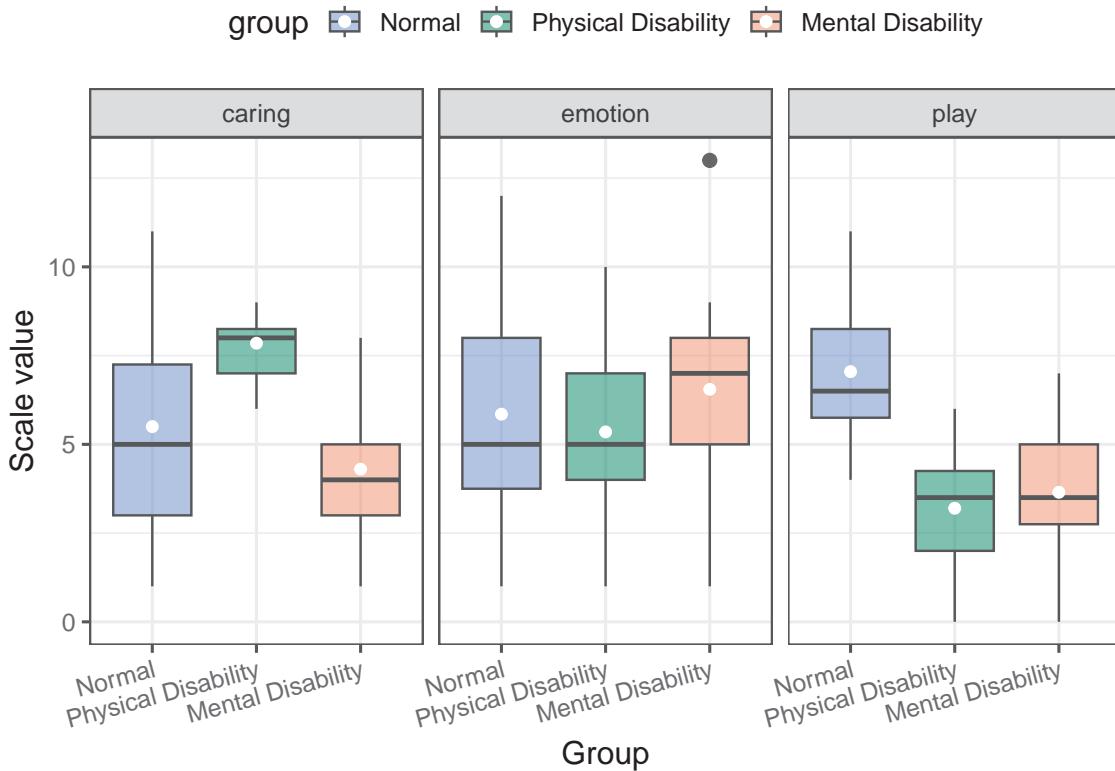


Figure 10.5: Faceted boxplots of scores on the three parenting scales, showing also the mean for each.

If the covariance matrices were all the same, the data ellipses would have roughly the same size and orientation, but that is not the case in Figure 10.6. The normal group shows greater variability overall and the correlations among the measures differ somewhat from group to group. We'll assess later whether this makes a difference in the conclusions that can be drawn (Chapter 12). The group centroids also differ, but the pattern is not particularly clear. We'll see an easier to understand view in HE plots and their canonical discriminant cousins.

10.4.1.1 Testing the model

Let's proceed to fit the multivariate model predicting all three scales from the `group` factor. `lm()` for a multivariate response returns an object of class "`m1m`", for which there are many methods (use `methods(class="m1m")` to find them).

```
parenting.m1m <- lm(cbind(caring, play, emotion) ~ group,
                      data=Parenting) |> print()
#>
#> Call:
#> lm(formula = cbind(caring, play, emotion) ~ group, data = Parenting)
#>
#> Coefficients:
#>            caring     play     emotion
#> (Intercept) 5.8833  4.6333  5.9167
#> group1      -0.3833  2.4167 -0.0667
#> group2       1.7750 -0.2250 -0.6000
```

The coefficients in this model are the values of the contrasts set up above. `group1` is the mean of the typical

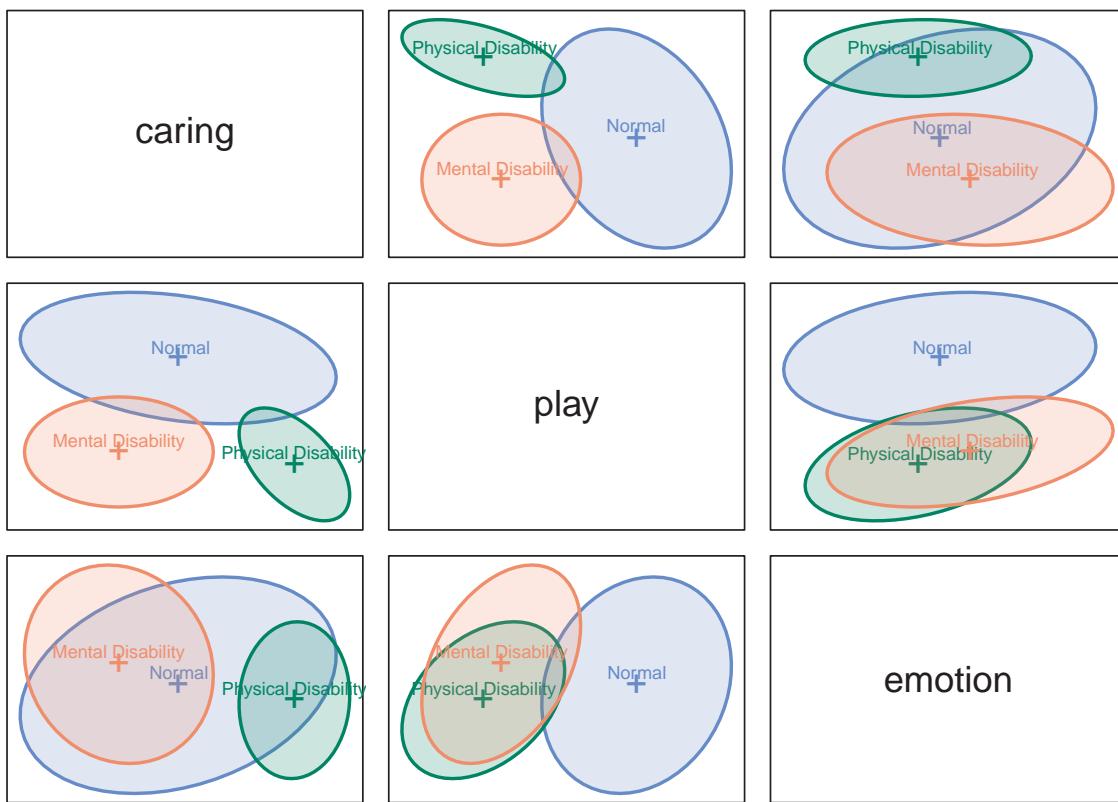


Figure 10.6: Bivariate data ellipses for pairs of the three responses, showing the means, correlations and variances for the three groups.

group minus the average of the other two, which is negative on `caring` and `emotion` but positive for `play`. `group2` is the difference in means for the physical vs. mental groups.

Before doing multivariate tests, it is useful to see what would happen if we ran univariate ANOVAs on each of the responses. These can be extracted from an MLM using `stats::summary.aov()` and they give tests of the model terms for each response variable separately:

```
summary.aov(parenting.mlm)
#> Response caring :
#>              Df Sum Sq Mean Sq F value Pr(>F)
#> group          2    130    65.2    18.6  6e-07 ***
#> Residuals      57    200     3.5
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Response play :
#>              Df Sum Sq Mean Sq F value Pr(>F)
#> group          2    177    88.6    27.6  4e-09 ***
#> Residuals      57    183     3.2
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Response emotion :
```

```
#>           Df Sum Sq Mean Sq F value Pr(>F)
#> group        2    15   7.27    1.02   0.37
#> Residuals    57   408   7.16
```

For a more condensed summary, you can instead extract the univariate model fit statistics from the "`m1m`" object using the `heplots::glance()` method for a multivariate model object. The code below selects just the R^2 and F -statistic for the overall model for each response, together with the associated p -value.

```
glance(parenting.m1m) |>
  select(response, r.squared, fstatistic, p.value)
#> # A tibble: 3 x 4
#>   response r.squared fstatistic      p.value
#>   <chr>     <dbl>       <dbl>       <dbl>
#> 1 caring     0.395      18.6  0.000000602
#> 2 play       0.492      27.6  0.00000000405
#> 3 emotion    0.0344     1.02  0.369
```

From this, one might conclude that there are differences only in `caring` and `play` and therefore ignore `emotion`, but this would be short-sighted. `car::Anova()`, shown below, gives the overall multivariate test $\mathcal{H}_0 : \mathbf{B} = 0$ of the `group` effect, that the groups don't differ on *any* of the response variables. Note that this has a much smaller p -value than any of the univariate F tests.

```
Anova(parenting.m1m)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> group     2    0.948      16.8      6    112  9e-14 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`Anova()` returns an object of class "`Anova.m1m`" which has various methods. The `summary()` method for this gives more details, including all four test statistics. With $p = 3$ responses, these tests have $s = \min(p, df_h) = \min(3, 2) = 2$ dimensions and the F approximations are *not* equivalent here. All four tests are highly significant, with Roy's test giving the largest F statistic.

```
parenting.summary <- Anova(parenting.m1m) |>  summary()
print(parenting.summary, SSP=FALSE)
#>
#> Type II MANOVA Tests:
#>
#> -----
#>
#> Term: group
#>
#> Multivariate Tests: group
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai        2    0.948      16.8      6    112 9.0e-14 ***
#> Wilks         2    0.274      16.7      6    110 1.3e-13 ***
#> Hotelling-Lawley 2    1.840      16.6      6    108 1.8e-13 ***
#> Roy          2    1.108      20.7      3     56 3.8e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `summary()` method by default prints the $\text{SSH} = \mathbf{H}$ and $\text{SSE} = \mathbf{E}$ matrices, but I suppressed them above. The data structure returned contains nested elements which can be extracted more easily from the object using `purrr::pluck()`:

```
H <- parenting.summary |>
  purrr::pluck("multivariate.tests", "group", "SSPH") |>
  print()
#>      caring play emotion
#> caring  130.4 -43.767 -41.833
#> play    -43.8 177.233  0.567
#> emotion -41.8   0.567 14.533
E <- parenting.summary |>
  purrr::pluck("multivariate.tests", "group", "SSPE") |>
  print()
#>      caring play emotion
#> caring  199.8 -45.8    35.2
#> play    -45.8 182.7   80.6
#> emotion 35.2   80.6 408.0
```

Linear hypotheses & contrasts

With three or more groups or with a more complex MANOVA design, contrasts provide a way of testing questions of substantive interest regarding differences among group means.

The test of the contrast comparing the typical group to the average of the others is the test using the contrast $c_1 = (1, -\frac{1}{2}, -\frac{1}{2})$ which produces the coefficients labeled "group1". The function `car::linearHypothesis()` carries out the multivariate test that this difference is zero. This is a 1 df test, so all four test statistics produce the same F and p -values.

```
coef(parenting.mlm)[["group1",]]
#>      caring play emotion
#> -0.3833 2.4167 -0.0667
linearHypothesis(parenting.mlm, "group1") |>
  print(SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai        1    0.521    19.9     3    55 7.1e-09 ***
#> Wilks         1    0.479    19.9     3    55 7.1e-09 ***
#> Hotelling-Lawley 1    1.088    19.9     3    55 7.1e-09 ***
#> Roy          1    1.088    19.9     3    55 7.1e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Similarly, the difference between the physical and mental groups uses the contrast $c_2 = (0, 1, -1)$ and the test that these means are equal is given by `linearHypothesis()` applied to `group2`.

```
coef(parenting.mlm)[["group2",]]
#>      caring play emotion
#>     1.775 -0.225 -0.600
linearHypothesis(parenting.mlm, "group2") |>
  print(SSP=FALSE)
#>
```

```
#> Multivariate Tests:
#>
#> Df test stat approx F num Df den Df Pr(>F)
#> Pillai 1 0.429 13.8 3 55 8e-07 ***
#> Wilks 1 0.571 13.8 3 55 8e-07 ***
#> Hotelling-Lawley 1 0.752 13.8 3 55 8e-07 ***
#> Roy 1 0.752 13.8 3 55 8e-07 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`linearHypothesis()` is very general. The second argument (`hypothesis.matrix`) corresponds to **C**, and can be specified as numeric matrix giving the linear combinations of coefficients by rows to be tested, or a character vector giving the hypothesis in symbolic form; "`group1`" is equivalent to "`group1 = 0`".

Because the two contrasts used here are orthogonal, they add together to give the overall test of **B = 0**, which implies that the means of the three groups are all equal. The test below gives the same results as `Anova(parenting.mlm)`.

```
linearHypothesis(parenting.mlm, c("group1", "group2")) |>
  print(SSP=FALSE)
#>
#> Multivariate Tests:
#>
#> Df test stat approx F num Df den Df Pr(>F)
#> Pillai 2 0.948 16.8 6 112 9.0e-14 ***
#> Wilks 2 0.274 16.7 6 110 1.3e-13 ***
#> Hotelling-Lawley 2 1.840 16.6 6 108 1.8e-13 ***
#> Roy 2 1.108 20.7 3 56 3.8e-09 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

10.4.2 Ordered factors

When groups are defined by an ordered factor, such as level of physical fitness (rated 1–5) or grade in school, it is tempting to treat that as a numeric variable and use a multivariate regression model. This would assume that the effect of that factor is linear and if not, we might consider adding polynomial terms. A different strategy, often preferable, is to make the group variable an *ordered factor*, for which R assigns *polynomial contrasts*. This gives separate tests of the linear, quadratic, cubic, ... trends of the response, without the need to specify them separately in the model.

10.4.3 Example: Adolescent mental health

The dataset `AddHealth` contains a large cross-sectional sample of participants from grades 7–12 from the National Longitudinal Study of Adolescent Health, described by Warne (2014). It contains responses to two Likert-scale (1–5) items, `anxiety` and `depression`. `grade` is an *ordered factor*, which means that the default contrasts are taken as orthogonal polynomials with linear (`grade.L`), quadratic (`grade.Q`), up to 5th degree (`grade^5`) trends, which decompose the total effect of grade.

```
data(AddHealth, package="heplots")
str(AddHealth)
#> 'data.frame': 4344 obs. of 3 variables:
#> $ grade : Ord.factor w/ 6 levels "7"<"8"<"9"<"10"<...: 5 4 6 1 2 2 2 3 3 3 ...
```

```
#> $ depression: int 0 0 0 0 0 0 0 1 2 ...
#> $ anxiety    : int 0 0 0 1 1 0 0 1 1 0 ...
```

The research questions are:

- How do the means for anxiety and depression vary separately with grade? Is there evidence for linear and nonlinear trends?
- How do anxiety and depression vary *jointly* with grade?
- How does the *association* (correlation, R^2) of anxiety and depression vary with age?

The first question can be answered by fitting separate linear models for each response (e.g., `lm(anxiety ~ grade)`). However the second question is more interesting because it considers the two responses together and takes their correlation into account. This would be fit as the MLM:

$$\mathbf{y} = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 x + \boldsymbol{\beta}_2 x^2 + \cdots \boldsymbol{\beta}_5 x^5 \quad (10.11)$$

or, expressed in terms of the variables,

$$\begin{bmatrix} y_{\text{anx}} \\ y_{\text{dep}} \end{bmatrix} = \begin{bmatrix} \beta_{0,\text{anx}} \\ \beta_{0,\text{dep}} \end{bmatrix} + \begin{bmatrix} \beta_{1,\text{anx}} \\ \beta_{1,\text{dep}} \end{bmatrix} \text{grade} + \begin{bmatrix} \beta_{2,\text{anx}} \\ \beta_{2,\text{dep}} \end{bmatrix} \text{grade}^2 + \cdots + \begin{bmatrix} \beta_{5,\text{anx}} \\ \beta_{5,\text{dep}} \end{bmatrix} \text{grade}^5$$

{#eq-AH-mod2}

With `grade` represented as an ordered factor, the values of x in Equation 10.11 are those of the orthogonal polynomials given by `poly(grade, 5)`.

Exploratory plots

Some exploratory analysis is useful before fitting and visualizing models. As a first step, we find the means, standard deviations, and standard errors of the means.

```
means <- AddHealth |>
  group_by(grade) |>
  summarise(
    n = n(),
    dep_sd = sd(depression, na.rm = TRUE),
    anx_sd = sd(anxiety, na.rm = TRUE),
    dep_se = dep_sd / sqrt(n),
    anx_se = anx_sd / sqrt(n),
    depression = mean(depression),
    anxiety = mean(anxiety) ) |>
  relocate(depression, anxiety, .after = grade) |>
  print()
#> # A tibble: 6 x 8
#>   grade depression anxiety     n dep_sd anx_sd dep_se anx_se
#>   <ord>      <dbl>   <dbl> <int>  <dbl>  <dbl>  <dbl>  <dbl>
#> 1 7          0.881   0.751   622   1.11   1.05  0.0447  0.0420
#> 2 8          1.08    0.804   664   1.19   1.06  0.0461  0.0411
#> 3 9          1.17    0.934   778   1.19   1.08  0.0426  0.0387
#> 4 10         1.27    0.956   817   1.23   1.11  0.0431  0.0388
#> 5 11         1.37    1.12    790   1.20   1.16  0.0428  0.0411
#> 6 12         1.34    1.10    673   1.14   1.11  0.0439  0.0426
```

Now, plot the means with ± 1 error bars. It appears that average level of both depression and anxiety increase steadily with grade, except for grades 11 and 12 which don't differ much. Alternatively, we could describe this as relationships that seem largely linear, with a hint of curvature at the upper end.

```
p1 <- ggplot(data = means, aes(x = grade, y = anxiety)) +
  geom_point(size = 4) +
  geom_line(aes(group = 1), linewidth = 1.2) +
  geom_errorbar(aes(ymin = anxiety - anx_se,
                     ymax = anxiety + anx_se),
                width = .2)

p2 <- ggplot(data = means, aes(x = grade, y = depression)) +
  geom_point(size = 4) +
  geom_line(aes(group = 1), linewidth = 1.2) +
  geom_errorbar(aes(ymin = depression - dep_se,
                     ymax = depression + dep_se),
                width = .2)

p1 + p2
```

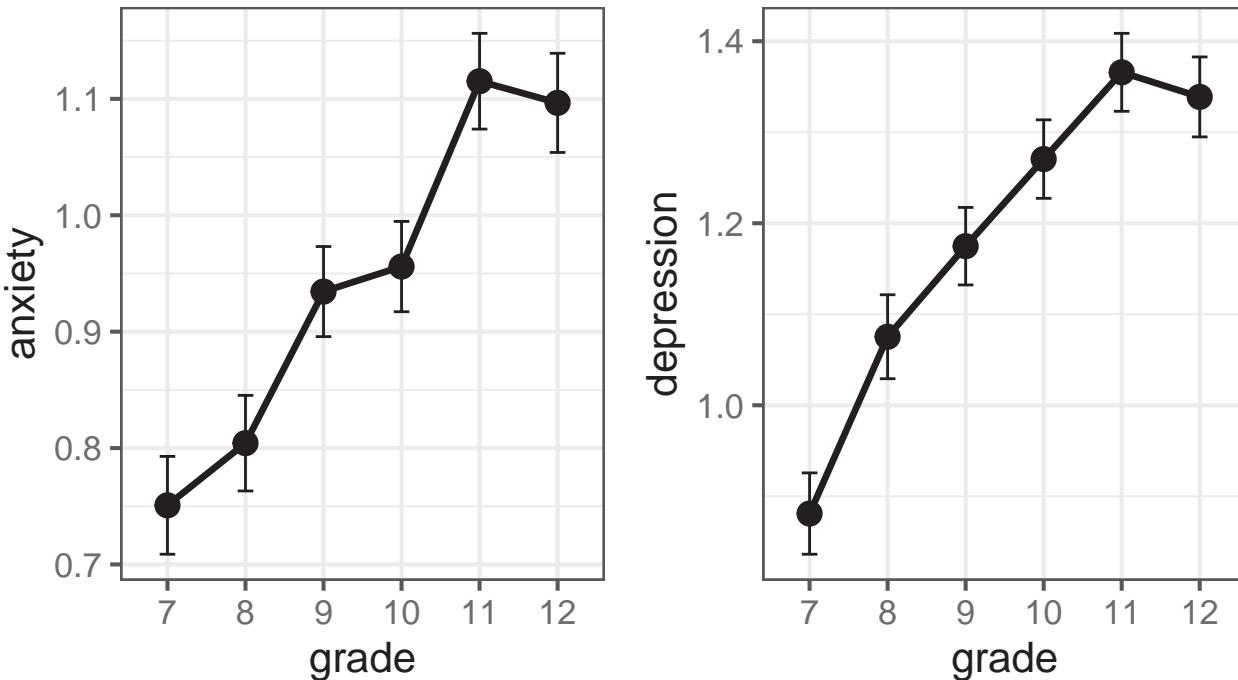


Figure 10.7: Means of anxiety and depression by grade, with ± 1 standard error bars.

It is also useful to within-group correlations using `covEllipses()`, as shown in Figure 10.8. This also plots the bivariate means showing the form of the association , treating anxiety and depression as multivariate outcomes. (Because the variability of the scores within groups is so large compared to the range of the means, I show the data ellipses with coverage of only 10%.)

```
covEllipses(AddHealth[, 3:2], group = AddHealth$grade,
            pooled = FALSE, level = 0.1,
```

```
center.cex = 2.5, cex = 1.5, cex.lab = 1.5,
fill = TRUE, fill.alpha = 0.05)
```

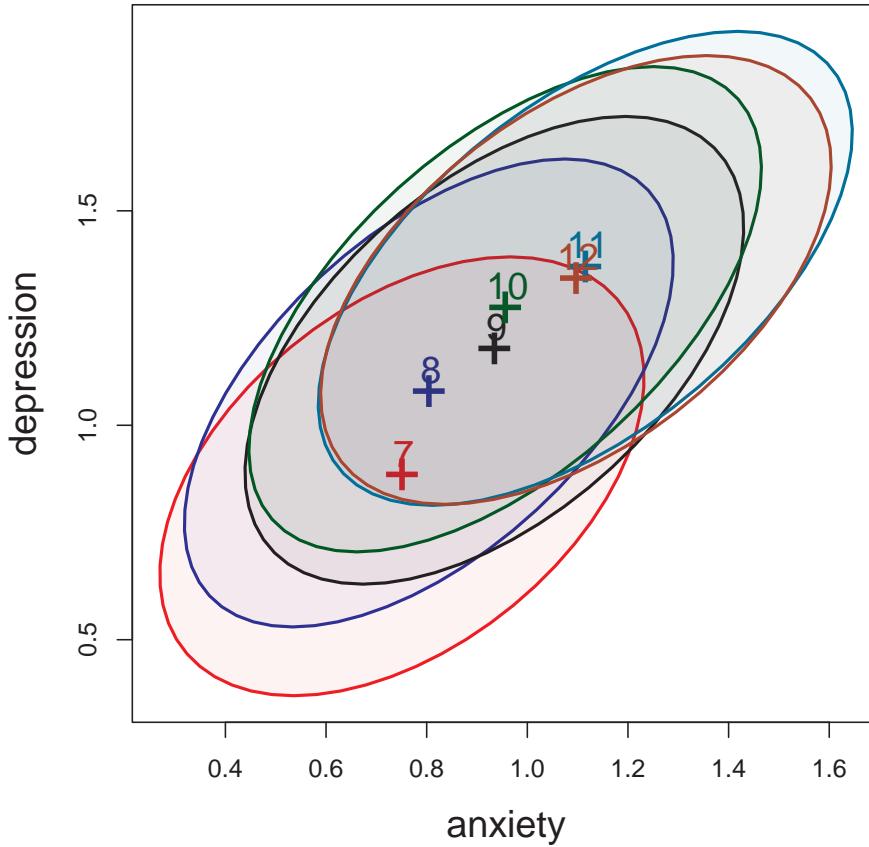


Figure 10.8: Within-group covariance ellipses for the `grade` groups.

Fit the MLM

Now, let's fit the MLM for both responses jointly in relation to `grade`. The null hypothesis is that the means for anxiety and depression are the same at all six grades,

$$\mathcal{H}_0 : \mu_7 = \mu_8 = \dots = \mu_{12} ,$$

or equivalently, that all coefficients except the intercept in the model Equation 10.11 are zero,

$$\mathcal{H}_0 : \beta_1 = \beta_2 = \dots = \beta_5 = \mathbf{0} .$$

We fit the MANOVA model, and test the grade effect using `car::Anova()`. The effect of `grade` is highly significant, as we could tell from Figure 10.7.

```
AH.mlm <- lm(cbind(anxiety, depression) ~ grade, data = AddHealth)

# overall test of `grade`
Anova(AH.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
```

```
#>      Df test stat approx F num Df den Df Pr(>F)
#> grade  5     0.0224    9.83     10    8676 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

However, the overall test, with 5 degrees of freedom is diffuse, in that it can be rejected if any pair of means differ. Given that `grade` is an ordered factor, it makes sense to examine narrower hypotheses of linear and nonlinear trends, `car::linearHypothesis()` on the coefficients of model `AH.mlm`.

```
coef(AH.mlm) |> rownames()
#> [1] "(Intercept)" "grade.L"      "grade.Q"      "grade.C"
#> [5] "grade^4"      "grade^5"
```

The joint test of the linear coefficients $\beta_1 = (\beta_{1,\text{anx}}, \beta_{1,\text{dep}})^T$ for anxiety and depression, $\mathcal{H}_0 : \beta_1 = \mathbf{0}$ is highly significant,

```
## linear effect
linearHypothesis(AH.mlm, "grade.L") |> print(SSP = FALSE)
#>
#> Multivariate Tests:
#>      Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1     0.019    42.5      2    4337 <2e-16 ***
#> Wilks       1     0.981    42.5      2    4337 <2e-16 ***
#> Hotelling-Lawley 1     0.020    42.5      2    4337 <2e-16 ***
#> Roy         1     0.020    42.5      2    4337 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test of the quadratic coefficients $\mathcal{H}_0 : \beta_2 = \mathbf{0}$ indicates significant curvature in trends across grade, as we saw in the plots of their means in Figure 10.7. One interpretation might be that depression and anxiety after increasing steadily up to grade eleven could level off thereafter.

```
## quadratic effect
linearHypothesis(AH.mlm, "grade.Q") |> print(SSP = FALSE)
#>
#> Multivariate Tests:
#>      Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1     0.002    4.24      2    4337  0.014 *
#> Wilks       1     0.998    4.24      2    4337  0.014 *
#> Hotelling-Lawley 1     0.002    4.24      2    4337  0.014 *
#> Roy         1     0.002    4.24      2    4337  0.014 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

An advantage of linear hypotheses is that we can test several terms *jointly*. Of interest here is the hypothesis that all higher order terms beyond the quadratic are zero, $\mathcal{H}_0 : \beta_3 = \beta_4 = \beta_5 = \mathbf{0}$. Using `linearHypothesis` you can supply a vector of coefficient names to be tested for their joint effect when dropped from the model.

```
coefs <- rownames(coef(AH.mlm)) |> print()
#> [1] "(Intercept)" "grade.L"      "grade.Q"      "grade.C"
#> [5] "grade^4"      "grade^5"
```

```

## joint test of all higher terms
linearHypothesis(AH.mlm, coefs[3:5],
                  title = "Higher-order terms") |>
  print(SSP = FALSE)
#>
#> Multivariate Tests: Higher-order terms
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      3   0.002    1.70      6   8676   0.12
#> Wilks       3   0.998    1.70      6   8674   0.12
#> Hotelling-Lawley 3   0.002    1.70      6   8672   0.12
#> Roy         3   0.002    2.98      3   4338   0.03 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

10.4.4 Factorial MANOVA

When there are two or more categorical factors, the general linear model provides a way to investigate the effects (differences in means) of each simultaneously. More importantly, this allows you to determine if factors *interact*, so the effect of one factor varies with the levels of another factor ...

Example: Penguins data

In Chapter 3 we examined the Palmer penguins data graphically, using a mosaic plot (Figure 3.36) of the frequencies of the three factors, `species`, `island` and `sex` and then `ggpairs()` scatterplot matrix (Figure 3.37).

TODO: Complete this

10.5 MRA → MMRA

When all predictor variables are quantitative, the MLM Equation 10.1 becomes the extension of univariate multiple regression analysis (MRA) to the situation where there are p response variables (RA). Just as in univariate models, we might want to test hypotheses about subsets of the predictors, for example when some predictors are meant as controls or things you might want to adjust for in assessing the effects of predictors of main interest.

But first, there are a couple of aspects of statistical practice that should be mentioned.

Model selection is one topic where univariate and multivariate approaches differ. When there are more than a few predictors, approaches like hierarchical regression, LASSO (Tibshirani, 1996) and stepwise selection can be used to eliminate uninformative predictors for each response.³ But this gives a different models for each response, based on the predictors included, each with its own interpretation. In contrast, the multivariate approach considers the outcome variables collectively. You can eliminate predictors that are unimportant, but the mechanics are geared toward removing them from the models for *all* responses.

Overall tests In a one-way ANOVA, to control for multiple testing, it is common practice to carry out an overall F -test to see if the group means differ collectively before testing comparison between specific groups.

³“Automatic” model selection procedures like stepwise regression, while seemingly attractive are dangerous in that can increase false positives or drop variables that should, on logical grounds, be included in the model. See [Stepwise selection of variables in regression is Evil](#) and Harrell ([Harrell, 2015](#)).

Similarly, in univariate multiple regression, researchers sometimes report an overall F -test or test of the R^2 so they can reject the hypothesis that *all* predictors have no effect, before considering them individually.

Similarly, in the case of multivariate linear models, some consider it necessary to reject the multivariate null hypothesis for a predictor term before considering how it contributes to each of the response variables. Some further suggest that the individual univariate models be tested after an overall significant effect. I believe the first of these is wise, but the second might be too much to require when the general goal is to understand the data.

10.5.1 Example: NLSY data

The dataset `NLSY` comes from a small part of the National Longitudinal Survey of Youth, a series of annual surveys conducted by the U.S. Department of Labor to examine the transition of young people into the labor force. This particular subset gives measures of 243 children on mathematics and reading achievement and also measures of behavioral problems (antisocial, hyperactivity). Also available are the yearly income and education of the child's father.

In this analysis the `math` and `read` scores are taken as the outcome variables.⁴ Among the remaining predictors, income and `educ` might be considered as background variables necessary to control for. Interest might then be focused on whether the behavioral variables `antisoc` and `hyperact` contribute beyond that.

```
data(NLSY, package = "heplots")
str(NLSY)
#> 'data.frame': 243 obs. of 6 variables:
#> $ math    : num 50 28.6 50 32.1 21.4 ...
#> $ read    : num 45.2 28.6 53.6 34.5 22.6 ...
#> $ antisoc : int 4 0 2 0 0 1 0 1 1 4 ...
#> $ hyperact: int 3 0 2 2 2 0 1 4 3 5 ...
#> $ income   : num 52.52 42.6 50 6.08 7.41 ...
#> $ educ     : int 14 12 12 12 14 12 12 12 12 9 ...
```

Exploratory plots

To begin, I would examine some scatterplots and univariate displays. I'll start with density plots for all the variables to see the shapes of their distributions, with rug plots at the bottom to show where the observations are located. From Figure 10.9 we see that math and reading scores are positively skewed, anti-social and hyperactivity have distributions highly concentrated in the lower scores. As we would suspect, father's income is quite positively skewed. Father's education is reasonably symmetric, but highly peaked at 12 years of schooling in this sample. The spikes reflect the fact that education is measured in discrete years.

```
NLSY_long <- NLSY |>
  tidyverse::pivot_longer(math:educ, names_to = "variable") |>
  dplyr::mutate(variable = forcats::fct_inorder(variable))

ggplot(NLSY_long, aes(x=value, fill=variable)) +
  geom_density(alpha = 0.5) +
  geom_rug() +
  facet_wrap(~variable, scales="free") +
  theme_bw(base_size = 14) +
  theme(legend.position = "none")
```

⁴Other choices are possible: we could instead try to model the behavioral variables, `antisocial` and `hyperact` first, and then determine if the parental variables add appreciably to this. Modeling choices aren't arbitrary. They should reflect the aims of a study and the story you want to tell about the result.

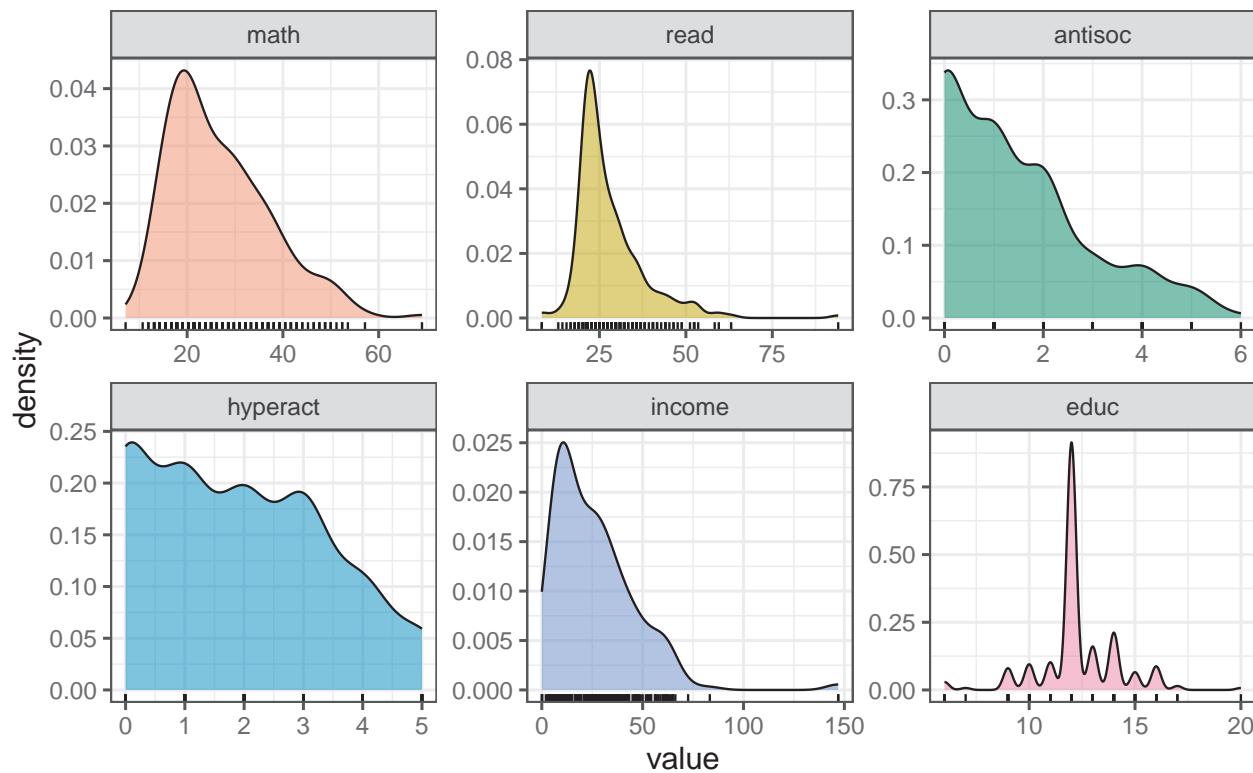


Figure 10.9: Density plots for the variables in the NLSY dataset.

In terms of an analysis focused on `math` and `read` as outcomes, a scatterplot of one against the other is useful, as is collection of scatterplots of each against the remaining variables. The second of these is left as an exercise to the reader.

```
set.seed(47)
ggplot(NLSY, aes(x = read, y = math)) +
  geom_jitter()+
  geom_smooth(method = lm, formula = y~x, fill = "blue", alpha = 0.2) +
  geom_smooth(method = loess, se = FALSE, color = "red", linewidth = 2)
```

The non-linear trend in Figure 10.10 may be due to the sparsity of data in the upper range of reading, and there are also a few unusual points shown in this plot. The function `heplots::noteworthy()` provides a variety of methods to identify such noteworthy points in scatterplots. The default `method` uses Mahalanobis D^2 . The plot below labels the five largest observations.

```
ids <- heplots::noteworthy(NLSY[, 1:2], method = "mahal", n=5)
ggplot(NLSY, aes(x = read, y = math)) +
  geom_jitter()+
  geom_smooth(method = lm, formula = y~x, fill = "blue", alpha = 0.2) +
  geom_text(data = NLSY[ids, ], label = ids, size = 5, nudge_y = 2)
```

Fitting models

We could of course include all of the predictors in a single model, and perhaps be done with it. To develop some model-thinking, it is more useful to proceed in smaller steps to see what we can learn from each. If we

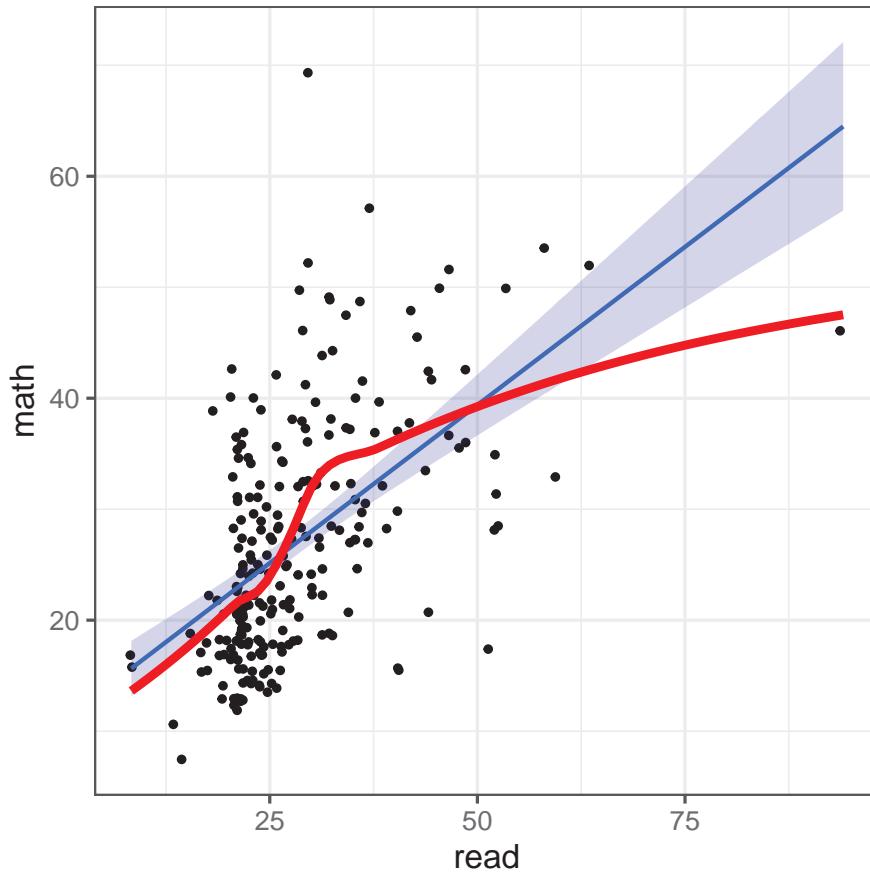


Figure 10.10: Scatterplot of mathematics score against reading score in the NLSY data

view parents' income and education as the most obvious predictors of reading and mathematics scores, those are the variables to fit first.

```
NLSY.mod1 <- lm(cbind(read, math) ~ income + educ,
                  data = NLSY)

Anova(NLSY.mod1) # Type II, partial test
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> income    1   0.0345     4.27      2    239 0.0151 *
#> educ      1   0.0515     6.49      2    239 0.0018 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Overall test

The `Anova()` results above give multivariate tests of the contributions of each predictor *separately* to explaining reading and math and how they vary together. To get an overall test of the global null hypothesis $\mathcal{H}_0 : \mathbf{B} = (\beta_{\text{inc}}, \beta_{\text{educ}}) = \mathbf{0}$ for *all* predictors together, you can use `linearHypothesis()`:

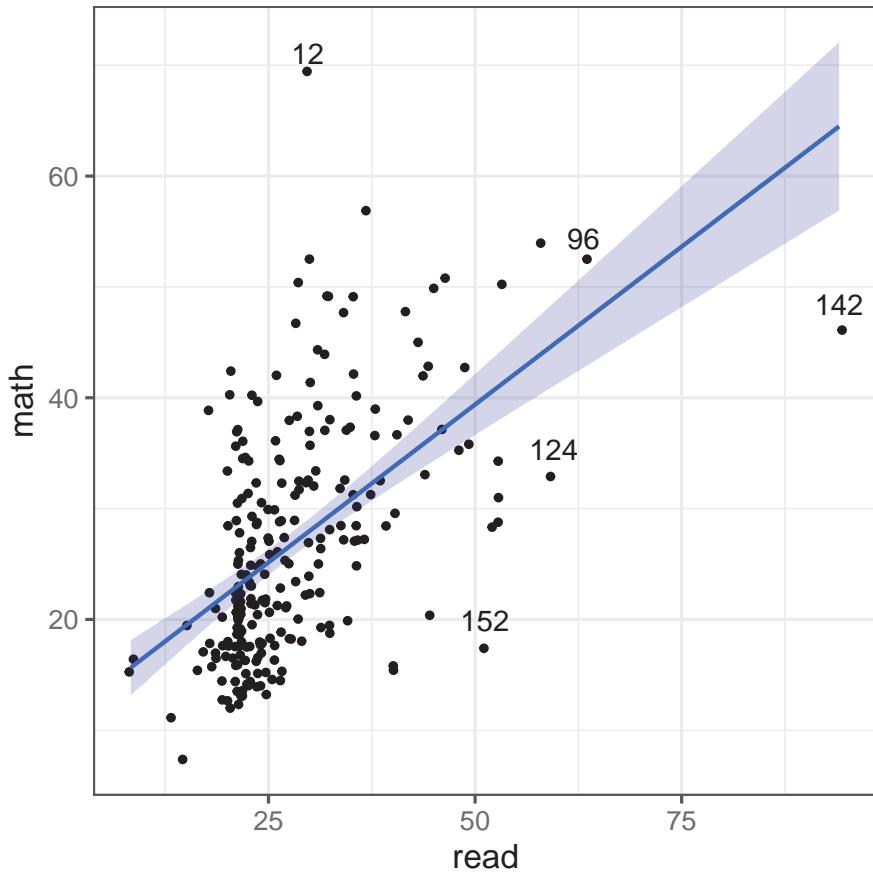


Figure 10.11: Scatterplot of mathematics score against reading score in the NLSY data, highlighting noteworthy points

```

coefs <- rownames(coef(NLSY.mod1))[-1]
linearHypothesis(NLSY.mod1, coefs, title = "income, educ = 0") |>
  print(SSP = FALSE)
#>
#> Multivariate Tests: income, educ = 0
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai        2   0.117    7.44      4     480 8.1e-06 ***
#> Wilks         2   0.884    7.59      4     478 6.2e-06 ***
#> Hotelling-Lawley 2   0.130    7.75      4     476 4.7e-06 ***
#> Roy          2   0.123   14.79      2     240 8.7e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

This joint multivariate test is more highly significant than either of those for the separate effects of the predictors, again because it pools strength.

Coefficient plots

As usual, you can display the coefficients using `coef()`. The `tidy` method for "`mle`" objects defined in `heplots` shows these in a tidy format with *t*-tests for each coefficient., arranged by the response variable.

```

coef(NLSY.mod1)
#>             read   math
#> (Intercept) 15.8848 8.7829
#> income       0.0137 0.0893
#> educ         0.9495 1.2755

tidy(NLSY.mod1)
#> # A tibble: 6 x 6
#>   response term      estimate std.error statistic p.value
#>   <chr>     <chr>      <dbl>     <dbl>      <dbl>     <dbl>
#> 1 read      (Intercept) 15.9       4.25      3.74  0.000233
#> 2 read      income      0.0137    0.0325     0.420 0.675
#> 3 read      educ        0.949     0.360      2.64  0.00894
#> 4 math      (Intercept) 8.78       4.33      2.03  0.0437
#> 5 math      income      0.0893    0.0331     2.70  0.00749
#> 6 math      educ        1.28       0.367     3.47  0.000607

```

However, a bivariate plot of these coefficients is more useful, because it provides visual tests of multivariate hypotheses. `heplots:::coefplot.mlm()` gives displays of the coefficients for a given *pair* of response variables. For interpretation, it adds the bivariate confidence ellipse for the coefficients, as well as univariate confidence intervals for each response. The univariate intervals are simply the horizontal and vertical shadows of the ellipses on the response variable axes.

A wrinkle here, as in Section 6.3, is that the coefficients are measured in different units and so coefficient plots for different predictors are more easily compared for standardized variables. To do this, I first re-fit the model using `scale(NLSY) ...`

```

NLSY_std <- scale(NLSY) |>
  as.data.frame()

NLSY_std.mod1 <- lm(cbind(read, math) ~ income + educ,
                     data = NLSY_std)

```

```

coefplot(NLSY_std.mod1, fill = TRUE,
         col = c("darkgreen", "brown"),
         lwd = 2,
         cex.lab = 1.5,
         ylim = c(-0.1, 0.5),
         xlab = "read coefficient (std)",
         ylab = "math coefficient (std)")

```

In Figure 10.12, the confidence ellipses for `income` and `educ` both exclude the origin, which represents the multivariate hypothesis $\mathcal{H}_0 : (\beta_{\text{read}}, \beta_{\text{math}}) = (0, 0)$, so this hypothesis is rejected. Note that if we only examined the univariate tests for each of the four parameters, we would conclude that for reading, `income` is not a significant predictor. The orientation of the confidence ellipses indicates the positive correlation between reading and mathematics scores.

10.5.1.1 Behavioral measures

Given that the parental background variables are highly predictive of student performance, we might want to know if the behavioral measures `antisoc` and `hyperact` add importantly to this. One way to do this is to add these predictors to the model and test for their additional contributions over and above the baseline model.

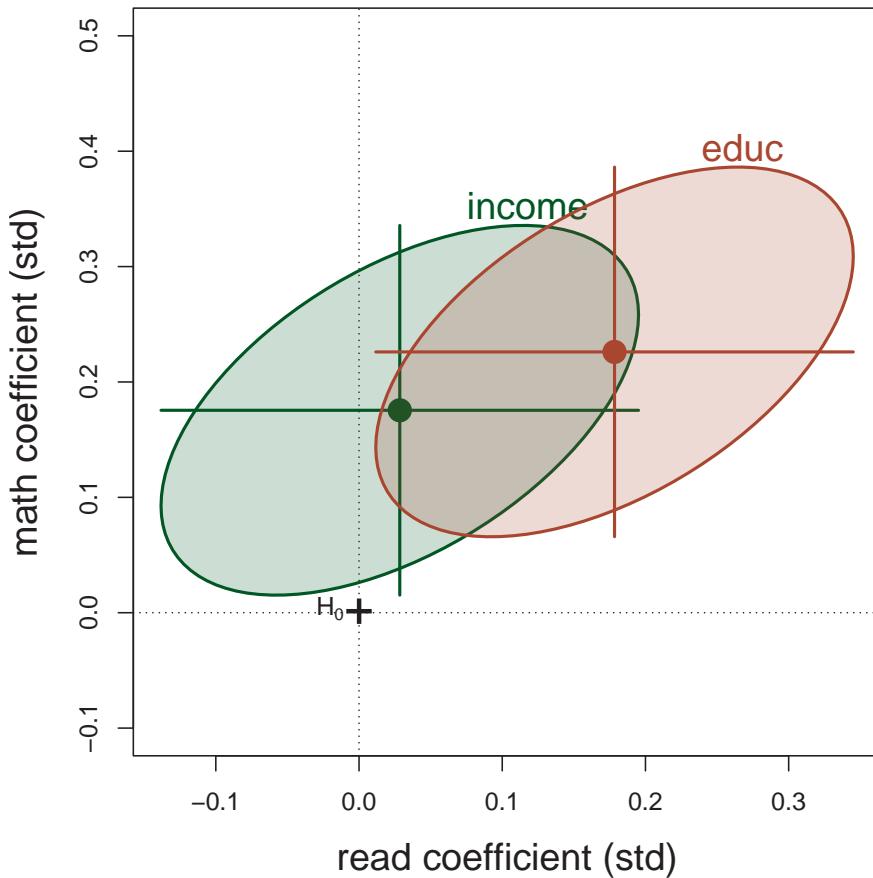


Figure 10.12: Bivariate coefficient plot for reading and math with 95% confidence ellipses. The variables have been standardized to make their units comparable.

You can do this using `update()`. In the model formula, “.” on the left hand side corresponds to the previous y variables; on the right-hand side it refers to the xs in the previous model, so I just add the new predictors to that.

```
NLSY.mod2 <- update(NLSY.mod1, . ~ . + antisoc + hyperact)
Anova(NLSY.mod2)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> income     1   0.0383    4.72      2    237 0.0098 ***
#> educ       1   0.0532    6.65      2    237 0.0015 ***
#> antisoc    1   0.0193    2.34      2    237 0.0988 .
#> hyperact   1   0.0144    1.74      2    237 0.1784
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Each of these new predictors are individually non-significant according to the Type II tests. Using `linearHypothesis()` you can test them jointly:

```

coefs <- rownames(coef(NLSY.mod2))[-1] |> print()
#> [1] "income"    "educ"       "antisoc"    "hyperact"

linearHypothesis(NLSY.mod2, coefs[3:4],
                  title = "NLSY.mod2 | NLSY.mod1") |>
  print(SSP = FALSE)
#>
#> Multivariate Tests: NLSY.mod2 | NLSY.mod1
#>          Df test stat approx F num Df den Df Pr(>F)
#> Pillai      2   0.024   1.45      4    476  0.218
#> Wilks       2   0.976   1.44      4    474  0.218
#> Hotelling-Lawley 2   0.024   1.44      4    472  0.218
#> Roy         2   0.022   2.64      2    238  0.073 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

10.5.2 Example: School data

Charnes et al. (1981) describe a large scale “social experiment” in public school education. Seventy school sites across the U.S. participated and a number of variables related to attributes of parents and teachers were used to predict aspects of students’ success in academic indicators (reading, mathematics), but also in their self-esteem. It was conceived in the late 1960’s in relation to a federally sponsored program charged with providing remedial assistance to educationally disadvantaged early primary school students.

The study was focused on the management styles used to guide educational planning across schools. In particular, it was primarily designed to compare schools using Program Follow Through (PFT) management methods of taking actions to achieve goals with those of Non Follow Through (NFT).

Here, I simply focus on the relations between outcome scores on tests of reading, mathematics and self-esteem in relation to five explanatory variables related to parents and teachers:

- **education** level of mother as measured by the percentage of high school graduates among female parents.
- **occupation**, highest occupation of a family member on a rating scale.
- **visit**, an index of the number of parental visits to the school site.
- **counseling**, a measure calculated from data on time spent with child on school-related topics such as reading together, etc.
- **teacher**, number of teachers at the given site.

The dataset, given in `schooldata` contains observations for 70 schools.⁵

Exploratory plots

There are eight variables in this example, so a scatterplot matrix or even a corrgram might not be sufficiently revealing. As usual, I tried a number of different methods and found a couple that were interesting and useful.

Multivariate normality is **not** required for all the variables in \mathbf{Y} and \mathbf{X} — it is only required for the residuals, $\boldsymbol{\varepsilon} = \mathbf{Y} - \widehat{\mathbf{Y}}$. Yet, for MMRA problems, sometimes an initial χ^2 QQ plot provides a handy way to flag possibly unusual values to pay attention to as the analysis proceeds. In Figure 10.13 we see five cases outside the 95% confidence envelope.

⁵In this, schools 1–49 were PFT sites and the remaining sites 50–70 were NFT. A separate dataset `schoolsites` provides other information on the schools, such as the general education style, region of the U.S., size of the city and that of the student population.

```

data(schooldata, package = "heplots")
res <- cqplot(schooldata, id.n = 5) |> print()
#>      DSQ quantile      p
#> 59 44.6      21.0 0.00714
#> 44 38.8      18.0 0.02143
#> 33 27.9      16.5 0.03571
#> 66 24.0      15.5 0.05000
#> 35 21.7      14.7 0.06429

# save the case ID numbers
outliers <- rownames(res) |> as.numeric() |> print()
#> [1] 59 44 33 66 35

```

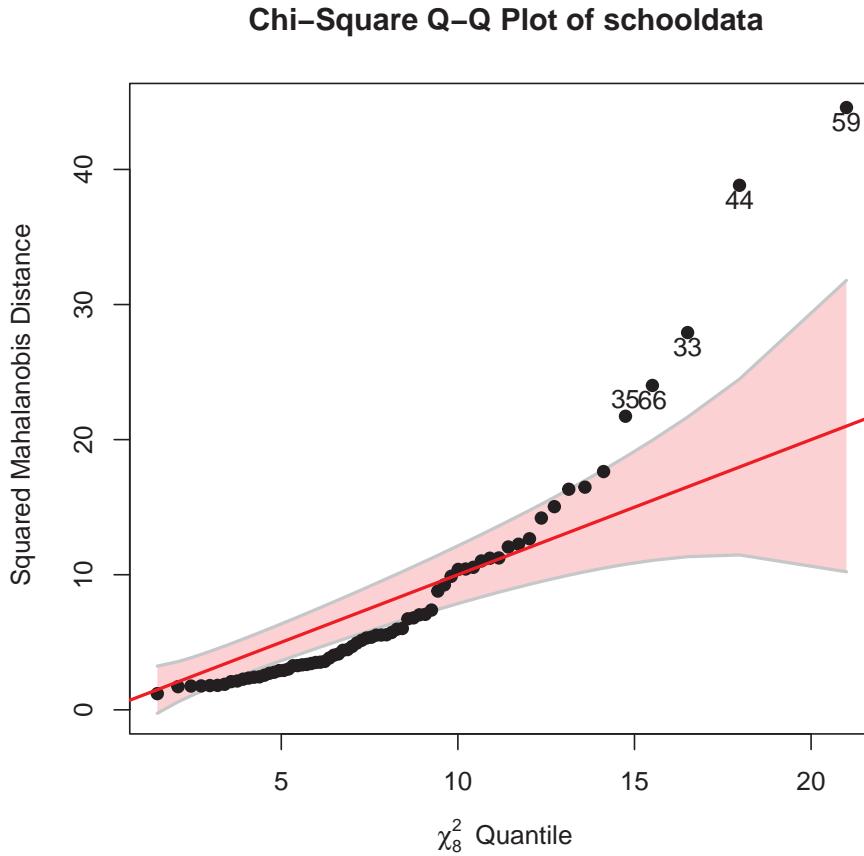


Figure 10.13: χ^2 QQ plot of the `schooldata` variables.

Rather than a complete 8×8 scatterplot matrix, it is useful here to examine the scatterplots *only* for each y variable against each of the predictors in \mathbf{X} .⁶ I'll take steps to flag some of these possibly unusual cases to see where they appear in these pairwise relations.

To prepare for this with `ggplot2`, it is necessary to reshape the data to long format *twice*—once for the ($q = 5$) x variables and again for the ($p = 3$) y responses to get all of their $q \times p$ combinations. That way, we get a data set with variables `x` and `y` whose variable names are by `xvar` and `yvar`.

⁶[GGally](#) has a function `ggduo()` that does something similar, plotting each of one set of variables against another. Like `ggpairs()` it allows for generalizations of a scatterplot where combinations of discrete factors and continuous variables can be displayed with appropriate visualizations for each.

```
# plot predictors vs each response
xvars <- names(schooldata)[1:5]
yvars <- names(schooldata)[6:8]

school_long <- schooldata |>
  tibble::rownames_to_column(var = "site") |>
  pivot_longer(cols = all_of(xvars),
                names_to = "xvar", values_to = "x") |>
  pivot_longer(cols = all_of(yvars),
                names_to = "yvar", values_to = "y") |>
  mutate(xvar = factor(xvar, xvars),
         yvar = factor(yvar, yvars))

car::some(school_long, n=8)
#> # A tibble: 8 x 5
#>   site   xvar      x  yvar      y
#>   <chr> <fct>    <dbl> <fct>    <dbl>
#> 1 1     teacher    9  reading   54.5
#> 2 18    education  28 mathematics 38.2
#> 3 26    teacher    7  selfesteem 31.2
#> 4 49    occupation 5.29 reading   12.2
#> 5 53    counseling 26.3 mathematics 22.0
#> 6 61    occupation 2.59 mathematics 7.1
#> 7 62    visit      9.89 reading   9.35
#> 8 64    occupation 8.91 mathematics 24.5
```

With this data structure, each scatterplot is a plot of a y against and x , and we can facet this using `facet_grid(yvar ~ xvar)`, giving Figure 10.14.

```
p1 <- ggplot(school_long, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, formula = y ~ x) +
  stat_ellipse(geom = "polygon",
               level = 0.95, fill = "blue", alpha = 0.2) +
  facet_grid(yvar ~ xvar, scales = "free") +
  labs(x = "predictor", y = "response") +
  theme_bw(base_size = 16)

# label the 3 most unusual points in each panel
p1 + geom_text_repel(data = school_long |>
                        filter(site %in% outliers[1:3]),
                        aes(label = site))
```

All of the predictors except for number of teachers show very strong linear relations with the outcome scores. Among the identified points, cases 44 and 59 stand out in all the plots, case 59 being particularly high on all the measures. As well, there is a small cluster of unusual points in the plots for number of teachers.

Fiting models

Let's proceed to fit the multivariate regression model. Here “.” on the right-hand side of the model formula means all the other variables in the dataset.

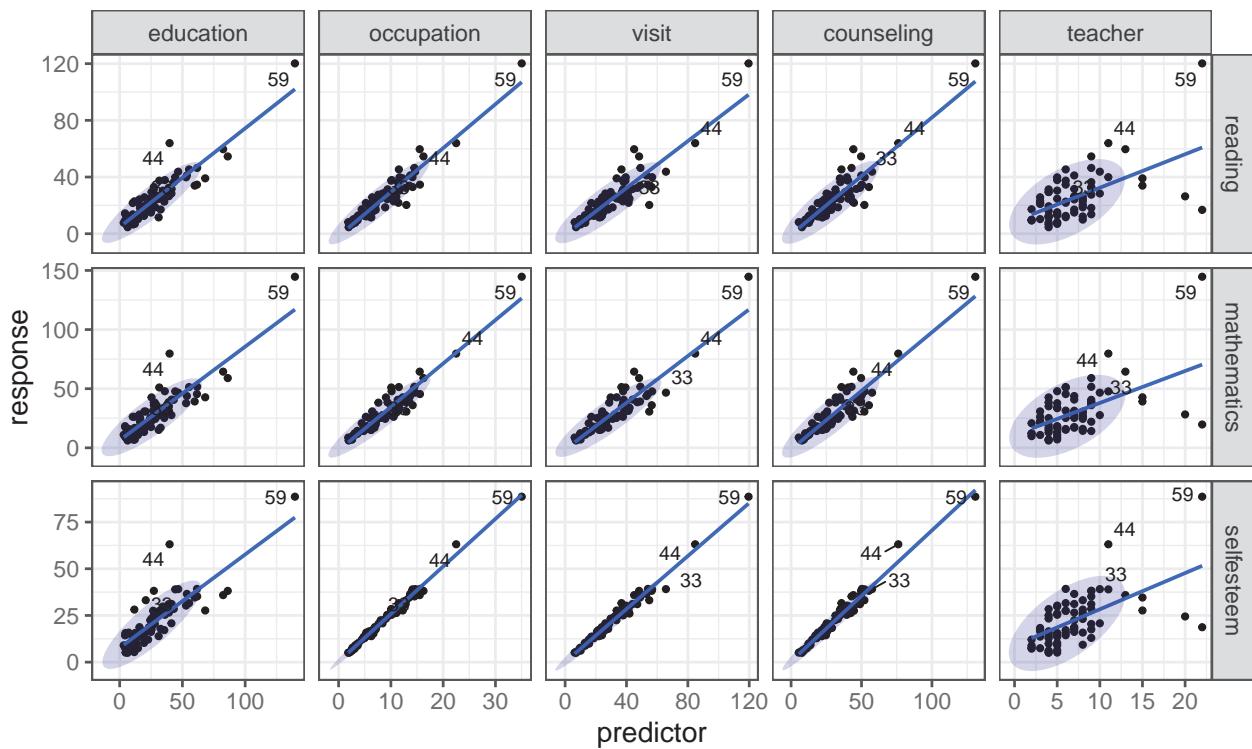


Figure 10.14: Scatterplots of each of the three response variables against each of the five predictors in the schooldata dataset. Three of the points identified as possible multivariate outliers are labeled.

```

school.mod <- lm(cbind(reading, mathematics, selfesteem) ~ .,
                  data=schooldata)
car::Anova(school.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> education   1    0.376     12.43      3     62 1.8e-06 ***
#> occupation  1    0.567     27.02      3     62 2.7e-11 ***
#> visit        1    0.260      7.27      3     62 0.00029 ***
#> counseling   1    0.065      1.43      3     62 0.24297
#> teacher      1    0.049      1.07      3     62 0.37003
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

These multivariate tests have a seemingly simple interpretation: parent's education and occupation and their visits to the schools are highly predictive of student's outcomes; their counseling efforts and the number of teachers in the schools, not so much.

You can get an assessment of the *strength* of multivariate association from the R^2 for each of the responses using `glance()` for the MLM. All of these are very high.

```

glance(school.mod)
#> # A tibble: 3 x 8
#>   response    r.squared sigma fstatistic numdf dendf  p.value  nobs
#>   <chr>        <dbl>    <dbl>       <dbl>    <dbl>    <dbl>    <dbl>    <int>

```

```
#> 1 reading      0.929  4.83      167.     5    64 2.34e-35    70
#> 2 mathematics 0.917  6.16      141.     5    64 3.37e-33    70
#> 3 selfesteem   0.993  1.17     1852.     5    64 8.47e-68    70
```

Similarly `etasq()` for an MLM gives an R^2 -like measure called η^2 of the *partial association* accounted for each of the *predictor terms* in the model. These are analogous to the Type II tests from `Anova()`, which test the additional contribution of each term in the model beyond all the others.

In spite of the overwhelming significance of the first three predictors, their variance accounted for is more modest. It is highest for parent's occupation, followed by education. Parent counseling and teachers contribute very little.

```
etasq(school.mod)
#>                 eta^2
#> education    0.3756
#> occupation   0.5666
#> visit        0.2603
#> counseling   0.0647
#> teacher       0.0491
```

10.6 Model diagnostics for MLMs

Model building, visualization and interpretation is often an iterative process. You fit a model and calculate some goodness of fit measures (R^2 for responses, η^2 for predictors). If these are reasonably strong, you feel happy and proceed to graphical displays to help you understand what you've found and explain it to others.

But **wait**: did you check the assumptions of the MLM? As in univariate models, diagnostic plots can help you spot problems in the data (unusual cases) or in the model (nonlinear relations, omitted predictors or interactions). You sometimes need to go circle back and fit a revised model, starting the process again.

For multivariate regression models, I consider the assumed multivariate normality of residuals and multivariate influence here. For MANOVA models question of homogeneity of covariance matrices is deferred until Chapter 12.

10.6.1 Multivariate normality of residuals

One easy thing to do is to check for multivariate normality of the residuals. Given that we found a few noteworthy points in Figure 10.13, a χ^2 QQ plot of the residuals in the model will tell us if any of these are really problematic. The pattern of points relative to the confidence band gives a rough indication of overall multivariate normality.

```
cqplot(school.mod, id.n = 5)
```

So, you can see that among the cases that stood out in the `cqplot()` of the observed variables (Figure 10.13), only case 35 attracts attention here, and it is well within the confidence band. Case 59, which was the largest in all the pairwise scatterplots (Figure 10.14) seems not unusual in the fitted model. It is a high-leverage point, but appeared to be well-fitted in all the simple regressions, except in those for `teacher`.

It is useful to contrast this with what we get from formal tests that the residuals are *strictly* multivariate normal. The **MVN** package (Korkmaz et al., 2025) provides `mvn()` for this, which performs a wide variety of

Chi-Square Q-Q Plot of school.mod

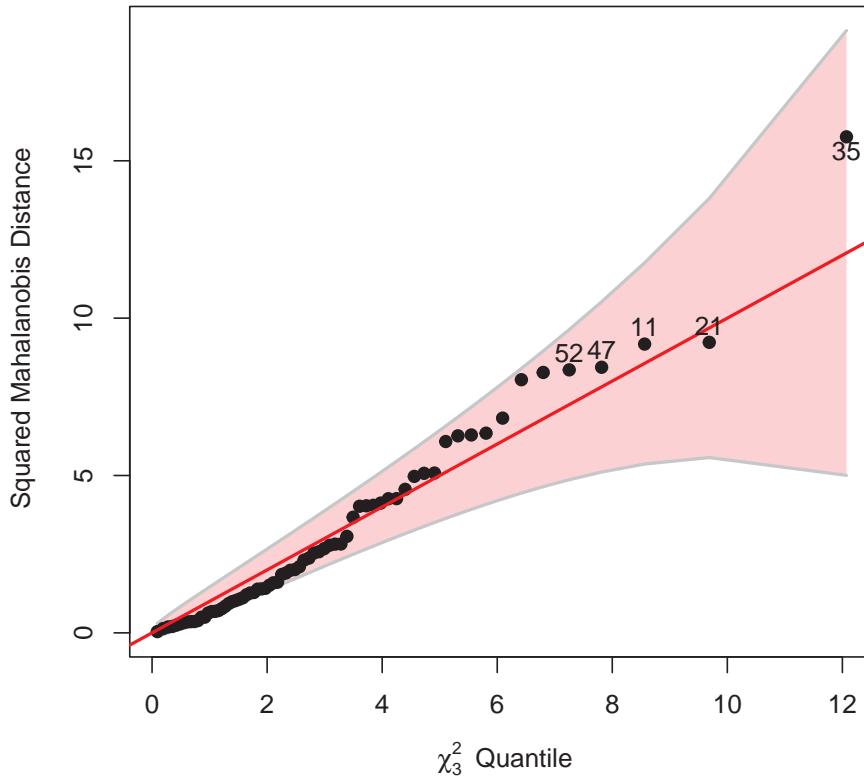


Figure 10.15: χ^2 QQ plot of the residuals in the schooldat multivariate regression model.

normality tests. The most widely used of these is due to Mardia (1974), which gives multivariate tests of skewness (lack of symmetry) and kurtosis (length of the tails).

Applying this to the **residuals** from the schools multivariate regression shows that multivariate normality is rejected here. Based on other evidence, this doesn't seem particularly troubling.

```

school.mvn <- mvn(residuals, mvn_test = "mardia")
# print multivariate and univariate tests
summary(school.mvn, select = "mvn")
#>           Test Statistic p.value     Method      MVN
#> 1 Mardia Skewness     1.92  0.750 asymptotic Normal
#> 2 Mardia Kurtosis    -1.16  0.244 asymptotic Normal
summary(school.mvn, select = "univariate")
#>           Test Variable Statistic p.value Normality
#> 1 Anderson-Darling start     0.307  0.524   Normal
#> 2 Anderson-Darling amount    0.625  0.085   Normal

```

`mvn()` also provides a variety of tests for univariate normality for each of the response variables. These are all OK.

10.6.2 Distance plot

Another useful screening plot (suggested by Peter J. Rousseeuw et al. (2004)) is a plot of Mahalanobis distances of the predictors against the Mahalanobis distances of the corresponding residuals for a fitted model. This diagnostic plot combines the information on leverage points and regression outliers in an interesting way.

It is much more useful than plotting them individually (against the theoretical values) as in a χ^2 QQ plot. Moreover, plotting them against each other is visually informative, because it places the leverage points (unusual in \mathbf{X}) and the outliers (unusual in \mathbf{Y}) in *different regions* of the plot. To judge notably “large” values, they suggest using cutoffs of $\sqrt{\chi_p^2(0.975)}$ for the predictor distances and $\sqrt{\chi_q^2(0.975)}$ for the residuals.

Such plots are produced by `heplots::distPlot()`. Running this on the `school.mod` model gives Figure 10.16. Points beyond the horizontal and vertical cutoff values are labeled with their case numbers.

```
distancePlot(school.mod, cex = 1.5, cex.lab = 1.2)
#> 0.975 X, Y distance cutoffs: 3.58 3.06
```

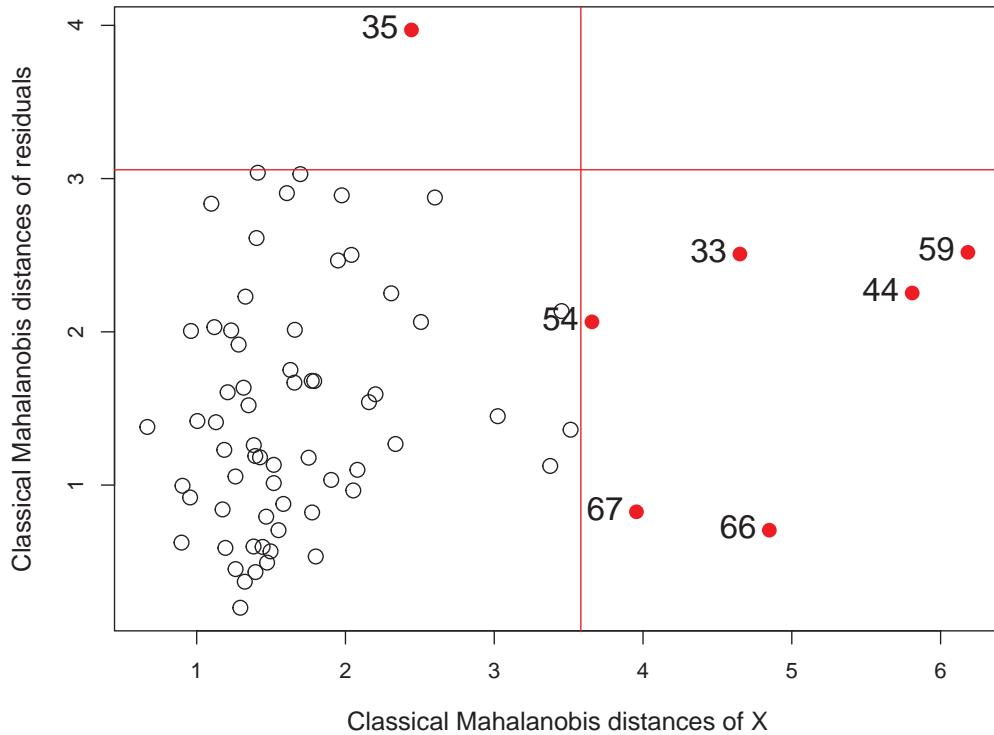


Figure 10.16: Plot of Mahalanobis distances of the least squares residuals vs. Mahalanobis distances of the predictors in the model

Most of the points identified in the χ^2 QQ plot Figure 10.13 are labeled here. Cases 44 and 59 are high leverage points with large \mathbf{X} distances. Case 35 is the only one beyond the cutoff for residuals. Interestingly, case 66 appeared near 35 in Figure 10.13, but is unusual for a different reason as we can see in Figure 10.16.

Peter J. Rousseeuw et al. (2004) also suggested methods of **robust multivariate regression** using robust estimates of location and scatter rather than the classical $\bar{\mathbf{y}}$ and \mathbf{S} . The *minimum covariance determinant* (*MCD*) estimator is a robust estimator with high breakdown value and bounded influence. It looks for the subset of size h , whose covariance matrix has the smallest determinant, where $h : \lceil n/2 \rceil < h < n$ controls the robustness. The `distPlot()` function implements this, as well as a *minimum variance ellipse* (*MVE*) method. Some robust methods are illustrated in Section 13.5.

10.6.3 Multivariate influence

TODO Sort out coverage here vs. Chapter 13

Again, what we see in simple scatterplots can be misleading because they ignore all the other variables in a model. But looking back after fitting a model and examining diagnostic plots can often be illuminating. Among the ideas we inherit from univariate models, the influence of particular observations on the results of analysis should be high on your list.

The multivariate extension of the diagnostic measures of leverage and influence, and influence plots (Section 6.6) is provided by the `mvinfluence` package (Friendly, 2025a). The theory behind this is due to Barrett & Ling (1992) and better illustrated in Barrett (2003). Mathematical details of this generalization are given in `help("mvinfluence-package")`.

Figure 10.17 shows one form of an influence plot for the `school.mod` model. Because multiple response variables are involved, this plots a measure of the *squared* studentized residuals for each observation against a generalized version of hat values, so potentially “bad” observations appear in the upper left corner. The size of the bubble symbol is proportional to a generalization of Cook’s distance, the measure of influence based on the change in all coefficients if each case was deleted from the analysis.

```
influencePlot(school.mod, id.n=4,
              type="stres",
              cex.lab = 1.5)
#>      H      Q CookD      L      R
#> 33 0.328 0.2017 0.7054 0.488 0.3001
#> 35 0.101 0.2825 0.3038 0.112 0.3142
#> 44 0.503 0.2984 1.6022 1.014 0.6009
#> 59 0.568 0.4937 2.9938 1.317 1.1441
#> 66 0.355 0.0174 0.0657 0.551 0.0269
```

That does not look good! You can see that cases 44 and 59 are actually quite troublesome here, but it turns out for different reasons. Take another look at Figure 10.14. You can see that case 59 is the most extreme on all the predictors, giving it very high leverage and therefore pulling the regression lines toward it in most of the plots, except those for number of teachers, where it also has large residuals. Case 44, on the other hand, stands out as high-leverage on only a few of the predictor-response combinations, but enough to give it a large multivariate hat value. It is also a point that is furthest from the regression lines.

What should be done? An appropriate action would be to re-fit the model, reducing the impact of these cases, in what I call a **sensitivity test**:

- Do the main conclusions change with those cases removed? That is, do any model terms change in significance tests or do coefficients change sign?
- Do the relative sizes of effects for predictors change enough to affect interpretation? That is, how much do the coefficients change?

The easiest solution is to just omit these troubling cases. You can do this using `update()`, specifying the `data` argument to be the subset of rows without the bad boys.

```
bad <- c(44, 59)
OK <- (1:nrow(schooldata)) |> setdiff(bad)
school.mod2 <- update(school.mod, data = schooldata[OK,])
Anova(school.mod2)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> education    1     0.211      5.35       3     60   0.0025 **
```

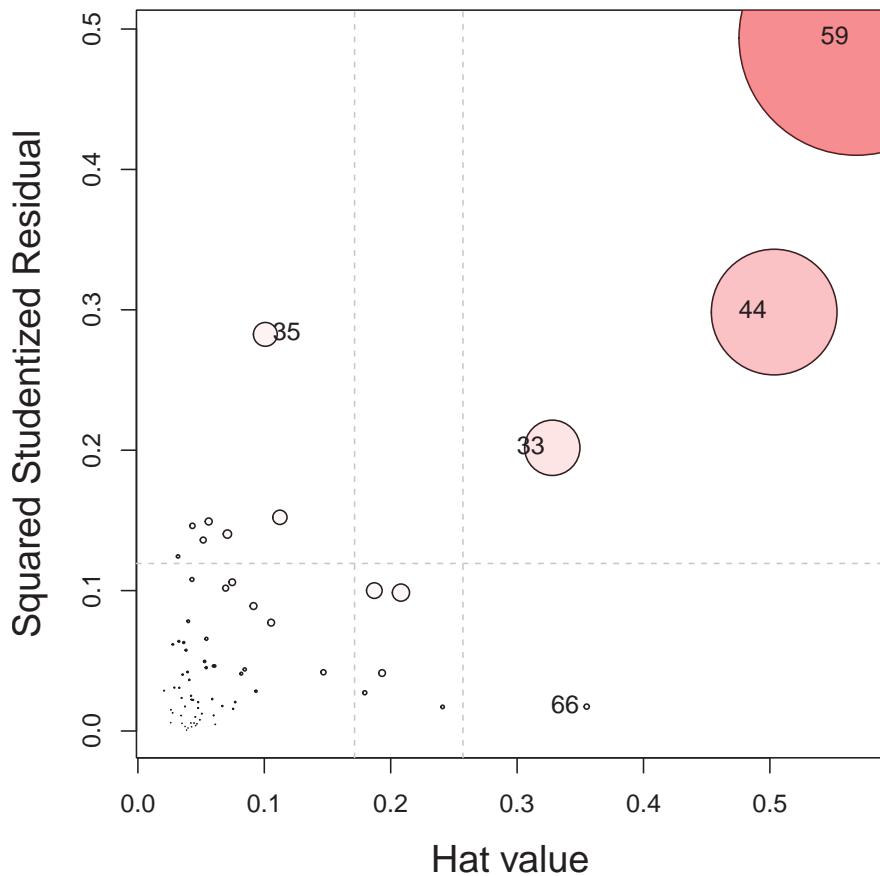


Figure 10.17: Influence plot for the `schooldat` multivariate regression model. Five cases are labeled as “noteworthy” on either axis.

```
#> occupation   1    0.422    14.62     3    60  2.9e-07 ***
#> visit        1    0.191     4.73     3    60   0.0050 **
#> counseling   1    0.053     1.13     3    60   0.3459
#> teacher      1    0.064     1.37     3    60   0.2618
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results of `Anova()` on this model tell us that the three significant predictors—occupation, education and visit—are still so, but slightly less so for the last two of these. To compare the coefficients in the new model compared to the old you can calculate the *relative difference*, $|\mathbf{B}_1 - \mathbf{B}_2|/\mathbf{B}_1$, which are

```
reldiff <- function(x, y, pct=TRUE) {
  res <- abs(x - y) / x
  if (pct) res <- 100 * res
  res
}

reldiff(coef(school.mod)[-1,], coef(school.mod2)[-1,]) |>
  round(1)
#>          reading mathematics selfesteem
```

```
#> education      10.1      19.8     -75.7
#> occupation    11.0      10.2      19.4
#> visit        -103.6     -76.8      2.8
#> counseling   -332.2     305.3     -115.4
#> teacher       -7.5      -5.6      7.0
```

As you can see, the effects on the coefficients for `visit` and `counseling` are dramatic.

10.7 ANCOVA → MANCOVA

TODO: Consider moving this to Chapter 11 and use much of the heplots MMRA vignette.

In univariate linear models, analysis of covariance (ANCOVA) is most often used in a situation where we want to compare the mean response, \bar{y}_j for different groups (defined by one or more factors), but where there are one or more quantitative predictors \mathbf{x}_1, \dots that should be taken into account for our comparisons to make sense. The simplest case is when \mathbf{x} is a pre-test score on the same measure, or when it is a background measure like age or level of education that we want to control for, to adjust for differences among the groups.

More generally, ANCOVA and its' multivariate MANCOVA brother are used for situations where the model matrix \mathbf{X} contains a mixture of factor variables and quantitative predictors, called "covariates". In this wider context, there are two flavors of analysis with different emphasis on the factors or the covariates:

- **true ANCOVA/MANOVA:** Attention is centered on the differences between the group means, but controlling for any difference in the covariate(s). This requires assuming that the slopes for the groups are all the same.
- **homogeneity of regression:** Here the focus is on the regression relations between the `ys` and the predictor `xs`, but we might also want to determine if the regression slopes are the same for all groups defined by the factors.

In the ANCOVA flavor, the model fits additive effects of the group factor(s) and the covariate(s), while the homogeneity of regression flavor adds interaction terms between groups and the `xs`. The test for homogeneity of regression is the added effect of the interaction terms:

```
mod1 <- lm(y ~ Group + x)          # ANCOVA model
mod2 <- lm(y ~ Group + x + Group:x) # allow separate slopes
mod2 <- lm(y ~ Group * x)           # same as above

anova(mod1, mod2)                  # test homogeneity of regression
```

Figure 10.18 illustrates these cases for a hypothetical two-group design studying the effect of an exercise program treatment on weight, recorded pre- (x) and post- (y) compared to a control group given no treatment. In panel (a) the slopes for the two groups are approximately equal, so the effect of treatment can be estimated by the difference in the fitted values of \hat{y}_i at the average value of x . In panel (b), the slope for the treated group is considerably greater than that for the control group, so the difference between the groups varies with x .

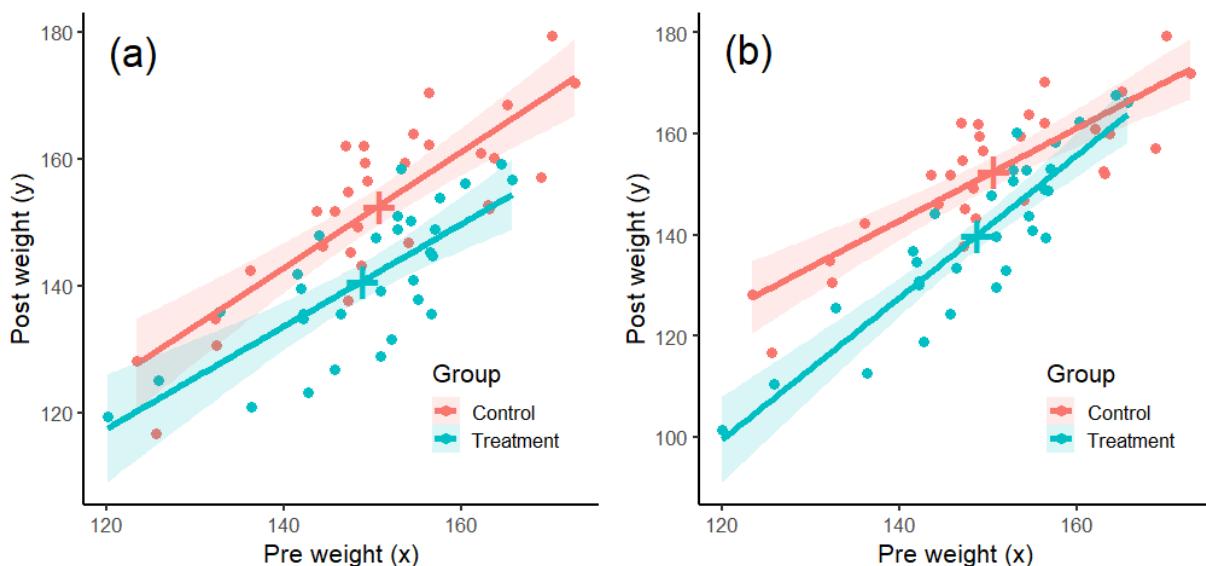


Figure 10.18: Two possible outcome patterns for a two-group design assessing the effect of a treatment on weight, measured pre- and post-treatment. (a) Additive effects of Group and x ; (b) Different slopes for the two groups. Plus signs show the means (\bar{x}_i, \bar{y}_i) for the two groups.

10.7.1 Example: Paired-associate tasks and academic performance

To what extent can simple tests of paired-associate learning⁷ predict measures of aptitude and achievement in kindergarten children? This was the question behind an experiment by William Rohwer at the University of California, Berkeley.

There were three outcome measures, one verbal and two visually-based:

- A student achievement test (**SAT**),
- Peabody Picture Vocabulary test (**PPVT**),
- Raven Progressive Matrices test (**Raven**).

Four paired-associate tasks were used, which differed in the syntactic and semantic relationship between the stimulus and response terms in each pair. These are called *named* (**n**), *still* (**s**), *named still* (**ns**), *named action* (**na**), and *sentence still* (**ss**).

Rohwer's data, taken from Timm (1975), is given in **Rohwer**. But there's a MANCOVA wrinkle: Performance on the academic tasks is well-known to vary with socioeconomic status of the parents or the school they attend. A simple design was to collect data from children in two schools, one in a low SES neighborhood ($n = 37$) and the other an upper-class high SES one ($n = 32$). The data look like this:

```
data(Rohwer, package = "heplots")
set.seed(42)
Rohwer |> dplyr::sample_n(6)
#>   group SES SAT PPVT Raven n  s ns na ss
#> 49     2  Hi  88  105     21 2 11 10 26 22
#> 65     2  Hi  50   96     13 5  8 20 28 26
#> 25     1  Lo   6   57     10 0  1 16 15 17
```

⁷Paired-associate learning are among the simplest tests of memory and learning. The subject is given a list of pairs of words or nonsense syllables, like “banana - house” or “YYZ - Toronto” to learn. On subsequent trials she is given the stimulus term of each pair (“banana”, “YYZ”) and asked to reply with the correct response (“house”, “Toronto”).

```
#> 18      1  Lo  45  54     10 0  6  6 14 16
#> 69      2  Hi  50  78     19 5 10 18 27 26
#> 64      2  Hi  24 102    16 4 17 21 27 31
```

Following the scheme for reshaping the data used in Figure 10.14, a set of scatterplots of each predictor against each response will give a useful initial look at the data. There's a lot to see here, so the plot in Figure 10.19 focuses attention on the regression lines for the two groups and their data ellipses.

```
yvars <- c("SAT", "PPVT", "Raven")      # outcome variables
xvars <- c("n", "s", "ns", "na", "ss")    # predictors

Rohwer_long <- Rohwer %>%
  dplyr::select(-group) |>
  tidyr::pivot_longer(cols = all_of(xvars),
                       names_to = "xvar", values_to = "x") |>
  tidyr::pivot_longer(cols = all_of(yvars),
                       names_to = "yvar", values_to = "y") |>
  dplyr::mutate(xvar = factor(xvar, levels = xvars),
                 yvar = factor(yvar, levels = yvars))

ggplot(Rohwer_long, aes(x, y, color = SES, shape = SES, fill = SES)) +
  geom_jitter(size=0.8) +
  geom_smooth(method = "lm",
              se = FALSE,
              formula = y ~ x,
              linewidth = 1.5) +
  stat_ellipse(geom = "polygon", alpha = 0.1) +
  labs(x = "Predictor (PA task)",
       y = "Response (Academic)") +
  facet_grid(yvar ~ xvar,           # plot matrix of Y by X
             scales = "free") +
  theme_bw(base_size = 16) +
  theme(legend.position = "bottom")
```

You can see here that the high-SES group generally performs better than the low group. The regression lines have similar slopes in some of the panels, but not all. The low SES group also appears to have larger variance on most of the PA tasks.

MANCOVA model

Nevertheless, I fit the MANCOVA model that allows a test of different means for the two SES groups on the responses, but constrains the slopes for the PA covariates to be equal. Only two of the PA tasks (na and ns) show individually significant effects in the multivariate tests.

```
# Make SES == 'Lo' the reference category
Rohwer$SES <- relevel(Rohwer$SES, ref = "Lo")

Rohwer.mod1 <- lm(cbind(SAT, PPVT, Raven) ~ SES + n + s + ns + na + ss,
                   data=Rohwer)
Anova(Rohwer.mod1)
#>
#> Type II MANOVA Tests: Pillai test statistic
```

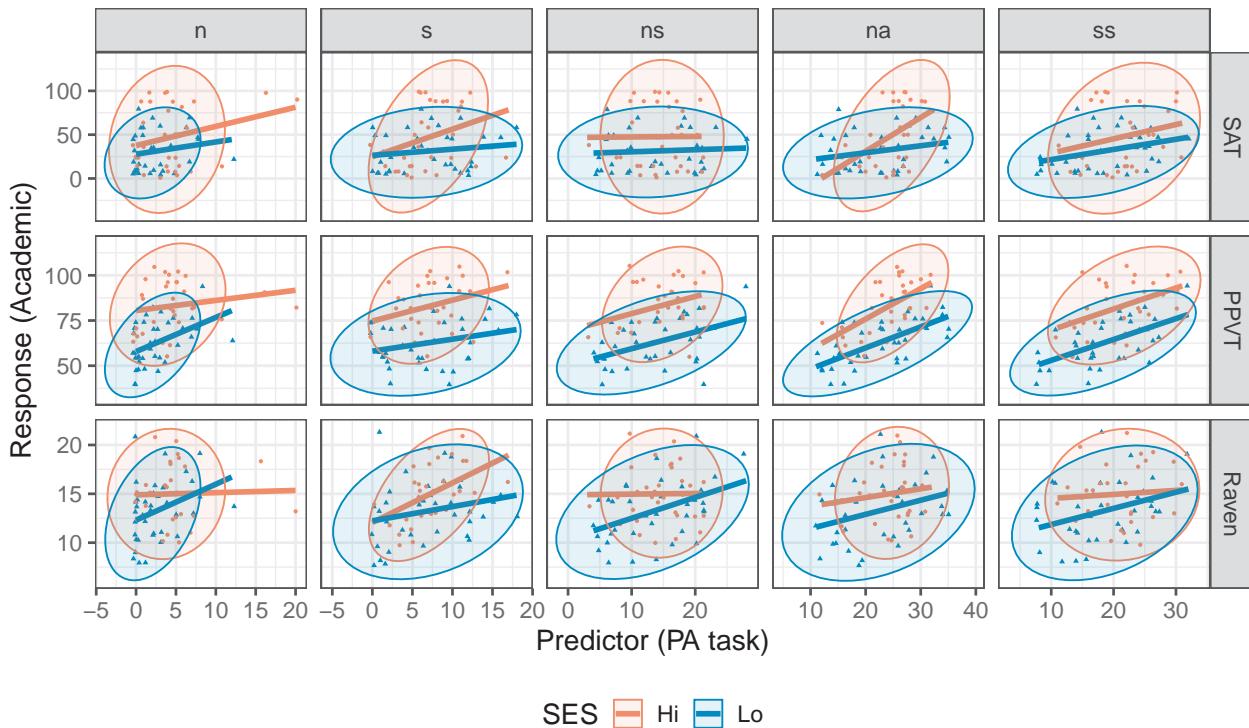


Figure 10.19: Scatterplots of each of the three response variables against each of the five predictors in the Rohwer dataset.

```
#>      Df test stat approx F num Df den Df Pr(>F)
#> SES   1    0.379    12.18     3    60 2.5e-06 ***
#> n     1    0.040     0.84     3    60  0.4773
#> s     1    0.093     2.04     3    60  0.1173
#> ns    1    0.193     4.78     3    60  0.0047 **
#> na    1    0.231     6.02     3    60  0.0012 **
#> ss    1    0.050     1.05     3    60  0.3770
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You can also examine the tests of the univariate ANCOVA models for each of the responses using `glance()` or `heplots::uniStats()`. All are significantly related, but the PPVT measure has the largest R^2 by far.

```
uniStats(Rohwer.mod1)
#> Univariate tests for responses in the multivariate linear model Rohwer.mod1
#>
#>          R^2      F df1 df2 Pr(>F)
#> SAT    0.295  4.33    6   62  0.001 **
#> PPVT   0.628 17.47    6   62  1e-11 ***
#> Raven  0.211  2.76    6   62  0.019 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To help interpret these effects, bivariate coefficient plots for the paired associate tasks are shown in Figure 10.20. (The coefficients for the group variable SES are on a different scale and so are omitted here.) From this you

can see that the named still and named action tasks have opposite signs: contrary to expectations, **ns** is negatively associated with the measures of aptitude and achievement (when the other predictors are adjusted for).

```
coefplot(Rohwer.mod1, parm = 2:6,  
         fill = TRUE,  
         level = 0.68,  
         cex.lab = 1.5)  
coefplot(Rohwer.mod1, parm = 2:6, variables = c(1,3),  
         fill = TRUE,  
         level = 0.68,  
         cex.lab = 1.5)
```

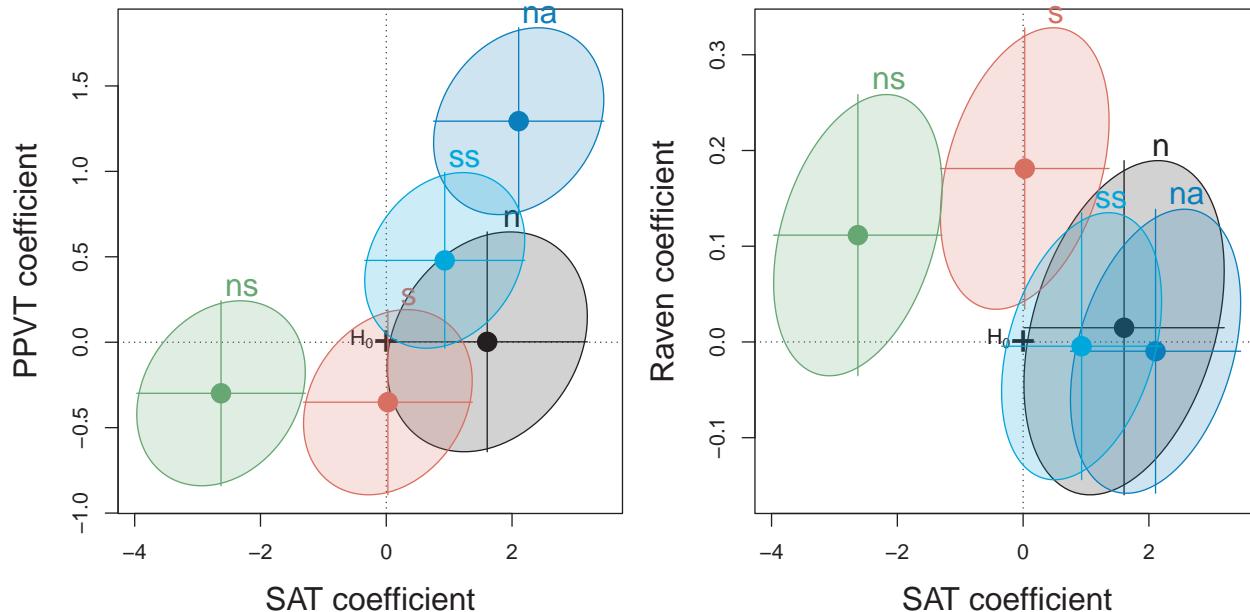


Figure 10.20: Bivariate coefficient plots for the MANCOVA model with confidence ellipses of 68% coverage.

TODO: For interpretation, it would be nice to know how many items were used for each of the PA tasks. The range of na goes up to 35, but others are less.

Adjusted means

From the analysis of covariance perspective, interest is often centered on estimating the differences between the group means, but adjusting or controlling for differences on the covariates. From the table of means below, you can see that the high SES group performs better on all three response variables, but this group also has higher scores on the paired associate tasks.

```
#> 1 Lo      1 31.3 62.6 13.2 2.54 6.92 13.5 22.4 18.4
#> 2 Hi      2 47.7 83.1 15     4.59 7.25 14.5 24.3 21.5
```

The adjusted mean differences are simply the values estimated by the coefficients for **SES** in the model. These are smaller than the differences between the observed means.

```
means[2, 3:5] - means[1, 3:5]
#>   SAT PPVT Raven
#> 1 16.4 20.4 1.76

# adjusted means
coef(Rohwer.mod1)[2,]
#>   SAT  PPVT Raven
#> 8.80 16.88 1.59
```

TODO: do this with a CI for the effects

Homogeneity of regression

The MANCOVA model, **Rohwer.mod1**, has relatively simple interpretations (a large effect of **SES**, with **ns** and **na** as the major predictors) but the test of the **SES** effect relies on the assumption of homogeneity of slopes for the predictors. We can test this assumption as follows, by adding interactions of **SES** with each of the covariates:

```
Rohwer.mod2 <- lm(cbind(SAT, PPVT, Raven) ~ SES * (n + s + ns + na + ss),
                     data=Rohwer)
Anova(Rohwer.mod2)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> SES       1    0.391    11.78      3     55 4.5e-06 ***
#> n         1    0.079     1.57      3     55 0.20638
#> s         1    0.125     2.62      3     55 0.05952 .
#> ns        1    0.254     6.25      3     55 0.00100 ***
#> na        1    0.307     8.11      3     55 0.00015 ***
#> ss         1    0.060     1.17      3     55 0.32813
#> SES:n     1    0.072     1.43      3     55 0.24417
#> SES:s     1    0.099     2.02      3     55 0.12117
#> SES:ns    1    0.118     2.44      3     55 0.07383 .
#> SES:na    1    0.148     3.18      3     55 0.03081 *
#> SES:ss    1    0.057     1.12      3     55 0.35094
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It appears from the above that there is only weak evidence of unequal slopes from the separate **SES**: terms; only that for **SES:na** is individually significant. The evidence for heterogeneity is stronger, however, when these terms are tested *collectively* using **linearHypothesis()**. I use a small **grep()** trick here to find the interaction terms, which have a ":" in their names.

```
# test interaction terms jointly
coefs <- rownames(coef(Rohwer.mod2))
interactions <- coefs[grep(":", coefs)]
```

```

print(linearHypothesis(Rohwer.mod2, interactions), SSP=FALSE)
#>
#> Multivariate Tests:
#>          Df test stat approx F num Df den Df Pr(>F)
#> Pillai      5   0.418    1.85     15    171 0.0321 *
#> Wilks       5   0.624    1.89     15    152 0.0277 *
#> Hotelling-Lawley 5   0.539    1.93     15    161 0.0240 *
#> Roy         5   0.385    4.38      5    57 0.0019 **
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Separate models

Model `Rohwer.mod2` with all interaction terms essentially fits a separate slope for each of the low and high SES groups for all responses with each of the predictor PA tasks. This is similar in spirit to what we would get if we fit a separate multivariate regression model for each of the groups, but parameterized differently: The heterogeneous regression model gives, for the interaction terms estimates of the difference in slopes between groups, while the separate-regressions approach gives separate slope estimates for each of the groups. These are equivalent, in the sense that the estimates for each approach can be derived from the other.

They are not equivalent in testing however, because the full model uses a combined pooled within-group error covariance, allows hypotheses about equality of slopes and intercepts to be tested directly and has greater power because it uses the total sample size. Here, I simply illustrate the mechanics of fitting separate models using the `subset` argument to `lm()`.

```

Rohwer.sesHi <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Hi")
Anova(Rohwer.sesHi)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> n     1   0.202    2.02      3    24 0.1376
#> s     1   0.310    3.59      3    24 0.0284 *
#> ns    1   0.358    4.46      3    24 0.0126 *
#> na    1   0.465    6.96      3    24 0.0016 ***
#> ss    1   0.089    0.78      3    24 0.5173
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Rohwer.sesLo <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Lo")
Anova(Rohwer.sesLo)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> n     1   0.0384    0.39      3    29 0.764
#> s     1   0.1118    1.22      3    29 0.321
#> ns    1   0.2252    2.81      3    29 0.057 .
#> na    1   0.2675    3.53      3    29 0.027 *
#> ss    1   0.1390    1.56      3    29 0.220
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The strength of evidence for the predictors `na` and `ns` is weaker here than when tested in the full heterogeneous model.

10.8 What we've learned

- The multivariate linear model is an extension of the (univariate) general linear model. Three major forms of the multivariate linear model are the MANOVA, MMRA and MANCOVA, all of which have their equivalents in ANOVA, MRA and ANCOVA.
- What the multivariate linear model offers over the univariate linear model is not just more powerful statistical tests, but also examinations of how the associations amongst the response variables varies with the explanatory variables.
- Just as with the general linear model, model diagnostics transfer to the multivariate linear model and are important for identifying potential issues in the validity of the estimated statistical model.
- Given that the multivariate linear model can do everything that the general linear model does, there is no reason to not consider the multivariate linear model. Confining discussions of multivariate response models to structural equation modeling may pedagogically create a cognitive blind spot that the association amongst response variables even *can* vary with the explanatory variables. Rather than something separate, for the purposes of causal inference, it would be better to use structural equation modeling as a more confirmatory statistical model than the multivariate linear model, which is perfectly well suited for exploratory purposes.
- Overall, we also see a glimpse of the importance of data visualization in the analysis of data using the multivariate linear model. The following chapter will explore this issue in more depth.

Packages used here:

13 packages used here: `broom`, `car`, `carData`, `dplyr`, `ggplot2`, `ggrepel`, `heplots`, `knitr`, `matlib`, `mvinfluence`, `MVN`, `patchwork`, `tidyR`

11

Visualizing Multivariate Models

The methods discussed in Chapter 10 provide the basis for a rather complete multivariate analysis of traditional univariate methods for the same designs. You can carry out multiple regression, ANOVA, or indeed, any classical linear model with the standard collection of analysis tools you use for a single outcome variable, but naturally extended in most cases to having several outcomes to analyse together. The key points are:

- Everything you know about the usual univariate models—regression coefficients, main effects and contrasts for factors, interactions of model terms, . . . applies here.
- By a rather clever design, called “matrix algebra” the separate univariate models can be combined, by turning vectors of responses, $\mathbf{y}_1, \mathbf{y}_2, \dots$ into a matrix \mathbf{Y} . Bingo! We get a multivariate extension.
- You can treat this as a collection of separate models, one for each response, because the coefficients are the same. But, the benefit of a multivariate approach is that you also get an overall multivariate test for each term in the model.

As nice as these mathematical and statistical ideas might be, the fact that the analysis is conducted for the response variables *collectively*, means that it may be harder to interpret and explain what this means about the separate responses. Here’s where multivariate model visualization comes to the rescue!

- **HE plots:** The tests of multivariate models, including multivariate analysis of variance (MANOVA) for group differences and multivariate multiple regression (MMRA) can be easily visualized by plots of a hypothesis (“H”) data ellipse for the fitted values, relative to the corresponding plot of the error ellipse (“E”) of the residuals, which I call the HE plot framework.
- **contrasts:** For factors in MANOVA, contrasts and linear hypotheses (Section 10.3.1) provide a way to decompose an overall multivariate test into portions directed at meaningful *specific* research questions regarding **how** the groups differ.
- **CDA:** For more than a few response variables, these result can be projected onto a lower-dimensional “canonical” space providing an even simpler description, accounting for most of the “juice”. Vectors for the response variables in this space show how these relate to the canonical dimension, facilitating interpretation.

Huang (2019) and others have criticized these methods as (a) difficult to understand because they are framed in terms of linear combinations of the the responses; (b) more complicated and limited in interpreting MANOVA effects and (c) unwieldy post hoc strategies often employed for interpretation.

These difficulties in understanding can, I believe, be cured by accessible graphical methods for visualizing hypothesis tests and for visualizing what these linear combinations reflect in terms of the observed variables. The HE plot framework described below provides powerful graphic methods available in easy used software.

This chapter describes this framework and illustrates some concrete examples, first for MANOVA designs which are conceptually and visually simpler, and then for MMRA designs with quantitative predictors and finally for MANCOVA models. Many more worked examples are available in vignettes for the `heplots`.¹

Packages

In this chapter I use the following packages. Load them now.

¹See the heplots vignettes [HE plot MANOVA Examples](#) and [HE plot MMRA Examples](#).

```
library(car)
library(heplots)
library(candisc)
library(ggplot2)
library(dplyr)
library(tidyr)
library(ggpubr)
library(patchwork)
```

11.1 HE plot framework

Chapter 9 illustrated the basic ideas of the framework for visualizing multivariate linear models in the context of a simple two group design, using Hotelling's T^2 . The main ideas were illustrated in Figure 9.9.

Having described the statistical ideas behind the MLM in Chapter 10, we can proceed to extend this framework to larger designs. Figure 11.1 illustrates these ideas using the simple one-way MANOVA design of the dogfood data from Section 10.2.1.

- In (a) **data space**, each group is summarized in (b) by its **data ellipse**, representing the means and covariances.
- Variation against the hypothesis of equal means can be seen by the **H** ellipse in the (c) **HE plot**, representing the data ellipse of the fitted values. Error variance is shown in the **E** ellipse, representing the pooled within-group covariance matrix, \mathbf{S}_p and the data ellipse of the residuals from the model. For the dogfood data, the group means have a negative relation: longer time to start eating is associated with a smaller amount eaten.
- The MANOVA (or Hotelling's T^2) is formally equivalent to a **discriminant analysis**, predicting group membership from the response variables which can be seen in data space. (The main difference is emphasis and goals: MANOVA seeks to test differences among group means, while discriminant analysis aims at classification of the observations into groups.)
- This effectively projects the p -dimensional space of the predictors into the smaller (d) **canonical space** that shows the greatest differences among the groups. As in a biplot, vectors show the relations of the response variables with the canonical dimensions.

For more complex models such as MANOVA with multiple factors or multivariate multivariate regression with several predictors, there is one sum of squares and products matrix (SSP), and therefore one **H** ellipse for *each term* in the model. For example, in a two-way MANOVA design with the model formula $(y_1, y_2) \sim A + B + A*B$ and equal sample sizes in the groups, the total sum of squares accounted for by the model is the sum of their separate effects,

$$\begin{aligned} \text{SSP}_{\text{Model}} &= \text{SSP}_A + \text{SSP}_B + \text{SSP}_{AB} \\ &= \mathbf{H}_A + \mathbf{H}_B + \mathbf{H}_{AB} . \end{aligned}$$

{#eq-HE-model}

All of these hypotheses can be overlaid in a single HE plot showing their effects together in a comprehensive view.

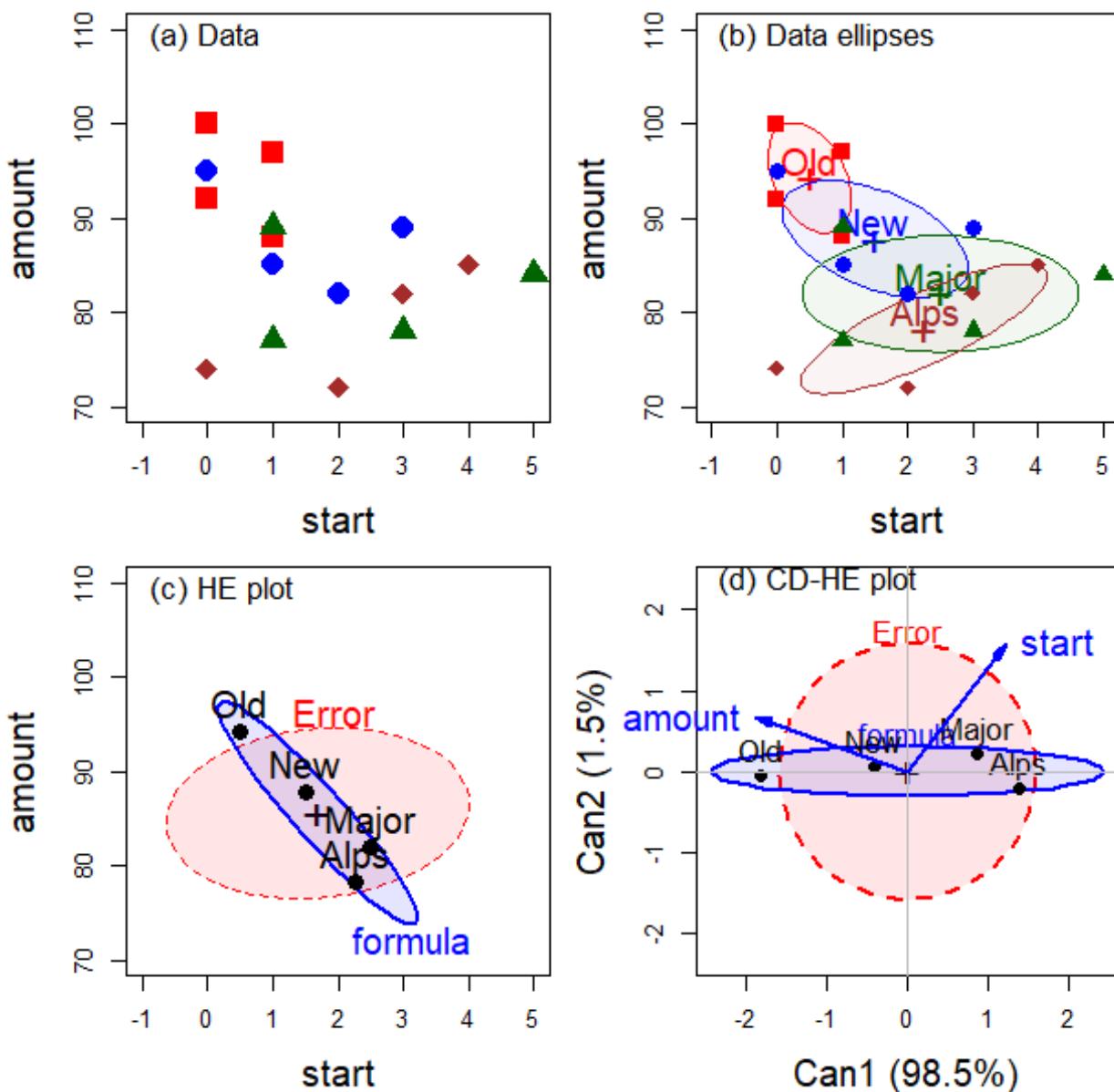


Figure 11.1: Dogfood quartet: Illustration of the conceptual ideas of the HE plot framework for the dogfood data. (a) Scatterplot of the data; (b) Summary using data ellipses; (c) HE plot shows the variation in the means in relation to pooled within group variance; (d) Transformation from data space to canonical space

11.2 HE plot construction

The HE plot is constructed to allow a direct visualization of the “size” of hypothesized terms in a multivariate linear model in relation to unexplained error variation. These can be displayed in 2D or 3D plots, so I use the term “ellipsoid” below to cover all cases.

Error variation is represented by a standard 68% data ellipsoid of the \mathbf{E} matrix of the residuals in $\boldsymbol{\varepsilon}$. This

is divided by the residual degrees of freedom, so the size of \mathbf{E}/df_e is analogous to a mean square error in univariate tests. The choice of 68% coverage allows you to “read” the residual standard deviation as the half-length of the shadow of the \mathbf{E} ellipsoid on any axis (see Figure 3.11).

The \mathbf{E} ellipsoid is then translated to the overall (grand) means $\bar{\mathbf{y}}$ of the variables plotted, which allows us to show the means for factor levels on the same scale, facilitating interpretation. In the notation of Equation 3.2, the error ellipsoid \mathcal{E}_c of size c is given by

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{E}) = \bar{\mathbf{y}} \oplus c\mathbf{E}^{1/2}, \quad (11.1)$$

where $c = \chi^2_2(0.68)$ for 2D plots and $c = \chi^2_3(0.68)$ for 3D plots of standard 68% coverage.² Ellipses of various coverage were shown in Figure 3.9.

An ellipsoid representing variation in the means of a factor (or any other term reflected in a general linear hypothesis test, Equation 10.9) uses the corresponding \mathbf{H} matrix is simply the data ellipse of the fitted values for that term. But there is a question of the relative scaling of the \mathbf{H} and \mathbf{E} ellipsoids for interpretation.

Dividing the hypothesis matrix by the error degrees of freedom, giving \mathbf{H}/df_e , puts this on the same scale as the \mathbf{E} ellipse. I refer to this as *effect size scaling*, because it is similar to an effect size index used in univariate models, e.g., $ES = (\bar{y}_1 - \bar{y}_2)/s_e$ in a two-group, univariate design. An alternative, *significance scaling* (Section 11.4) provides a visual test of significance of a model \mathbf{H} term.

To illustrate this concretely, consider the HE plot for the `dogfood` shown in Figure 11.1 (c), reproduced here as Figure 11.2.

```
data(dogfood, package="heplots")
dogfood.mod <- lm(cbind(start, amount) ~ formula, data=dogfood)

heplot(dogfood.mod,
       fill = TRUE, fill.alpha = 0.1,
       cex.lab = 1.5, cex = 1.5,
       xlim = c(-1, 4.5),
       ylim = c(70, 100))
```

From the analysis in Section 10.2.2, we found the \mathbf{H} matrix for the `formula` effect in the `dogfood.mod` model to be as shown below. The negative covariance, -70.94, reflects a correlation of -0.94 between the means of start time and amount eaten.

```
dogfood.aov <- Anova(dogfood.mod)
SSP_H <- dogfood.aov$SSP[[1]] |> print()
#>           start amount
#> start     9.69   -70.9
#> amount   -70.94  585.7
```

Similarly, the \mathbf{E} matrix, shown below, reflects a slight positive correlation, 0.12, for dogs fed the same formula.

```
SSP_E <- dogfood.aov$SSPE |> print()
#>           start amount
#> start     25.8    11.8
#> amount    11.8   390.3
```

Example 11.1. Iris data

Perhaps the most famous (or infamous) dataset in the history of multivariate data analysis is that of

²In smallish samples ($n < 30$) we use the better approximations, $c = \sqrt{2F_{2,n-2}^{0.68}}$ for 2D plots and $c = \sqrt{3F_{3,n-3}^{0.68}}$ for 3D.

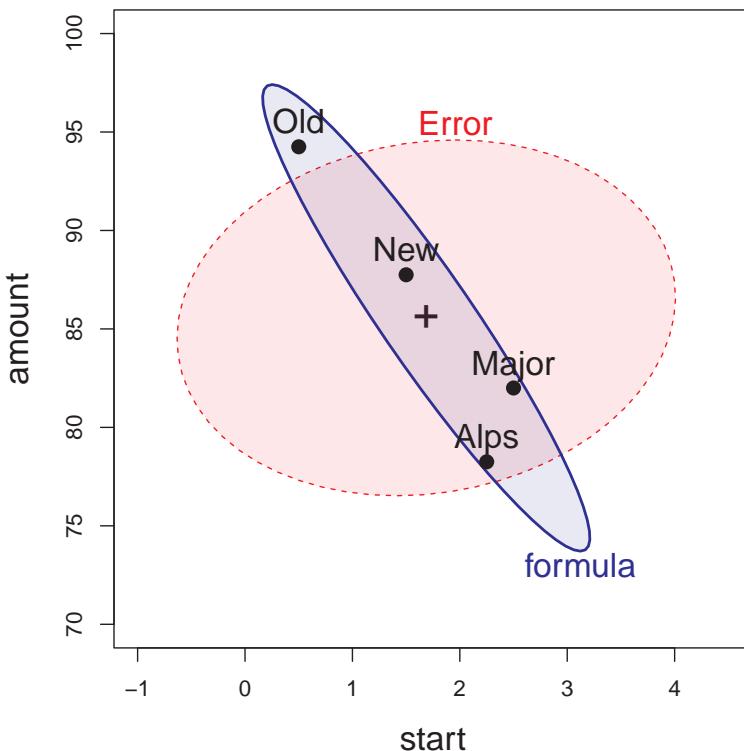


Figure 11.2: HE plot for the dogfood data, showing the means of the four groups, which generates the **H** ellipse for the effect of `formula`. The **E** ellipse labeled ‘Error’ shows the within-group variances and covariance.

measurements on three species of Iris flowers collected by Edgar Anderson (1935) in the Gaspé Peninsula of Québec, Canada. Anderson wanted to quantify the outward appearance (“morphology”: shape, structure, color, pattern, size) of species as a method to study variation within and between such groups. Although Anderson published in the obscure *Bulletin of the American Iris Society*, R. A. Fisher (1936) saw this as a challenge and opportunity to introduce the method now called discriminant analysis—how to find a weighted composite of variables to best discriminate among existing groups.

i History corner

I said “infamous” above because Fisher published in the *Annals of Eugenics*. He was an ardent eugenicist himself, and the work of eugenacists was often pervaded by prejudice against racial, ethnic and disabled groups. Through guilt by association, the Iris data, having mistakenly been called “Fisher’s Iris Data”, has become deprecated, even called “racist data”.³ The voices of the *Setosa*, *Versicolor* and *Virginica* of Gaspé protest: we don’t have a racist bone in our body and nor prejudice against any other species, to no avail.

Bodmer et al. (2021) present a careful account of Fisher’s views on eugenics within the context of his time and his contributions to modern statistical theory and practice. Fisher’s views on race were largely formed by Darwin and Galton, but “nearly all of Fisher’s statements were about populations, groups of populations, or the human species as a whole”. Regardless, the `iris` data were Anderson’s and should not be blamed. After all, if Anderson had gave his car to Fisher, would the car be tainted by Fisher’s eugenicist leanings?

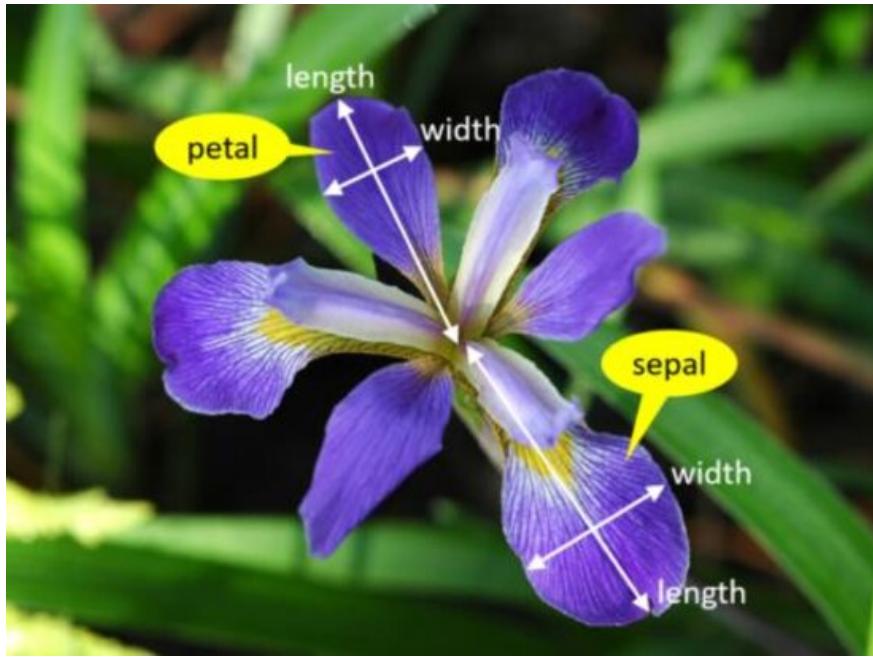


Figure 11.3: Diagram of an iris flower showing the measurements of petal and sepal size. Each flower has three sepals and three alternating petals. The sepals have brightly colored central sections. *Source:* Gayan De Silva (2020)

So that we understand what the measurements represent, Figure 11.3 superposes labels on a typical iris flower, having three sepals which alternate with three petals. Sepals are like ostentatious petals, with attractive decorations in the central section. Length is the distance from the center to the tip and width is the transverse dimension.

As always, it is useful to start with overview displays to see the data. A scatterplot matrix (Figure 11.4) shows that *versicolor* and *virginica* are more similar to each other than either is to *setosa*, both in their pairwise means (*setosa* are smaller) and in the slopes of regression lines. Further, the ellipses suggest that the assumption of constant within-group covariance matrices is problematic: While the shapes and sizes of the concentration ellipses for *versicolor* and *virginica* are reasonably similar, the shapes and sizes of the ellipses for *setosa* are different from the other two.

```
iris_colors <- c("blue", "darkgreen", "brown4")
scatterplotMatrix(~ Sepal.Length + Sepal.Width +
                  Petal.Length + Petal.Width | Species,
  data = iris,
  col = iris_colors,
  pch = 15:17,
  smooth=FALSE,
  regLine = TRUE,
  ellipse=list(levels=0.68, fill.alpha=0.1),
  diagonal = FALSE,
  legend = list(coords = "bottomleft",
                cex = 1.3, pt.cex = 1.2))
```

³For example, Megan Stodel in a blog post [Stop using iris](#) says, “*It is clear to me that knowingly using work that was itself used in pursuit of racist ideals is totally unacceptable.*” A Reddit discussion on this topic, [Is it socially acceptable to use the Iris dataset?](#) has some interesting replies.

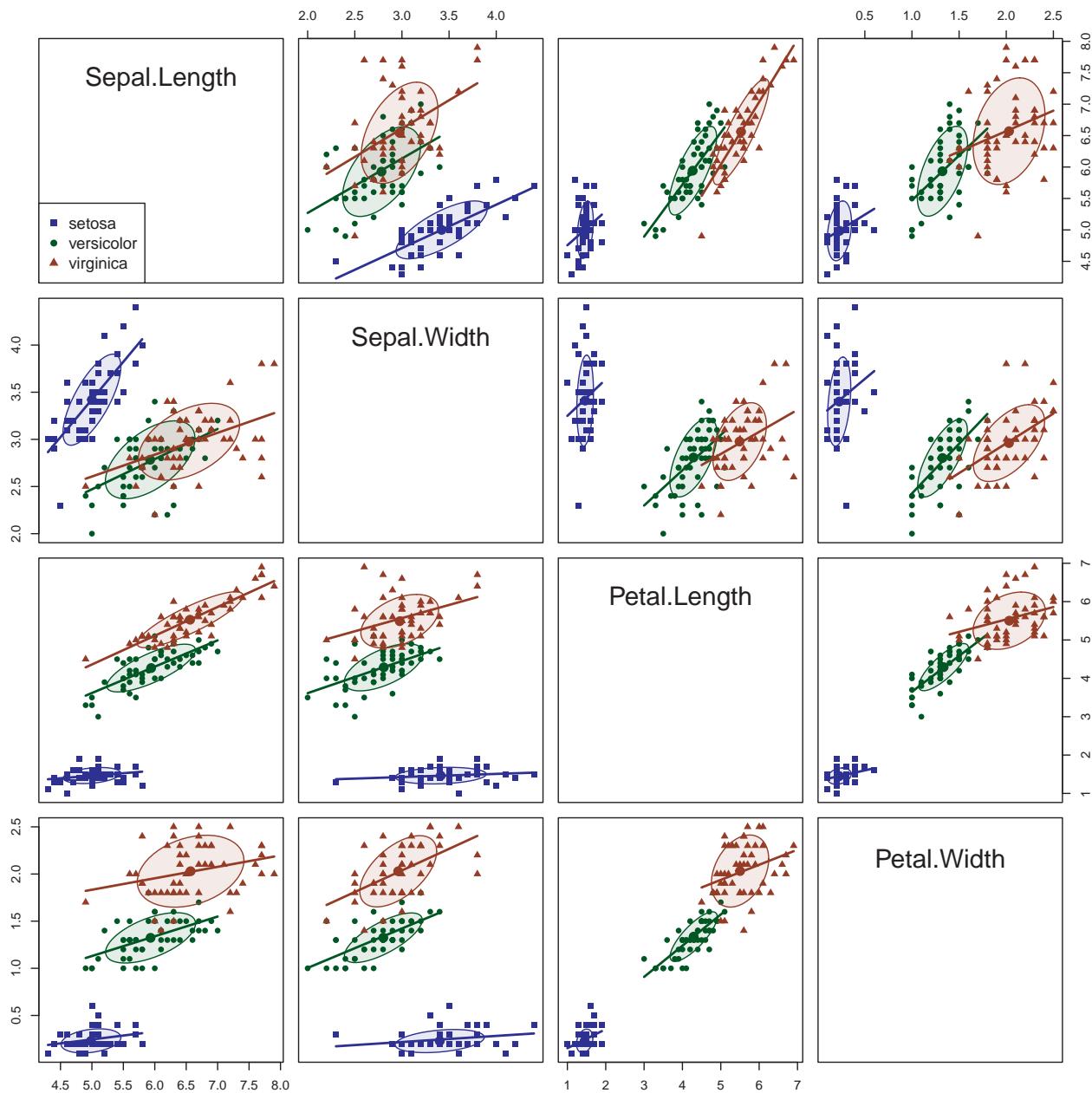


Figure 11.4: Scatterplot matrix of the `iris` dataset. The species are summarized by 68% data ellipses and linear regression lines in each pairwise plot.

**TODO*: Should this be a numbered section?

11.2.1 MANOVA model

We proceed nevertheless to fit a multivariate one-way ANOVA model to the iris data. The MANOVA model for these data addresses the question: “Do the means of the Species differ significantly for the sepal and petal variables taken together?”

$$\mathcal{H}_0 : \mu_{\text{setosa}} = \mu_{\text{versicolor}} = \mu_{\text{virginica}}$$

Because there are three species, the test involves $s = \min(p, g - 1) = 2$ degrees of freedom, and we are entitled

to represent this by two 1-df contrasts, or sub-questions. From the separation among the groups shown in Figure 11.4 (or more botanical knowledge), it makes sense to compare:

- Setosa vs. others: $\mathbf{c}_1 = (1, -\frac{1}{2}, -\frac{1}{2})$
- Versicolor vs. Virginica: : $\mathbf{c}_1 = (0, 1, -1)$

You can do this by putting these vectors as columns in a matrix and assigning this to the `contrasts()` of `Species`. It is important to do this *before* fitting with `lm()`, because the contrasts in effect determine how the `X` matrix is setup, and hence the names of the coefficients representing `Species`.

```
C <- matrix(c(1,-1/2,-1/2,
             0,   1,  -1), nrow=3, ncol=2)
contrasts(iris$Species) <- C
contrasts(iris$Species)
#>           [,1] [,2]
#> setosa      1.0   0
#> versicolor -0.5   1
#> virginica  -0.5  -1
```

Now let's fit the model. As you would expect from Figure 11.4, the differences among groups are highly significant.

```
iris.mod <- lm(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~
                 Species, data=iris)
Anova(iris.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> Species  2     1.19      53.5     8    290 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As a quick follow-up, it is useful to examine the univariate tests for each of the iris variables, using `heplots::glance()` or `heplots::uniStats()`. It is of interest that the univariate R^2 values are much larger for the petal variables than the sepal length and width.⁴ For comparison, `heplots::etasq()` gives the overall η^2 proportion of variance accounted for in all responses.

```
glance(iris.mod)
#> # A tibble: 4 x 8
#>   response    r.squared sigma fstatistic numdf dendf p.value   nobs
#>   <chr>        <dbl> <dbl>       <dbl> <dbl> <dbl> <dbl> <int>
#> 1 Sepal.Length  0.619  0.515      119.    2     147 1.67e-31   150
#> 2 Sepal.Width   0.401  0.340       49.2    2     147 4.49e-17   150
#> 3 Petal.Length  0.941  0.430      1180.   2     147 2.86e-91   150
#> 4 Petal.Width   0.929  0.205      960.    2     147 4.17e-85   150

etasq(iris.mod)
#>      eta^2
#> Species 0.596
```

But these statistics don't help to understand *how* the species differ. For this, we turn to HE plots.

⁴Recall that R^2 for a linear model is the the proportion of variation in the response that is explained by the model, calculated as $R^2 = SS_H/SS_T = SS_H/(SS_H + SSE)$. For a multivariate model, these are obtained from the diagonal elements of \mathbf{H} and \mathbf{E} .

11.3 HE plots

The `heplot()` function takes a "mle" object and produces an HE plot for one pair of variables specified by the `variables` argument. By default, it plots the first two. Figure 11.5 shows the HE plots for the two sepal and the two petal variables.

```
heplot(iris.mod, size = "effect",
       cex = 1.5, cex.lab = 1.5,
       fill = TRUE, fill.alpha = c(0.3, 0.1))
heplot(iris.mod, size = "effect", variables = 3:4,
       cex = 1.5, cex.lab = 1.5,
       fill = TRUE, fill.alpha = c(0.3, 0.1))
```

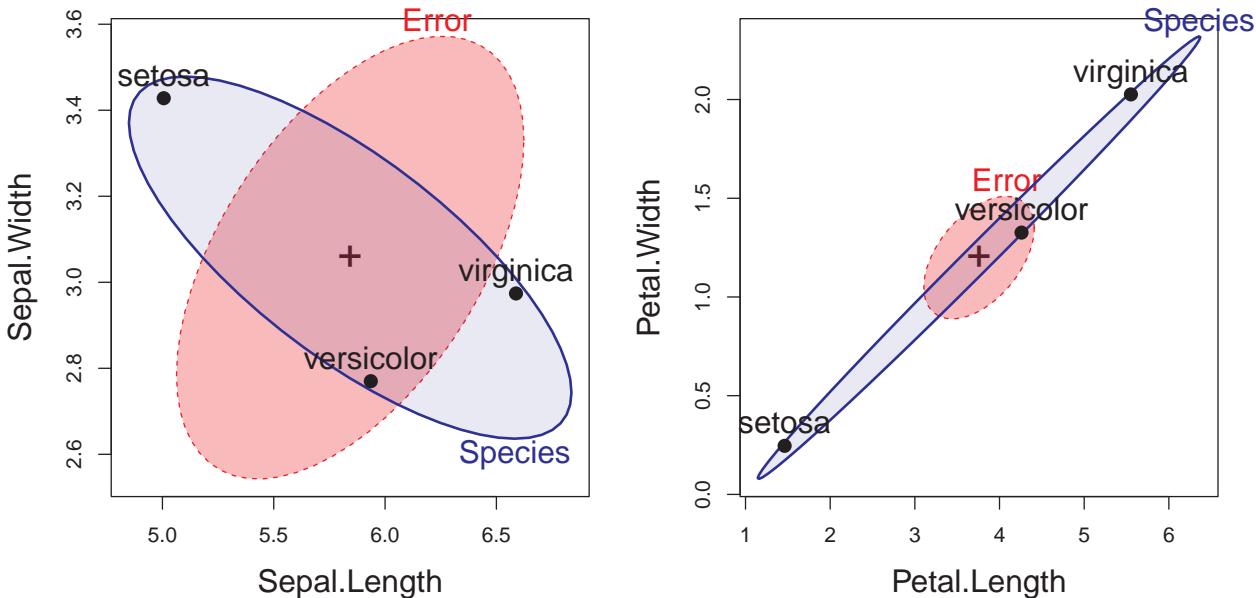


Figure 11.5: HE plots for the multivariate model `iris.mod`. The left panel shows the plot for the Sepal variables; the right panel plots the Petal variables.

The interpretation of the plots in Figure 11.5 is as follows:

- For the Sepal variables, length and width are positively correlated *within* species (the **E** = “Error” ellipsoid). The means of the groups (the **H** = “Species” ellipsoid), however, are negatively correlated. This plot is the HE plot representation of the data shown in row 2, column 1 of Figure 11.4. It reflects the relative **shape** of the iris sepals: shorter and wider for *setosa* than the other two species.
- For the Petal variables length and width are again positively correlated *within* species, but now the means of the groups are positively correlated: longer petals go with wider ones across species. This reflects the relative **size** of the iris petals. The analogous data plot appears in row 4, column 3 of Figure 11.4.

11.4 Significance scaling

The geometry of ellipsoids and multivariate tests allow us to go further with another re-scaling of the \mathbf{H} ellipsoid that gives a *visual test of significance* for any term in a MLM. This is done simply by dividing \mathbf{H}/df_e further by the α -critical value of the corresponding test statistic to show the strength of evidence against the null hypothesis.

Among the various multivariate test statistics, Roy's maximum root test, based on the largest eigenvalue λ_1 of \mathbf{HE}^{-1} , gives $\mathbf{H}/(\lambda_\alpha df_e)$ which has the visual property that the scaled \mathbf{H} ellipsoid will protrude *somewhere* outside the standard \mathbf{E} ellipsoid if and only if Roy's test is significant at significance level α . The critical value λ_α for Roy's test is

$$\lambda_\alpha = \left(\frac{df_1}{df_2} \right) F_{df_1, df_2}^{1-\alpha},$$

where $df_1 = \max(p, df_h)$ and $df_2 = df_h + df_e - df_1$.

For these data, the HE plot using significance scaling is shown in the right panel of Figure 11.6. The left panel is the same as that shown for sepal width and length in Figure 11.5, but with axis limits to make the two plots directly comparable.

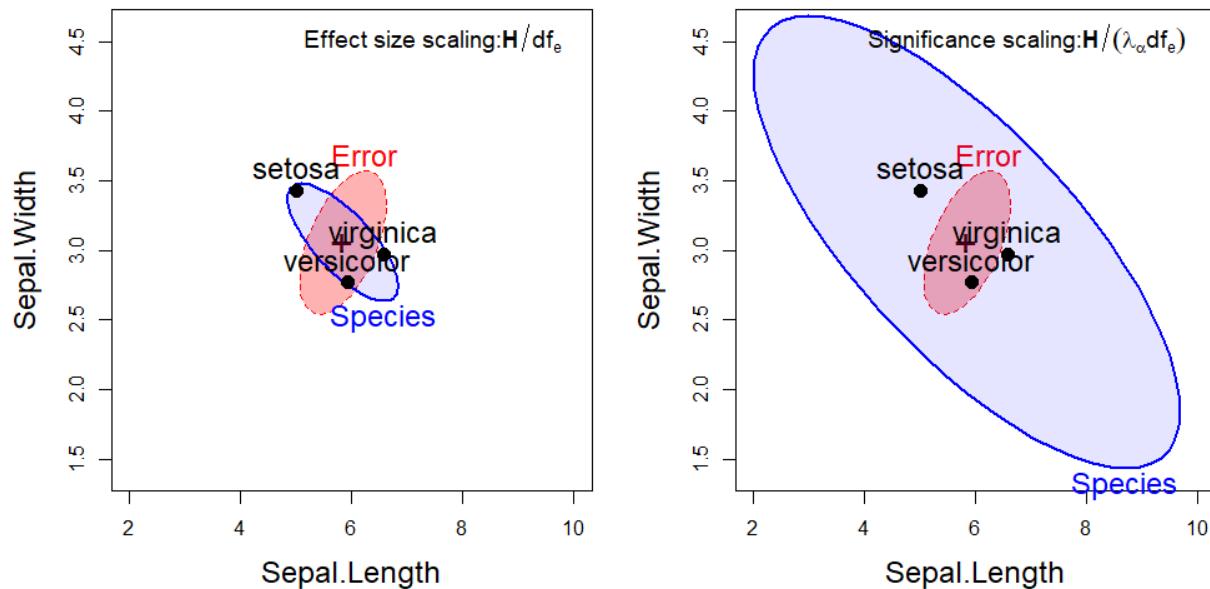


Figure 11.6: HE plots for sepal width and sepal length in the iris dataset. Left: *effect* scaling of the \mathbf{H} matrix; right: *significance* scaling, where protrusion of \mathbf{H} outside \mathbf{E} indicates a significant effect by Roy's test.

You can interpret the plot using effect scaling to indicate that the overall “size” of variation of the group means is roughly the same as that of within-group variation for the two sepal variables. Significance scaling weights the evidence against the null hypothesis that a given effect is zero. Clearly, the species vary significantly on the sepal variables, and the direction of the \mathbf{H} ellipse suggests that those whose sepals are longer are also less wide.

11.5 Visualizing contrasts and linear hypotheses

As described in Section 5.1.3, tests of linear hypotheses and contrasts represented by the general linear test $\mathcal{H}_0 : \mathbf{C} \mathbf{B} = \mathbf{0}$ provide a powerful way to probe the *specific* effects represented within the global null hypothesis, $\mathcal{H}_0 : \mathbf{B} = \mathbf{0}$, that all effects are zero.

In this example the contrasts \mathbf{c}_1 (`Species1`) and \mathbf{c}_2 (`Species2`) among the iris species are orthogonal, i.e., $\mathbf{c}_1^\top \mathbf{c}_2 = 0$. Therefore, their tests are statistically independent, and their \mathbf{H} matrices are additive. They fully decompose the general question of differences among the groups into two independent questions regarding the contrasts.

$$\mathbf{H}_{\text{Species}} = \mathbf{H}_{\text{Species1}} + \mathbf{H}_{\text{Species2}} \quad (11.2)$$

`car::linearHypothesis()` is the means for testing these statistically, and `heplot()` provides the way to show these tests visually. Using the contrasts set up in Section 11.2.1, \mathbf{c}_1 , representing the difference between `setosa` and the other species is labeled `Species1` and the comparison of `versicolor` with `virginica` is `Species2`. The coefficients for these in \mathbf{B} give the differences in the means. The line for `(Intercept)` gives grand means of the variables.

```
coef(iris.mod)
#>           Sepal.Length Sepal.Width Petal.Length Petal.Width
#> (Intercept)      5.843       3.057      3.758       1.199
#> Species1        -0.837       0.371     -2.296      -0.953
#> Species2        -0.326      -0.102      -0.646      -0.350
```

Numerical tests of hypotheses using `linearHypothesis()` can be specified in a very general way: A matrix (or vector) \mathbf{C} giving linear combinations of coefficients by rows, or a character vector giving the hypothesis in symbolic form. A character variable or vector tests whether the named coefficients are different from zero for all responses.

```
linearHypothesis(iris.mod, "Species1") |> print(SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.97    1064     4   144 <2e-16 ***
#> Wilks       1   0.03    1064     4   144 <2e-16 ***
#> Hotelling-Lawley 1   29.55   1064     4   144 <2e-16 ***
#> Roy         1   29.55   1064     4   144 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

linearHypothesis(iris.mod, "Species2") |> print(SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.745    105     4   144 <2e-16 ***
#> Wilks       1   0.255    105     4   144 <2e-16 ***
#> Hotelling-Lawley 1   2.925   105     4   144 <2e-16 ***
#> Roy         1   2.925   105     4   144 <2e-16 ***
```

The various test statistics are all equivalent here—they give the same F statistics—because they have 1 degree of freedom.

In passing, from Equation 11.2, note that the *joint* test of these contrasts is exactly equivalent to the overall test of **Species** (results not shown).

```
linearHypothesis(iris.mod, c("Species1", "Species2"))
```

We can show these contrasts in an HE plot by supplying a named list for the **hypotheses** argument. The names are used as labels in the plot. In the case of a 1-df multivariate test, the **H** ellipses plot as a degenerate line.

```
hyp <- list("S:Vv" = "Species1", "V:v" = "Species2")
heplot(iris.mod, hypotheses=hyp,
       cex = 1.5, cex.lab = 1.5,
       fill = TRUE, fill.alpha = c(0.3, 0.1),
       col = c("red", "blue", "darkgreen", "darkgreen"),
       lty = c(0,0,1,1), label.pos = c(3, 1, 2, 1),
       xlim = c(2, 10), ylim = c(1.4, 4.6))
```

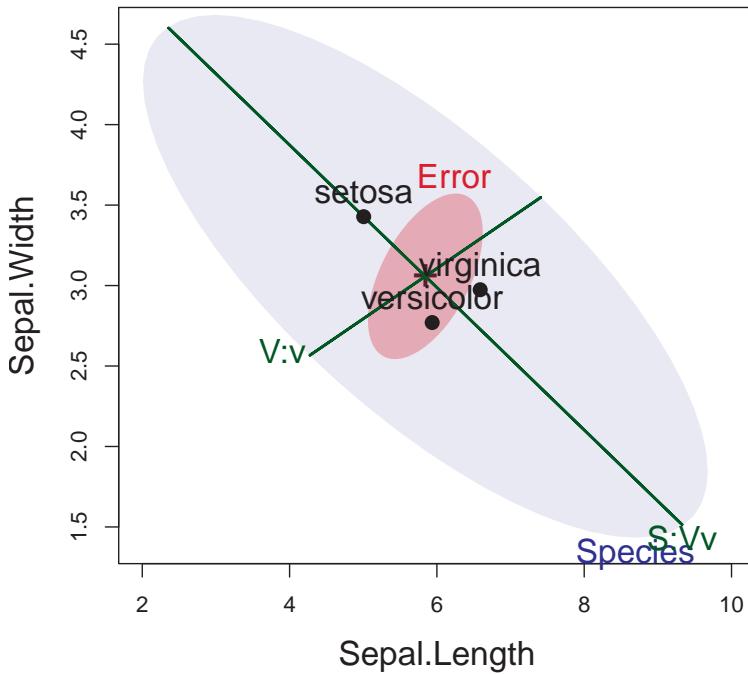


Figure 11.7: HE plot for sepal length and width in the `iris` data showing the tests of the two contrasts, using significance scaling.

This HE plot shows that, for the two sepal variables, the greatest between-species variation is accounted for by the contrast (**S:Vv**) between *setosa* and the others, for which the effect is very large in relation to error variation. The second contrast (**V:v**), between the *versicolor* and *virginica* species is relatively smaller, but still explains significant variation of the sepal variables among the species.

The directions of these hypotheses in a given plot show *how* the group means differ in terms of a given

contrast.⁵ For example, the contrast $\mathbf{S}:\mathbf{Vv}$ is the line that separates *setosa* from the others and indicates that *setosa* flowers have shorter but wider sepals.

11.6 HE plot matrices

In base R graphics, 2D scatterplots are extended to all pairwise views of multivariate data with a `pairs()` method. For multivariate linear models, the `heplots` defines a `pairs.mlm()` method to display HE plots for all pairs of the response variables.

```
pairs(iris.mod,
      fill=TRUE, fill.alpha=c(0.3, 0.1))
```

Figure 11.8 provides a fairly complete visualization of the results of the multivariate tests and answers the question: **how** do the species differ? Sepal length and the two petal variables have the group means nearly perfectly correlated, in the order *setosa* < *versicolor* < *virginica*. For Sepal width, however, *setosa* has the largest mean, and so the **H** ellipses show a negative correlation in the second row and column.

11.7 Low-D views: Canonical analysis

The HE plot framework so far provides views of all the effects in a MLM in *variable* space. We can view this in 2D for selected pairs of response variables, or for all pairwise views in scatterplot matrix format, as in Figure 11.8.

There is also an `heplot3d()` function giving plots for three response variables together. The 3D plots are interactive, in that they can be rotated and zoomed by mouse control, and dynamic, in that they can be made to spin and saved as movies. To save space, these plots are not shown here.

However in a one-way MANOVA design with more than response three variables, it is difficult to visualize how the groups vary on *all* responses together, and how the different variables contribute to discrimination among groups. In this situation, **canonical discriminant analysis** (CDA) is often used, to provide a low-D visualization of between-group variation.

When the predictors are also continuous, the analogous term is **canonical correlation analysis** (CCA). The advantage in both cases is that we can also show the relations of the response variables to these dimensions, similar to a biplot (Section 4.3) for a PCA of purely quantitative variables.

The key to this is the eigenvalue decomposition, $\mathbf{H}\mathbf{E}^{-1}\lambda_i = \lambda_i\mathbf{v}_i$ (Equation 10.7) of **H** relative to **E**. The eigenvalues, λ_i , give the “size” of each s orthogonal dimensions on which the multivariate tests are based (Section 10.2.3). But the corresponding eigenvectors, \mathbf{v}_i , give the *weights* for the response variables in s linear combinations that maximally discriminate among the groups or equivalently maximize the (canonical) R^2 of a linear combination of the predictor **X**s with a linear combination of the response **Y**s.

Thus, CDA amounts to a transformation of the p responses $\mathbf{Y}_{n \times p}$ into scores \mathbf{Z} in the canonical space, $\mathbf{Z}_{n \times s} = \mathbf{Y} \mathbf{E}^{-1/2} \mathbf{V}$,

where **V** contains the eigenvectors of $\mathbf{H}\mathbf{E}^{-1}$ and $s = \min(p, df_h)$ dimensions, the degrees of freedom for the hypothesis. It is well-known (e.g., Gittins (1985)) that *canonical discriminant plots* of the first two (or three,

⁵That the **H** ellipses for the contrasts subtend that for the overall test of **Species** is no accident. In fact, this is true in p -dimensional space for *any* linear hypothesis, and orthogonal contrasts have the additional geometric property that they form *conjugate axes* for the overall **H** ellipsoid relative to the **E** ellipsoid (Friendly et al., 2013).

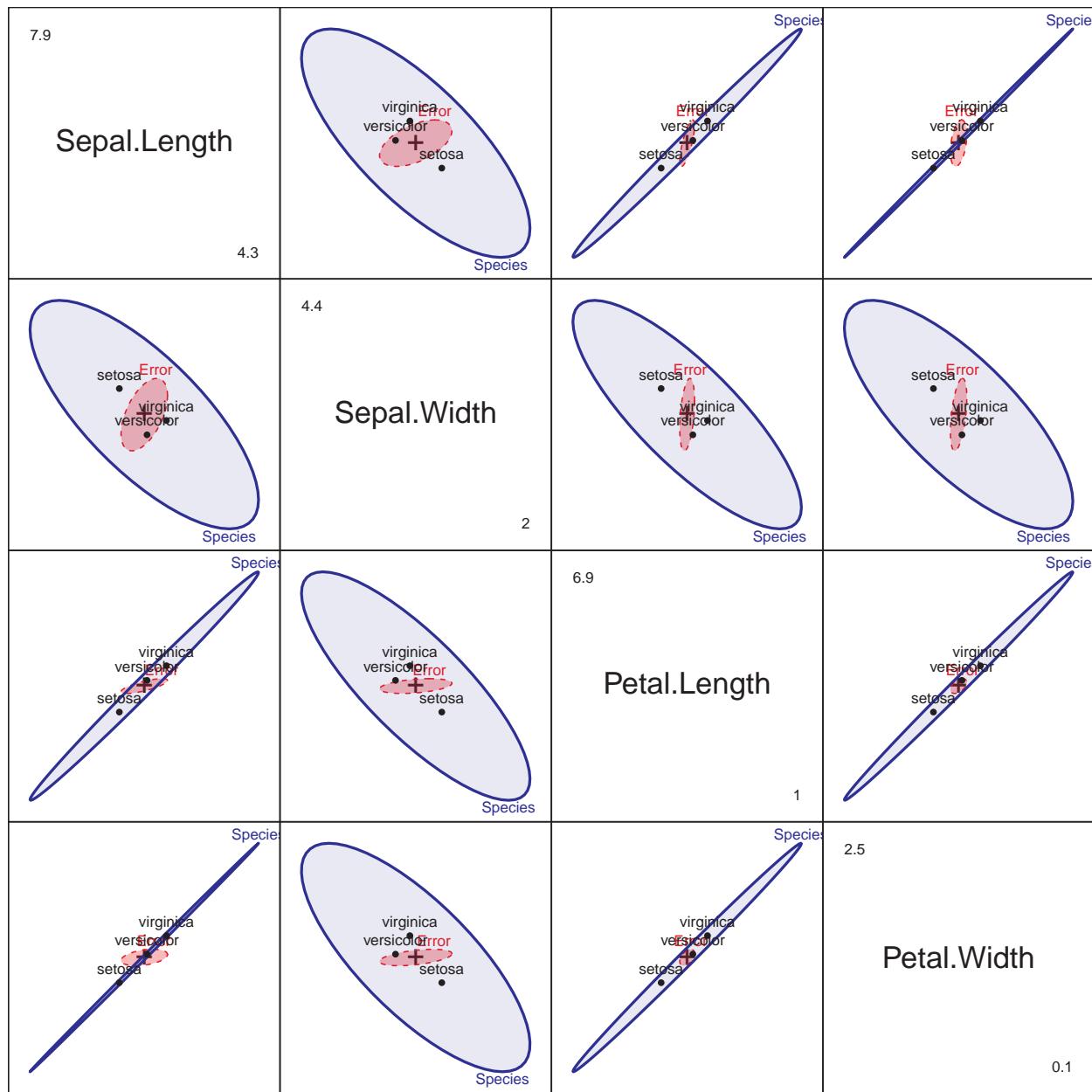


Figure 11.8: All pairwise HE plots for the iris data.

in 3D) columns of \mathbf{Z} corresponding to the largest canonical correlations provide an optimal low-D display of the variation between groups relative to variation within groups.

Canonical discriminant analysis is typically carried out in conjunction with a one-way MANOVA design. The `candisc` package (Friendly & Fox, 2025) generalizes this to multi-factor designs in the `candisc()` function. For any given term in a "mlm", the *generalized canonical discriminant analysis* amounts to a standard discriminant analysis based on the \mathbf{H} matrix for *that* term in relation to the full-model \mathbf{E} matrix.⁶

Tests based on the eigenvalues λ_i , initially stated by Bartlett (1938), use Wilks' Λ likelihood ratio tests of

⁶The related `candiscList()` function performs a generalized canonical discriminant analysis for *all* terms in a multivariate linear model, returning a list of the results for each factor.

these. This allow you to determine the number of significant canonical dimensions, or the number of different aspects to consider for the relations between the responses and predictors. This is a big win over univariate analyses for each dependent variable separately as follow-ups for a significant MANOVA result.

These take the form of *sequential* global tests of the hypothesis that the canonical correlation in the current row and all that follow are zero. Thus, if you find The canonical R^2 , `CanRsq`, gives the R-squared value of fitting the i th response canonical variate to the corresponding i th canonical variate for the predictors.

For the iris data, we get the following printed summary:

```
iris.can <- candisc(iris.mod) |> print()
#>
#> Canonical Discriminant Analysis for Species:
#>
#>   CanRsq Eigenvalue Difference Percent Cumulative
#> 1  0.970      32.192       31.9  99.121      99.1
#> 2  0.222       0.285       31.9   0.879     100.0
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>   LR test stat approx F numDF denDF Pr(> F)
#> 1       0.023     199.1      8    288 < 2e-16 ***
#> 2       0.778     13.8      3    145 5.8e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This analysis shows a very simple result: The differences among the iris species can be nearly entirely accounted for by the first canonical dimension (99.1%). Interestingly, the second dimension is also highly significant, even though it accounts for only 0.88%.

11.7.1 Coeficients

The `coef()` method for “`candisc`” objects returns a matrix of weights for the response variables in the canonical dimensions. By default, these are given for the response variables *standardized* to $\bar{y} = 0$ and $s_y^2 = 1$.

The `type` argument also allows for *raw score* weights (`type = "raw"`) used to compute scores for the observations on the canonical variables `Can1`, `Can2`, Using `type = "structure"` gives the canonical structure coefficients, which are the correlations between each response and the canonical scores.

```
coef(iris.can, type = "std")
#>           Can1   Can2
#> Sepal.Length  0.427  0.0124
#> Sepal.Width   0.521  0.7353
#> Petal.Length -0.947 -0.4010
#> Petal.Width   -0.575  0.5810
coef(iris.can, type = "structure")
#>           Can1   Can2
#> Sepal.Length -0.792  0.218
#> Sepal.Width   0.531  0.758
#> Petal.Length -0.985  0.046
#> Petal.Width   -0.973  0.223
```

The standardized (or raw score) weights are interpreted in terms of their signs and magnitudes, just as in

coefficient weights in a multiple regression. From the numbers, `Can1` seems to be a contrast between the sepal and petal variables. For `Can2`, sepal length doesn't matter and the result contrasts the two width variables against petal length.

I find it easier to interpret the correlations between the observed and canonical variables, given as the canonical structure coefficients. These are easily visualized as vectors in canonical space (similar to biplots for a PCA), as shown below (Figure 11.9).

11.7.2 Canonical scores plot

The `plot()` method for "candisc" objects gives a plot of these observation scores. `ellipse=TRUE` overlays this with their standard data ellipses for each species, as shown in Figure 11.9. The response variables are shown as vectors, using the structure coefficients, as in a biplot. Thus, the relative size of the projection of these vectors on the canonical axes reflects the correlation of the observed response on the canonical dimension. For ease of interpretation I flipped the sign of the first canonical dimension (`rev.axes`), so that the positive `Can1` direction corresponds to larger flowers.

```
vars <- names(iris)[1:4] |>
  stringr::str_replace("\\\\.", "\\n")
plot(iris.can,
  var.labels = vars,
  var.col = "black",
  var.lwd = 2,
  ellipse=TRUE,
  scale = 9,
  col = iris_colors,
  pch = 15:17,
  cex = 0.7, var.cex = 1.25,
  rev.axes = c(TRUE, FALSE),
  xlim = c(-10, 10),
  cex.lab = 1.5)
```

The interpretation of this plot is simple: in canonical space, variation of the means for the iris species is essentially one-dimensional (99.1% of the effect of `Species`), and this dimension corresponds to overall size of the iris flowers: The Setosas have smaller flowers.

All variables except for `Sepal.Width` are positively aligned with this axis, but the two petal variables show the greatest discrimination. The negative direction for `Sepal.Width` reflects the pattern seen in Figure 11.8, where *setosa* have wider sepals.

For the second dimension, look at the projections of the variable vectors on the `Can2` axis. All are positive, but this is dominated by `Sepal.Width`. We could call this a flower shape dimension.

11.7.3 Canonical HE plot

For a one-way design, the *canonical HE plot* is simply the HE plot of the canonical scores in the analogous MLM model that substitutes **Z** for **Y**. In effect, it is a more compact visual summary of the plot shown in Figure 11.9.

This is shown in Figure 11.10 for the `iris` data. In canonical space, the residuals are always uncorrelated, so the **E** ellipse plots as a circle. The **H** ellipse for `Species` here reflects a data ellipse for the fitted values—group means—shown as labeled points in the plot. The differences among `Species` are so large that this plot uses `size = "effect"` scaling, making the axes comparable to those in Figure 11.9.⁷

⁷If significance scaling was used the interpretation of the canonical HE plot would be the same as before: if the hypothesis ellipse extends beyond the error ellipse, then that dimension is significant.

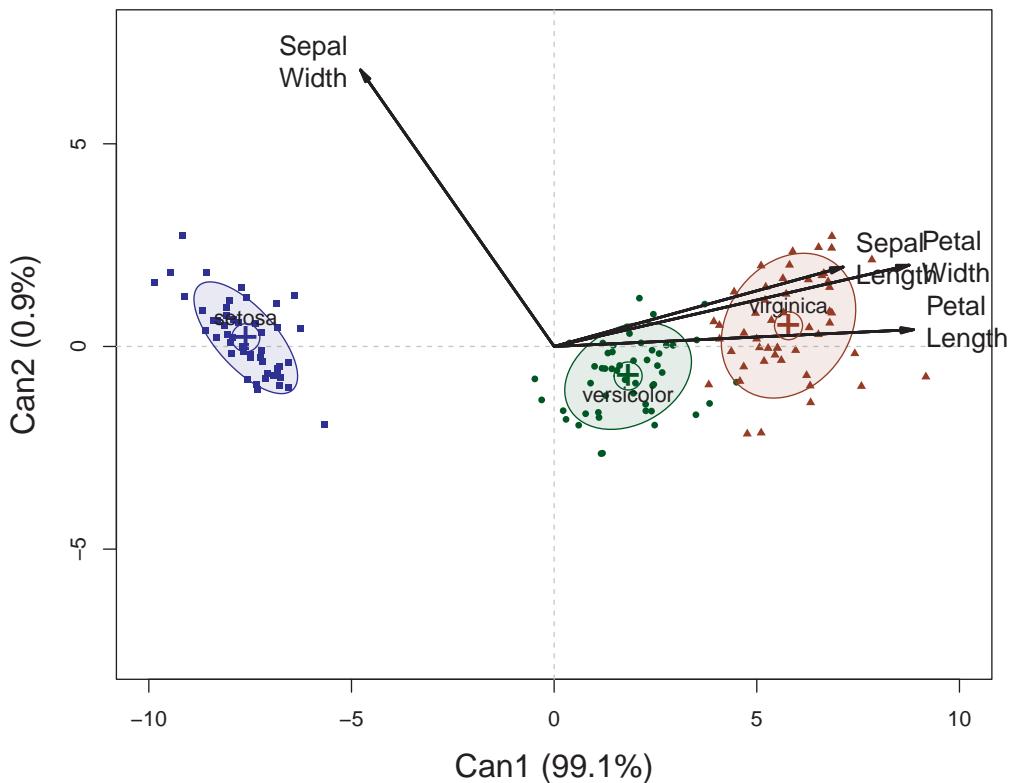


Figure 11.9: Plot of canonical scores for the iris data. Ellipses give 68% coverage data ellipses for the canonical scores. Variable vectors make angles with the Can1 and Can2 axes indicating their correlations.

The vectors for each predictor are the same structure coefficients as in the ordinary canonical plot. They can again be reflected for easier interpretation and scaled in length to fill the plot window.

```
heplot(iris.can,
      size = "effect",
      scale = 8,
      var.labels = vars,
      var.col = "black",
      var.lwd = 2,
      var.cex = 1.25,
      fill = TRUE, fill.alpha = 0.2,
      rev.axes = c(TRUE, FALSE),
      xlim = c(-10, 10),
      cex.lab = 1.5)
```

The collection of plots shown for the iris data here can be seen as progressive visual summaries of the data and increased visual understanding of

- The scatterplot matrix in Figure 11.4 shows the iris flowers in the data space of the sepal and petal variables.
- Canonical analysis substitutes for these the two linear combinations reflected in Can1 and Can2 . The plot in Figure 11.9 portrays *exactly* the same relations among the species, but in the reduced canonical space of only two dimensions.
- The HE plot version, shown in Figure 11.10 summarizes the separate data ellipses for the species with pooled, within-group variance of the \mathbf{E} matrix for the canonical variables, which are always uncorrelated. The variation among the group means is reflected in the size and shape of the ellipse for the \mathbf{E} matrix.

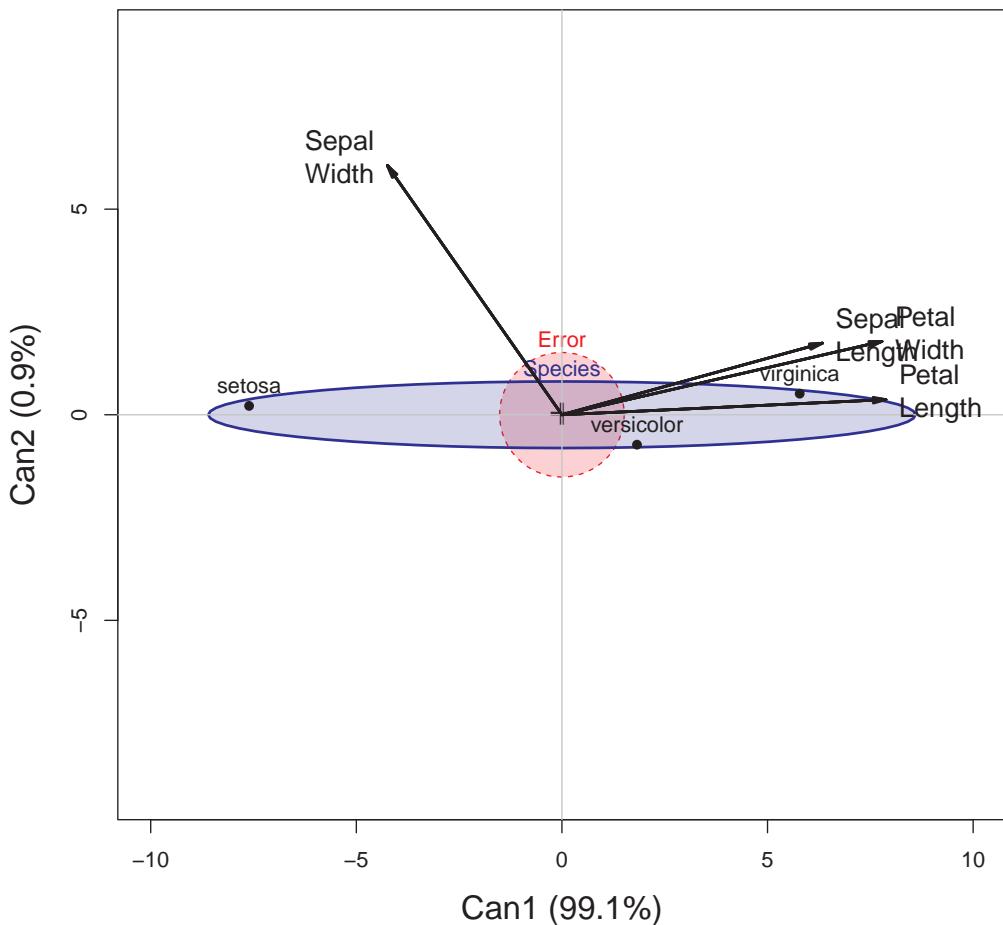


Figure 11.10: Canonical HE plot for the iris data. Compared with Figure 11.9, it substitutes canonical \mathbf{H} and \mathbf{E} ellipses for the canonical scores shown there.

11.8 Factorial MANOVA

When there are two or more factors, the overall model is comprised of main effects and possible interactions as shown in [?@eq-HE-model](#). A significant advantage of HE plots is that the main effects and interactions can be overlaid in the same plot showing how each term contributes to assessment of differences among the groups.

I illustrate this below for a simple 2×2 factorial design with three response variables.

Example 11.2. Plastic film data

An industrial experiment was conducted to determine the optimal conditions for extruding plastic film. Of interest were three responses: resistance to `tear`, film `gloss` and the `opacity` of the film. Two factors were manipulated, both at two levels, labeled High and Low: change in `rate` of extrusion (-10%, +10%) and amount of some `additive` (1%, 1.5%), with $n = 5$ runs at each combination of the factor levels. The dataset `heplots::Plastic` comes from Johnson & Wichern (1998), Example 6.12.

```

data(Plastic, package="heplots")
str(Plastic)
#> 'data.frame':   20 obs. of  5 variables:
#> $ tear    : num  6.5 6.2 5.8 6.5 6.5 6.9 7.2 6.9 6.1 6.3 ...
#> $ gloss   : num  9.5 9.9 9.6 9.6 9.2 9.1 10 9.9 9.5 9.4 ...
#> $ opacity : num  4.4 6.4 3 4.1 0.8 5.7 2 3.9 1.9 5.7 ...
#> $ rate    : Factor w/ 2 levels "Low","High": 1 1 1 1 1 1 1 1 1 ...
#> $ additive: Factor w/ 2 levels "Low","High": 1 1 1 1 1 2 2 2 2 ...

```

Multivariate tests

The MANOVA model `plastic.mod` fits the main effects of `rate` and `additive` and the interaction `rate:additive`. The `Anova()` summary shows a strong effect for `rate` of extrusions for the three responses jointly, a lesser, but still significant effect of `additive` and a non-significant interaction.

```

plastic.mod <- lm(cbind(tear, gloss, opacity) ~ rate*additive, data=Plastic)
Anova(plastic.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> rate       1    0.618    7.55      3     14   0.003 **
#> additive    1    0.477    4.26      3     14   0.025 *
#> rate:additive 1    0.223    1.34      3     14   0.302
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

As a reminder, if you want to see the results of univariate tests for the responses separately, `heplots::uniStats()` (or `heplots::glance.mlm()`) gives some answers, including R^2 and univariate F tests.

```

uniStats(plastic.mod)
#> Univariate tests for responses in the multivariate linear model plastic.mod
#>
#>          R^2    F df1 df2 Pr(>F)
#> tear    0.586 7.56   3  16 0.0023 **
#> gloss   0.483 4.99   3  16 0.0125 *
#> opacity 0.125 0.76   3  16 0.5315
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

HE plots

We can visualize *all* these effects for pairs of variables in an HE plot, showing the “size” and orientation of hypothesis variation (**H**) for each model term, in relation to error variation (**E**), as ellipsoids. When, as here, the model terms have 1 degree of freedom, the **H** ellipsoids for `rate`, `additive` and `rate:additive` each degenerate to a line.

Figure 11.11 shows the HE plot for the responses `tear` and `gloss`, the strongest by univariate tests. This plot takes advantage of another feature of `heplot()`: You can overlay plots using `add = TRUE`, as is done here to show both significance and effect size scaling in a single plot.

```

colors = c("red", "darkblue", "darkgreen", "brown4")
heplot(plastic.mod, size="significance",

```

```

col=colors, cex=1.5, cex.lab = 1.5,
fill=TRUE, fill.alpha=0.1)
heplot(plastic.mod, size="effect",
col=colors, lwd=6,
add=TRUE, term.labels=FALSE)

```

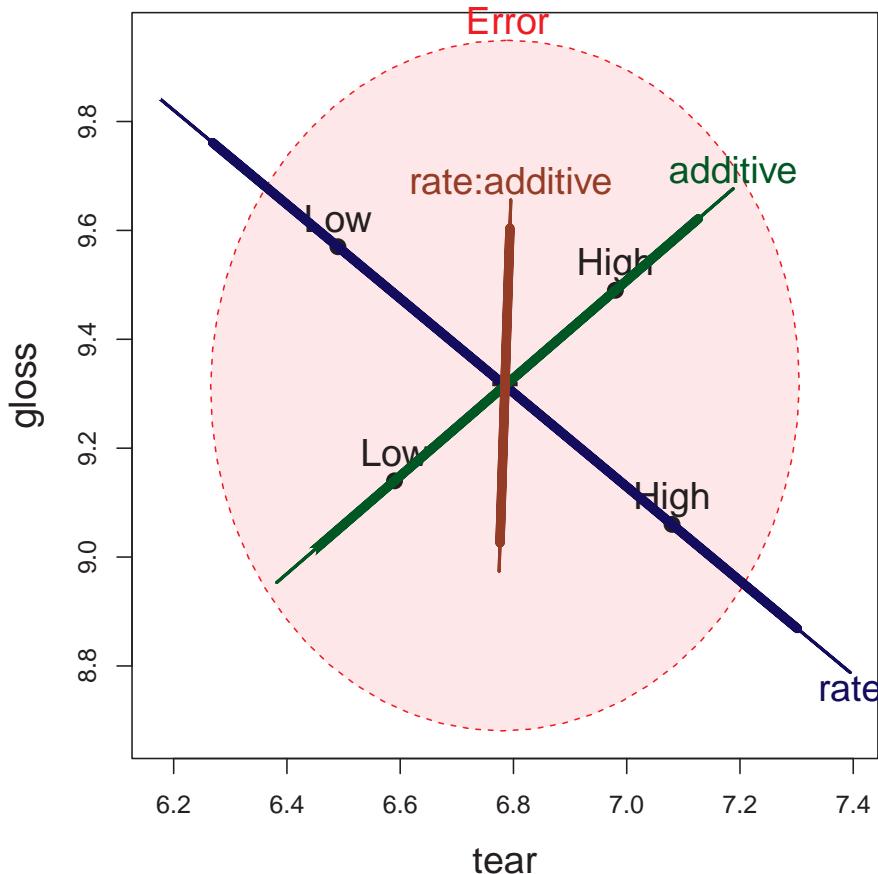


Figure 11.11: HE plot for effects on `tear` and `gloss` according to the factors `rate`, `additive` and their interaction, `rate:additive`. The thicker lines show effect size scaling; thinner lines show significance scaling.

In this view, the effect of extrusion `rate` is highly significant, with the high level giving larger tear resistance and lower gloss on average. High level of `additive` produces greater tear resistance and higher gloss. The interaction effect, `rate:additive`, while non-significant, points nearly entirely in the direction of `gloss`.

Understanding interactions

To understand an interaction effect, the simplest thing is to plot the cell means and error bars in a traditional line plot with one factor on the horizontal axis and the other as separate lines within the panel. Here (Figure 11.12) I use `ggpubr::gglne()`, which simplifies such plots, though at the expense of a bit of control for `ggplot2`.⁸

```

p1 <- gglne(Plastic,
  x = "rate", y = "tear",

```

⁸Line plots like this are nearly always more understandable with labels directly on the lines, rather than in a legend. Wrappers for `ggplot2` simplify some things but can't accommodate this kind of customization.

```

color = "additive", shape = "additive",
add = c("mean_se"),
position = "dodge",
point.size = 5, linewidth = 1.5,
ggtheme = theme_pubr(base_size = 16)
)

p2 <- ggline(Plastic,
  x = "rate", y = "gloss",
  color = "additive", shape = "additive",
  add = c("mean_se"),
  position = "dodge",
  point.size = 5, linewidth = 1.5,
  ggtheme = theme_pubr(base_size = 16)
)

p1 + p2

```

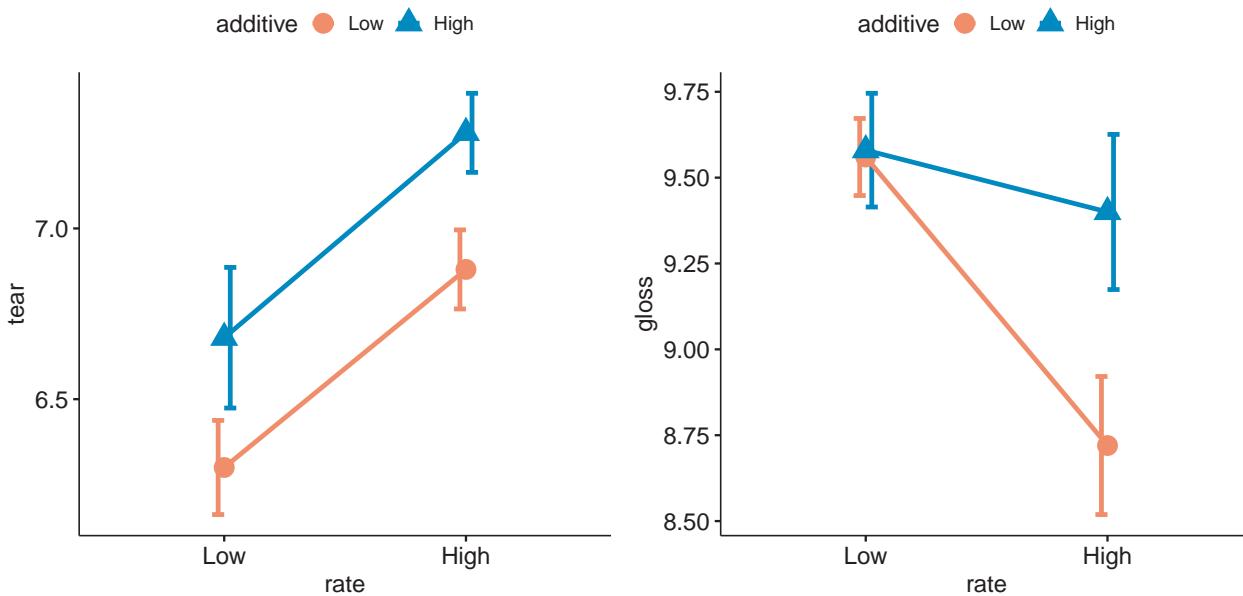


Figure 11.12: Line plots of means and their standard errors for the response `tear` (left) and `gloss` (right)"

For resistance to `tear`, the lines are parallel, so there is no interaction, which is why the `rate:additive` effect had no horizontal component in Figure 11.11. In the panel for `gloss`, the means for `additive` don't differ at high `rate`, but do substantially at the low extrusion rate.

But what if you really wanted to show the means for the combinations of rate and additive in an HE plot? By design, means for the levels of interaction terms are not shown in the HE plot, because doing so in general can lead to messy displays.

We can add them here for the term `rate:additive` as shown in Figure 11.13. This uses `heplots::termMeans()` to find the cell means for the combinations of the two factors and then `lines()` to connect the pairs of points for the low and high :levels of `additive`.

```

par(mar = c(4,4,1,1)+.1)
heplot(plastic.mod, size="evidence",
       col=colors, cex=1.5, cex.lab = 1.5,
       lwd = c(1, 5),
       fill=TRUE, fill.alpha=0.05)

## add interaction means
intMeans <- termMeans(plastic.mod, 'rate:additive',
                       abbrev.levels=3)
points(intMeans[,1], intMeans[,2], pch=18, cex=1.2, col="brown4")
text(intMeans[,1], intMeans[,2], rownames(intMeans),
     adj=c(0.5, 1), col="brown4")
lines(intMeans[c(1,3),1], intMeans[c(1,3),2],
      col="brown4", lwd = 3)
lines(intMeans[c(2,4),1], intMeans[c(2,4),2],
      col="brown4", lwd = 3)

```

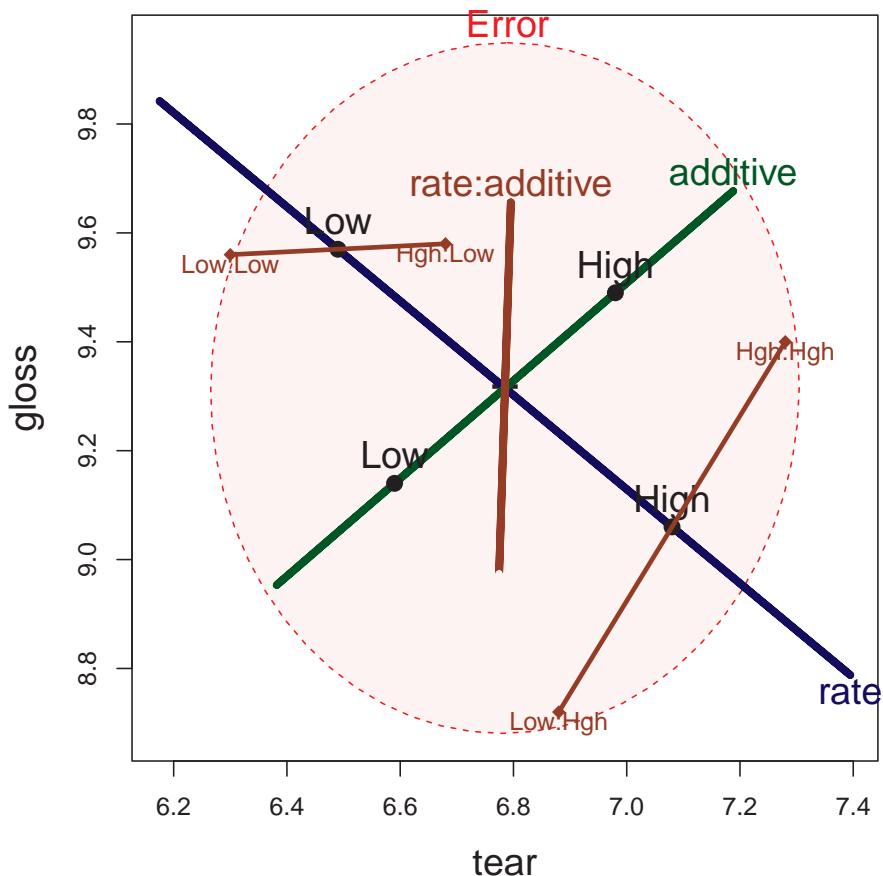


Figure 11.13: HE plot for effects on `tear` and `gloss` using significance scaling. To this is added points showing the means for the combinations of `rate` and `additive`.

11.9 Quantitative predictors: MMRA

The ideas behind HE plots extend naturally to multivariate multiple regression (MMRA). A purely visual feature of HE plots in these cases is that the **H** ellipse for a quantitative predictor with 1 df appears as a degenerate line. But consequently, the angles between these for different predictors has a simple interpretation as as the correlation between their predicted effects. Moreover, it is easy to show visual overall tests of *joint* linear hypotheses for two or more predictors together.

TODO: Use these examples below ? * `heplots::Hernior`: Recovery from Elective Herniorrhaphy -> HE_mmra vignette – Use for exercise

Example 11.3. NLSY data

Here I'll continue the analysis of the NLSY data from Section 10.5.1. In the model `NLSY.mod1`, I used only father's income and education to predict scores in reading and math, and both of these demographic variables were highly significant. Figure 11.14 shows what this looks like in an HE plot.

```
data(NLSY, package = "heplots")
NLSY.mod1 <- lm(cbind(read, math) ~ income + educ,
                 data = NLSY)

heplot(NLSY.mod1,
       fill=TRUE, fill.alpha = 0.2,
       cex = 1.5, cex.lab = 1.5,
       lwd=c(2, 3, 3),
       label.pos = c("bottom", "top", "top")
     )
```

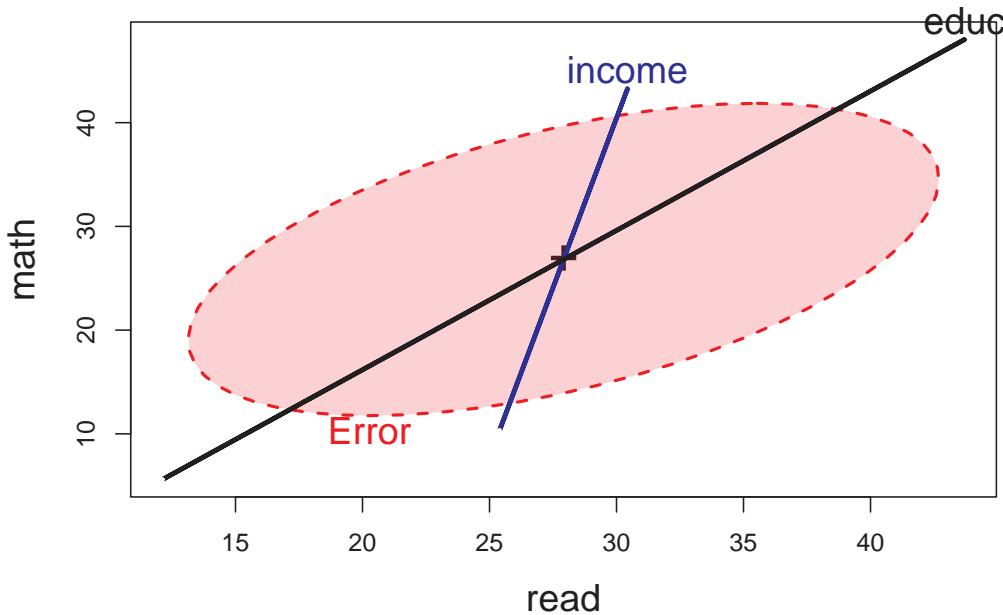


Figure 11.14: HE plot for the simple model for the NLSY data fitting reading and math scores from income and education.

Fathers income and education are positively correlated in their effects on the outcome scores. From the

angles in the plot, income is most related to the math score, while education is related to both, but slightly more to the reading score.

The overall joint test for both predictors can then be visualized as the test of the linear hypothesis $\mathcal{H}_0 : \mathbf{B} = [\beta_{\text{income}}, \beta_{\text{educ}}] = \mathbf{0}$. For `heplot()`, we specify the names of the coefficients to be tested with the `hypotheses` argument.

```
coefs <- rownames(coef(NLSY.mod1))[-1] |> print()
#> [1] "income" "educ"

heplot(NLSY.mod1,
       hypotheses = list("Overall" = coefs),
       fill=TRUE, fill.alpha = 0.2,
       cex = 1.5, cex.lab = 1.5,
       lwd=c(2, 3, 3, 2),
       label.pos = c("bottom", "top"))
```

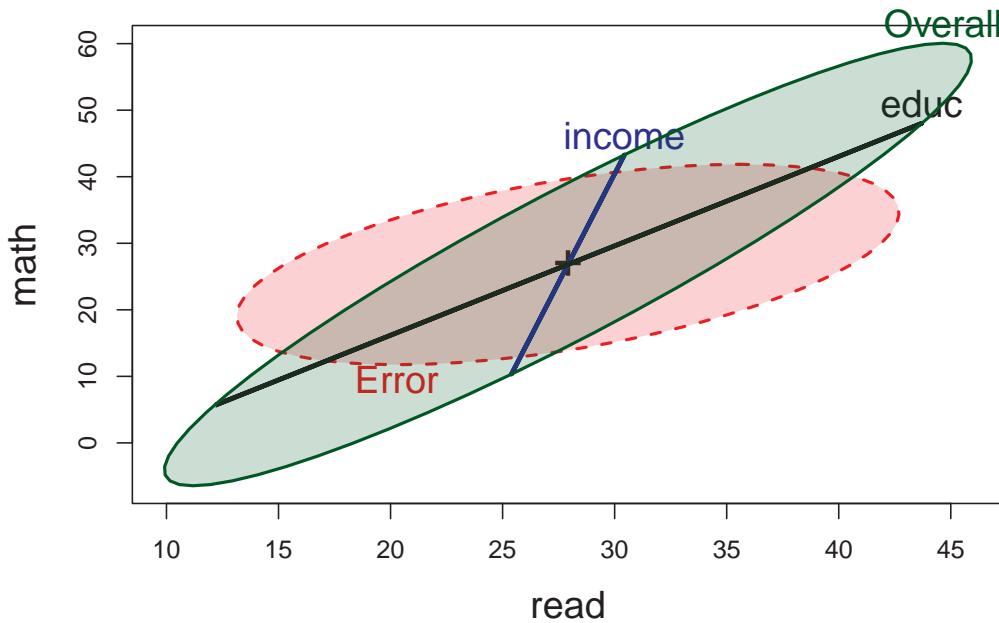


Figure 11.15: HE plot adding the **H** ellipse for the overall test that both predictors have no effect on the outcome scores.

The geometric relations of the **H** ellipses for the overall test and the individual predictors in Figure 11.15 is of worth noting here. Those for the separate coefficients always lie within the overall ellipse. The contribution for income makes the overall ellipse larger in the direction of the `math` score, while the contribution of education makes it larger in both directions.

Example 11.4. School data: HE plots

The `schooldata` dataset analyzed in Section 10.5.2 can also be illuminated by the methods of this chapter. There I fit the multivariate regression model predicting students scores on reading, mathematics and a measure of self-esteem using as predictors measures of parents' education, occupation, school visits, counseling help with school assignments and number of teachers per school.

But I also found two highly influential observations (cases 44, 59; see Figure 10.17) whose effect on the coefficients is rather large; so, I remove them from the analysis here.⁹

```
data(schooldata, package = "heplots")

bad <- c(44, 59)
OK <- (1:nrow(schooldata)) |> setdiff(bad)
school.mod2 <- lm(cbind(reading, mathematics, selfesteem) ~ .,
                   data=schooldata[OK, ])
```

In this model, parent's education and occupation and their visits to the schools were highly predictive of student's outcomes but their counseling efforts and the number of teachers in the schools did not contribute much. However, the nature of these relationships was largely uninterpreted in that analysis.

Here is where HE plots can help. You can think of this as a way to visualize what is entailed in the coefficients for this model by showing the *magnitude* of the predictor effects by their size and their relations to the outcome variable by their *direction*. The table of raw score coefficients isn't very helpful in this regard.

```
coef(school.mod2)
#>           reading mathematics selfesteem
#> (Intercept) 2.7096      3.561     0.39751
#> education    0.2233      0.132    -0.01088
#> occupation   3.3336      4.284     1.79574
#> visit        0.0101     -0.123     0.20005
#> counseling   -0.3953     -0.293     0.00868
#> teacher       -0.1945     -0.360     0.01129
```

Figure 11.16 shows the HE plot for reading and mathematics scores in this model, using the default significance scaling.

```
heplot(school.mod2,
       fill=TRUE, fill.alpha=0.1,
       cex = 1.5,
       cex.lab = 1.5,
       label.pos = c(rep("top", 4), "bottom", "bottom"))
```

Parent's occupation and education are both significant in this view, but what is more important is their orientation. Both are positively associated with reading and math scores, but education is somewhat more related to reading than to mathematics. Number of teachers and degree of parental counseling have a similar orientation, with teachers having a greater relation to mathematics scores. Visits to school and number of teachers are not significant in this plot, but both are positively correlated with reading and math and are coincident in the plot. The parent time counseling measure, while also insignificant, tilts in the opposite direction, having different signs for reading and math.

In the `pairs()` plot for all three responses (Figure 11.17), we see something different in the relations for self-esteem. While occupation has a large positive relation in all the plots in the third row and column, education, counseling and teachers have negative relations in these plots, particularly with mathematics scores.

⁹An alternative to fitting the model removing specific cases deemed troublesome is to use a `robust` method, such as `heplots::roblm()`. This uses re-weighted least squares to down-weight observations with large residuals or other problems. These methods are illustrated in Section 13.5.

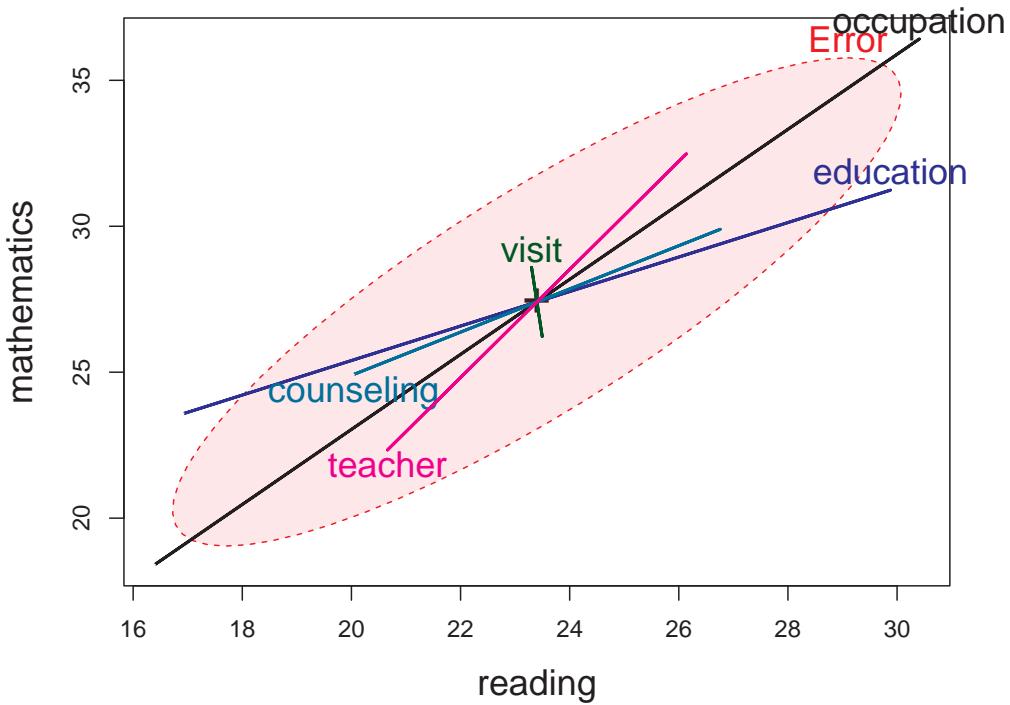


Figure 11.16: HE plot for reading and mathematics scores in the multivariate regression model...

```
pairs(school.mod2,
      fill=TRUE, fill.alpha=0.1,
      var.cex = 2.5,
      cex = 1.3)
```

The analysis of this data is continued below in Example 11.5.

11.10 Canonical correlation analysis

Just as we saw for MANOVA designs, a canonical analysis for multivariate regression involves finding a low-D view of the relations between predictors and outcomes that maximally explains their relations in terms of linear combinations of each. That is, the goal is to find weights for one set of variables, say \mathbf{X} *not* to predict each of the other set $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots]$ *individually*, but rather to also find weights for the \mathbf{y} s which is most highly correlated with the linear combination of the \mathbf{x} s.

In this sense, canonical correlation analysis (CCA) is *symmetric* in the x and y variables: the y set is not considered responses. Rather the goal is simply to explain the correlations between the two sets. For a thorough treatment of this topic, see Gittins (1985).

Geometrically, these linear combinations are vectors representing projections in the observation space of the x and y variables, and CCA can also be thought of as minimizing the angle between these vectors or maximizing the cosine of this angle. This is illustrated in Figure 11.18.

Specifically, we want to find one set of weights \mathbf{a}_1 for the x variables and another for the y variables to give the linear combinations \mathbf{u}_1 and \mathbf{v}_1 ,

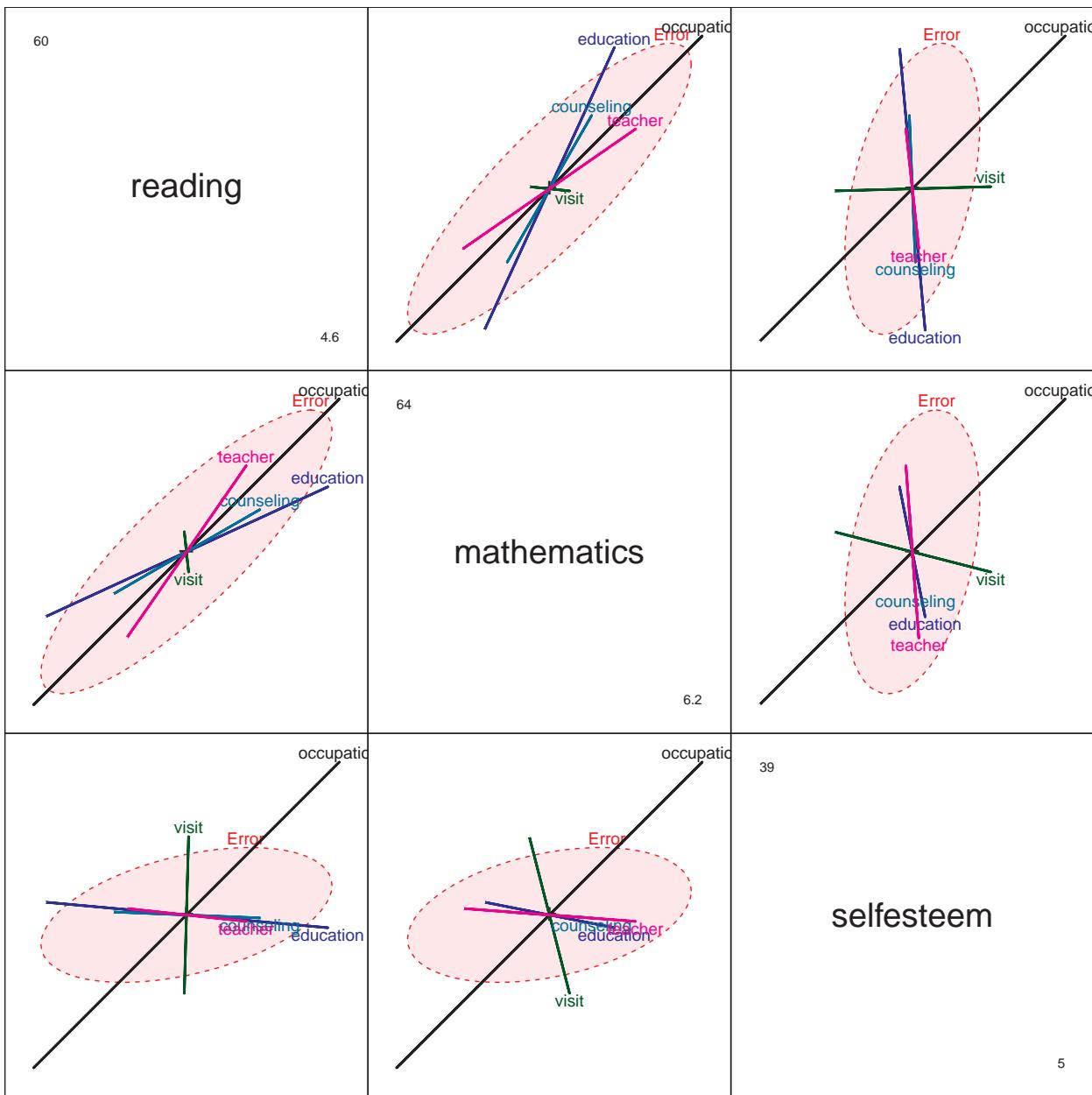


Figure 11.17: Pairwise HE plots for the three outcome variables in the multivariate regression model ...

$$\begin{aligned}\mathbf{u}_1 &= \mathbf{X} \mathbf{a}_1 = a_{11}\mathbf{x}_1 + a_{12}\mathbf{x}_2 + \cdots + a_{1q}\mathbf{x}_q \\ \mathbf{v}_1 &= \mathbf{Y} \mathbf{b}_1 = b_{11}\mathbf{y}_1 + b_{12}\mathbf{y}_2 + \cdots + b_{1p},\end{aligned}$$

such that the correlation $\rho_1 = \text{corr}(\mathbf{u}_1, \mathbf{v}_1)$ is maximized, or equivalently, minimizing the angle between them.

Using \mathbf{S}_{xx} , \mathbf{S}_{yy} to represent the covariance matrices of the x and y variables, and \mathbf{S}_{xy} for the cross-covariances between the two sets, the correlation between the linear combinations of each can be expressed as

$$\begin{aligned}\rho_1 &= \text{corr}(\mathbf{u}_1, \mathbf{v}_1) = \text{corr}(\mathbf{X} \mathbf{a}_1, \mathbf{Y} \mathbf{b}_1) \\ &= \frac{\mathbf{a}_1^\top \mathbf{S}_{xy} \mathbf{b}_1}{\sqrt{\mathbf{a}_1^\top \mathbf{S}_{xx} \mathbf{a}_1} \sqrt{\mathbf{b}_1^\top \mathbf{S}_{yy} \mathbf{b}_1}}\end{aligned}$$

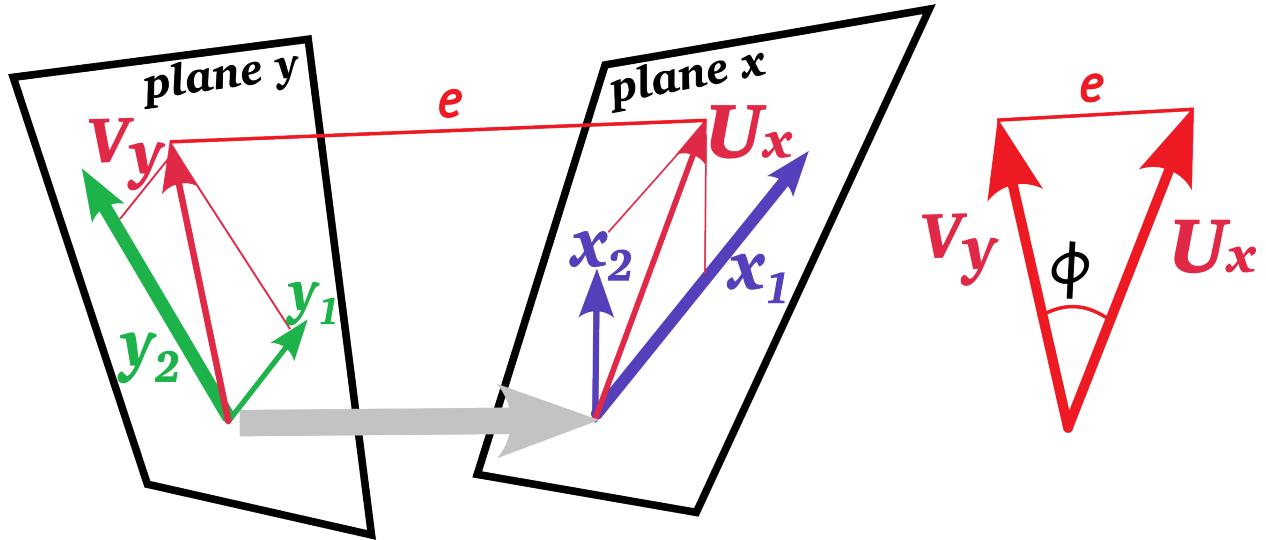


Figure 11.18: Diagram illustrating canonical correlation. For two y variables, all linear combinations are vectors in their plane, and similarly for the x variables. Maximizing the correlation between linear combinations of each is equivalent to making the angle ϕ between them as small as possible, or maximizing $\cos(\theta)$, shown in the diagram at the right. The thick grey arrow indicates that the two planes should be overlaid at a common origin. *Source:* Re-drawn by Udi Alter following a Cross-Validated discussion by user ‘ttnphns’, <https://bit.ly/4dgq2cp>

But, the y variables lie in a p -dimensional (observation) space, and the x in q dimensions, so what they have in common is a space of $s = \min(p, q)$ dimensions. Therefore, we can find additional pairs of canonical variables,

$$\begin{aligned} \mathbf{u}_2 &= \mathbf{X} \mathbf{a}_2 & \mathbf{v}_2 &= \mathbf{Y} \mathbf{b}_2 \\ &\vdots && \\ \mathbf{u}_s &= \mathbf{X} \mathbf{a}_s & \mathbf{v}_s &= \mathbf{Y} \mathbf{b}_s \end{aligned}$$

such that each pair $(\mathbf{u}_i, \mathbf{v}_i)$ has the maximum possible correlation and all distinct pairs are uncorrelated:

$$\begin{aligned} \rho_i &= \max_{\mathbf{a}_i, \mathbf{b}_i} \{ \mathbf{u}_i^\top \mathbf{v}_i \} = \\ \|\mathbf{u}_i\| &= 1, \quad \|\mathbf{v}_i\| = 1, \\ \mathbf{u}_i^\top \mathbf{u}_j &= 0, \quad \mathbf{v}_i^\top \mathbf{v}_j = 0 \quad \forall j \neq i : i, j \in \{1, 2, \dots, s\}. \end{aligned}$$

In words, the correlations among canonical variables are zero except when they are associated with the same canonical correlation or the weights $(\mathbf{a}_i, \mathbf{b}_i)$ for the same pair. Alternatively, all $p \times q$ correlations the variables in \mathbf{Y} and \mathbf{X} are fully summarized in the $s = \min(p, q)$ canonical correlations ρ_i for $i = 1, 2, \dots, s$.

The solution, developed by Hotelling (1936), is a form of a generalized eigenvalue problem, that can be stated in two equivalent ways,

$$\begin{aligned} (\mathbf{S}_{yx} \mathbf{S}_{xx}^{-1} \mathbf{S}_{xy} - \rho^2 \mathbf{S}_{yy}) \mathbf{b} &= \mathbf{0} \\ (\mathbf{S}_{xy} \mathbf{S}_{yy}^{-1} \mathbf{S}_{yx} - \rho^2 \mathbf{S}_{xx}) \mathbf{a} &= \mathbf{0}. \end{aligned}$$

Both equations have the same form and have the same eigenvalues. And, given the eigenvectors for one of these equations, we can find the eigenvectors for the other.

TODO: Fill in details of canonical correlations

Example 11.5. School data: Canonical analysis

With $p = 3$ responses and $q = 5$ predictors there are three possible sets of canonical variables which together account for 100% of the total linear relations between them. `heplots::cancor()` gives the percentage associated with each of the eigenvalues and the canonical correlations.

For this dataset, the first canonical variates, with $\text{Can } R = 0.995$, accounts for 98.6%, so you might think that that is sufficient. Yet the likelihood ratio tests show that the second set, with $\text{Can } R = 0.774$, is also significant, even though it only accounts for 1.3%.

```
# bad <- c(44, 59)
# OK <- (1:nrow(schooldata)) |> setdiff(bad)
school.can2 <- cancor(cbind(reading, mathematics, selfesteem) ~
                       education + occupation + visit + counseling + teacher,
                       data=schooldata[OK, ])
school.can2
#>
#> Canonical correlation analysis of:
#>   5   X  variables: education, occupation, visit, counseling, teacher
#>   with    3   Y  variables: reading, mathematics, selfesteem
#>
#>     CanR  CanRSQ   Eigen percent   cum
#> 1 0.9946 0.9892 91.41999 98.57540 98.58
#> 2 0.7444 0.5541  1.24267  1.33994 99.92
#> 3 0.2698 0.0728  0.07852  0.08466 100.00
#>                               scree
#> 1 ****
#> 2
#> 3
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>   CanR LR test stat approx F numDF denDF Pr(> F)
#> 1 0.995      0.004     67.5     15    166 < 2e-16 ***
#> 2 0.744      0.413      8.5      8    122 4.1e-09 ***
#> 3 0.270      0.927      1.6      3     62    0.19
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The virtue of CCA is that *all* correlations between the X and Y variables are completely captured in the correlations between the *pairs* of canonical scores: The $p \times q$ correlations between the sets are entirely represented by the $s = \min(p, q)$ canonical ones. Whether the second dimension is useful here depends on whether it adds some interpretable increment to what is going on in these relations. One could be justifiably happy with an explanation based on the first dimension that accounts for nearly all the total association between the sets.

The class "cancor" object returned by `cancor()` contains the canonical coefficients, for which there is a `coef()` method as in `candisc()`, and also a `scores()` method to return the scores on the canonical variables, called `Xcan1`, `Xcan2`, ... and `Ycan1`, `Ycan2`.

```
names(school.can2)
#> [1] "cancor"      "names"       "ndim"        "dim"         "coef"
```

```
#> [6] "scores"      "X"           "Y"           "weights"     "structure"
#> [11] "call"        "terms"
```

You can use the `plot()` method or `heplot()` method to visualize and help interpret the results. The `plot()` method plots the canonical `scores$X` against the `scores$Y` for a given dimension (selected by the `which` argument). The `id.n` argument gives a way to flag noteworthy observations.

```
plot(school.can2,
      pch=16, id.n = 3,
      cex.lab = 1.5, id.cex = 1.5,
      ellipse.args = list(fill = TRUE, fill.alpha = 0.1))
text(-2, 1.5, paste("Can R =", round(school.can2$cancor[1], 3)),
      cex = 1.4, pos = 4)

plot(school.can2, which = 2,
      pch=16, id.n = 3,
      cex.lab = 1.5, id.cex = 1.5,
      ellipse.args = list(fill = TRUE, fill.alpha = 0.1))
text(-3, 3, paste("Can R =", round(school.can2$cancor[2], 3)),
      cex = 1.4, pos = 4)
par(op)
```

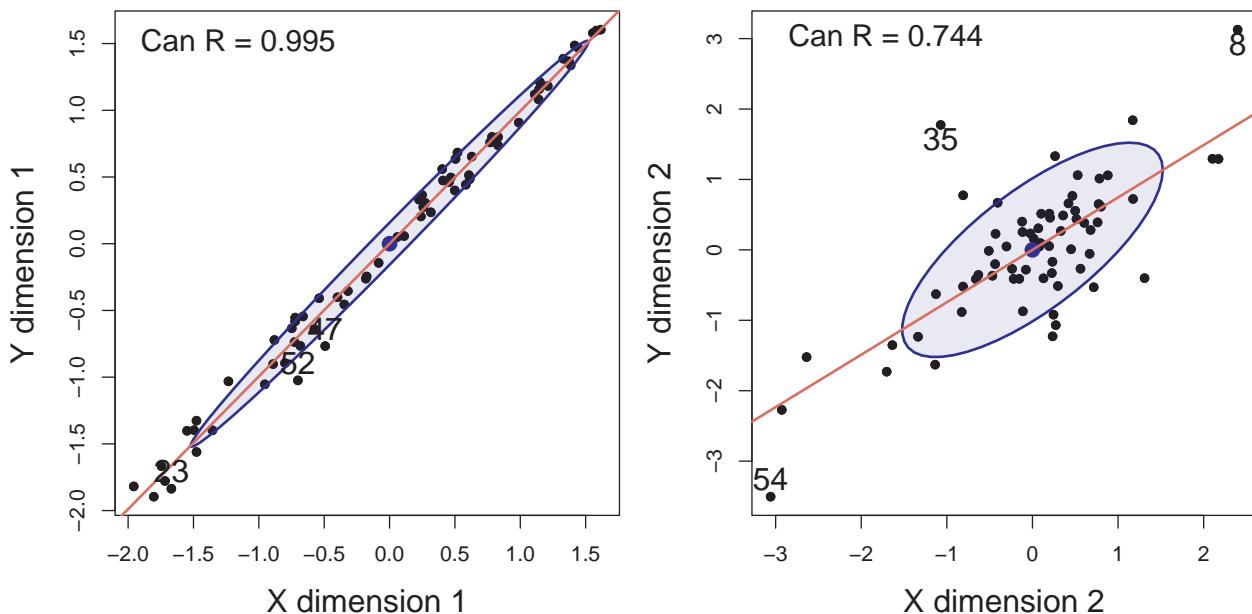


Figure 11.19: Plots of canonical scores for the first two canonical dimensions of the `schooldata` dataset, omitting the two highly influential cases.

It is worthwhile to look at an analogous plot of canonical scores for the original dataset including the two highly influential cases. As you can see in Figure 11.20, cases 44 and 59 are way outside the range of the rest of the data. Their influence increases the canonical correlation to a near perfect $\rho = 0.997$.

```
school.can <- cancor(cbind(reading, mathematics, selfesteem) ~
                      education + occupation + visit + counseling + teacher,
                      data=schooldata)
```

```
plot(school.can,
      pch=16, id.n = 3,
      cex.lab = 1.5, id.cex = 1.5,
      ellipse.args = list(fill = TRUE, fill.alpha = 0.1))
text(-5, 1, paste("Can R =", round(school.can$cancor[1], 3)),
      cex = 1.4, pos = 4)
```

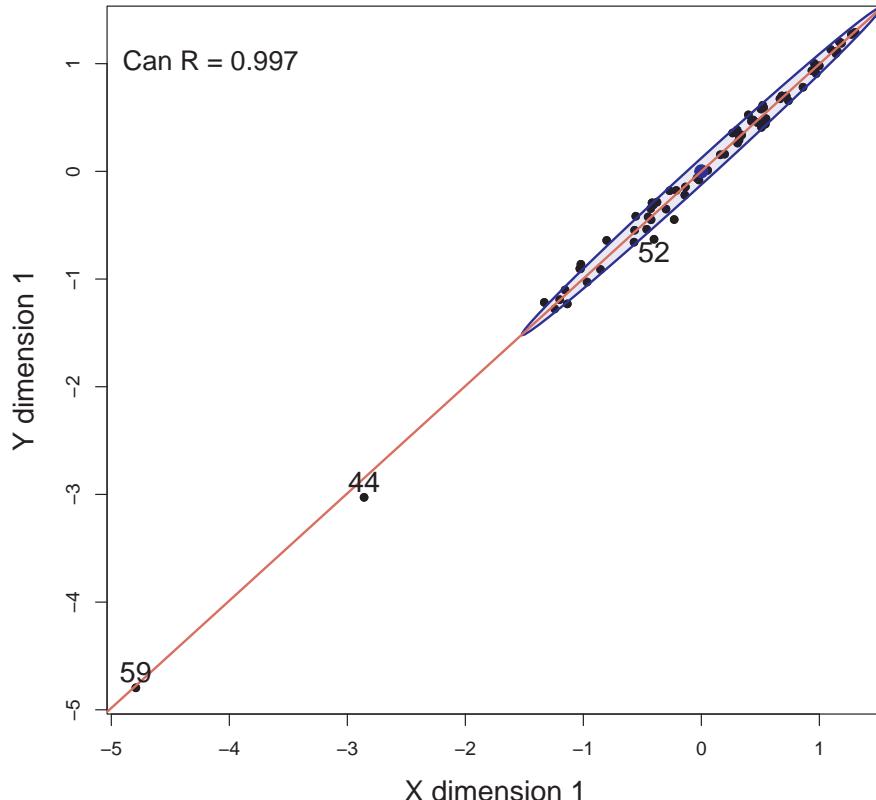


Figure 11.20: Plots of canonical scores on the first canonical dimension for the `schooldata`, including the influential cases, which stand out as so far from the rest of the observations.

Plots of canonical scores tell us of the strength of the canonical dimensions, but do not help interpreting the analysis in relation to the original variables. The HE plot version for canonical correlation analysis re-fits a multivariate regression model for the Y variables against the Xs, but substitutes the canonical scores for each, essentially projecting the data into canonical space.

TODO: Check out signs of structure coeffs from `cancor()`. Would be better to reflect the vectors for `Ycan1`.

```
heplot(school.can2,
       fill = TRUE, fill.alpha = 0.2,
       var.col = "red",
       asp = NA, scale = 0.25,
       cex.lab = 1.5, cex = 1.25,
       prefix="Y canonical dimension ")
```

The `red` variable vectors shown in these plots are intended only to show the correlations of Y variables with the canonical dimensions. The fact that they are so closely aligned reflects the fact that the first dimension

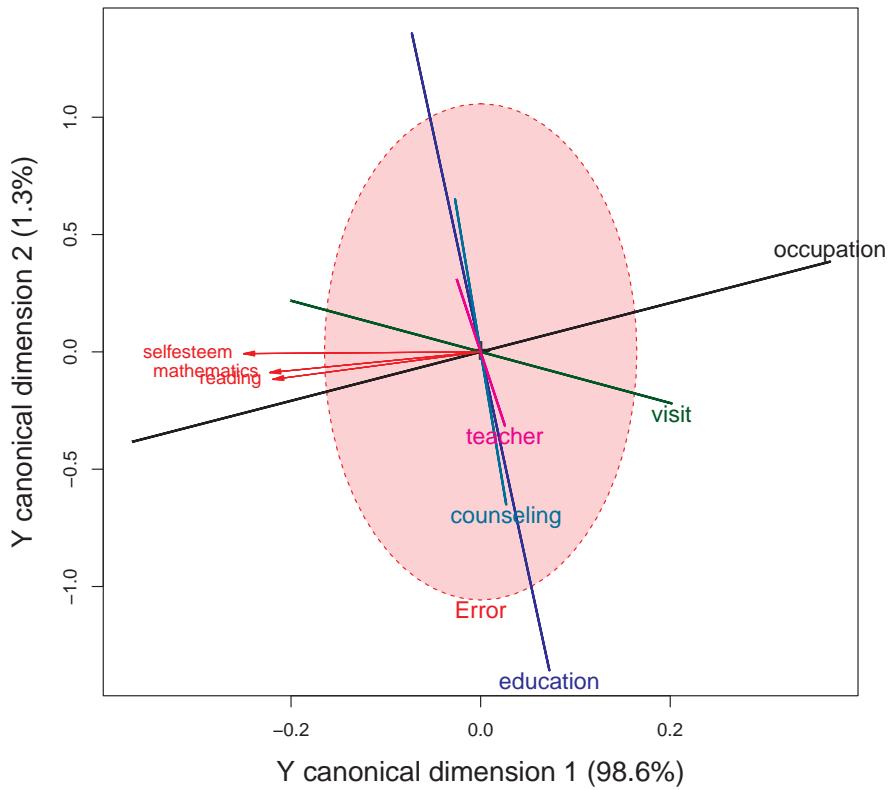


Figure 11.21: HE plot for the canonical correlation analysis of the schooldata. Vectors for the variables indicate their correlations with the canonical dimensions.

accounts for nearly all of their associations with the predictors. The orientation of the **H** ellipses/lines reflects the projection of those from Figure 11.17 into canonical space

Only their *relative lengths* and angles with respect to the Y canonical dimensions have meaning in these plots. Relative lengths correspond to proportions of variance accounted for in the Y canonical dimensions plotted; angles between the variable vectors and the canonical axes correspond to the structure correlations. The absolute lengths of these vectors are arbitrary and are typically manipulated by the `scale` argument to provide better visual resolution and labeling for the variables.

11.11 MANCOVA models

HE plots for designs containing a collection of quantitative predictors and one or more factors are quite simple in MANCOVA models where the effects are additive, i.e., don't involve interactions. They are a bit more challenging when you allow *separate* slopes for groups on all quantitative variables, because there get to be too many terms to usefully display. But these models are more complicated!

If the evidence for heterogeneity of regressions is not very strong, it is still useful to fit the MANCOVA model and display it in an HE plot.

An alternative is to fit separate models for the groups and display these as HE plots. As noted earlier (Section 10.7.1), this is not ideal for testing hypotheses, but provides a useful and informative display of the relations between the predictors and responses and the groups effect. I illustrate these approaches for the Rohwer data, encountered in Section 10.7.1, below.

Example 11.6. Rohwer data

In Section 10.7.1 I fit several models for Rohwer's data on the relations between paired-associate tasks and scholastic performance. The first model was the MANCOVA model testing the difference between the high and low SES groups, controlling for, or taking into account differences on the paired-associate task.

```
Rohwer.mod1 <- lm(cbind(SAT, PPVT, Raven) ~ SES + n + s + ns + na + ss,
                    data=Rohwer)
```

HE plots for this model for the pairs (SAT, PPVT) and (SAT, Raven) is shown in Figure 11.22. The result of an overall test for *all* predictors, $\mathcal{H}_0 : \mathbf{B} = \mathbf{0}$, is added to the basic plot using the `hypotheses` argument.

```
colors <- c("red", "blue", rep("black", 5), "#969696")
covariates <- rownames(coef(Rohwer.mod1))[-(1:2)]
pairs(Rohwer.mod1,
      col=colors,
      hypotheses=list("Regr" = covariates),
      fill = TRUE, fill.alpha = 0.1,
      cex=1.5, cex.lab = 1.5, var.cex = 3,
      lwd=c(2, rep(3,5), 4))
```

The positive effect of SES on the outcome measures is seen in all pairwise plots: the high SES group is better on all responses. The positive orientation of the `Regr` ellipses for the covariates shows that the predicted values for all three responses are positively correlated (more so for SAT and PPVT): higher performance on the paired associate tasks, in general, is associated with higher academic performance. The two significant predictors, `na` and `ns` are the only ones that extend outside the error ellipses, but their orientations differ.

Homogeneity of regression

A second model tested the assumption of homogeneity of regression by adding interactions of SES with the PA tasks, allowing separate slopes for the two groups on each of the other predictors.

```
Rohwer.mod2 <- lm(cbind(SAT, PPVT, Raven) ~ SES * (n + s + ns + na + ss),
                    data=Rohwer)
```

This model has 11 terms, excluding the intercept: `SES`, plus 5 main effects (`xs`) for the predictors and 5 interactions (slope differences), too many for an understandable display. To visualize this in an HE plot (Figure 11.23), I simplify, by showing the interaction terms *collectively* by a single ellipse, representing their joint effect, and specified as a linear hypothesis called `slopes` that picks out the interaction effects.

The argument `terms` limits the `H` ellipses for the right-hand-side of the model which are shown to just those terms specified. The combined effect of the interaction terms is specified as an hypothesis (`slopes`) testing the interaction terms (which have a ":" in their name). Because `SES` is "treatment-coded" in this model, the interaction terms reflect the difference in slopes for the high SES group compared to the low.

```
(coefs <- rownames(coef(Rohwer.mod2)))
#> [1] "(Intercept)" "SESLo"        "n"          "s"
#> [5] "ns"          "na"          "ss"         "SESLo:n"
#> [9] "SESLo:s"     "SESLo:ns"    "SESLo:na"   "SESLo:ss"

colors <- c("red", "blue", rep("black", 5), "#969696")
heplot(Rohwer.mod2, col=c(colors, "darkgreen"),
       terms=c("SES", "n", "s", "ns", "na", "ss"),
```

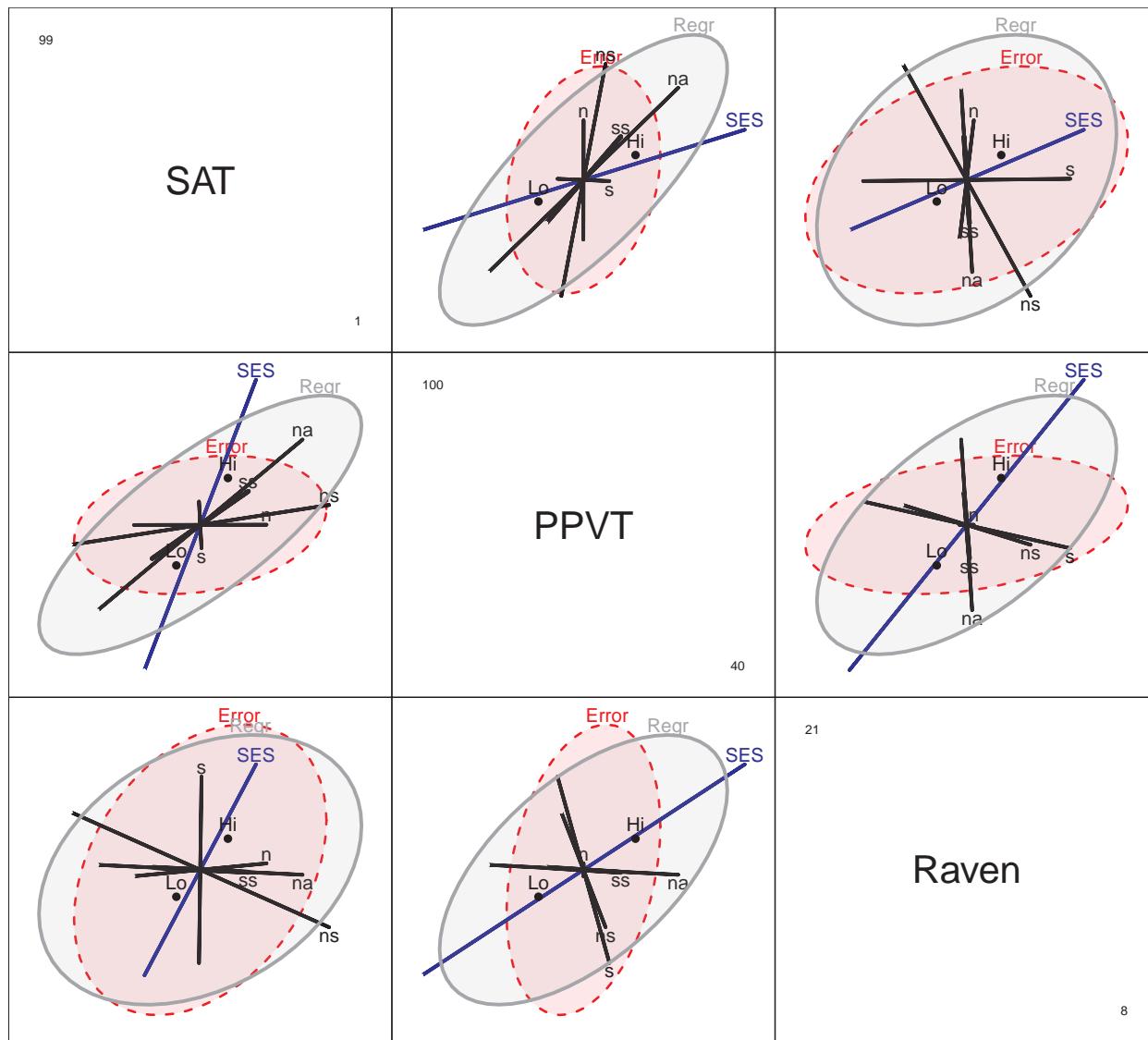


Figure 11.22: All-pairs HE plot for SAT, PPVT and Raven using the MANCOVA model. The ellipses labeled ‘Regr’ show the test of the overall effect of the quantitative predictors.

```

hypotheses=list("Regr" = c("n", "s", "ns", "na", "ss"),
                 "Slopes" = coefs[grep(":", coefs)]),
fill = TRUE, fill.alpha = 0.2, cex.lab = 1.5)

```

Separate models

When there is heterogeneity of regressions, using submodels for each of the groups has the advantage that you can easily visualize the slopes for the predictors in each of the groups, particularly if you overlay the individual HE plots. In this example, I'm using the models `Rohwer.sesLo` and `Rohwer.sesHi` fit to each of the groups.

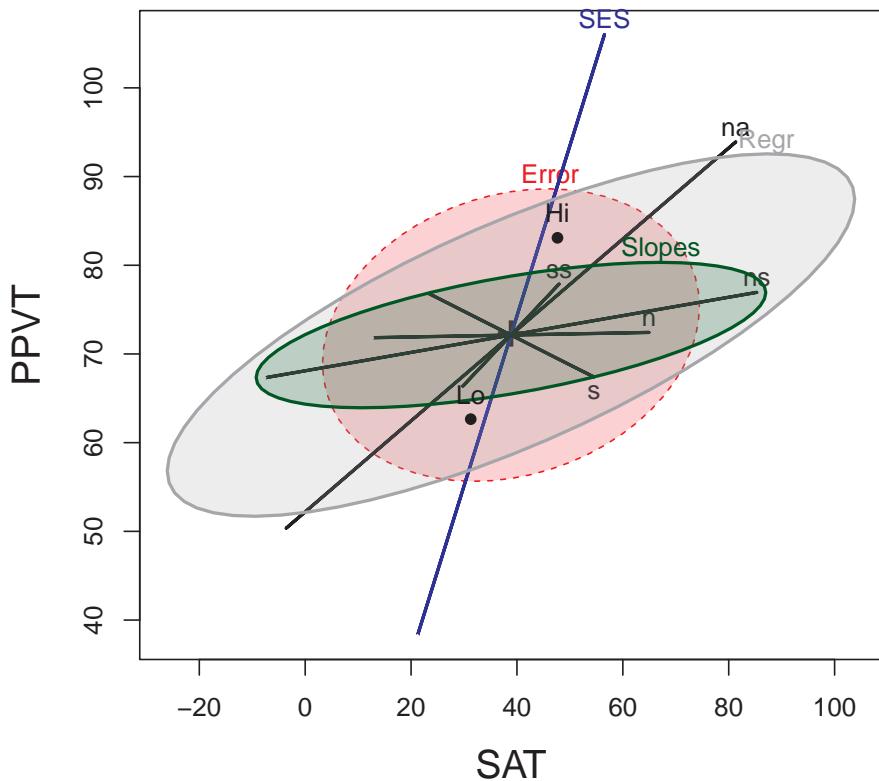


Figure 11.23: HE plot for SAT and PPVT using the heterogeneous regression model. The ellipse labeled ‘Regr’ shows the test of the covariates combined, and the ellipse labeled ‘slopes’ shows the combined difference in slopes between the two groups.

```
Rohwer.sesLo <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Lo")
Rohwer.sesHi <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Hi")
```

Here I make use of the fact that several HE plots can be overlaid using the option `add=TRUE` as shown in Figure 11.24. The axis limits may need adjustment in the first plot so that the second one will fit.

```
heplot(Rohwer.sesLo,
       xlim = c(0,100),                      # adjust axis limits
       ylim = c(40,110),
       col=c("red", "black"),
       fill = TRUE, fill.alpha = 0.1,
       lwd=2, cex=1.2, cex.lab = 1.5)
heplot(Rohwer.sesHi,
       add=TRUE,
       col=c("brown", "black"),
       grand.mean=TRUE,
       error.ellipse=TRUE,                   # not shown by default when add=TRUE
       fill = TRUE, fill.alpha = 0.1,
       lwd=2, cex=1.2)
```

We can readily see the difference in means for the two SES groups (Hi has greater scores on both variables)

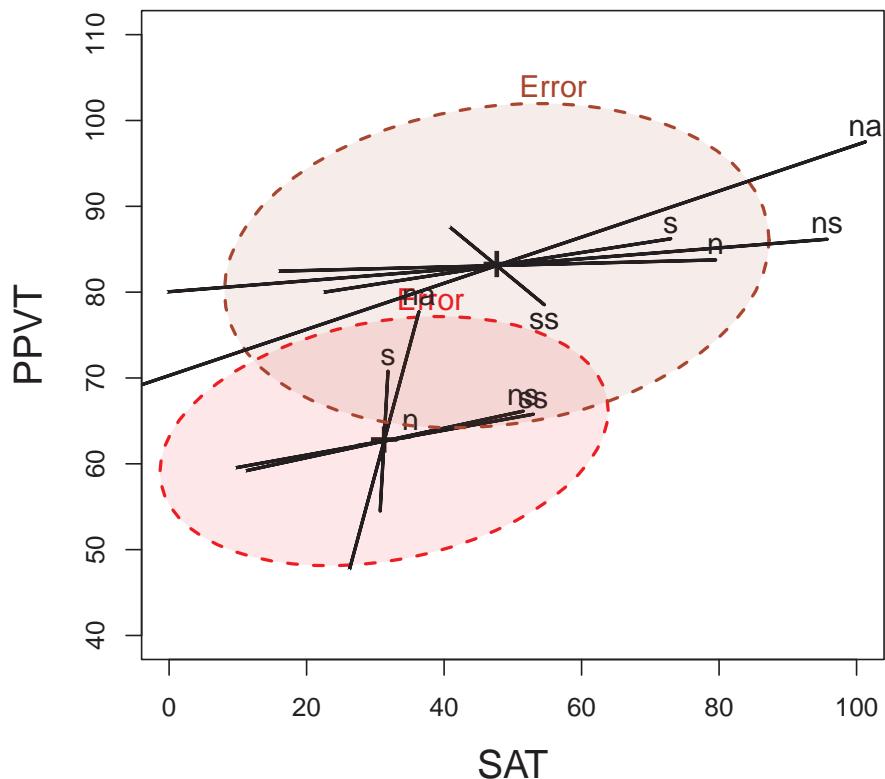


Figure 11.24: Overlaid HE plots for SAT and PPVT, for the low and high SES groups, when each group is fit separately.

and it also appears that the slopes of the **s** and **n** predictor ellipses are shallower for the High than the Low group, indicating greater relation with the SAT score. As well, the error ellipses show that on these measures, error variation is somewhat smaller in the low SES group.

11.12 What have we learned?

- **HE plots clarify complex multivariate models into enlightening visualizations** - The HE plot framework brilliantly addresses the interpretability problem of multivariate models by visualizing hypothesis (H) ellipses against error (E) ellipses. Rather than navigate a confusing maze of tables of coefficients and test statistics, with HE plots you can see which effects matter, how they relate to each other, and whether they're statistically significant. All of these benefits are given in a single, intuitive plot that reveals the geometric structure underlying your multivariate analysis.
- **Canonical space is your secret weapon for high-dimensional visualization** - When you have many response variables, canonical discriminant analysis and canonical correlation analysis project the complex multivariate relationships into a lower-dimensional space that captures the essential patterns. This isn't just dimension reduction—it's insight amplification, revealing the fundamental directions of variation that matter most while showing how your original variables contribute to these meaningful dimensions.
- **Visual hypothesis testing beats p-value hunting every time** - HE plots make hypothesis testing immediate and intuitive: if a hypothesis ellipse extends outside the error ellipse (under significance scaling), the effect is significant. No more scanning tables of p-values or wrestling with multiple comparisons—the

geometry tells the story directly. You can even decompose overall effects into meaningful contrasts and see their individual contributions as separate ellipses.

- **Ellipse orientations reveal the hidden correlational structure of your effects** - The angles between hypothesis ellipses in HE plots directly show how different predictors relate to your response variables and to each other. When effect ellipses point in similar directions, those predictors have similar multivariate signatures; when they're orthogonal, they capture independent aspects of variation. This geometric insight goes far beyond what correlation matrices can reveal.
- **Multivariate models become tractable through progressive visual summaries** - The chapter demonstrates a powerful visualization strategy: start with scatterplot matrices to see the raw data structure, move to HE plots to understand model effects, then project to canonical space for the clearest possible view of multivariate relationships. Each step preserves the essential information while making it more interpretable, turning the complexity of multivariate analysis into a comprehensible visual narrative.

12

Visualizing Equality of Covariance Matrices

To make the preliminary test on variances is rather like putting to sea in a rowing boat to find out whether conditions are sufficiently calm for an ocean liner to leave port. — G. E. P. Box (1953)

This chapter concerns the extension of tests of homogeneity of variance from the classical univariate ANOVA setting to the analogous multivariate (MANOVA) setting. Such tests are a routine but important aspect of data analysis, as particular violations can drastically impact model estimates and appropriate conclusions that can be drawn (Lix & Keselman, 1996).

Beyond issues of model assumptions, the question of equality of covariance matrices is often of general interest itself. For instance, variability is often an important issue in studies of strict equivalence in laboratories comparing across multiple patient measurements and in other applied contexts (see Gastwirth et al., 2009 for other exemplars).

Moreover the outcome of such tests often have important consequences for the details of a main method of analysis. Just as the Welsh t -test (Welch, 1947) is now commonly used and reported for a two-group test of differences in means under *unequal* variances, a preliminary test of equality of covariance matrices is often used in discriminant analysis to decide whether linear (LDA) or quadratic discriminant analysis (QDA) should be applied in a given problem. In such cases, the data at hand should inform the choice of statistical analysis to utilize.

I provide some answers to the following questions:

- **Visualization:** How can we visualize differences among group variances and covariance matrices, perhaps in a way that is analogous to what is done to visualize differences among group means? As will be illustrated (Section 12.6), differences among covariance matrices can be comprised of spread in overall size (“scatter”) and shape (“orientation”). These can be seen in data space with data ellipses, particularly if the data is centered by shifting all groups to the grand mean.
- **Low-D views:** When there are more than a few response variables, what low-dimensional views can show the most interesting properties related to the equality of covariance matrices? Projecting the data into the space of the principal components serves well again here (Section 12.7). Surprisingly, we will see that the *small dimensions* contain useful information about differences among the group covariance matrices, similar to what we find for outlier detection.
- **Other statistics:** Box’s M -test is most widely used. Are there other worthwhile test statistics? You will see that graphical methods suggest alternatives in Section 12.8. Similarly, the univariate Levene’s test has a simple multivariate generalization which can be visualized using HE plots (Section 12.9).

The following sections provide a capsule summary of the issues in this topic. Most of the discussion is couched in terms of a one-way design for simplicity, but the same ideas can apply to two-way (and higher) designs, where a “group” factor is defined as the product combination (interaction) of two or more factor variables.

When there are also numeric covariates, this topic can also be extended to the multivariate analysis of covariance (MANCOVA) setting. This is accomplished simply by applying these techniques to the *residuals* from predictions by the covariates alone.

Packages

In this chapter I use the following packages. Load them now

```
library(car)
library(helplots)
library(candisc)
library(ggplot2)
library(dplyr)
library(tidyr)
```

The main methods described here are implemented in the `helplots` package.

12.1 Homogeneity of Variance in Univariate ANOVA

In classical (Gaussian) univariate ANOVA models, the main interest is typically on tests of mean differences in a response y according to one or more factors. The validity of the typical F test, however, relies on the assumption of *homogeneity of variance*: all groups have the same (or similar) variance,

$$\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_g^2 .$$

It turns out that the F test for differences in means is relatively robust to violation of this assumption (Harwell et al., 1992), as long as the group sample sizes are roughly equal.¹ This applies to Type I error α rates, which are not much affected. However, unequal variance makes the ANOVA tests less efficient: you lose power to detect significant differences.

A variety of classical test statistics for homogeneity of variance are available, including Hartley's F_{max} (Hartley, 1950), Cochran's C (Cochran, 1941), and Bartlett's test (Bartlett, 1937), but these have been found to have terrible statistical properties (Rogan & Keselman, 1977), which prompted Box's famous quote.

Levene (1960) introduced a different form of test, based on the simple idea that when variances are equal across groups, the average *absolute values* of differences between the observations and group means will also be equal, i.e., substituting an L_1 norm for the L_2 norm of variance. In a one-way design, this is equivalent to a test of group differences in the means of the auxilliary variable $z_{ij} = |y_{ij} - \bar{y}_i|$.

More robust versions of this test were proposed by M. B. Brown & Forsythe (1974). These tests substitute the group mean by either the group **median** or a **trimmed mean** in the ANOVA of the absolute deviations. Some suggest these should be almost always preferred to Levene's version using the mean deviation. See Conover et al. (1981) for an early review and Gastwirth et al. (2009) for a general discussion of these tests. In what follows, we refer to this class of tests as "Levene-type" tests and suggest a multivariate extension described below (Section 12.2).

These deviations from a group central can be calculated using `helplots::colDevs()` and the central value can be a function, like `mean`, `median` or an anonymous one like `function(x) mean(x, trim = 0.1)`) that trims 10% off each side of the distribution. With a response `Y` Levene-type tests then be performed "by hand" as follows:

```
# Levine
Z.mean <- abs( colDevs(Y, group) )
lm(Z.mean ~ group)

# Brown-Forsythe
```

¹If group sizes are greatly unequal **and** homogeneity of variance is violated, then the F statistic is too liberal (actual rejection rate greater than the nominal α) when large sample variances are associated with small group sizes. Conversely, the F statistic is too conservative (actual $< \alpha$) if large variances are associated with large group sizes.

```
Z.med <- abs( colDevs(Y, group, median) )
lm(Z.med ~ group)
```

The function `car::leveneTest()` does this, so we could examine whether the variances are equal in the Penguin variables, one at a time, like so:

```
data(peng, package = "heplots")
leveneTest(bill_length ~ species, data=peng)
#> Levene's Test for Homogeneity of Variance (center = median)
#>          Df F value Pr(>F)
#> group     2   2.29   0.1
#>           330

leveneTest(bill_depth ~ species, data=peng)
#> Levene's Test for Homogeneity of Variance (center = median)
#>          Df F value Pr(>F)
#> group     2   1.91   0.15
#>           330

# ...

leveneTest(body_mass ~ species, data=peng)
#> Levene's Test for Homogeneity of Variance (center = median)
#>          Df F value Pr(>F)
#> group     2   5.13 0.0064 **
#>           330
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

More conveniently, `heplots:leveneTests()` with an “s”, does this for each of a set of response variables, specified in a data frame, a model formula or a "`mlm`" object. It also formats the results in a more pleasing way:

```
peng.mod <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~ species,
                 data = peng)
leveneTests(peng.mod)
#> Levene's Tests for Homogeneity of Variance (center = median)
#>
#>          df1 df2 F value Pr(>F)
#> bill_length     2 330   2.29 0.1033
#> bill_depth      2 330   1.91 0.1494
#> flipper_length  2 330   0.44 0.6426
#> body_mass       2 330   5.13 0.0064 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So, this tells us that the groups do not differ in variances on first three variables, but they do for `body_mass`.

12.2 Visualizing Levene's test

To gain some insight into the problem of homogeneity of variance it is helpful to see how the situation looks in terms of data. For the Penguin data, it might be simplest just to look at boxplots of the variables and try to see whether the **widths** of the central 50% boxes seem to be the same, as in Figure 12.1. However, it is perceptually difficult to focus on differences with widths of the boxes within each panel when their centers also differ from group to group.

```
source("R/penguin/penguin-colors.R")
col <- peng.colors("dark")
clr <- c(col, gray(.20))
peng_long <- peng |>
  pivot_longer(bill_length:body_mass,
               names_to = "variable",
               values_to = "value")

peng_long |>
  group_by(species) |>
  ggplot(aes(value, species, fill = species)) +
  geom_boxplot() +
  facet_wrap(~ variable, scales = 'free_x') +
  theme_penguins() +
  theme_bw(base_size = 14) +
  theme(legend.position = 'none')
```

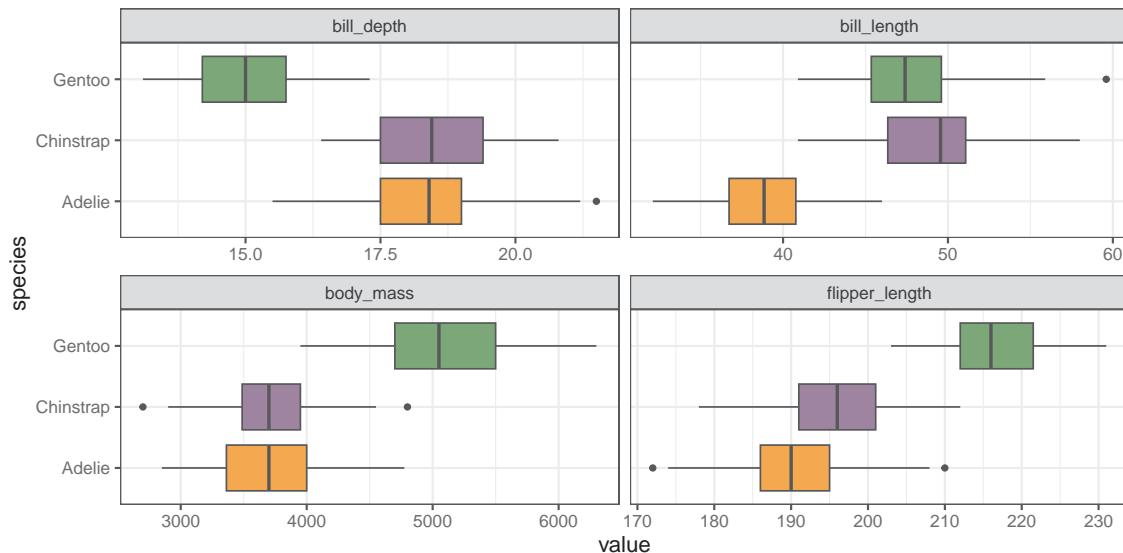


Figure 12.1: Boxplots for the Penguin variables. For assessing homogeneity of variance, we should be looking for differences in **width** of the central 50% boxes in each panel, rather than difference in central tendency.

Instead, you can see more *directly* what is tested by the Levene test by graphing the *absolute deviations* from the group means or medians. This is another example of the graphic idea that you can make visual comparisons easier by plotting quantities of direct interest. You can calculate the median deviation values as follows:

```
vars <- c("bill_length", "bill_depth", "flipper_length", "body_mass")
pengDevs <- colDevs(peng[, vars], peng$species, median) |>
  abs()
```

From a boxplot of the absolute deviations in Figure 12.2 your eye can now focus on the central value, shown by the **median** ‘|’ line, because Levene’s method is testing whether these differ across groups.

```
# calculate absolute differences from median
dev_long <- data.frame(species = peng$species, pengDevs) |>
  pivot_longer(bill_length:body_mass,
               names_to = "variable",
               values_to = "value")

dev_long |>
  group_by(species) |>
  ggplot(aes(value, species, fill = species)) +
  geom_boxplot() +
  facet_wrap(~ variable, scales = 'free_x') +
  xlab("absolute median deviation") +
  theme_penguins() +
  theme_bw(base_size = 14) +
  theme(legend.position = 'none')
```

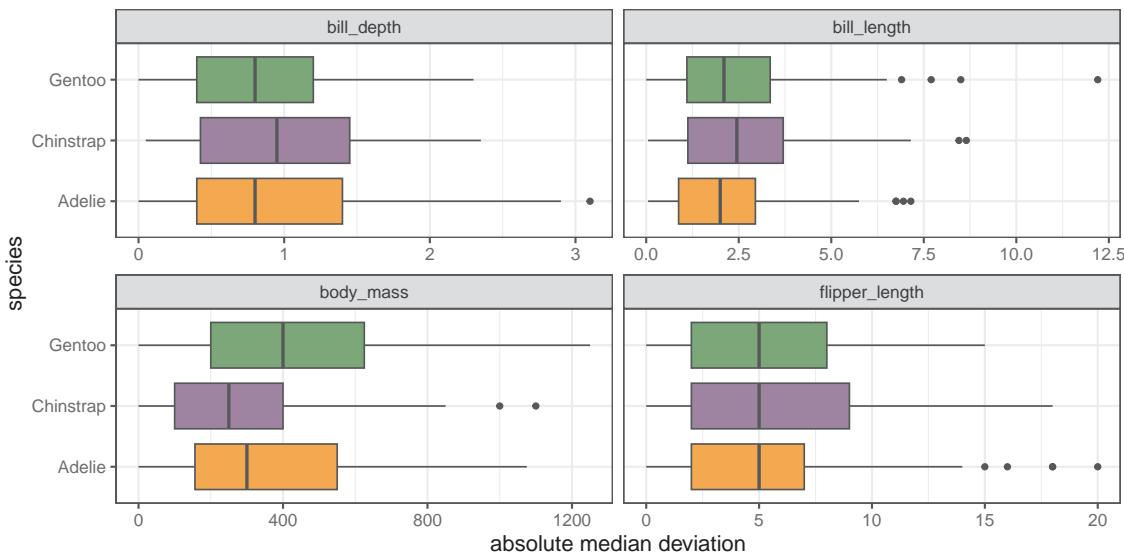


Figure 12.2: Boxplots for absolute differences from group medians for the Penguin data. The visual test of equality of variance is whether the median lines in the boxplots align.

It is now easy to see that the medians largely align for all the variables except for `body_mass`.

12.3 Homogeneity of variance in MANOVA

In the MANOVA context, the main emphasis, of course, is on differences among mean vectors, testing

$$\mathcal{H}_0 : \mu_1 = \mu_2 = \cdots = \mu_g .$$

However, the standard test statistics (Wilks' Lambda, Hotelling-Lawley trace, Pillai-Bartlett trace, Roy's maximum root) rely upon the analogous assumption that the within-group covariance **matrices** Σ_i are equal for all groups,

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_g .$$

This is much stronger than in the univariate case, because it also requires that all the correlations between pairs of variables are the same for all groups. For example, for two responses, there are three parameters $(\rho, \sigma_1^2, \sigma_2^2)$ assumed equal across all groups; for p responses, there are $p(p+1)/2$ assumed equal. The variances relate to *size* differences among data ellipses while the differences in the covariances appear as differences in *shape*.

Penguin data: Covariance ellipses

To preview a main example, Figure 12.3 shows data ellipses for the main size variables in the Penguins data (`peng`). `heplots::covEllipses()` is specialized for viewing the relations among the data ellipsoids representing the sample covariance matrices, $\mathbf{S}_1 = \mathbf{S}_2 = \cdots = \mathbf{S}_g$. It draws the data ellipse for each group, and also for the pooled within-group \mathbf{S}_p , as shown in Figure 12.3 for bill length and bill depth.

You can see that the sizes and shapes of the data ellipses are sort of similar in the left panel. The visual comparison becomes more precise when the data ellipses are all shifted to a common origin at the grand means (using `center = TRUE`). From this you can see that the Adelie group differs most from the others.

```
op <- par(mar = c(4, 4, 1, 1) + .5,
           mfrow = c(c(1,2)))
covEllipses(cbind(bill_length, bill_depth) ~ species, data=peng,
            fill = TRUE,
            fill.alpha = 0.1,
            lwd = 3,
            col = clr)

covEllipses(cbind(bill_length, bill_depth) ~ species, data=peng,
            center = TRUE,
            fill = c(rep(FALSE,3), TRUE),
            fill.alpha = .1,
            lwd = 3,
            col = clr,
            label.pos = c(1:3,0))
par(op)
```

All such pairwise plots in scatterplot matrix format are produced using the `variables` argument to `covEllipses()`, giving Figure 12.4.

```
clr <- c(peng.colors(), "black")
covEllipses(peng[,3:6], peng$species,
            variables=1:4,
            col = clr,
            fill=TRUE,
            fill.alpha=.1)
```

The covariance ellipses in Figure 12.4 look pretty similar in size, shape and orientation. But what does Box's M test (described below) say? As you can see, it concludes strongly against the null hypothesis, because the test is highly sensitive to small differences among the covariance matrices.

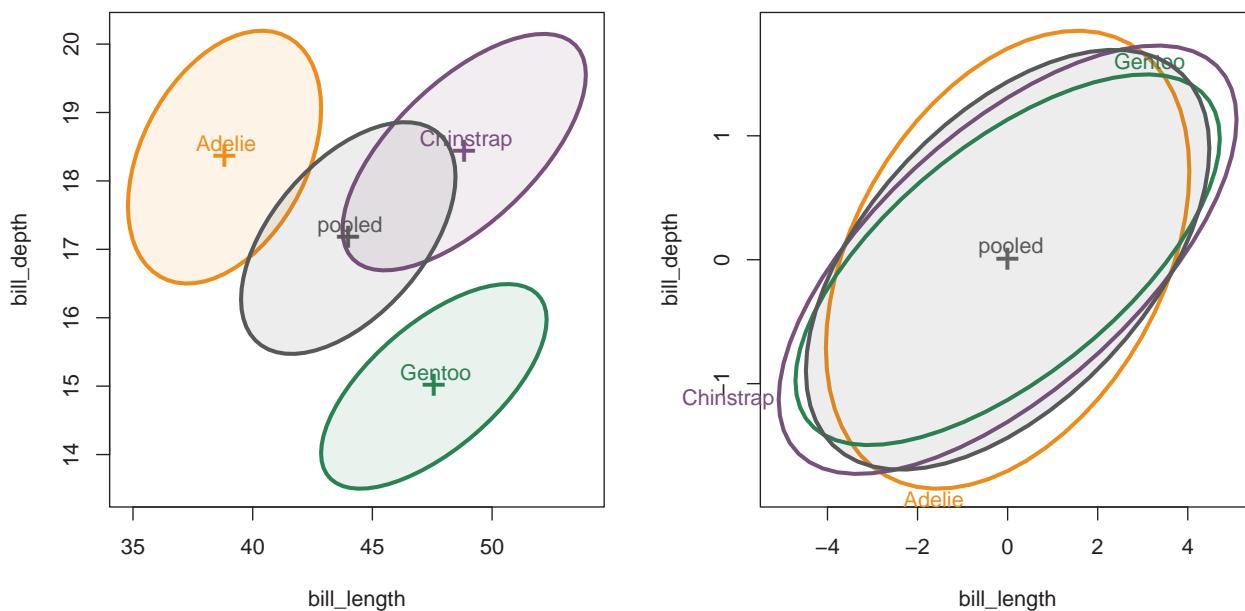


Figure 12.3: Data ellipses for bill length and bill depth in the penguins data, also showing the pooled covariance. Left: As is; right: these are centered at the grand means for easier comparison.

```

peng.boxm <- boxM(cbind(bill_length, bill_depth, flipper_length, body_mass)
  ~ species,
  data=peng) |>
  print()
#>
#> Box's M-test for Homogeneity of Covariance Matrices
#>
#> data: peng
#> Chi-Sq (approx.) = 75, df = 20, p-value = 3e-08

```

Iris data: Covariance ellipses

It will be useful to have another example as we proceed, so Figure 12.5 shows an analogous plot for the iris data we examined in Section 11.6.

Even when these are shown uncentered, the differences in size, shape and orientation are much more apparent. Iris *setosa* stands out as having smaller variance on some of the variables, while the ellipses for *virginica* tend to be larger. Their orientation (slopes) also differ quite a bit.

```

iris.colors <- c("red", "darkgreen", "blue")
covEllipses(iris[,1:4], iris$Species,
  variables=1:4,
  fill = TRUE,
  fill.alpha=.1,
  col = c(iris.colors, "black"),
  label.pos=c(1:3,0))

#>
#> Box's M-test for Homogeneity of Covariance Matrices
#>

```

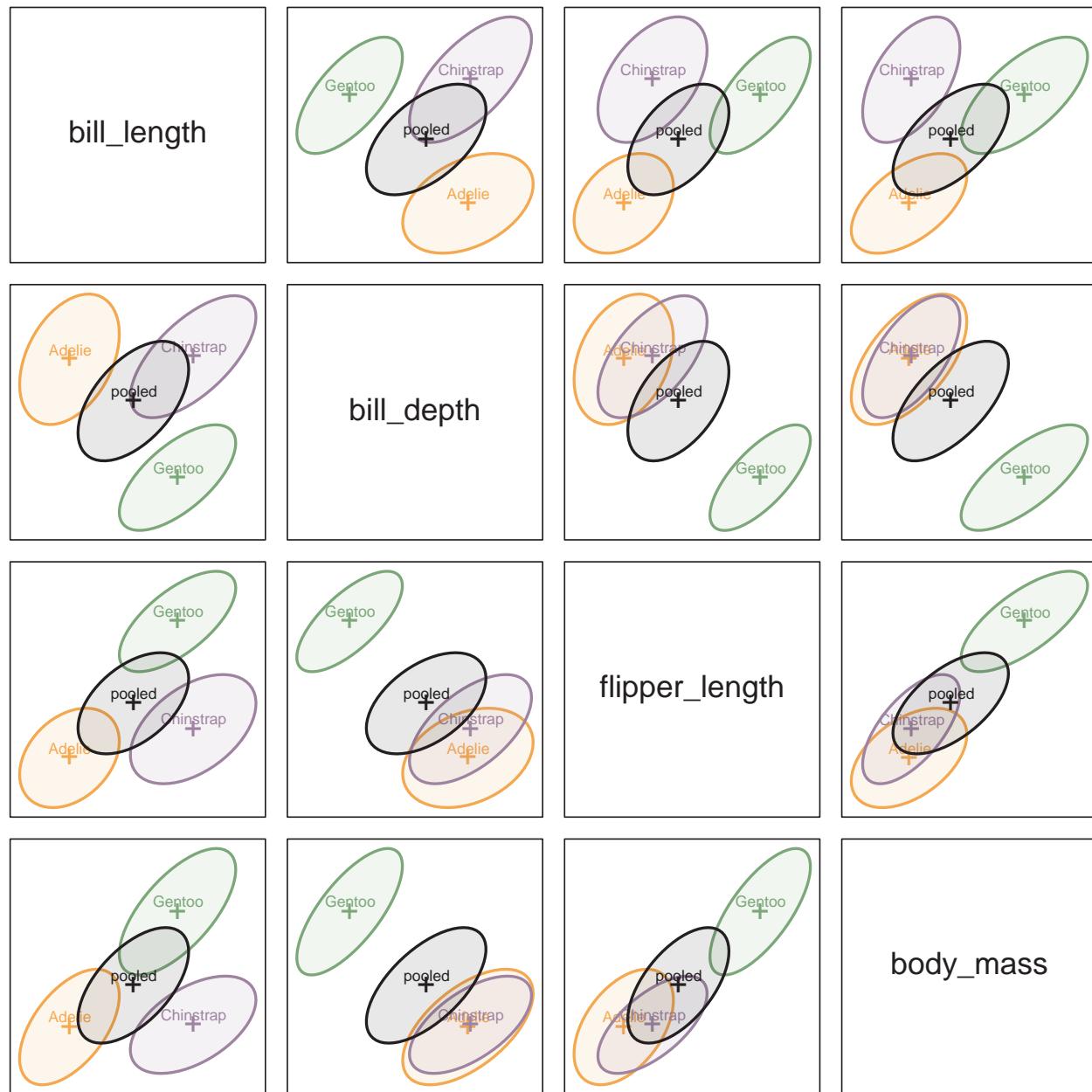


Figure 12.4: All pairwise covariance ellipses for the penguins data. The covariance matrices are homogeneous when the ellipses for the groups all have the same size and shape as that for the mean-centered pooled data (shown in black).

```
#> data: iris
#> Chi-Sq (approx.) = 141, df = 20, p-value <2e-16
```

Unsurprisingly, Box's \mathcal{M} tests gives $\chi^2_{20} = 140.94$ with a p -value $< 2.2\text{e-}16$, putting numbers to the visual conclusion from Figure 12.5.

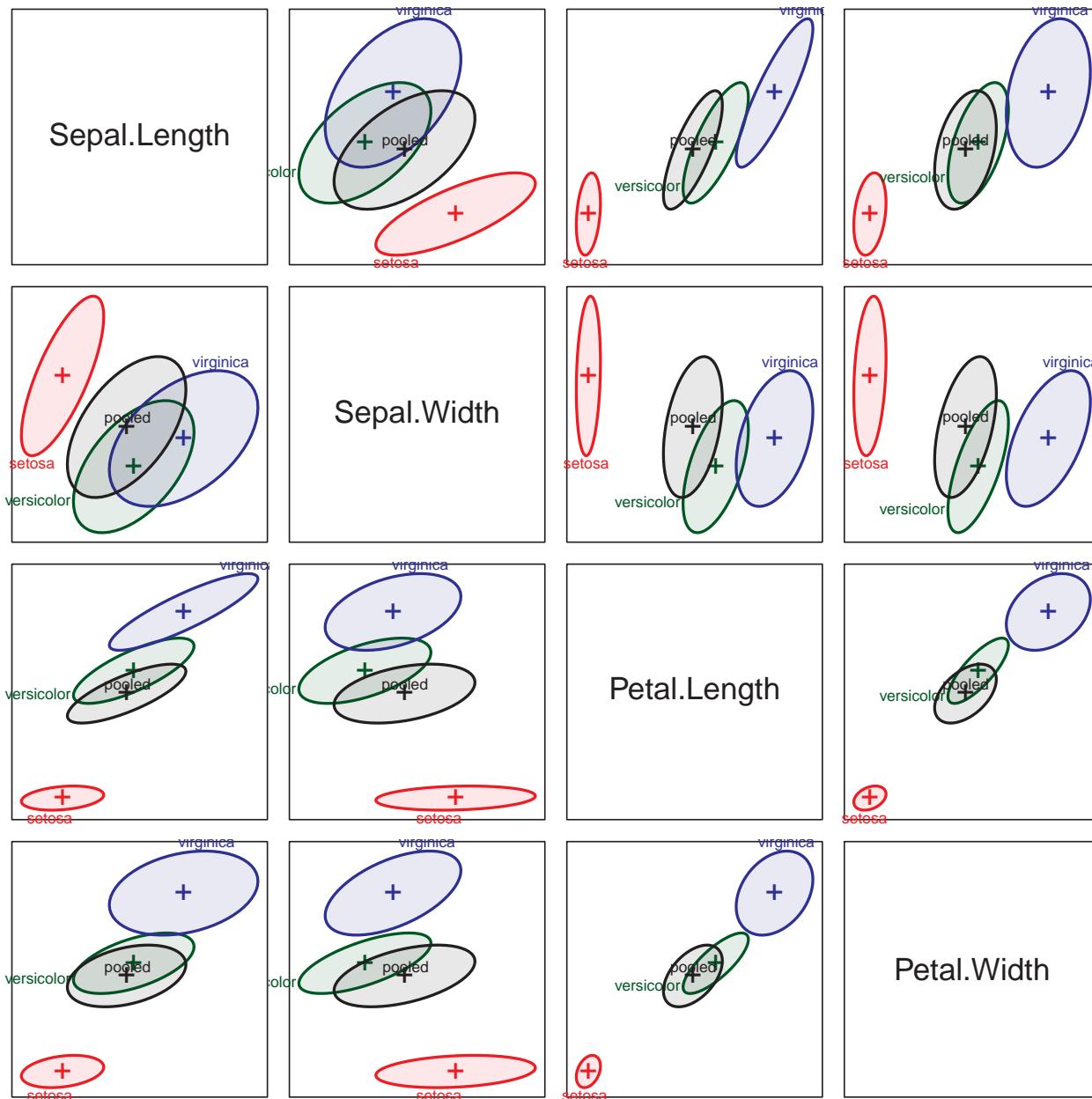


Figure 12.5: All pairwise covariance ellipses for the iris data. The large differences in size and shape indicate substantial heterogeneity in variances and covariances.

12.4 Box's \mathcal{M} test

Take a moment and think, “How could we generalize a test of equality of variances, $s_1^2 = s_2^2 = \dots = s_g^2$, to the multivariate case, where we have $(p \times p)$ matrices, $\mathbf{S}_1 = \mathbf{S}_2 = \dots = \mathbf{S}_g$ for each of g groups?”. Multivariate thinking suggests that that we calculate some measure of “size” of each \mathbf{S}_i , in a similar way to what is done in multivariate tests comparing \mathbf{H} and \mathbf{E} matrices.

Box (1949) proposed the following likelihood-ratio test (LRT) statistic \mathcal{M} for testing the hypothesis of equal covariance matrices, using the log of the determinant $|\mathbf{S}_i|$ as the measure of size.

$$\mathcal{M} = (N - g) \ln |\mathbf{S}_p| - \sum_{i=1}^g (n_i - 1) \ln |\mathbf{S}_i| , \quad (12.1)$$

where $N = \sum n_i$ is the total sample size and

$$\mathbf{S}_p = (N - g)^{-1} \sum_{i=1}^g (n_i - 1) \mathbf{S}_i$$

is the pooled covariance matrix.

\mathcal{M} can thus be thought of as a ratio of the determinant of the pooled \mathbf{S}_p to the geometric mean of the determinants of the separate \mathbf{S}_i .

In practice, there are various transformations of the value of M to yield a test statistic with an approximately known distribution (Timm, 1975). Roughly speaking, when each $n_i > 20$, a χ^2 approximation is often used; otherwise an F approximation is known to be more accurate.

Asymptotically, $-2 \ln(\mathcal{M})$ has a χ^2 distribution. The χ^2 approximation due to Box (1949, 1950) is that

$$X^2 = -2(1 - c_1) \ln(\mathcal{M}) \sim \chi_{df}^2$$

with $df = (g - 1)p(p + 1)/2$ degrees of freedom, and a bias correction constant:

$$c_1 = \left(\sum_i \frac{1}{n_i - 1} - \frac{1}{N - g} \right) \frac{2p^2 + 3p - 1}{6(p + 1)(g - 1)} .$$

In this form, Bartlett's test for equality of variances in the univariate case is the special case of Box's \mathcal{M} when there is only one response variable, so Bartlett's test is sometimes used as univariate follow-up to determine which response variables show heterogeneity of variance.

Yet, like its univariate counterparts, Box's test is well-known to be highly sensitive to violation of (multivariate) normality and the presence of outliers², as Box (1953) suggested in the opening chapter quote. Yet, it provides a nice framework for thinking about this problem more generally. . .

12.5 Visualizing heterogeneity

A larger goal of this chapter is to use this background as another illustration of multivariate thinking, here, for visualizing and testing the heterogeneity of covariance matrices in multivariate designs.

While researchers often rely on a single number to determine if their data have met a particular threshold, such compression will often obscure interesting information, particularly when a test concludes that differences exist, and one is left to wonder “why?”. It is within this context where, again, visualizations often reign supreme.

We have already seen one useful method in Section 12.3, which uses direct visualization of the information

²For example, Tiku & Balakrishnan (1984) concluded from simulation studies that the normal-theory LRT provides poor control of Type I error under even modest departures from normality. O'Brien (1992) proposed some robust alternatives, and showed that Box's normal theory approximation suffered both in controlling the null size of the test and in power. Zhang & Boos (1992) also carried out simulation studies with similar conclusions and used bootstrap methods to obtain corrected critical values.

in the \mathbf{S}_i and \mathbf{S}_p using *data ellipsoids* to show size and shape as minimal schematic summaries; In what follows, I propose three additional visualization-based approaches to questions of heterogeneity of covariance in MANOVA designs:

- (a) a simple dotplot of the components of Box's \mathcal{M} test: the log determinants of the \mathbf{S}_i together with that of the pooled \mathbf{S}_p . Extensions of these simple plots raise the question of whether measures of heterogeneity other than that captured in Box's test might also be useful; and,
- (b) PCA low-rank views to highlight features more easily seen there than in the full data space.
- (c) the connection between Levene-type tests and an ANOVA (of centered absolute differences) suggests a parallel with a multivariate extension of Levene-type tests and a MANOVA. We explore this with a version of Hypothesis-Error (HE) plots we have found useful for visualizing mean differences in MANOVA designs.

These methods are described and illustrated in Friendly & Sigal (2018).

12.6 Visualizing Box's \mathcal{M}

Box's test is based on a comparison of the log determinants of the \mathbf{S}_i relative to that of the pooled \mathbf{S}_p , so the simplest thing to do is just plot them!

`boxM()` produces a "boxm" object, for which there are `summary()` (details) and `plot()` methods. The `plot()` method gives a dot plot of the log determinants $\ln |\mathbf{S}_i|$ together with that for the pooled covariance $\ln |\mathbf{S}_p|$. Cai et al. (2015) provide the theory for the (asymptotic) confidence intervals shown.

```
plot(peng.boxm, gplabel="species", cex.lab = 1.5)
plot(iris.boxm, gplabel="Species", cex.lab = 1.5)
```

In these plots (Figure 12.6), the value for the pooled covariance appears within the range of the groups, because it is a weighted average. If you take a moment to look back at Figure 12.4, you'll see that the data ellipses for Gentoo are slightly smaller in most pairwise views, but it is much easier to see this in a plot that summarizes this, like Figure 12.6.

From Figure 12.5, it is clear that *setosa* shows the smallest within-group variability. The numeric scale values on the horizontal axis give a sense that the range across groups is considerably greater for the iris data than for the Penguins. Some other generalizations of Box's test using other measures are illustrated in Section 12.8.

12.7 Low-rank views

With $p > 3$ response variables, a simple alternative to the pairwise 2D plots in data space (shown in Figure 12.4 and Figure 12.5) is the projection into the principal component space accounting for the greatest amounts of total variance in the data (Friendly & Sigal (2018)). For the Iris data, a simple PCA of the covariance matrix shows that nearly 96% of total variance in the data is accounted for in the first two dimensions.

```
iris.pca <- prcomp(iris[,1:4], scale. = TRUE)
summary(iris.pca)
```

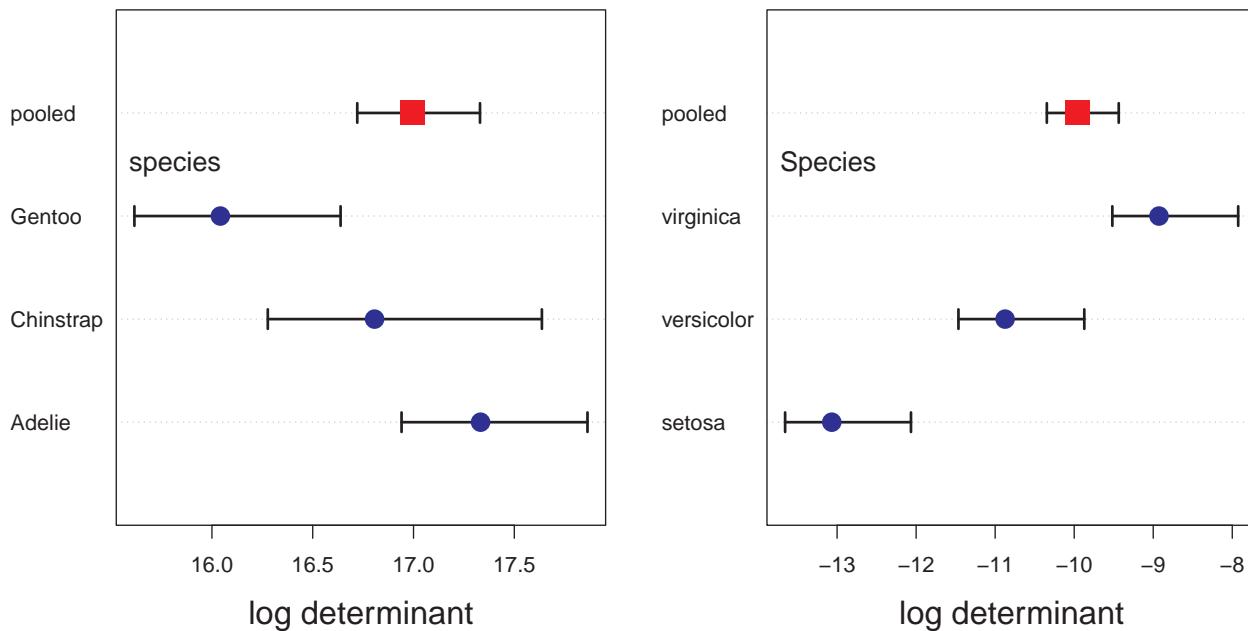


Figure 12.6: Plots of the contributions to Box's \mathcal{M} statistic for the Penguin and iris data.

```
#> Importance of components:
#>          PC1    PC2    PC3    PC4
#> Standard deviation   1.71 0.956 0.3831 0.14393
#> Proportion of Variance 0.73 0.229 0.0367 0.00518
#> Cumulative Proportion  0.73 0.958 0.9948 1.00000
```

Figure 12.7 shows the plots of the covariance ellipsoids for the first two principal component scores, uncentered (left panel) and centered (right panel). The dominant PC1 (73% of total variance) essentially orders the species by a measure of overall size of their sepals and petals. In the centered view, it can again be seen how *Setosa* differs in covariance from the other two species, and that while *Virginica* and *Versicolor* both have similar shapes to the pooled covariance matrix, *Versicolor* has somewhat greater variance on PC1.

```
source(here::here("R/util/text.usr.R"))

covEllipses(iris.pca$x, iris$Species,
  col = c(iris.colors, "black"),
  fill=TRUE, fill.alpha=.1,
  cex.lab = 1.5,
  label.pos = c(1, 3, 3, 0), asp=1)
text.usr(0.05, 0.95, "(a) Uncentered", pos = 4)

covEllipses(iris.pca$x, iris$Species,
  center=TRUE,
  col = c(iris.colors, "black"),
  fill=TRUE, fill.alpha=.1,
  cex.lab = 1.5,
  label.pos = c(1, 3, 3, 0), asp=1)
text.usr(0.95, 0.95, "(b) Centered", pos = 2)
```

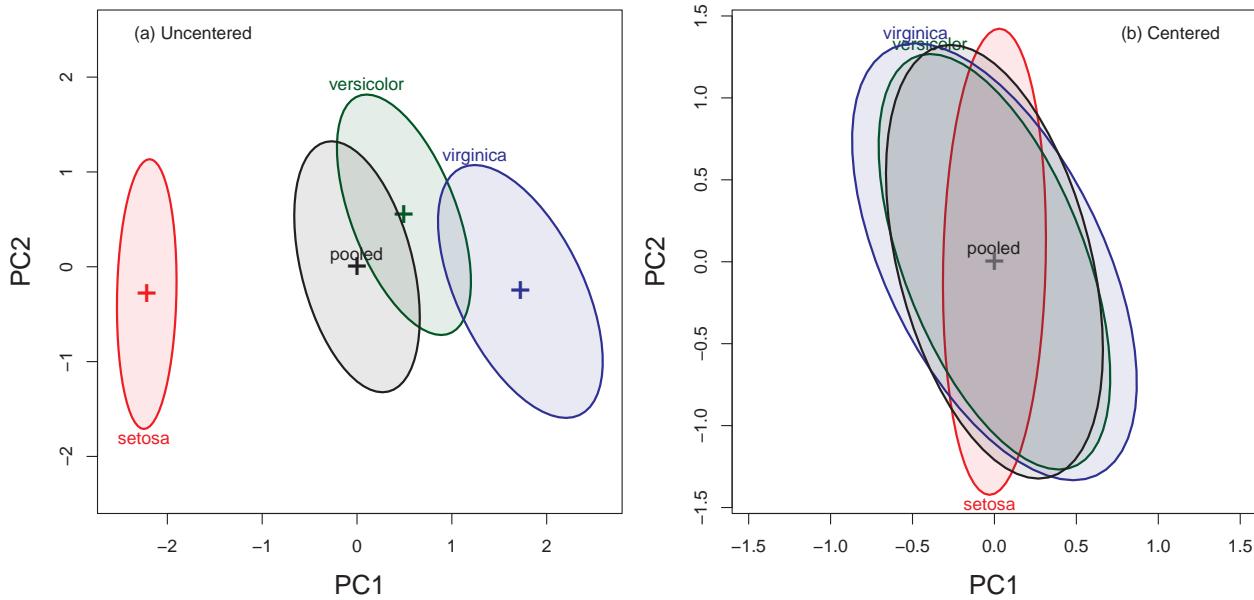


Figure 12.7: Covariance ellipsoids for the first two principal components of the iris data. Left (a): uncentered, showing group means on the principal components; right (b): centered at the origin.

12.7.1 Small dimensions can matter

For the Iris data, the first two principal components account for 96% of total variance, so we might think we are done here. Yet, as we've seen in other problems (outliers, collinearity), important information also exists in the space of the *smallest* principal component dimensions.

This is also true, as we will see for Box's \mathcal{M} test, because it is a (linear) function of all the eigenvalues of the between and within group covariance matrices, is therefore also subject to the influence of the smaller dimensions, where differences among \mathbf{S}_i and of \mathbf{S}_p can lurk.

```
covEllipses(iris.pca$x, iris$Species,
            variables = 3:4,
            center=TRUE,
            col = c(iris.colors, "black"),
            fill=TRUE, fill.alpha=.1,
            cex.lab = 1.5,
            label.pos = c(1, 3, 3, 4), asp=1)
```

Figure 12.8 shows the covariance ellipsoids in (PC3, PC4) space. Even though these dimensions contribute little to total variance, there are more pronounced differences in the within-group shapes (correlations) relative to the pooled covariance, and these contribute to a rejection of homogeneity by Box's \mathcal{M} test. Here we see that the correlation for *Virginica* is of opposite sign from the other two groups.

12.8 Other measures of heterogeneity

As we saw above Section 12.3, the question of equality of covariance matrices can be expressed in terms of the similarity in size and shape of the data ellipses for the individual group \mathbf{S}_i relative to that of \mathbf{S}_p . Box's \mathcal{M} test uses just one possible function to describe this size: the logs of their determinants.

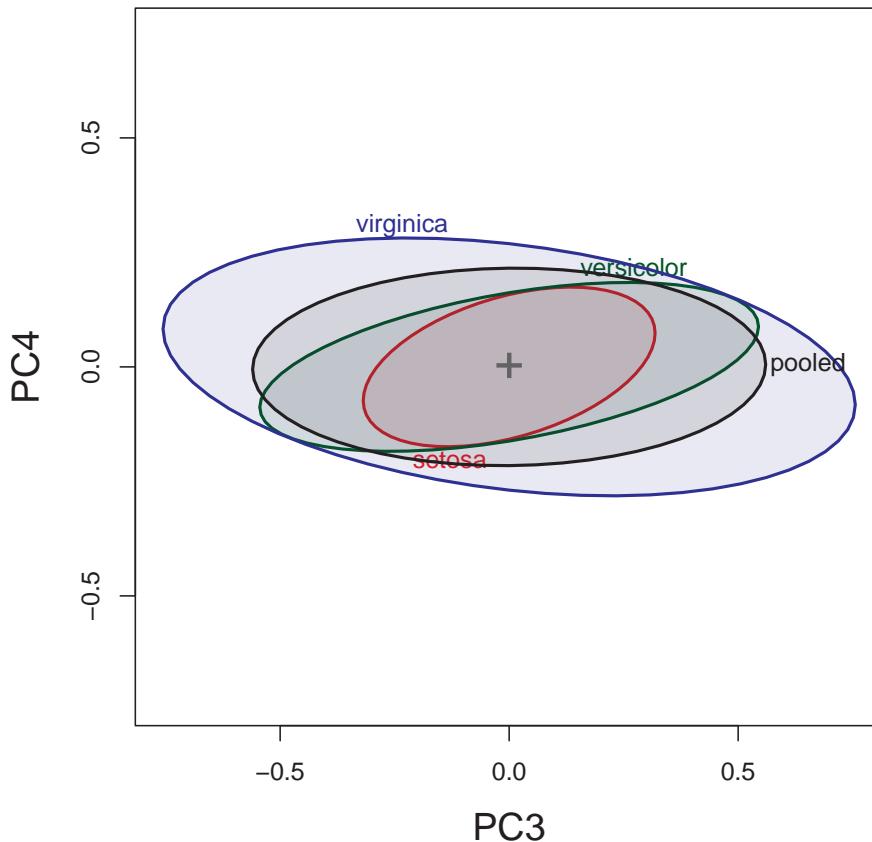


Figure 12.8: Covariance ellipses for the smallest principal components of the iris data.

When Σ is the covariance matrix of a multivariate vector \mathbf{y} with eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \lambda_p$, the properties shown in Table 12.1 represent methods of describing the size and shape of the ellipsoid in \mathbb{R}^p .³ Just as is the case for tests of the MLM itself where different functions of these give test statistics (Wilks' Λ , Pillai trace, etc.), one could construct other test statistics for homogeneity of covariance matrices.

Table 12.1: Statistical and geometrical properties of “size” of an ellipsoid

Size	Conceptual formula	Geometry	Function
(a) Generalized variance:	$\det \Sigma = \prod_i \lambda_i$	area, (hyper)volume	geometric mean
(b) Average variance:	$\text{tr}(\Sigma) = \sum_i \lambda_i$	linear sum	arithmetic mean
(c) Average precision:	$1/\text{tr}(\Sigma^{-1}) = 1/\sum_i (1/\lambda_i)$		harmonic mean
(d) Maximal variance:	λ_1	maximum dimension	supremum

Hence, for a sample covariance matrix \mathbf{S} , $|\mathbf{S}|$ is a measure of generalized variance and $\ln |\mathbf{S}|$ is a measure of average variance across the p dimensions.

The "boxM" plot methods in r"heplots" can compute and plot all of the functions of the eigenvalues in Table 12.1. The results are shown in Figure 12.9.

```
plot(peng.boxm, which="product", gplabel="species")
plot(peng.boxm, which="sum", gplabel="species")
```

³More general theory and statistical applications of the geometry of ellipsoids is given by Friendly et al. (2013).

```
plot(peng.boxm, which="precision", gplabel="species")
plot(peng.boxm, which="max", gplabel="species")
```

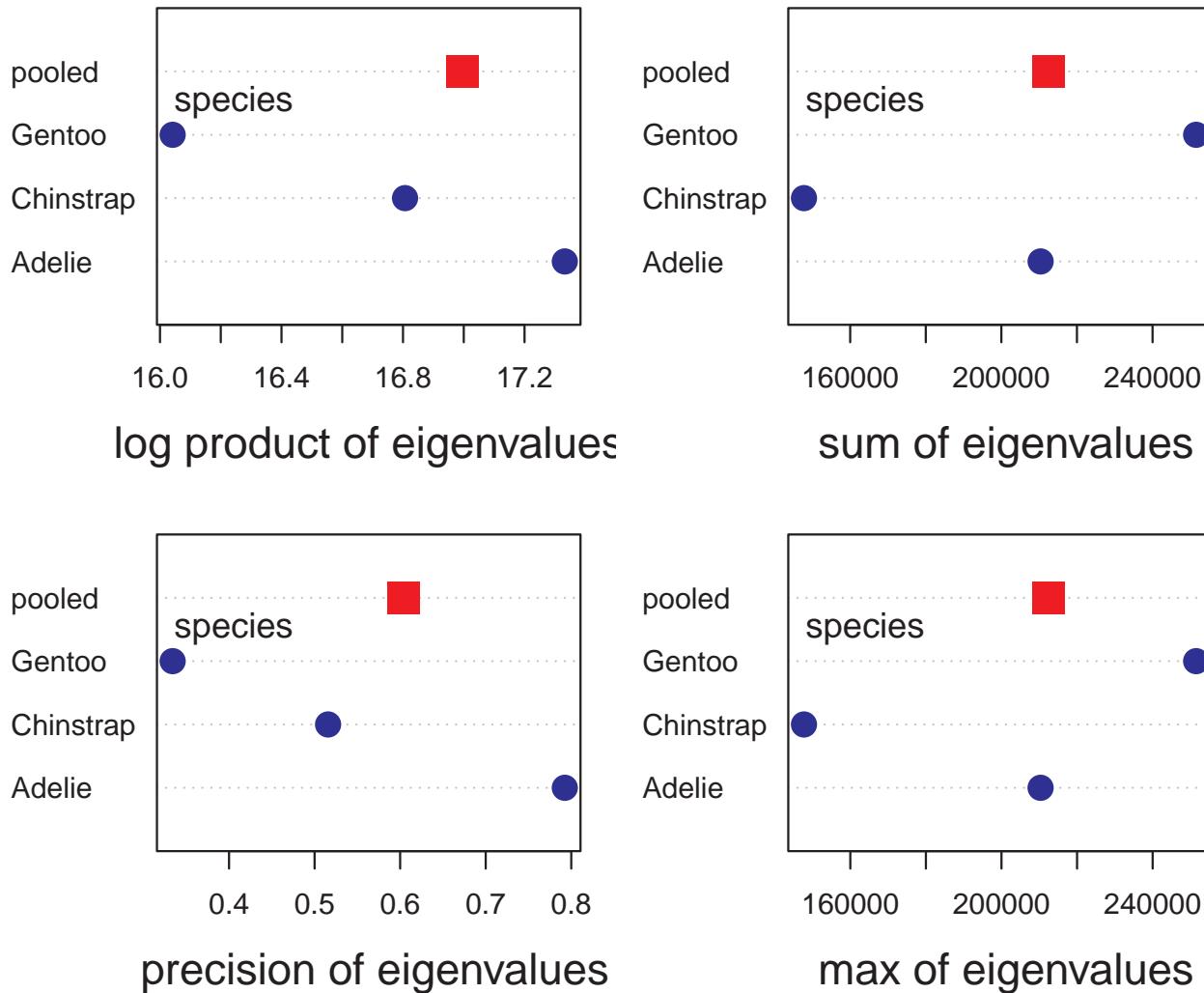


Figure 12.9: Plot of eigenvalue statistics of the covariance matrices for the Penguin data

Except for the absence of error bars, the plot for log product in the upper left panel of Figure 12.9 is the same as that in Figure 12.6, but with the sign reversed. In principle, it is possible to add such confidence intervals for all these measures through the use of bootstrapping, but this has not yet been implemented in the `heplots`

For this data set, the pattern of points in the plot for Box's \mathcal{M} is also more or less the same as that for the precision measure. The plots for the sum of and maximum eigenvalue are also similar to each other, but differ from those of the two measures in the left column of Figure 12.9. The main point is that these are not *all the same*, so different functions reflect different patterns of the eigenvalues, and could be used to define new statistical tests, perhaps with greater power or sensitivity to outliers.

12.9 Multivariate analog of Levine's test

The fact that Levine's test is just a simple ANOVA of the absolute deviations from the group means or medians suggests an easy multivariate generalization (which has not been well-studied) —Simply do a MANOVA of the absolute differences, $|\mathbf{Y} - \bar{\mathbf{Y}}_j|$, centering on the means or medians.⁴ For the `iris` data, this gives:

```
irisdev <- colDevs(iris[,1:4], iris$Species, median,
                     group.var = "Species") |>
  mutate(across(where(is.numeric), abs))

irisdev.mod <- lm(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~
                     Species, data=irisdev)
Anova(irisdev.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> Species   2     0.394      8.89     8    290 6.7e-11 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `irisdev.mod` model is just another MANOVA, so we can visualize it in an HE plot (Figure 12.10). But remember that this is showing differences among groups in multivariate dispersion rather than in means.

```
pairs(irisdev.mod, variables=1:4, fill=TRUE, fill.alpha=.1)
```

In this plot, the **H** ellipses for the groups show the variation in the overall *dispersion*—the size of absolute differences from the group medians. These are very highly correlated across groups in most of the subplots, particularly so for those involving sepal width.

12.9.1 Canonical discriminant analysis

But we can go further. Just as canonical discriminant analysis provides a low-dimensional view of differences in means, the same method can be used to visualize heterogeneity of dispersion expressed by the the Levene-type MANOVA in 1D or 2D.

For the model `irisdev.mod`, this shows that 98% of groups differences shown in the HE plots (Figure 12.10) can be found in a single canonical dimension. The 4D information can be reduced to 1D.

```
library(candisc)
irisdev.can <- candisc(irisdev.mod) |>
  print()
#>
#> Canonical Discriminant Analysis for Species:
#>
#>   CanRsq Eigenvalue Difference Percent Cumulative
#>   1  0.381      0.6154      0.602     97.9      97.9
```

⁴M. J. Anderson (2006) describes other procedures based on Euclidean distances $D_{ij} = [\sum(y_{ij} - \bar{y}_j)^2]^{1/2}$ of the observations from their group centroids (means or spatial medians). Rather than fit a parametric MANOVA model, he suggests a permutation test method (PERMDIST) which randomly permutes the group assignments and yields non-parametric tests.

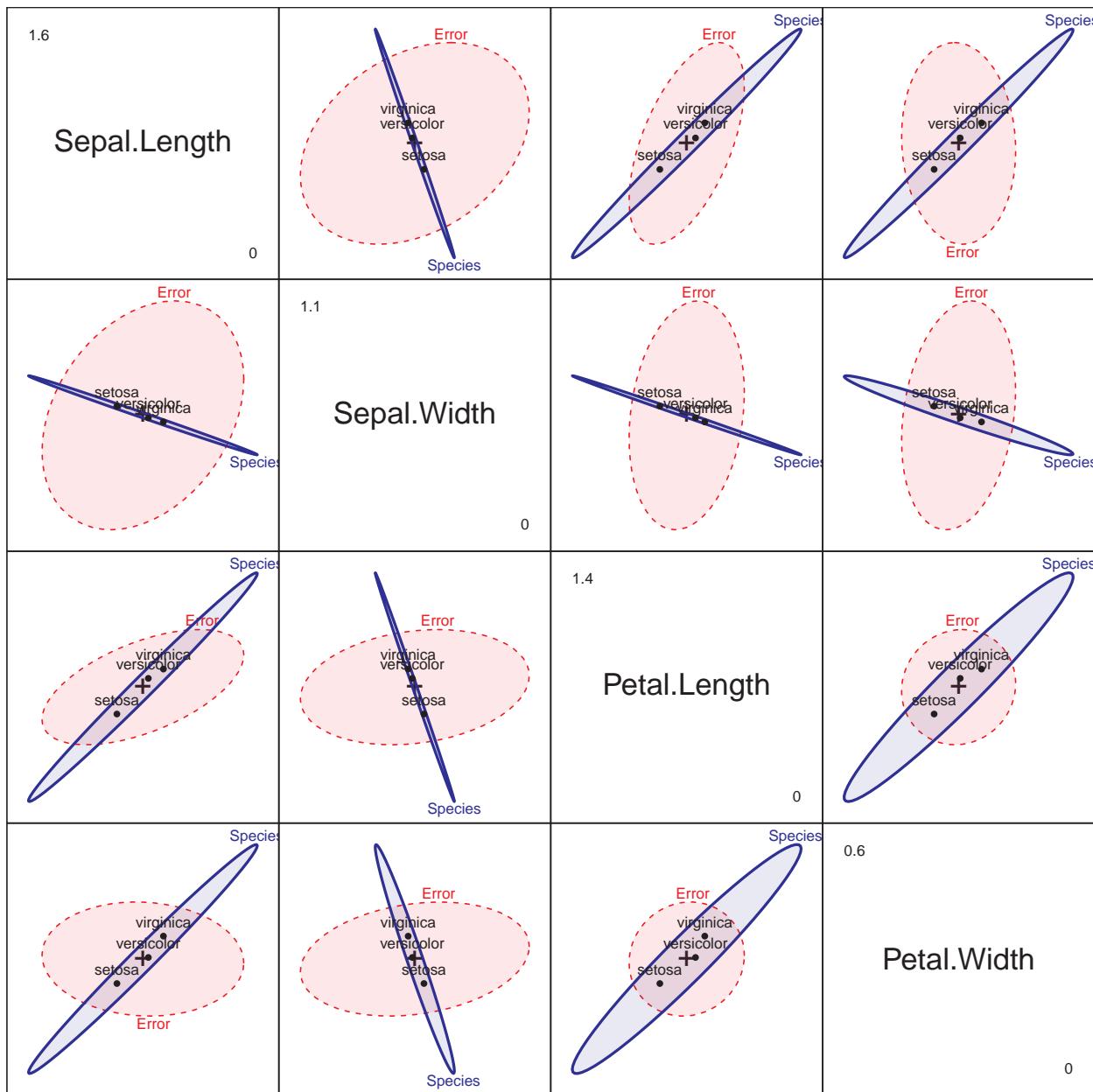


Figure 12.10: HE plots for the multivariate generalization of Levine's test based on MANOVA of absolute deviations from group medians.

```
#> 2 0.013      0.0132      0.602      2.1      100.0
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#> LR test stat approx F numDF denDF Pr(> F)
#> 1      0.611     10.06      8    288 2.3e-12 ***
#> 2      0.987     0.64      3    145   0.59
```

```
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Plotting the first canonical dimension gives boxplots for the scores and vectors for the weights of the iris variables.

```
plot(irisdev.can, which=1,
      fill.alpha = .3,
      col = iris.colors, lwd = 2,
      var.cex = 1.2,
      cex.lab = 1.5, cex.axis = 1.1)
```

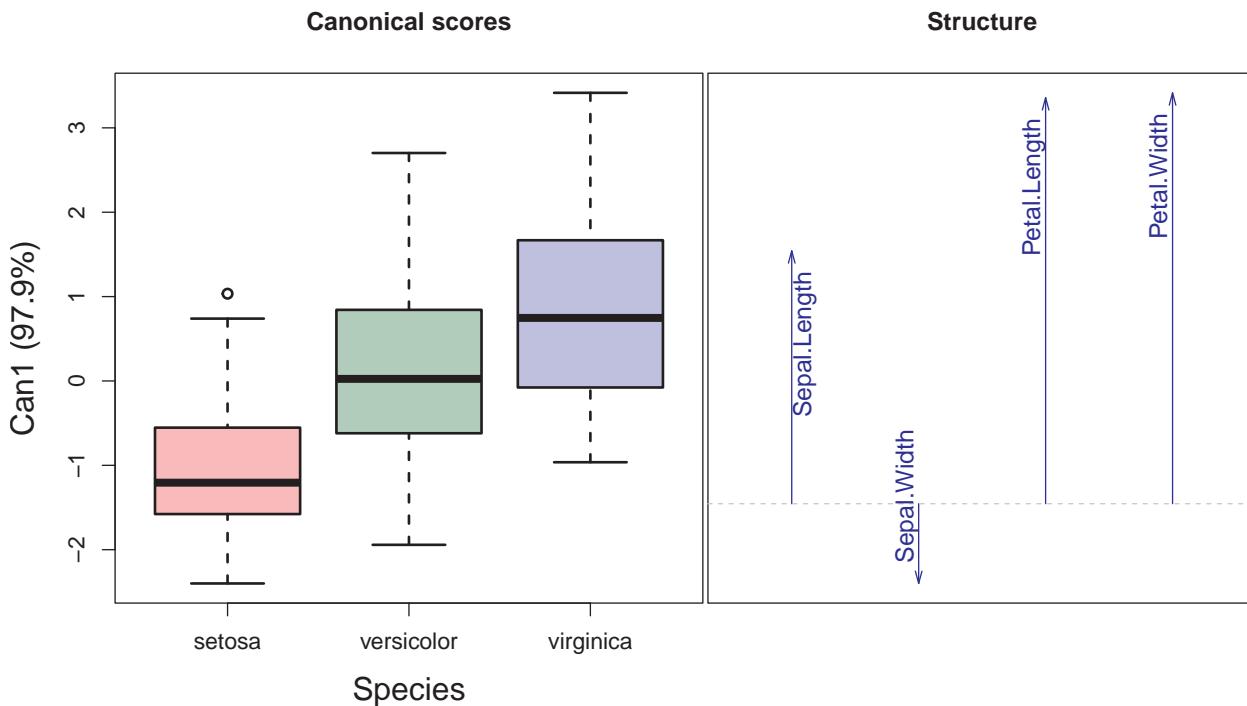


Figure 12.11: Canonical discriminant plot for the multivariate generalization of Levine's test based on MANOVA of absolute deviations from group medians.

Overall, the covariance matrices are smallest for setosa and largest for virginica. The structure coefficients for the variables reflect the pattern shown in the HE plots (Figure 12.10), where the direction of the \mathbf{H} ellipses is negative for sepal width but positive for the others.

12.10 What Have We Learned?

The quest to understand equality of covariance matrices in multivariate models has taken us on a journey from Box's famous row boat metaphor to some novel visualization techniques. Here are the essential insights that will transform how you think about and explore heterogeneity in your multivariate data:

- **Visualization beats test statistics alone** - While Box's \mathcal{M} -test gives you a single p -value, covariance

ellipses reveal the *why* behind heterogeneity. You can literally see differences in size (scatter) and shape (orientation) of group covariances, making the abstract concrete and interpretable.

- **Centering makes differences more apparent** - Shifting all group ellipses to a common center (the grand mean) is like removing visual noise to see the signal. This simple transformation makes differences in covariance structure apparent, turning subtle variations into obvious visual patterns.
- **Small dimensions often hold big secrets** - Don't ignore those "unimportant" principal components! The smallest PC dimensions frequently contain the most discriminating information about covariance differences. It's often where the action is hiding, away from the obvious patterns in major dimensions.
- **Multiple measures tell richer stories** - Box's \mathcal{M} -test uses log determinants, but why stop there? Exploring different eigenvalue functions (sum, precision, maximum) can reveal distinct patterns of heterogeneity, like having multiple lenses to examine the same phenomenon.
- **Levene meets MANOVA in beautiful harmony** - The multivariate extension of Levene's test (MANOVA on absolute deviations) creates a natural bridge between univariate and multivariate thinking, complete with interpretable HE plots that make variance differences as visual as mean differences.
- **Graphics simplify complex concepts** - These visualization tools transform an intimidating mathematical concept (equality of $p \times p$ matrices) into something any researcher can explore, understand, and communicate to others.

The chapter's strength lies in showing that covariance matrices aren't just mathematical abstractions—they're visual stories waiting to be told, patterns waiting to be discovered, and insights illustrating the power of multivariate thinking.

13

Multivariate Influence and Robust Estimation

Note

This chapter may be moved from the printed PDF copy to an online Appendix.

In the analysis of linear models, the identification and treatment of outliers and influential observations represents one of the most critical yet challenging aspects of statistical modeling. As you saw earlier (Section 6.6), even a single “bad” observation can completely alter the results of a linear model fit by ordinary least squares.

Univariate influence diagnostics have been well-established since the pioneering work of R. D. Cook (1977) and others (Belsley et al. (1980); R. D. Cook & Weisberg (1982)) and their wide implementation in R packages such as `stats` and `car` makes these readily accessible in statistical practice. If you seek statistical advice regarding a perplexing model, the consultant may well ask:

Did you make any influence or other diagnostic plots?

However, the extension to multivariate response models introduces additional complexity that goes far beyond simply applying univariate methods to each response variable separately. The multivariate case requires consideration of the *joint influence* structure across all responses simultaneously, accounting for the correlation patterns among dependent variables and the potential for observations to be influential in some linear combinations of responses while appearing benign when examined multivariate.

This multivariate perspective can reveal influence patterns that would otherwise remain hidden, as an observation might exert substantial leverage on the overall model fit through subtle but systematic effects across multiple responses. Detecting outliers and influential observations has now progressed to the point where the methods described below (Section 13.1) can usefully be applied to multivariate linear models.

But having found some troublesome cases, the question arises, what to do about them? We are generally reluctant to simply ignore them, unless there is evidence of a gross data error (as in the `Davis` data, Section 2.1.2). Instead, a large class of **robust** methods, which reduce the impact of outliers on the analysis, have been developed. These are described in Section 13.5 below.

Packages

In this chapter I use the following packages. Load them now.

```
library(dplyr)
library(tidyr)
library(car)
library(helplots)
library(candisc)
library(mvinfluence)
library(ggplot2)
library(patchwork)
```

13.1 Multivariate influence

An elegant extension of the ideas behind leverage, studentized residuals and measures of influence to the case of multivariate response data is due to Barrett & Ling (1992) (see also: Barrett (2003)). These methods have been implemented in the `mvinfluence` package (Friendly, 2025a) which makes available several forms of influence plots to visualize the results.

As in the univariate case, the measures of multivariate influence stem from case-deletion idea of comparing some statistic calculated from the full sample to that statistic calculated when case i is deleted. The Barrett-Ling approach generalized this to the case of deleting a set I of $m \geq 1$ cases. This can be useful because some cases can “mask” the influence of others in the sense that when one is deleted, others become much more influential. However, in most cases the default of deleting individual observations ($m = 1$) is sufficient.

13.1.1 Notation

It is useful to define some notation used to designate terms in the model calculated from the *complete* dataset versus those calculated with one or more observations *excluded*. As before, let \mathbf{X} be the model matrix in the multivariate linear model, $\mathbf{Y}_{n \times p} = \mathbf{X}_{n \times q} \mathbf{B}_{q \times p} + \mathbf{E}_{n \times p}$. As we know, the usual least squares estimate of \mathbf{B} is given by $\mathbf{B} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$.

Then let:

- \mathbf{X}_I be the *submatrix* of \mathbf{X} whose m rows are indexed by I ,
- $\mathbf{X}_{(-I)}$ is the *complement*, the submatrix of \mathbf{X} with the m rows in I deleted,

Matrices \mathbf{Y}_I , $\mathbf{Y}_{(-I)}$ are defined similarly, denoting the submatrix of m rows of \mathbf{Y} and the submatrix with those rows deleted, respectively.

The calculation of regression coefficients when the cases indexed by I have been removed has the form $\mathbf{B}_{(-I)} = (\mathbf{X}_{(-I)}^\top \mathbf{X}_{(-I)})^{-1} \mathbf{X}_{(-I)}^\top \mathbf{Y}_I$. The corresponding residuals are expressed as $\mathbf{E}_{(-I)} = \mathbf{Y}_{(-I)} - \mathbf{X}_{(-I)} \mathbf{B}_{(-I)}$.

13.1.2 Hat values and residuals

The influence measures defined by Barrett & Ling (1992) are functions of two matrices \mathbf{H}_I and \mathbf{Q}_I corresponding to hat values and residuals, defined as follows:

- For the full data set, the “hat matrix”, \mathbf{H} , is given by $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$,
- \mathbf{H}_I is the $m \times m$ the submatrix of \mathbf{H} corresponding to the index set I , $\mathbf{H}_I = \mathbf{X}(\mathbf{X}_I^\top \mathbf{X}_I)^{-1} \mathbf{X}_I^\top$,
- \mathbf{Q} is the analog of \mathbf{H} defined for the residual matrix \mathbf{E} , that is, $\mathbf{Q} = \mathbf{E}(\mathbf{E}^\top \mathbf{E})^{-1} \mathbf{E}^\top$, with corresponding submatrix $\mathbf{Q}_I = \mathbf{E}_I(\mathbf{E}_I^\top \mathbf{E}_I)^{-1} \mathbf{E}_I^\top$,

13.1.3 Cook's distance

Multivariate analogs of all the usual influence diagnostics (Cook's D, CovRatio, ...) can be defined in terms of \mathbf{H} and \mathbf{Q} . For instance, Cook's distance is defined for a univariate response by

$$D_I = (\mathbf{b} - \mathbf{b}_{(-I)})^\top (\mathbf{X}^\top \mathbf{X})(\mathbf{b} - \mathbf{b}_{(-I)}) / ps^2 ,$$

a measure of the squared distance between the coefficients \mathbf{b} for the full data set and those $\mathbf{b}_{(-I)}$ obtained when the cases in I are deleted.

In the multivariate case, Cook's distance is obtained by replacing the vector of coefficients \mathbf{b} by $\text{vec}(\mathbf{B})$, the result of stringing out the coefficients for all responses in a single $(n \times p)$ -length vector.

$$D_I = \frac{1}{p} [\text{vec}(\mathbf{B} - \mathbf{B}_{(-I)})]^T (S^{-1} \otimes \mathbf{X}^T \mathbf{X}) \text{vec}(\mathbf{B} - \mathbf{B}_{(-I)}) ,$$

where \otimes is the Kronecker (direct) product and $\mathbf{S} = \mathbf{E}^T \mathbf{E} / (n - p)$ is the covariance matrix of the residuals.

13.1.4 Leverage and residual components

We gain further insight by considering how far we can generalize from the case for a univariate response. When $m = 1$, Cook's distance can be re-written as a product of leverage and residual components as

$$D_i = \left(\frac{n-p}{p} \right) \frac{h_{ii} q_{ii}}{(1-h_{ii})^2} .$$

This suggests that we define a *leverage component* L_i and *residual component* R_i as

$$L_i = \frac{h_{ii}}{1-h_{ii}} \quad R_i = \frac{q_{ii}}{1-h_{ii}} .$$

R_i is the studentized residual here, and $D_i \propto L_i \times R_i$.

In the general, multivariate case there are analogous matrix expressions for \mathbf{L} and \mathbf{R} . When $m > 1$, the quantities \mathbf{H}_I , \mathbf{Q}_I , \mathbf{L}_I , and \mathbf{R}_I are $m \times m$ matrices. Where scalar quantities are needed, the `mvinfluence` functions apply a function, `FUN`, either `det()` or `tr()` to calculate a measure of "size". This is done as:

```
H <- sapply(x$H, FUN)
Q <- sapply(x$Q, FUN)
L <- sapply(x$L, FUN)
R <- sapply(x$R, FUN)
```

This is the same trick used in the calculation of the various multivariate test statistics like Wilks' Lambda and Pillai's trace. In this way, the full range of multivariate influence measures discussed by Barrett (2003) can be calculated.

13.2 The Mysterious Case 9

To illustrate these ideas, this example, from Barrett (2003), considers the simplest case, of one predictor (`x`) and two response variables, `y1` and `y2`.

```
Toy <- tibble(
  case = 1:9,
  x = c(1, 1, 2, 2, 3, 3, 4, 4, 10),
  y1 = c(0.10, 1.90, 1.00, 2.95, 2.10, 4.00, 2.95, 4.95, 10.00),
  y2 = c(0.10, 1.80, 1.00, 2.93, 2.00, 4.10, 3.05, 4.93, 10.00)
)
```

A quick peek (Figure 13.1) at the data indicates that `y1` and `y2` are nearly perfectly correlated with each other. Both of these are also strongly linear with `x` and there is one extreme point (case 9). The data is peculiar, but looking at these pairwise plots doesn't suggest that anything is terribly wrong. In the plots of `y1` and `y2` against `x`, case 9 simply looks like a good leverage point (Section 6.6).

```
scatterplotMatrix(~ y1 + y2 + x, data=Toy,
  cex=2,
```

```
col = "blue", pch = 16,
id = list(n=1, cex=2),
regLine = list(lwd = 2, col="red"),
smooth = FALSE)
```

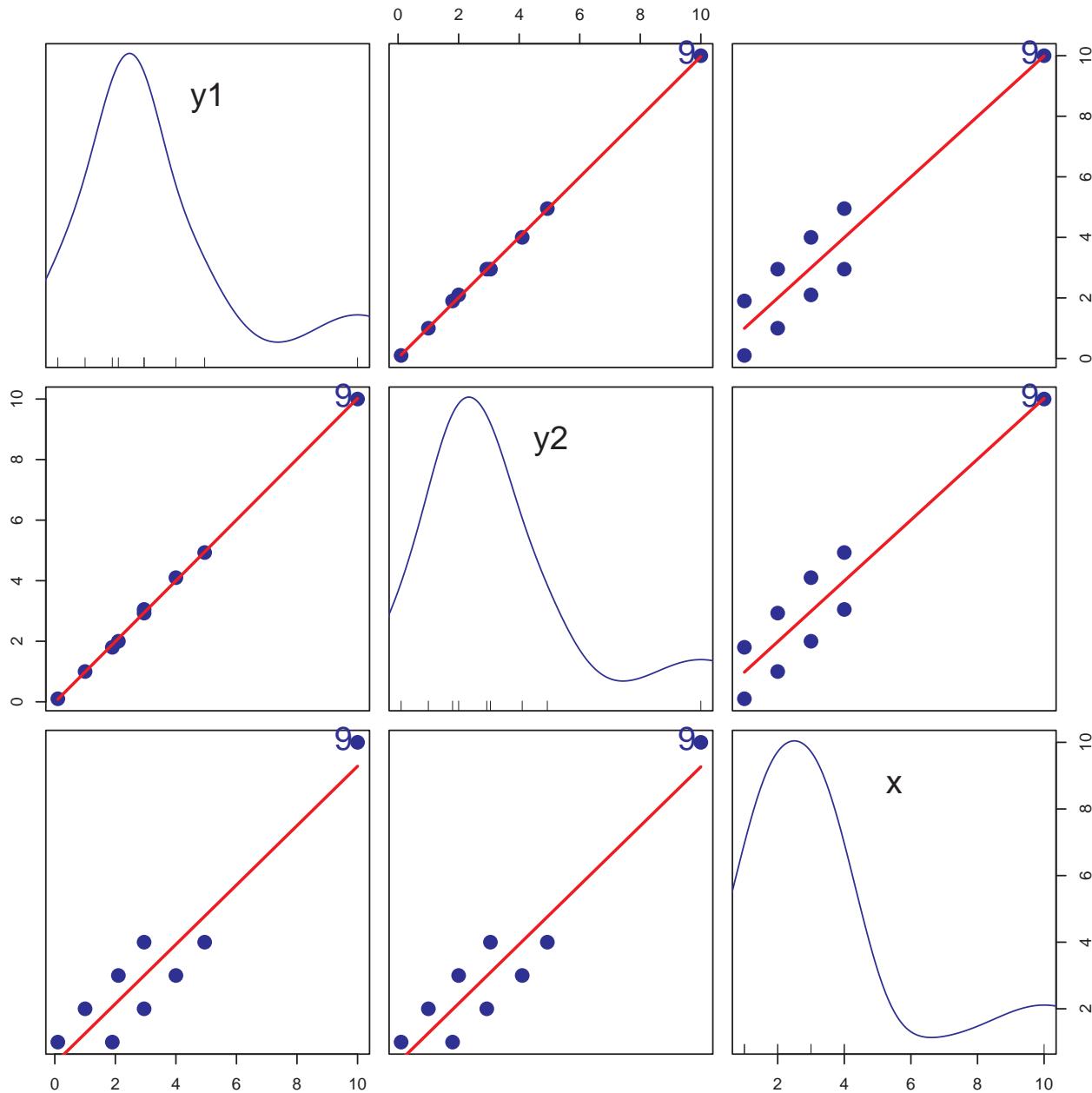


Figure 13.1: Scatterplot matrix for the toy example.

For this example, we fit the univariate models with y_1 and y_2 separately and then the multivariate model.

```
Toy.lm1 <- lm(y1 ~ x, data=Toy)
Toylm2 <- lm(y2 ~ x, data=Toy)
Toymlm <- lm(cbind(y1, y2) ~ x, data=Toy)
```

13.2.1 Cook's D

First, let's examine the Cook's D statistics for the models. Note that the function `cooks.distance()` invokes `stats::cooks.distance.lm()` for the univariate response models, but invokes `mvinfluence::cooks.distance.mlm()` for the multivariate model.

```
df <- Toy
df$D1 <- cooks.distance(Toy.lm1)
df$D2 <- cooks.distance(Toy.lm2)
df$D12 <- cooks.distance(Toy.mlm)

df
#> # A tibble: 9 x 7
#>   case     x     y1     y2      D1      D2     D12
#>   <int> <dbl> <dbl> <dbl>    <dbl>    <dbl>   <dbl>
#> 1     1     1   0.1   0.1   0.121   0.118   0.125
#> 2     2     1   1.9   1.8   0.124   0.103   0.298
#> 3     3     2     1     1   0.0901   0.0886   0.0906
#> 4     4     2   2.95   2.93   0.0829   0.0819   0.0831
#> 5     5     3   2.1     2   0.0548   0.0673   0.182
#> 6     6     3     4     4.1   0.0692   0.0852   0.232
#> 7     7     4   2.95   3.05   0.0793   0.0651   0.203
#> 8     8     4   4.95   4.93   0.0665   0.0643   0.0690
#> 9     9    10    10    10   0.00159   0.00830  3.22
```

The only thing remarkable here is for case 9: The univariate Cook's Ds, D1 and D2 are very small, yet the multivariate statistic, D12=3.22 is over 10 times the next largest value.

Let's see how case 9 stands out in the influence plots (Figure 13.2). It has an extreme hat value. But, because it's residual is very small, it does not have much influence on the fitted models for either y_1 or y_2 . Neither of these plots suggest that anything is terribly wrong with the univariate models—none of the points are in the “danger” zones of the upper- and lower-right corners.

```
ip1 <- car::influencePlot(Toy.lm1,
                           id = list(cex=1.5), cex.lab = 1.5)
ip2 <- car::influencePlot(Toy.lm2,
                           id = list(cex=1.5), cex.lab = 1.5)
```

TODO: Check how these are defined in `mvinfluence`

Contrast these results with what we get for the model for y_1 and y_2 jointly (Figure 13.3) In the multivariate version, `mvinfluence::influencePlot.mlm()` plots the squared studentized residual (denoted R in the output) against the hat value; this is referred to as a ‘type = “stres” plot.¹ Case 9 stands in Figure 13.3 out as wildly influential on the joint regression model. But there's more: The cases in Figure 13.2 with large Cook's D (bubble size) have only tiny influence in the multivariate model.

```
influencePlot(Toy.mlm, type = "stres",
              id.n=2, id.cex = 1.3,
              cex.lab = 1.5)
#>      H      Q CookD      L      R
#> 1 0.202 0.177 0.125 0.253 0.222
```

¹Similar to the univariate version, hat values greater than 2 or 3 times their average, $\bar{h} = p/n$ here, are considered large in the multivariate case. Values of the squared studentized residual R_i are calibrated by the Beta distribution, $\text{Beta}(\alpha = 0.95, q/2, (n - p - q)/2)$.

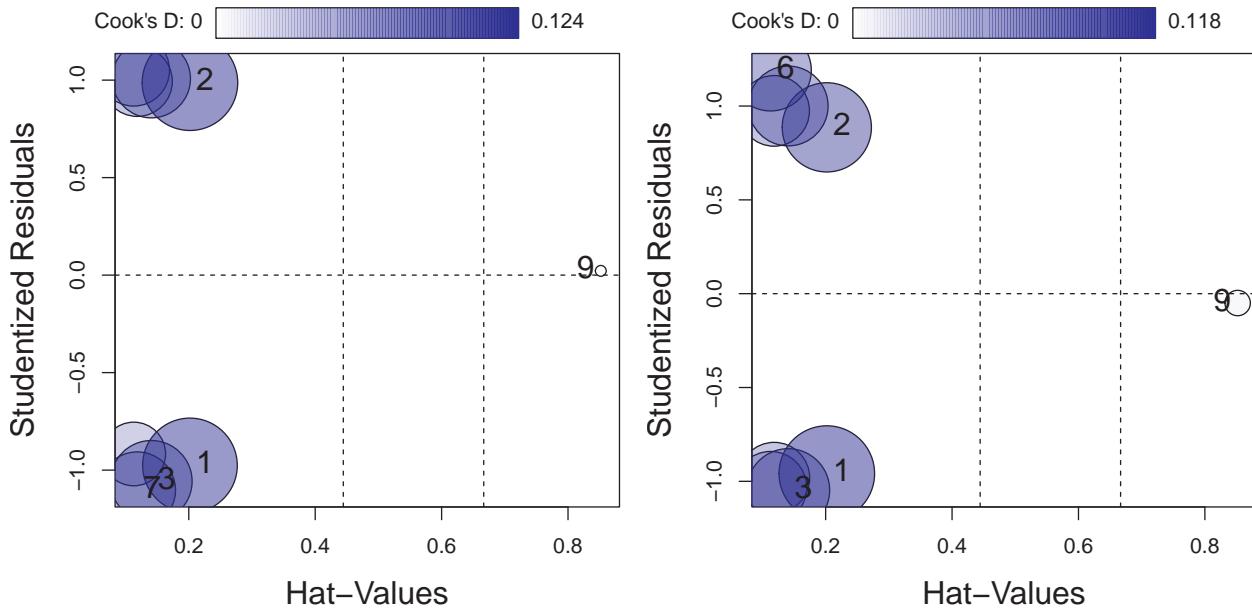


Figure 13.2: Influence plots for the univariate models for y_1 and y_2

```
#> 2 0.202 0.422 0.298 0.253 0.529
#> 6 0.113 0.587 0.232 0.127 0.662
#> 9 0.852 1.082 3.225 5.750 7.301
```

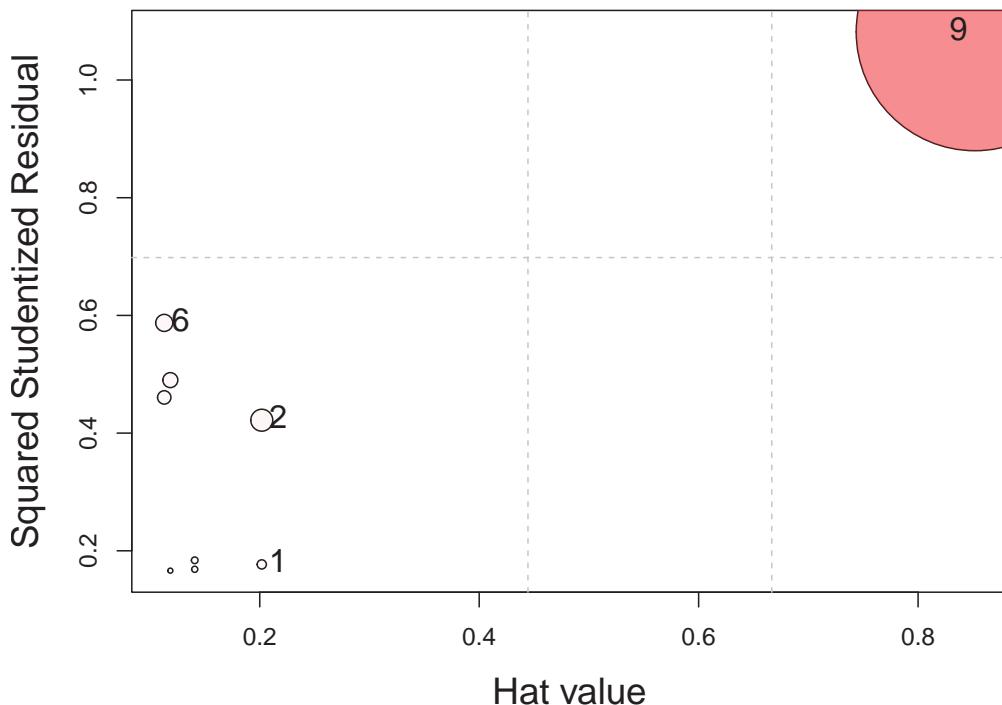


Figure 13.3: Studentized residual influence plot for the multivariate model $(y_1, y_2) \sim x$. Dotted vertical lines mark large hat values, $H > 2, 3p/n$. The dotted horizontal line marks large values of the squared studentized residual.

Theory into Practice

Chairman Mao said, “Theory into practice”, but Tukey (1959) said that, “The practical power of a statistical test is the product of its’ statistical power and the probability of use”. The story for multivariate influence here illustrates a nice feature of the connections between statistical theory, graphic development and implemented in software here.

A statistical development proposes a new way of thinking about a problem. People with a graphical bent look at this and think, “How can I visualize this?”. A software developer then solves the remaining problem of how to incorporate that into easy-to-use functions or applications. If only this was easy, but sometimes, all three roles appear within a given person.

The general formulation of Barrett (2003) suggests an alternative form of the multivariate influence plot (Figure 13.4) that uses the leverage (L) and residual (R) components (`type = "LR"`) directly.

Because influence is the product of leverage and residual, a plot of $\log(L)$ versus $\log(R)$ has the attractive property that contours of constant Cook’s distance fall on diagonal lines with slope = -1. Adjacent reference lines represent constant *multiples* of influence.

```
influencePlot(Toy.mlm, type="LR",
              id.n=2, id.cex = 1.3,
              cex.lab = 1.5) -> infl
```

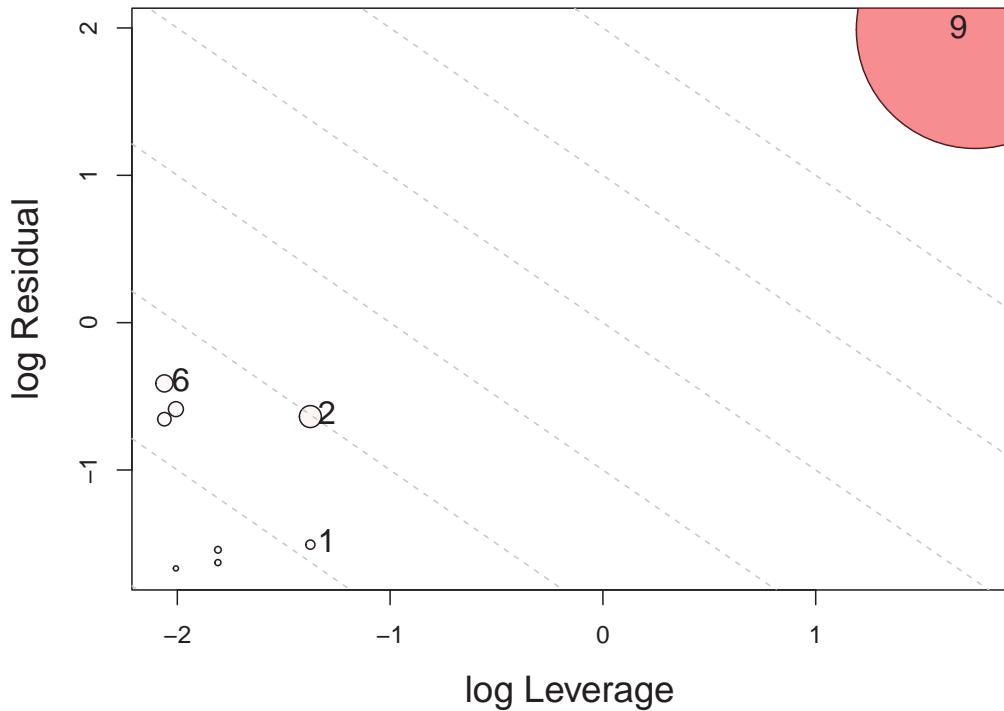


Figure 13.4: LR plot of $\log(L)$ versus $\log(R)$ for the multivariate model $(y_1, y_2) \sim x$. Dotted lines show contours of constant Cook’s distance.

13.2.2 DFBETAS

The DFBETAS statistics give the estimated change in the regression coefficients when each case is deleted in turn. We can gain some insight as to why case 9 is unremarkable in the univariate regressions by plotting these, shown in Figure 13.5. The values come from `stats::dfbetas()` and return the standardized values.

```

db1 <- as.data.frame(dfbetas(Toy.lm1))
gg1 <- ggplot(data = db1, aes(x=`(Intercept)`, y=x, label=rownames(db1))) +
  geom_point(size=1.5) +
  geom_label(size=6, fill="pink") +
  xlab(expression(paste("Deletion Intercept ", b[0]))) +
  ylab(expression(paste("Deletion Slope ", b[1]))) +
  ggtitle("dfbetas for y1") +
  theme_bw(base_size = 16)

db2 <- as.data.frame(dfbetas(Toy.lm2))
gg2 <- ggplot(data = db2, aes(x=`(Intercept)`, y=x, label=rownames(db2))) +
  geom_point(size=1.5) +
  geom_label(size=6, fill="pink") +
  xlab(expression(paste("Deletion Intercept ", b[0]))) +
  ylab(expression(paste("Deletion Slope ", b[1]))) +
  ggtitle("dfbetas for y2") +
  theme_bw(base_size = 16)

gg1 + gg2

```

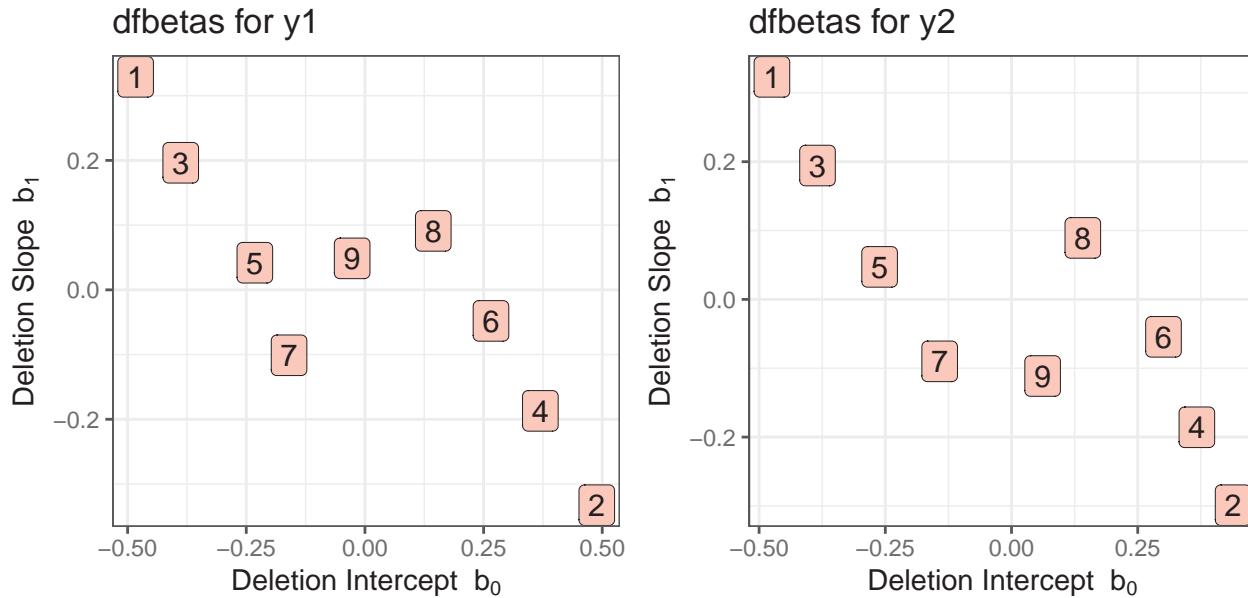


Figure 13.5

The values for case 9 are nearly $(0, 0)$ in both plots, indicating that deleting this case has negligible effect in both *univariate* regressions. Yet, case 9 appeared very influential in the multivariate model. Why did this happen?

In this contrived example, the problem arose from the very high correlation between y_1 and y_2 , $r = 0.9997$ as can be seen in the (y_1, y_2) panel in Figure 13.1. Although each of the y_1 and y_2 values for the high-leverage cases are in-line with the univariate regressions (and thus have small univariate Cook's Ds), the ill-conditioning magnifies small discrepancies in their positions, making the multivariate Cook's D larger. And that's the solution to the Mysterious Case 9.

13.3 Example: NLSY data

The `heplots::NLSY` data introduced in Section 10.5.1 provides a more realistic example of how the *multivariate* influence measures and their plots contribute to understanding multivariate data. Some plots were suggested there (Figure 10.10) to identify “noteworthy” points. The model `NLSY.mod1` fits the child’s reading and math scores using parents’ education:

```
data(NLSY, package = "heplots")
NLSY.mod1 <- lm(cbind(read, math) ~ income + educ,
                 data = NLSY)
```

For influence plots, the `id.method = "noteworthy"` method for point labeling selects observations with large values for *any* of the standardized residual, hat value or Cook’s D, so 6 points are labeled in Figure 13.6.

```
influencePlot(NLSY.mod1,
              id.method = "noteworthy",
              id.n = 3, id.cex = 1.25,
              cex.lab = 1.5)
#>          H      Q   CookD      L      R
#> 12  0.01547 0.090101 0.11149 0.01571 0.091516
#> 19  0.14656 0.004801 0.05629 0.17173 0.005625
#> 54  0.16602 0.016554 0.21986 0.19907 0.019849
#> 142 0.00509 0.208258 0.08478 0.00511 0.209323
#> 152 0.00422 0.049821 0.01682 0.00424 0.050032
#> 221 0.05707 0.000873 0.00399 0.06053 0.000926
```

There is an interesting configuration of the points in Figure 13.6. One group of points (152, 12, 142) is apparent at the left side. These have relatively low leverage (hat value), and increasingly large residuals. Another group of points (221, 19, 54) have small residuals but increasingly large hat values. You should take a moment and compare these points with their positions in Figure 10.10. However, with `id.n = 2` only two points are flagged: 54 and 142.

The LR plot is also instructive here.

```
influencePlot(NLSY.mod1, type = "LR",
              id.method = "noteworthy",
              id.n = 3, id.cex = 1.25,
              cex.lab = 1.5)
#>          H      Q   CookD      L      R
#> 12  0.01547 0.090101 0.11149 0.01571 0.091516
#> 19  0.14656 0.004801 0.05629 0.17173 0.005625
#> 54  0.16602 0.016554 0.21986 0.19907 0.019849
#> 142 0.00509 0.208258 0.08478 0.00511 0.209323
#> 152 0.00422 0.049821 0.01682 0.00424 0.050032
#> 221 0.05707 0.000873 0.00399 0.06053 0.000926
```

In Figure 13.7 the noteworthy points are arrayed within a diagonal band corresponding to contours of equal Cook’s D, and this statistic increases multiplicatively with each step away from the origin. Cases 54 and 142 stand out somewhat here.

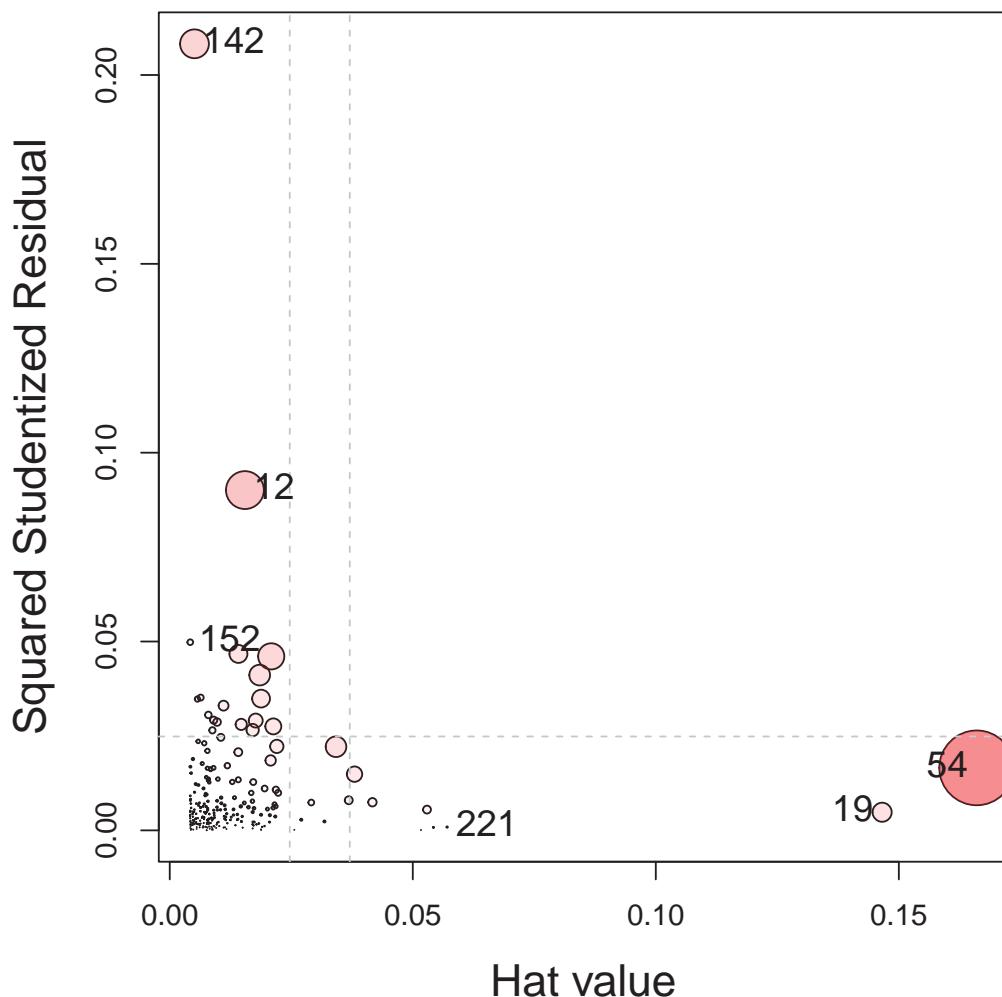


Figure 13.6: Influence plot for the NLSY data...

13.4 Example: Penguin data

Another example illustrates the appearance of influence plots with factors as predictors. Earlier chapters (Section 3.9.2, Section 4.7.1) presented a variety of plots (Figure 3.26, Figure 3.27, Figure 4.33) for the Penguin data in which a few cases were identified as unusual in exploratory analysis. But, are any of them influential in the context of a multivariate model?

Let's consider the simple model predicting `species` from the numeric variables

```
data(peng, package="heplots")
peng.mlm <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~
  species, data=peng)
```

In the influence plot (Figure 13.8) the predictor `species` is of course discrete, so there are only three distinct hat-values and the values for each species appear as columns of points in this plot. For ease of interpretation, I use a little `dplyr` magic to label the species and their sample sizes.

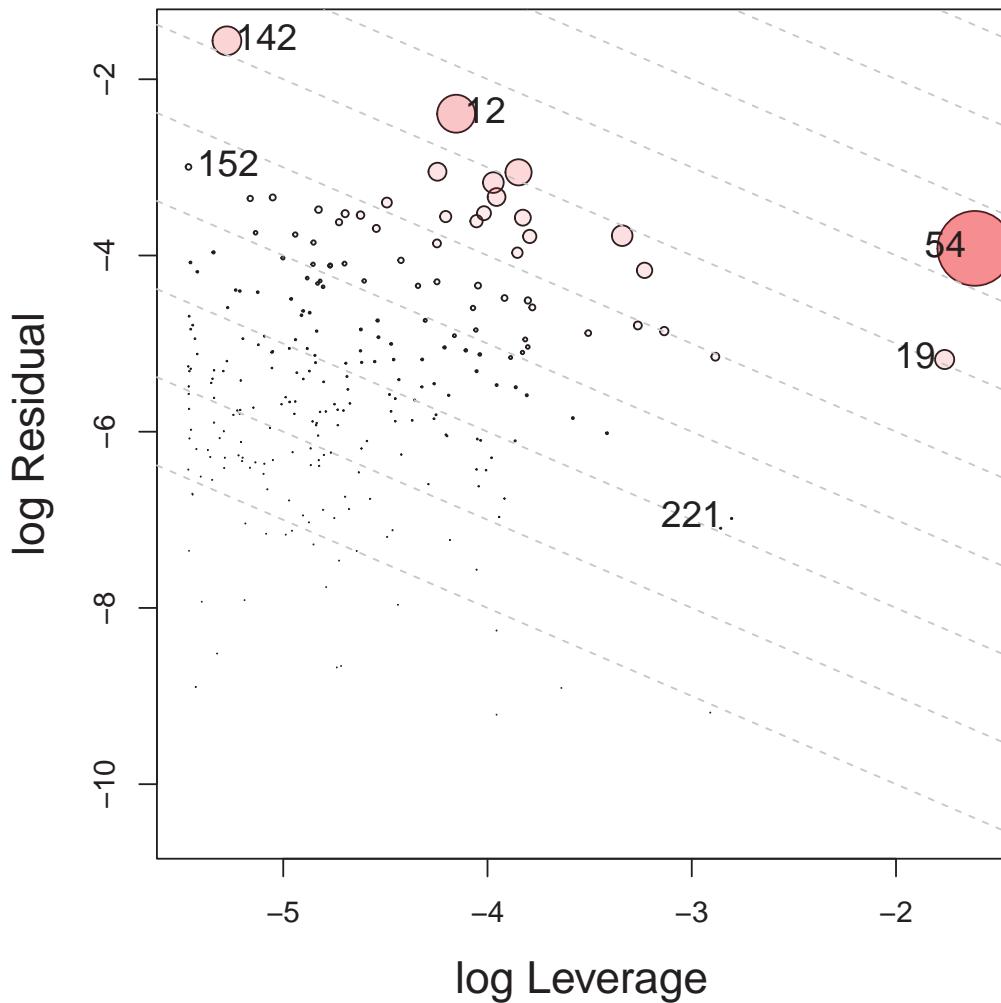


Figure 13.7: Influence plot for the NLSY data...

```

res <- influencePlot(peng.mlm, id.n=3, type="stres")
res |> arrange(desc(CookD)) |> print(digits = 2)
#>      H      Q   CookD      L      R
#> 283 0.0147 0.0919 0.1486 0.0149 0.0932
#> 286 0.0147 0.0322 0.0520 0.0149 0.0327
#> 179 0.0084 0.0512 0.0474 0.0085 0.0517
#> 10  0.0068 0.0562 0.0424 0.0069 0.0566
#> 268 0.0147 0.0079 0.0129 0.0149 0.0081
#> 267 0.0147 0.0037 0.0061 0.0149 0.0038
#> 266 0.0147 0.0021 0.0034 0.0149 0.0021

# labels for species, adding sample n
loc <- merge(peng |> add_count(species),
             res,
             by = "row.names") |>
group_by(species) |>
slice(1) |>

```

```
ungroup() |>
  select(species, H, n) |>
  mutate(label = glue::glue("{species}\n(n={n})"))
  text(loc$H, 0.10, loc$label, xpd=TRUE)
```

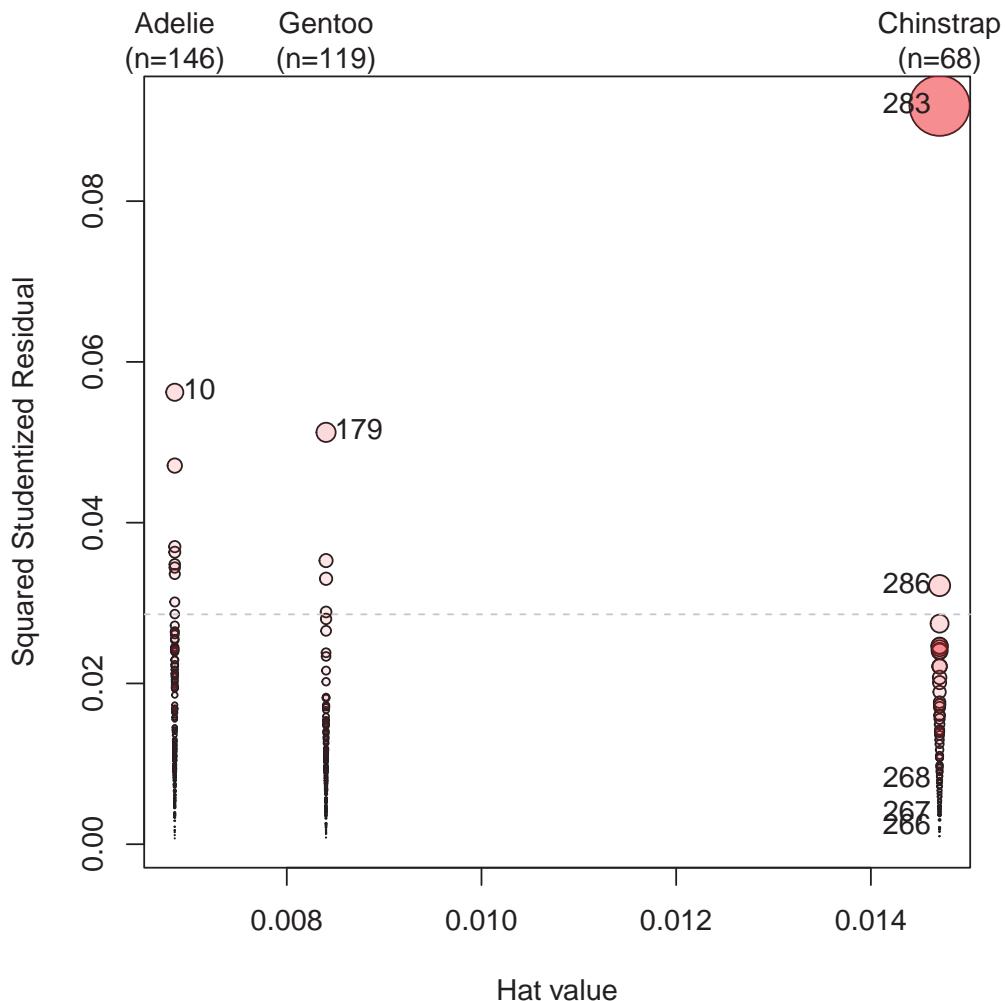


Figure 13.8: Influence plot for the Penguin data, showing the squared studentized residual vs. hat value. Unusual points on either variable or on the Cook's D statistic are identified with their case number.

We know that hat-values are proportional to how unusual the observations are from the means of the predictors, but what is this for a factor, like `species`? The answer is that $H_j \propto 1/n_j$, so Chinstrap with the smallest sample size ($n = 68$) is the most unusual.

The most influential case (283) here is our Chinstrap friend “Cyrano” (see Figure 3.27). Among the others, case 10 is the Adelie bird we labeled “Hook Nose”. To better understand *why* these cases are influential, we can make plots of the data in data space or in PCA space as we did earlier.

13.5 Robust Estimation

Robust methods for multivariate linear models aim to provide reliable parameter estimates and inference procedures that remain stable in the presence of outlying observations. As noted by Peter J. Rousseeuw et al. (2004), “*The main advantage of robust regression is that it provides reliable results even when some of the assumptions of classical regression are violated.*” But this advantage comes at the cost of increased computational complexity.

Robust regression is a compromise between excluding “bad” data points entirely from analysis vs. including all the data and treating them equally in classical OLS regression. The essential idea of robust regression is to weight the observations differently based on how well behaved these observations are. Roughly speaking, it is a form of *weighted* least squares regression. But because the weights are derived from the data, this must be done iteratively, requiring more computation.

Several general approaches have been developed for robust multivariate regression. These include M-estimators, S-estimators (Peter J. Rousseeuw & Yohai, 1984), and MM-estimators (Yohai, 1987). Each approach offers different trade-offs between robustness properties, computational efficiency, and statistical efficiency under ideal conditions. See the CRAN Task View: Robust Statistical Methods for an extensive list of robust methods in R and the vignette for the `rrcov` package for a general overview of multivariate robust methods.

The method implemented in the `heplots::robmlm()` function belongs to the class of **M-estimators**. This generalizes OLS estimation by replacing the least squares criterion with a more robust version. The key idea is to relax the least squares criterion of minimizing $Q(\mathbf{e}) = \sum e_i^2 = \sum (y_i - \hat{y}_i)^2$ by considering more general functions $Q(\mathbf{e}, \rho)$, where the function $\rho(e_i)$ can be chosen to reduce the impact of large outliers. In these terms,

- OLS uses $\rho(e_i) = e_i^2$
- L_1 estimation uses $\rho(e_i) = |e_i|$, the least absolute values
- A bit more complicated, the **biweight** function uses a squared measure of error up to some value c and then levels off thereafter,

$$\rho(e_i) = \begin{cases} \left[1 - \left(\frac{e_i}{c}\right)^2\right]^2 & |e_i| \leq c, \\ 1 & |e_i| > c. \end{cases}$$

These functions are more easily understood in a graph (Figure 13.9). The biweight function (ref?) has a property like Winsorizing—the squared error remains constant for residuals $e_i > c$, with the tuning constant $c = 4.685$ for `MASS::psi.bisquare()`.

But, there is a problem here, in that weighted least squares is designed for situations where the observation weights are *known in advance*, for instance, when data points have unequal variances or when some observations are more reliable than others.

The solution, called *iteratively reweighted least squares* (IRLS), is to substitute computation for theory, and iterate between estimating the coefficients (with given weights) and determining weights for the observations in the next iteration. This is implemented in `heplots::robmlm()`, which generalizes the methods used in `MASS::rlm()`. . . .

13.6 Example: Penguin data

In earlier analyses of the Penguin data, a few points appeared unusual for their species in various plots (Figure 3.21, Figure 3.27) and a few were deemed overly influential in Section 13.4 for the simple model

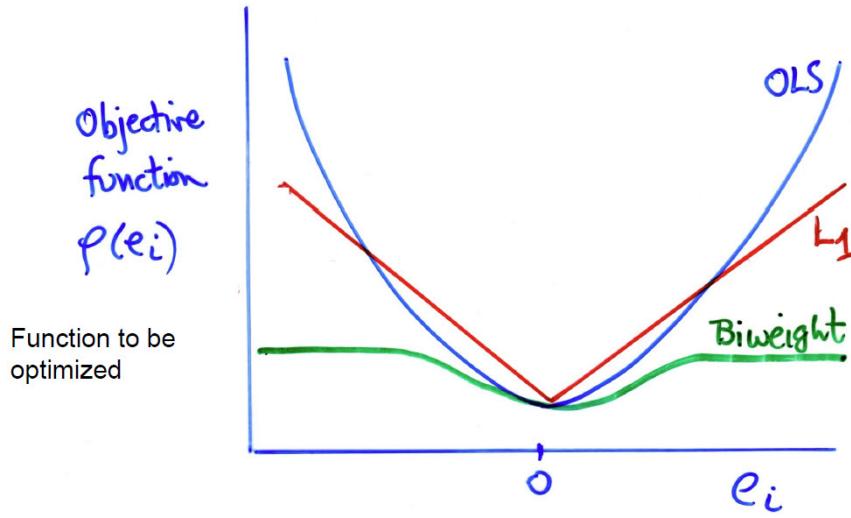


Figure 13.9: Diagram plotting the function $\rho(e_i)$ of the contributions of the residuals e_i to what is minimized in various fitting methods.

`peng.mlm` predicting `species`. What effect does fitting a robust MANOVA have on the results? We can just substitute `robmelm()` for `lm()`. to fit the robust model.

```
peng.rlm <- robmelm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~
                      species, data=peng)
```

The `plot()` method for a "robmelm" object give an index plot of the observation weights in the final iteration. I take some care here to color the points by species, label the points with the lowest weights and add my names for a couple of birds.

```
source(here::here("R", "penguin", "penguin-colors.R"))
col = peng.colors("dark")[peng$species]
plot(peng.rlm,
      segments = TRUE,
      id.weight = 0.6,
      col = col,
      cex.lab = 1.3)
# label old friends
notables <- tibble(
  id = c(10, 283),
  name = c("HookNose", "Cyrano"),
  wts = peng.rlm$weights[id])
text(notables$id, notables$wts,
     label = notables$name, pos = 3,
     xpd = TRUE)
# label species in axis at the top
ctr <- split(seq(nrow(peng)), peng$species) |> lapply(mean)
axis(side = 3, at=ctr, labels = names(ctr), cex.axis=1.2)
```

The observations that are labeled here are among those which stood out as notable in other plots. In Figure 13.10, the lowest four points are down-weighted considerably. Our friend "Cyrano" (283), the greatest

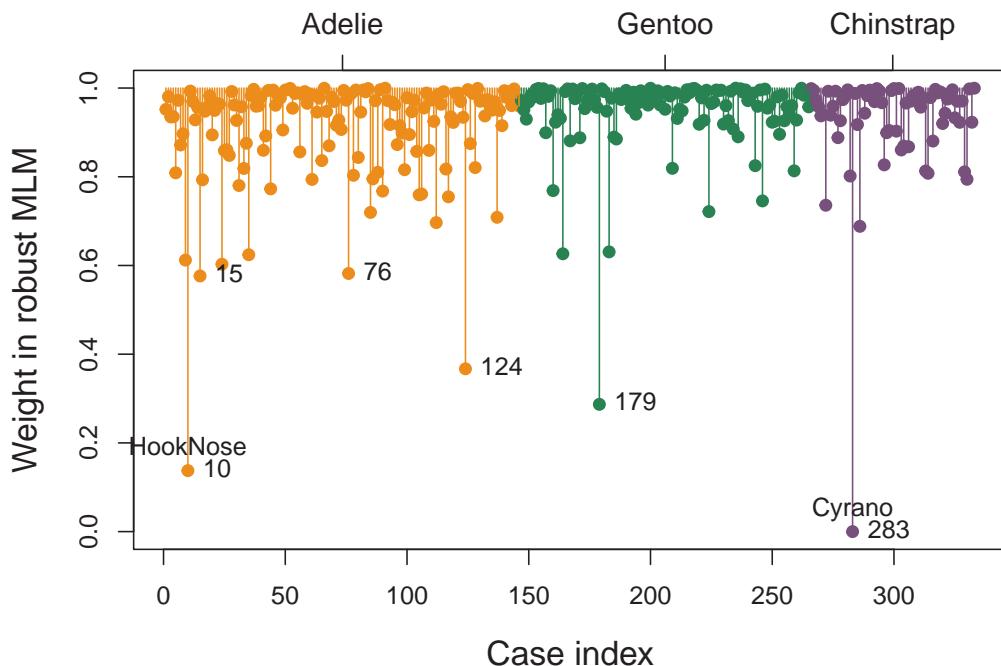


Figure 13.10: Index plot of the observation weights for the robust model, `peng.rlm`. Observations with weights < 0.5 are labeled with their case number.

outlier, gets a weight of exactly zero, so is effectively excluded from the model; “Hook Nose” (10) gets a weight of 0.13, and so he can’t do much damage.

```
cases <- c(10, 124, 179, 283)
lowwt <- peng.rlm$weights[cases]
names(lowwt) <- cases
lowwt
#>   10    124    179    283
#> 0.137 0.367 0.287 0.000
```

Another way to assess the effect of using a robust model is to examine its’ effects on the model coefficients. `heplots::rel_diff()` calculates the relative difference, $|x - y|/x$, expressed here as percent of change. As you can see below, the changes in most of the coefficients is rather small.

```
rel_diff(coef(peng.mlm), coef(peng.rlm)) |>
  print(digits=2)
#>           bill_length bill_depth flipper_length body_mass
#> (Intercept)      0.083       0.26       0.092      0.43
#> speciesChinstrap 0.572     78.10      8.053     76.20
#> speciesGentoo    0.857      -0.51      0.089      0.12
```

The largest differences are for the coefficients for `Chinstrap`, particularly for `bill_depth` and `body_mass`, which reflect the dramatic effect of effectively removing “Cyrano” from the analysis.

If you accept the premise of down-weighting cases with large residuals in robust methods, `robmle()` provides a principled way to handle them. Bye-bye Cyrano; it’s been fun, but we can get along nicely without you.

14

Case studies

This chapter presents some complete analyses of datasets that will be prominent in the book. Some of this material may later be moved to earlier chapters.

Packages

In this chapter I use the following packages. Load them now.

```
library(car)
library(helplots)
library(candisc)
library(ggplot2)
library(dplyr)
library(tidyr)
library(corrgram)
library(ggbiplot)

scatterplotMatrix(~ Speed + Attention + Memory + Verbal + Visual + ProbSolv | Dx,
  data=NeuroCog,
  plot.points = FALSE,
  smooth = FALSE,
  legend = FALSE,
  col = scales::hue_pal()(3),
  ellipse=list(levels=0.68))
```

In this figure, we can see that the regression lines have similar slopes and similar data ellipses for the groups, though with a few exceptions. You can also see that the **Normal** group has higher means on all the variables and that the **Schizophrenic** and **SchizoAffective** don't seem to differ very much.

Biplot

While still in exploratory mode, we can gain greater insight with our trusty multivariate juicer, the biplot (Section 4.3). A PCA of the cognitive variables shows that nearly 80% of the total variance is accounted for by two dimensions, of which the first dimension accounts for 69%.

```
neuro.pca <- NeuroCog |>
  select(Speed:Visual) |>
  prcomp(scale. = TRUE)
summary(neuro.pca)

#> Importance of components:
#>              PC1    PC2    PC3    PC4    PC5
#> Standard deviation   1.856  0.720  0.629  0.5761  0.5570
#> Proportion of Variance 0.689  0.104  0.079  0.0664  0.0621
#> Cumulative Proportion 0.689  0.793  0.872  0.9379  1.0000
```

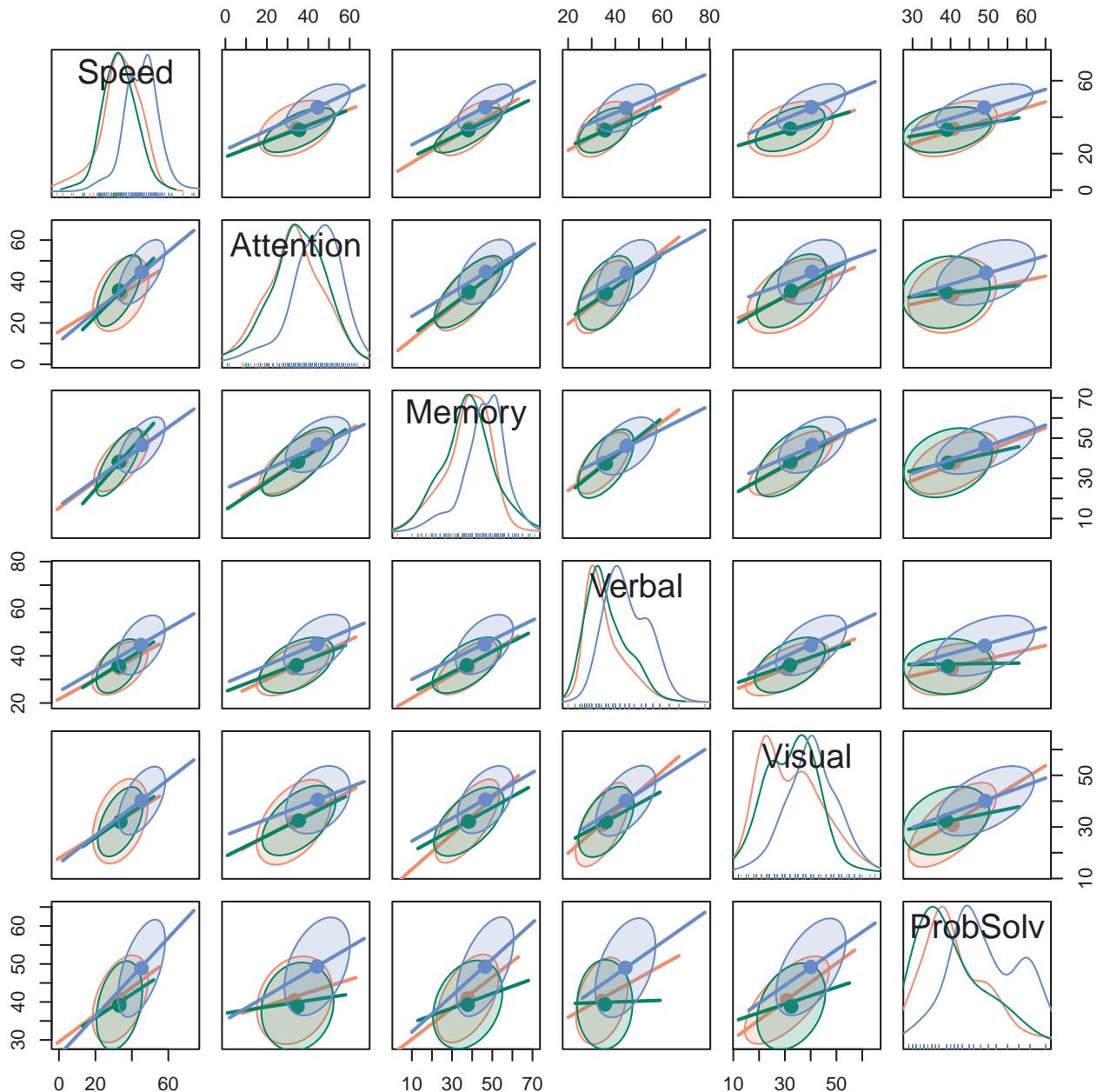


Figure 14.1: Scatterplot matrix of the NeuroCog data. Points are suppressed here, focusing on the data ellipses and regression lines. Colors for the groups: Schizophrenic (red), SchizoAffective (green), Control (blue).

The biplot for this analysis (Figure 14.2) provides a nice summary of what was seen in the scatterplot matrix (Figure 14.1): A large difference between the **Control** group and the others, which don't differ much from each other.

```
ggbiplot(neuro.pca,
  obs.scale = 1, var.scale = 1,
  groups = NeuroCog$Dx,
  varname.size = 5,
```

```

var.factor = 1.5,
ellipse = TRUE) +
scale_color_discrete(name = 'groups') +
theme_minimal() +
theme(legend.direction = 'horizontal',
legend.position = 'top')

```

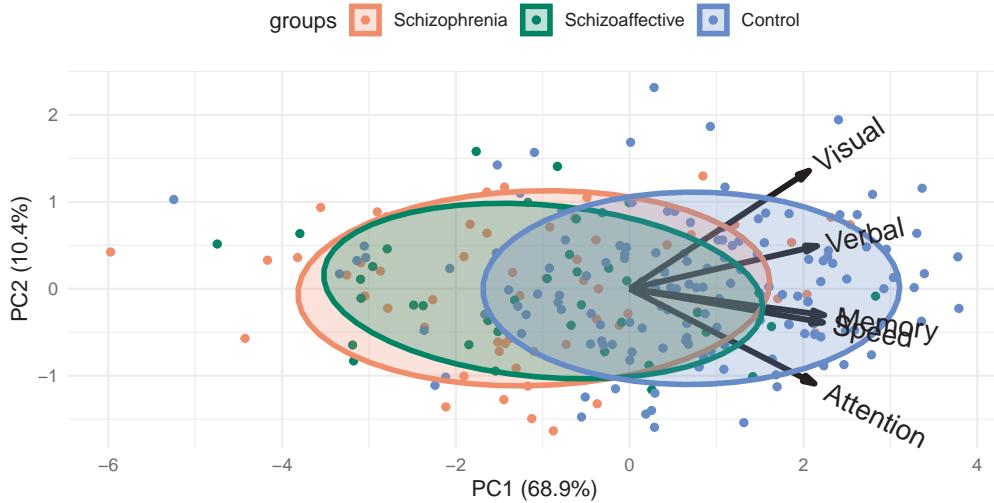


Figure 14.2: Biplot of the NeuroCog data. . .

The variable vectors in Figure 14.2 reflect the correlations of the response variables with each other and with the principal components, PC1 and PC2. All of the variables are positively correlated in this 2D view, **Memory** and **Speed** most highly so.

14.1 Fitting the MLM

We proceed to fit the one-way MANOVA model to answer the main research question of how well these variables distinguish among the groups.

```

NC.mlm <- lm(cbind(Speed, Attention, Memory, Verbal, Visual, ProbSolv) ~ Dx,
               data=NeuroCog)
Anova(NC.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>   Df test stat approx F num Df den Df Pr(>F)
#>   Dx  2      0.299      6.89      12     470 1.6e-11 ***
#>   ---
#>   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The first research question is captured by the contrasts for the **Dx** factor shown above. We can test these with `car::linearHypothesis()`. The contrast **Dx1** for control vs. the diagnosed groups is highly significant,

```
# control vs. patients
print(linearHypothesis(NC.mlm, "Dx1"), SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.289    15.9     6   234 2.8e-15 ***
#> Wilks       1   0.711    15.9     6   234 2.8e-15 ***
#> Hotelling-Lawley 1   0.407    15.9     6   234 2.8e-15 ***
#> Roy         1   0.407    15.9     6   234 2.8e-15 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

but the second contrast, Dx2, comparing the schizophrenic and schizoaffective group, is not.

```
# Schizo vs SchizAff
print(linearHypothesis(NC.mlm, "Dx2"), SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.006    0.249     6   234 0.96
#> Wilks       1   0.994    0.249     6   234 0.96
#> Hotelling-Lawley 1   0.006    0.249     6   234 0.96
#> Roy         1   0.006    0.249     6   234 0.96
```

As a quick check on the model, a χ^2 QQ plot (Figure 14.3) reveals no problems with multivariate normality of residuals nor potentially harmful residuals.

```
cqplot(NC.mlm, id.n = 3)
```

14.1.1 HE plot

So the question becomes: how to understand these results.

`heplot()` shows the visualization of the multivariate model in the space of two response variables (the first two by default). The result (Figure 14.4) tells a very simple story: The control group performs higher on higher measures than the diagnosed groups, which do not differ between themselves.

(For technical reasons, to abbreviate the group labels in the plot, we need to `update()` the MLM model after the labels are reassigned.)

```
# abbreviate levels for plots
NeuroCog$Dx <- factor(NeuroCog$Dx,
                        labels = c("Schiz", "SchAff", "Contr"))
NC.mlm <- update(NC.mlm)
```

Then, feed the model to `heplot()` for a plot of the first two response variables, `Speed` and `Attention`.

```
heplot(NC.mlm,
       fill=TRUE, fill.alpha=0.1,
       cex.lab=1.3, cex=1.25)
par(op)
```

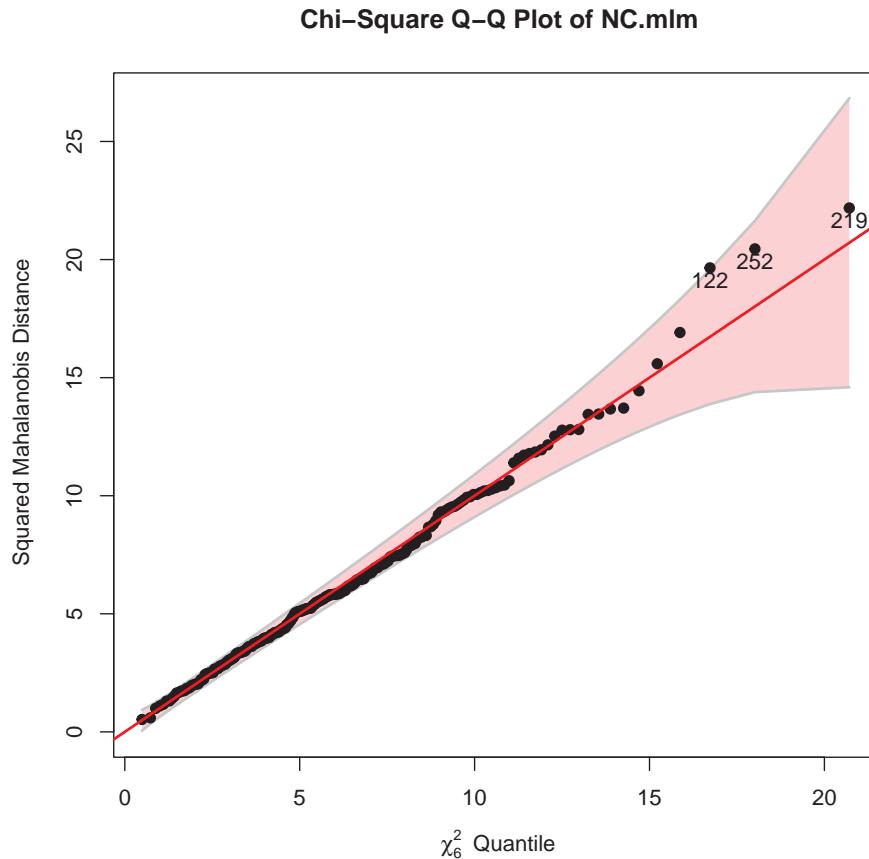


Figure 14.3: Chisquare QQ plot of the MANOVA model

This pattern, of the control group higher than the others, is consistent across all of the response variables, as we see from a plot of `pairs(NC.mlm)`:

```
pairs(NC.mlm,
      fill=TRUE, fill.alpha=0.1,
      var.cex=2)
```

It signals that we are likely to see a simpler representation of the data in canonical space.

14.1.2 Canonical space

We can gain further insight, and a simplified plot showing all the response variables by projecting the MANOVA into the canonical space, which is entirely 2-dimensional (because $df_h = 2$). However, the output from `candisc()` shows that 98.5% of the mean differences among groups can be accounted for in just one canonical dimension.

```
NC.can <- candisc(NC.mlm)
NC.can
#>
#> Canonical Discriminant Analysis for Dx:
#>
#>    CanRsq Eigenvalue Difference Percent Cumulative
```

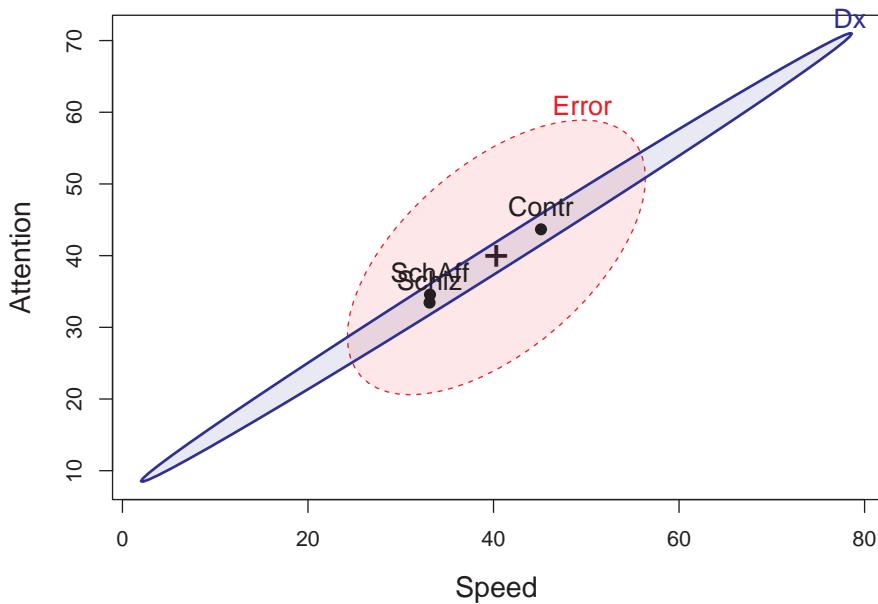


Figure 14.4: HE plot of Speed and Attention in the MLM for the NeuroCog data. The labeled points show the means of the groups on the two variables. The blue H ellipse for groups indicates the strong positive correlation of the group means.

```
#> 1 0.29295     0.41433      0.408     98.5      98.5
#> 2 0.00625     0.00629      0.408      1.5    100.0
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#> LR test stat approx F numDF denDF Pr(> F)
#> 1        0.703     7.53     12    468   9e-13 ***
#> 2        0.994     0.30      5    235     0.91
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 14.6 is the result of the `plot()` method for class "candisc" objects, that is, the result of calling `plot(NC.can, ...)`. It plots the two canonical scores, $Z_{n \times 2}$ for the subjects, together with data ellipses for each of the three groups.

```
pos <- c(4, 1, 4, 4, 1, 3)
col <- c("red", "darkgreen", "blue")
plot(NC.can,
      ellipse=TRUE,
      rev.axes=c(TRUE, FALSE),
      pch=c(7, 9, 10),
      var.cex=1.2, cex.lab=1.5, var.lwd=2, scale=4.5,
      col=col,
      var.col="black", var.pos=pos,
      prefix="Canonical dimension ")
```

The interpretation of Figure 14.6 is again fairly straightforward. As noted earlier (Section 11.7), the projections

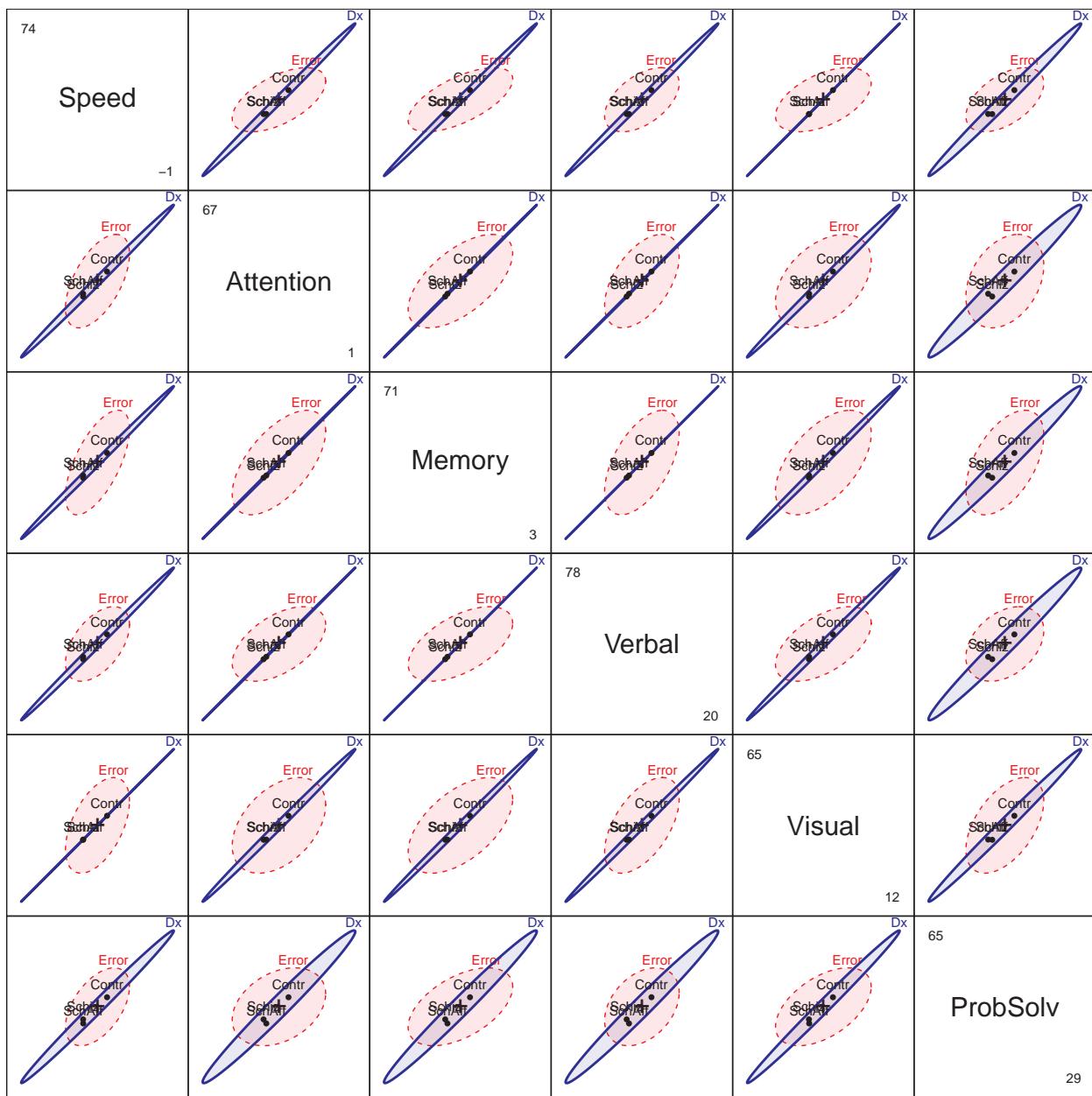


Figure 14.5: HE plot matrix of the MLM for NeuroCog data.

of the variable vectors in this plot on the coordinate axes are proportional to the correlations of the responses with the canonical scores. From this, we see that the normal group differs from the two patient groups, having higher scores on all the neurocognitive variables, most of which are highly correlated. The problem solving measure is slightly different, and this, compared to the cluster of memory, verbal and attention, is what distinguishes the schizophrenic group from the schizoaffectives.

The separation of the groups is essentially one-dimensional, with the control group higher on all measures. Moreover, the variables processing speed and visual memory are the purest measures of this dimension, but all variables contribute positively. The second canonical dimension accounts for only 1.5% of group mean differences and is non-significant (by a likelihood ratio test). Yet, if we were to interpret it, we would note

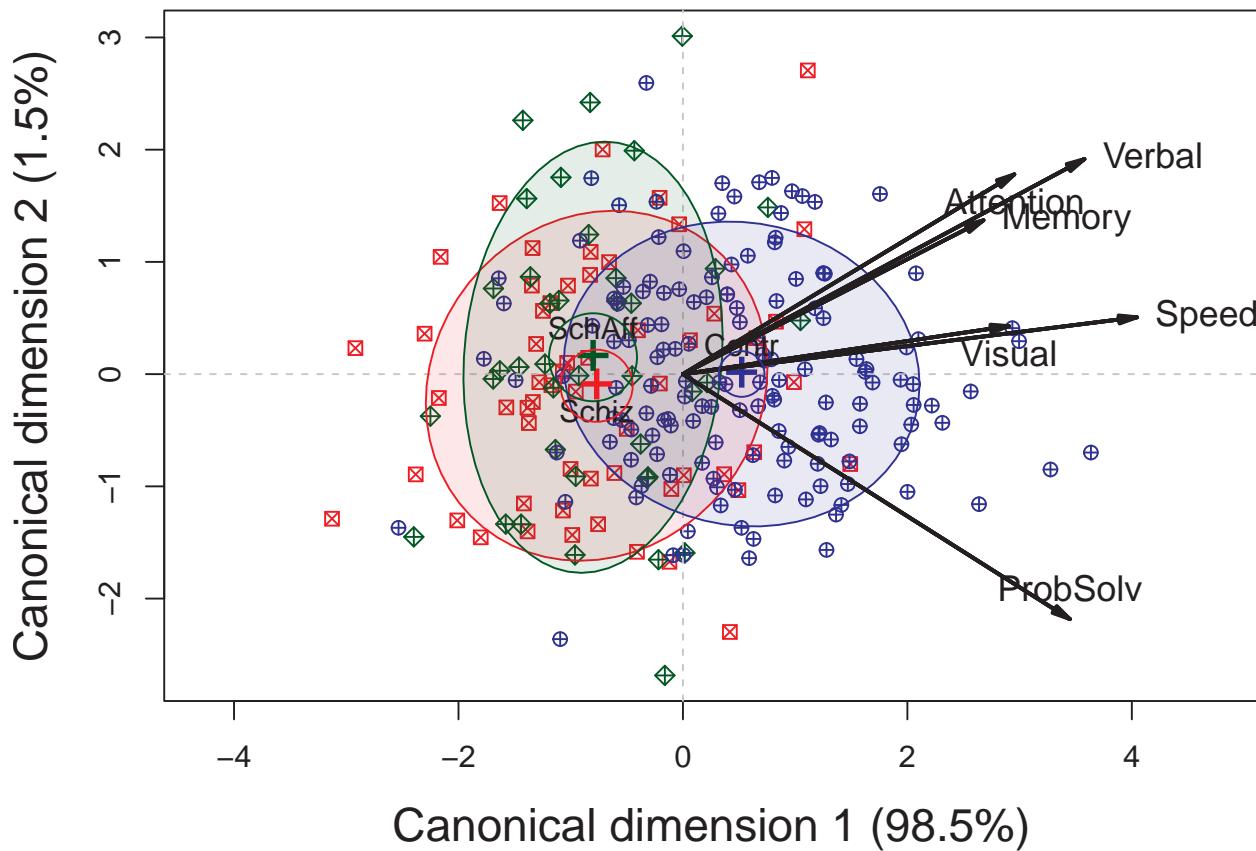


Figure 14.6: Canonical discriminant plot for the NeuroCog data MANOVA. Scores on the two canonical dimensions are plotted, together with 68% data ellipses for each group.

that the schizophrenia group is slightly higher on this dimension, scoring better in problem solving and slightly worse on working memory, attention, and verbal learning tasks.

Summary

This analysis gives a very simple description of the data, in relation to the research questions posed earlier:

- On the basis of these neurocognitive tests, the schizophrenic and schizoaffective groups do not differ significantly overall, but these groups differ greatly from the normal controls.
- All cognitive domains distinguish the groups in the same direction, with the greatest differences shown for the variables most closely aligned with the horizontal axis in Figure 14.6.

14.2 Social cognitive measures

The social cognitive measures were designed to tap various aspects of the perception and cognitive processing of emotions of others. Emotion perception was assessed using a Managing Emotions score from the MCCB. A “theory of mind” (ToM) score assessed ability to read the emotions of others from photographs of the eye region of male and female faces. Two other measures, externalizing bias (**ExtBias**) and personalizing bias (**PersBias**) were calculated from a scale measuring the degree to which individuals attribute internal, personal or situational causal attributions to positive and negative social events.

The analysis of the `SocialCog` data proceeds in a similar way: first we fit the MANOVA model, then test the overall differences among groups using `Anova()`. We find that the overall multivariate test is again significant,

```
data(SocialCog, package="heplots")
SC.mlm <- lm(cbind(MgeEmotions, ToM, ExtBias, PersBias) ~ Dx,
              data=SocialCog)
Anova(SC.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>   Df test stat approx F num Df den Df Pr(>F)
#>   Dx  2      0.212     3.97      8    268 0.00018 ***
#>   ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Testing the same two contrasts using `linearHypothesis()` (results not shown), we find that the overall multivariate test is again significant, but now *both* contrasts are significant ($Dx1: F(4, 133) = 5.21, p < 0.001$; $Dx2: F(4, 133) = 2.49, p = 0.0461$), the test for $Dx2$ just barely.

```
# control vs. patients
print(linearHypothesis(SC.mlm, "Dx1"), SSP=FALSE)
# Schizo vs. SchizAff
print(linearHypothesis(SC.mlm, "Dx2"), SSP=FALSE)
```

These results are important, because, if they are reliable and make sense substantively, they imply that patients with schizophrenia and schizoaffective diagnoses *can* be distinguished by their performance on tasks assessing social perception and cognition. This was potentially a new finding in the literature on schizophrenia.

As we did above, it is useful to visualize the nature of these differences among groups with HE plots for the `SC.mlm` model. Each contrast has a corresponding **H** ellipse, which we can show in the plot using the `hypotheses` argument. With a single degree of freedom, these degenerate ellipses plot as lines.

```
heplot(SC.mlm,
       hypotheses=list("Dx1"="Dx1", "Dx2"="Dx2"),
       fill=TRUE, fill.alpha=.1,
       cex.lab=1.5, cex=1.2)
```

It can be seen that the three group means are approximately equally spaced on the `ToM` measure, whereas for `MgeEmotions`, the control and schizoaffective groups are quite similar, and both are higher than the schizophrenic group. This ordering of the three groups was somewhat similar for the other responses, as we could see in a `pairs(SC.mlm)` plot.

14.2.1 Model checking

Normally, we would continue this analysis, and consider other HE and canonical discriminant plots to further interpret the results, in particular the relations of the cognitive measures to group differences, or perhaps an analysis of the relationships between the neuro- and social-cognitive measures. We don't pursue this here for reasons of length, but this example actually has a more important lesson to demonstrate.

Before beginning the MANOVA analyses, extensive data screening was done by the client using SPSS, in which all the response *and* predictor variables were checked for univariate normality and multivariate normality

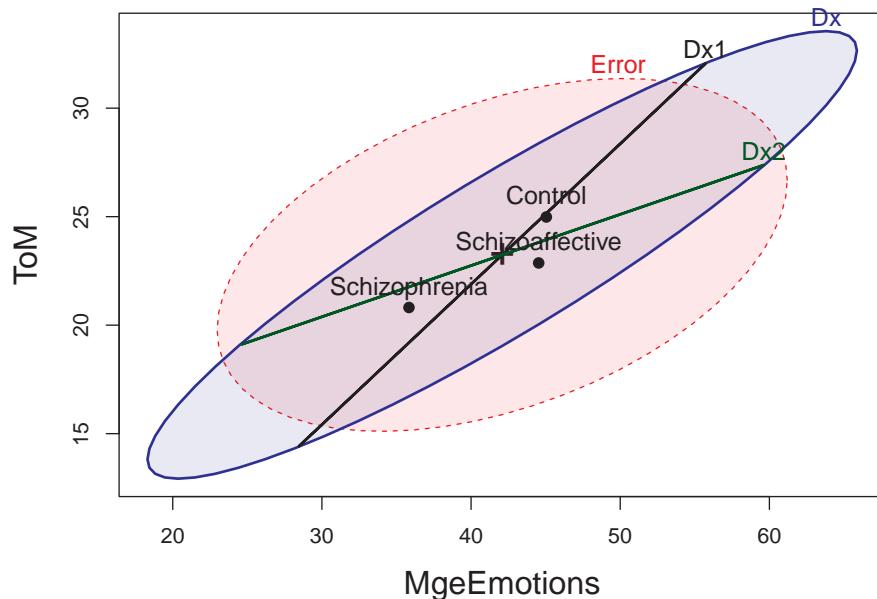


Figure 14.7: HE plot of Speed and Attention in the MLM for the `SocialCog` data. The labeled points show the means of the groups on the two variables. The lines for `Dx1` and `Dx2` show the tests of the contrasts among groups.

(MVN) for both sets. This traditional approach yielded a huge amount of tabular output and no graphs, and did not indicate any major violation of assumptions.¹

A simple visual test of MVN and the possible presence of multivariate outliers is related to the theory of the data ellipse: Under MVN, the squared Mahalanobis distances $D_M^2(\mathbf{y}) = (\mathbf{y} - \bar{\mathbf{y}})' \mathbf{S}^{-1} (\mathbf{y} - \bar{\mathbf{y}})$ should follow a χ_B^2 distribution. Thus, a quantile-quantile plot of the ordered D_M^2 values vs. corresponding quantiles of the χ_B^2 distribution should approximate a straight line (Cox, 1968; Healy, 1968). Note that this should be applied to the *residuals* from the model – `residuals(SC.mlm)` – and not to the response variables directly.

`heplots::cqplot()` implements this for "mle" objects Calling this function for the model `SC.mlm` produces Figure 14.8. It is immediately apparent that there is one extreme multivariate outlier; three other points are identified, but the remaining observations are nearly within the 95% confidence envelope (using a robust MVE estimate of \mathbf{S}).

```
cqplot(SC.mlm, method="mve",
       id.n=4,
       main="",
       cex.lab=1.25)
```

Further checking revealed that this was a data entry error where one case (15) in the schizophrenia group had a score of -33 recorded on the `ExtBias` measure, whose valid range was (-10, +10). In R, it is very easy to re-fit a model to a subset of observations (rather than modifying the dataset itself) using `update()`. The result of the overall Anova and the test of `Dx1` were unchanged; however, the multivariate test for the most interesting contrast `Dx2` comparing the schizophrenia and schizoaffective groups became non-significant at the $\alpha = 0.05$ level ($F(4, 133) = 2.18, p = 0.0742$).

¹Actually, multivariate normality of the predictors in \mathbf{X} is not required in the MLM. This assumption applies only to the conditional values $\mathbf{Y} | \mathbf{X}$, i.e., that the errors $\epsilon_i' \sim \mathcal{N}_p(\mathbf{0}, \Sigma)$ with constant covariance matrix. Moreover, the widely used MVN test statistics, such as Mardia's (1970) test based on multivariate skewness and kurtosis are known to be quite sensitive to mild departures in kurtosis (Mardia, 1974) which do not threaten the validity of the multivariate tests.

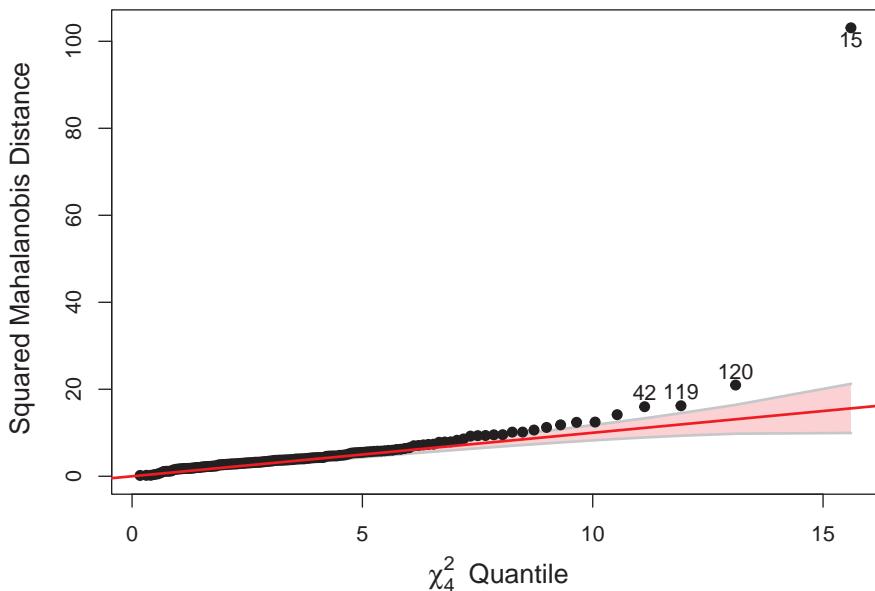


Figure 14.8: Chi-square quantile-quantile plot for residuals from the model `SC.mlm`. The confidence band gives a point-wise 95% envelope, providing information about uncertainty. One extreme multivariate outlier is highlighted.

```
SC.mlm1 <- update(SC.mlm,
                     subset=rownames(SocialCog) != "15")

Anova(SC.mlm1)
print(linearHypothesis(SC.mlm1, "Dx1"), SSP=FALSE)
print(linearHypothesis(SC.mlm1, "Dx2"), SSP=FALSE)
```

14.2.2 Canonical HE plot

This outcome creates a bit of a quandry for further analysis (do univariate follow-up tests? try a robust model?) and reporting (what to claim about the Dx2 contrast?) that we don't explore here. Rather, we proceed to attempt to interpret the MLM with the aid of canonical analysis and a canonical HE plot. The canonical analysis of the model `SC.mlm1` now shows that both canonical dimensions are significant, and account for 83.9% and 16.1% of between group mean differences respectively.

```
SC.can1 <- candisc(SC.mlm1)
SC.can1
#>
## Canonical Discriminant Analysis for Dx:
#>
##   CanRsq Eigenvalue Difference Percent Cumulative
## 1 0.1645      0.1969       0.159     83.9      83.9
## 2 0.0364      0.0378       0.159     16.1     100.0
#>
## Test of H0: The canonical correlations in the
## current row and all that follow are zero
#>
```

```
#>   LR test stat approx F numDF denDF Pr(> F)
#> 1      0.805     3.78     8   264 0.00032 ***
#> 2      0.964     1.68     3   133 0.17537
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
op <- par(mar=c(5,4,1,1)+.1)
heplot(SC.can1,
       fill=TRUE, fill.alpha=.1,
       hypotheses=list("Dx1"="Dx1", "Dx2"="Dx2"),
       lwd = c(1, 2, 3, 3),
       col=c("red", "blue", "darkgreen", "darkgreen"),
       var.lwd=2,
       var.col="black",
       label.pos=c(3,1),
       var.cex=1.2,
       cex=1.25, cex.lab=1.2,
       scale=2.8,
       prefix="Canonical dimension ")
par(op)
```

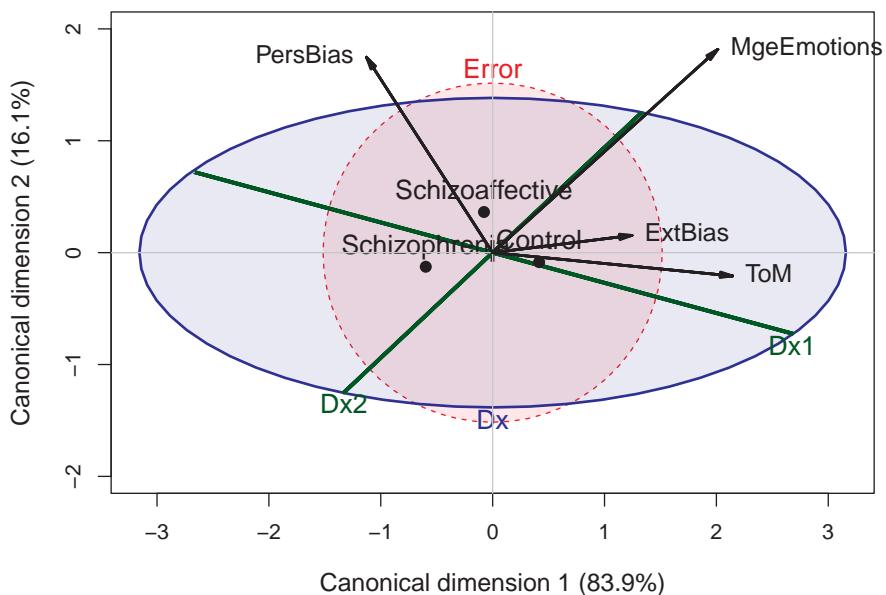


Figure 14.9: Canonical HE plot for the corrected SocialCog MANOVA. The variable vectors show the correlations of the responses with the canonical variables. The embedded green lines show the projections of the **H** ellipses for the contrasts **Dx1** and **Dx2** in canonical space.

The HE plot version of this canonical plot is shown in Figure 14.9. Because the `heplot()` method for a "`candisc`" object refits the original model to the **Z** canonical scores, it is easy to also project other linear hypotheses into this space. Note that in this view, both the **Dx1** and **Dx2** contrasts project outside **E** ellipse.².

This canonical HE plot has a very simple description:

²The direct application of significance tests to canonical scores probably requires some adjustment because these are computed to have the optimal between-group discrimination.

- Dimension 1 orders the groups from control to schizoaffective to schizophrenia, while dimension 2 separates the schizoaffective group from the others;
- Externalizing bias and theory of mind contributes most to the first dimension, while personal bias and managing emotions are more aligned with the second; and,
- The relations of the two contrasts to group differences and to the response variables can be easily read from this plot.

```
#cat("Packages used here:\n")
write_pkgs(file = .pkg_file)
#> 11 packages used here:
#> broom, candisc, car, carData, corrgram, dplyr, ggbiplot, ggplot2, heplots, knitr, tidyr
```


Part V

End matter

Colophon

This book was produced using [R version 4.5.1 \(2025-06-13 ucrt\)](#). Fundamental to this was the framework for reproducible documents provided by Yihui Xie's [knitr](#) package.

[Quarto](#) was used to compile and render the book in HTML and PDF formats. [** Don't really need all this**]

```
Quarto 1.7.29
[>] Checking environment information...
    Quarto cache location: C:\Users\friendly\AppData\Local\quarto
[>] Checking versions of quarto binary dependencies...
    Pandoc version 3.6.3: OK
    Dart Sass version 1.85.1: OK
    Deno version 1.46.3: OK
    Typst version 0.13.0: OK
[>] Checking versions of quarto dependencies.....OK
[>] Checking Quarto installation.....OK
    Version: 1.7.29
    CodePage: 1252
[>] Checking tools.....OK
    TinyTeX: (external install)
    Chromium: (not installed)
[>] Checking LaTeX.....OK
    Using: TinyTeX
    Version: 2025
[>] Checking Chrome Headless.....OK
    Using: Chrome found on system
    Source: Windows Registry
[>] Checking basic markdown render....OK
[>] Checking Python 3 installation....(None)
    Unable to locate an installed version of Python 3.
    Install Python 3 from https://www.python.org/downloads/
[>] Checking R installation.....OK
    Version: 4.5.1
    LibPaths:
        - C:/R/R-4.5.1/library
    knitr: 1.50
    rmarkdown: 2.29
[>] Checking Knitr engine render.....OK
```

Package versions

The principal R package versions used in examples and illustrations are listed below. These were captured via `sessioninfo:::package_info()` from all `library()` commands in the text, and scripts which also updated the references to packages.

At the time of writing, most of these were current on CRAN repositories but some development versions are indicated as “local” in the `source` column.

package	version	date	source
bayestestR	0.16.1	2025-07-01	CRAN
broom	1.0.9	2025-07-28	CRAN
candisc	0.9.2	2025-07-29	local
car	3.1-3	2024-09-27	CRAN
carData	3.0-5	2022-01-06	CRAN
corpcor	1.6.10	2021-09-16	CRAN
correlation	0.8.8	2025-07-08	CRAN
corrgram	1.14	2021-04-29	CRAN
corrplot	0.95	2024-10-14	CRAN
datawizard	1.2.0	2025-07-17	CRAN
dplyr	1.1.4	2023-11-17	CRAN
easystats	0.7.5	2025-07-11	CRAN
effects	4.2-4	2025-07-29	CRAN
effectsize	1.0.1	2025-05-27	CRAN
factoextra	1.0.7	2020-04-01	CRAN
FactoMineR	2.12	2025-07-23	CRAN
forcats	1.0.0	2023-01-29	CRAN
genridge	0.8.0	2024-12-02	CRAN
GGally	2.3.0	2025-07-18	CRAN
gganimate	1.0.10	2025-06-21	CRAN
ggbiplot	0.6.2	2024-01-08	CRAN
ggdensity	1.0.0	2023-02-09	CRAN
ggeffects	2.3.0	2025-06-13	CRAN
gggda	0.1.1	2025-08-05	Github (corybrunson/gggda@df6c7782e93f83c70c1cf43b11d06ffabc6c712a)
ggpcp	0.2.0	2022-11-28	CRAN
ggplot2	3.5.2	2025-04-09	CRAN
ggpubr	0.6.1	2025-06-27	CRAN
ggrepel	0.9.6	2024-09-07	CRAN
ggstats	0.10.0	2025-07-02	CRAN
heplots	1.7.8	2025-08-18	local
Hotelling	1.0-8	2021-09-09	CRAN
imager	1.0.5	2025-08-02	CRAN
insight	1.3.1	2025-06-30	CRAN
knitr	1.50	2025-03-16	CRAN
lubridate	1.9.4	2024-12-08	CRAN
magrittr	2.0.3	2022-03-30	CRAN
marginalEffects	0.28.0	2025-06-25	CRAN
MASS	7.3-65	2025-02-28	CRAN
matlib	1.0.1	2025-07-17	Github (friendly/matlib@d33440a7d98983c36f19f5cdc4d19d3f2ea8a50e)
modelbased	0.12.0	2025-07-10	CRAN
modelsummary	2.4.0	2025-06-08	CRAN
mvinfluence	0.9.3	2025-08-09	local
MVN	6.1	2025-06-10	CRAN
nestedLogit	0.3.2	2023-06-22	CRAN
parameters	0.27.0	2025-07-09	CRAN
patchwork	1.3.1	2025-06-21	CRAN
performance	0.15.0	2025-07-10	CRAN

package	version	date	source
purrr	1.1.0	2025-07-10	CRAN
qgraph	1.9.8	2023-11-03	CRAN
readr	2.1.5	2024-01-10	CRAN
report	0.6.1	2025-02-07	CRAN
Rtsne	0.17	2023-12-07	CRAN
see	0.11.0	2025-03-11	CRAN
stringr	1.5.1	2023-11-14	CRAN
tibble	3.3.0	2025-06-08	CRAN
tidyverse	1.3.1	2024-01-24	CRAN
tourr	2.0.0	2023-02-22	CRAN
vcd	1.4-13	2024-09-16	CRAN
VisCollin	0.1.2	2023-09-05	CRAN

References

- Abbott, E. A. (1884). *Flatland: A romance of many dimensions*. Buccaneer Books.
- Adler, D., & Murdoch, D. (2023). *Rgl: 3D visualization using OpenGL*. <https://CRAN.R-project.org/package=rgl>
- Aluja, T., Morineau, A., & Sanchez, G. (2018). *Principal component analysis for data science*. <https://pca4ds.github.io/>
- Anderson, E. (1935). The irises of the Gaspé peninsula. *Bulletin of the American Iris Society*, 35, 2–5.
- Anderson, M. J. (2006). Distance-based tests for homogeneity of multivariate dispersions. *Biometrics*, 62(1), 245–253. <https://doi.org/10.1111/j.1541-0420.2005.00440.x>
- Andrews, D. F. (1972). Plots of high dimensional data. *Biometrics*, 28, 123–136.
- Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician*, 27, 17–21.
- Arel-Bundock, V. (2025a). *Marginaleffects: Predictions, comparisons, slopes, marginal means, and hypothesis tests*. <https://marginaleffects.com/>
- Arel-Bundock, V. (2025b). *Modelsummary: Summary tables and plots for statistical models and data: Beautiful, customizable, and publication-ready*. <https://modelsummary.com>
- Asimov, D. (1985). Grand tour. *SIAM Journal of Scientific and Statistical Computing*, 6(1), 128–143.
- Barab'asi, A.-L. (2016). *Network science*. Cambridge University Press.
- Barrett, B. E. (2003). Understanding influence in multivariate regression. *Communications in Statistics - Theory and Methods*, 32(3), 667–680. <https://doi.org/10.1081/STA-120018557>
- Barrett, B. E., & Ling, R. F. (1992). General classes of influence measures for multivariate regression. *Journal of the American Statistical Association*, 87(417), 184–191. <https://www.jstor.org/stable/i314301>
- Bartlett, M. S. (1937). Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A*, 160(901), 268–282. <https://doi.org/10.2307/96803>
- Bartlett, M. S. (1938). Further aspects of the theory of multiple regression. *Mathematical Proceedings of the Cambridge Philosophical Society*, 34(1), 33–40. <https://doi.org/10.1017/s0305004100019897>
- Bashaw, W. L., & Findley, W. G. (Eds.). (1967). *Symposium on general linear model approach to the analysis of experimental data in educational research, final report*. <https://files.eric.ed.gov/fulltext/ED026737.pdf>
- Becker, R. A., Cleveland, W. S., & Shyu, M.-J. (1996). The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2), 123–155.
- Belsley, D. A. (1991). *Conditioning diagnostics: Collinearity and weak data in regression*. Wiley.
- Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley; Sons.
- Biecek, P., Baniecki, H., Krzyzinski, M., & Cook, D. (2023). *Performance is not enough: A story of the rashomon's quartet*. <https://arxiv.org/abs/2302.13356>
- Black, C., Southwell, C., Emmerson, L., Lunn, D., & Hart, T. (2018). Time-lapse imagery of adélie penguins reveals differential winter strategies and breeding site occupation. *PLOS ONE*, 13(3), e0193532. <https://doi.org/10.1371/journal.pone.0193532>
- Blishen, B., Carroll, W., & Moore, C. (1987). The 1981 socioeconomic index for occupations in canada. *Canadian Review of Sociology/Revue Canadienne de Sociologie*, 24(4), 465–488. <https://doi.org/10.1111/j.1755-618x.1987.tb00639.x>
- Bock, R. D. (1963). Programming univariate and multivariate analysis of variance. *Technometrics*, 5(1), 95–117. <https://doi.org/10.1080/00401706.1963.10490061>
- Bock, R. D. (1964). A computer program forunivariate and multivariate analysis of variance. *Proceedings of Scientific Symposium on Statistics*.
- Bodmer, W., Bailey, R. A., Charlesworth, B., Eyre-Walker, A., Farewell, V., Mead, A., & Senn, S. (2021).

- The outstanding scientist, r.a. Fisher: His views on eugenics and race. *Heredity*, 126(4), 565–576. <https://doi.org/10.1038/s41437-020-00394-6>
- Bondy, J. A., & Murty, U. S. R. (2008). *Graph theory*. Springer.
- Borg, I., & Groenen, P. J. F. (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.
- Borg, I., Groenen, P. J. F., & Mair, P. (2018). Applied multidimensional scaling and unfolding. In *SpringerBriefs in Statistics*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-73471-2>
- Box, G. E. P. (1949). A general distribution theory for a class of likelihood criteria. *Biometrika*, 36(3-4), 317–346. <https://doi.org/10.1093/biomet/36.3-4.317>
- Box, G. E. P. (1950). Problems in the analysis of growth and wear curves. *Biometrics*, 6, 362–389.
- Box, G. E. P. (1953). Non-normality and tests on variances. *Biometrika*, 40(3/4), 318–335. <https://doi.org/10.2307/2333350>
- Brown, M. B., & Forsythe, A. B. (1974). Robust tests for equality of variances. *Journal of the American Statistical Association*, 69(346), 364–367. <https://doi.org/10.1080/01621459.1974.10482955>
- Brown, P. J., & Zidek, J. V. (1980). Adaptive multivariate ridge regression. *The Annals of Statistics*, 8(1), 64–74. <http://www.jstor.org/stable/2240743>
- Brunson, J. C., & Gracey, J. (2025). *Gggda: A ggplot2 extension for geometric data analysis*. <https://github.com/corybrunson/gggda>
- Buja, A., Cook, D., Asimov, D., & Hurley, C. (2005). Computational methods for high-dimensional rotations in data visualization. In J. S. CR Rao EJ Wegman (Ed.), *Handbook of statistics* (pp. 391–413). Elsevier. [https://doi.org/10.1016/s0169-7161\(04\)24014-7](https://doi.org/10.1016/s0169-7161(04)24014-7)
- cagne, 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. <http://historical.library.cornell.edu/cgi-bin/cul.math/docviewer?did=00620001&seq=3>
- Cai, T. T., Liang, T., & Zhou, H. H. (2015). Law of log determinant of sample covariance matrix and optimal estimation of differential entropy for high-dimensional gaussian distributions. *Journal of Multivariate Analysis*, 137, 161–172. [https://doi.org/https://doi.org/10.1016/j.jmva.2015.02.003](https://doi.org/10.1016/j.jmva.2015.02.003)
- Caiani, F. (1926). Origins of fourth dimension concepts. *The American Mathematical Monthly*, 33(8), 397–406. <https://doi.org/10.1080/00029890.1926.11986607>
- Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2), 245–276. https://doi.org/10.1207/s15327906mbr0102_10
- Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. (1983). *Graphical methods for data analysis*. Wadsworth.
- Chambers, J. M., & Hastie, T. J. (1991). *Statistical models in s* (p. 624). Chapman & Hall/CRC.
- Charnes, A., Cooper, W. W., & Rhodes, E. (1981). Evaluating program and managerial efficiency: An application of data envelopment analysis to program follow through. *Management Science*, 27(6), 668–697. <http://www.jstor.org/stable/2631155>
- Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74, 829–836.
- Cleveland, W. S. (1985). *The elements of graphing data*. Wadsworth Advanced Books.
- Cleveland, W. S., & Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83, 596–610.
- Cleveland, W. S., & McGill, R. (1984). Graphical perception: Theory, experimentation and application to the development of graphical methods. *Journal of the American Statistical Association*, 79, 531–554.
- Cleveland, W. S., & McGill, R. (1985). Graphical perception and graphical methods for analyzing scientific data. *Science*, 229, 828–833.
- Clyde, D. J., Cramer, E. M., & Sherin, R. J. (1966). *Multivariate statistical programs*. Biometric Laboratory, University of Miami.
- Cochran, W. G. (1941). The distribution of the largest of a set of estimated variances as a fraction of their total. *Annals of Eugenics*, 11(1), 47–52. <https://doi.org/10.1111/j.1469-1809.1941.tb02271.x>
- Conover, W. J., Johnson, M. E., & Johnson, M. M. (1981). A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data. *Technometrics*, 23(4), 351–361. <https://doi.org/10.1080/00401706.1981.10487680>

- Cook, D., Buja, A., Cabrera, J., & Hurley, C. (1995). Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3), 155. <https://doi.org/10.2307/1390844>
- Cook, D., Buja, A., Lee, E.-K., & Wickham, H. (2008). Grand tours, projection pursuit guided tours, and manual controls. In *Handbook of data visualization* (pp. 295–314). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-33037-0_13
- Cook, D., & Laa, U. (2024). *Interactively exploring high-dimensional data and models in R*. Online. https://dicook.github.io/mulgar_book/
- Cook, D., & Swayne, D. F. (2007). *Interactive and dynamic graphics for data analysis : With R and GGobi*. Springer. <http://www.ggobi.org/book/>
- Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, 19(1), 15–18. <http://links.jstor.org/sici?&sici=0040-1706%28197702%2919%3A1%3C15%3ADOIOIL%3E2.0.CO%3B2-8>
- Cook, R. D. (1993). Exploring partial residual plots. *Technometrics*, 35(4), 351–362.
- Cook, R. D. (1996). Added-variable plots and curvature in linear regression. *Technometrics*, 38(3), 275–278. <https://doi.org/10.1080/00401706.1996.10484507>
- Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression*. Chapman; Hall.
- Cook, R. D., & Weisberg, S. (1994). ARES plots for generalized linear models. *Computational Statistics & Data Analysis*, 17(3), 303–315. [https://doi.org/10.1016/0167-9473\(92\)00075-3](https://doi.org/10.1016/0167-9473(92)00075-3)
- Costantini, G., Epskamp, S., Borsboom, D., Perugini, M., Möttus, R., Waldorp, L. J., & Cramer, A. O. J. (2015). State of the aRt personality research: A tutorial on network analysis of personality data in R. *Journal of Research in Personality*, 54, 13–29. <https://doi.org/10.1016/j.jrp.2014.07.003>
- Cotton, R. (2013). *Learning R*. O'Reilly Media.
- Cox, D. R. (1968). Notes on some aspects of regression analysis. *Journal of the Royal Statistical Society Series A*, 131, 265–279.
- Csárdi, G., Nepusz, T., Traag, V., Horvát, S., Zanini, F., Noom, D., & Müller, K. (2024). *igraph: Network analysis and visualization in r*. <https://doi.org/10.5281/zenodo.7682609>
- Curran, J., & Hersh, T. (2021). *Hotelling: Hotelling's t^2 test and variants*. <https://CRAN.R-project.org/package=Hotelling>
- Davies, R., Locke, S., & D'Agostino McGowan, L. (2022). *datasauRus: Datasets from the datasaurus dozen*. <https://CRAN.R-project.org/package=datasauRus>
- Davis, C. (1990). Body image and weight preoccupation: A comparison between exercising and non-exercising women. *Appetite*, 16(1), 84. [https://doi.org/10.1016/0195-6663\(91\)90115-9](https://doi.org/10.1016/0195-6663(91)90115-9)
- Dempster, A. P. (1969). *Elements of continuous multivariate analysis*. Addison-Wesley.
- Dempster, A. P. (1972). Covariance selection. *Biometrics*, 28(1), 157–175.
- Dixon, W. J. (1965). *BMD biomedical computer programs*. Health Sciences Computing Facility, School of Medicine, University of California; Health Sciences Computing Faculty.
- Dray, S., & Siberchicot, A. (2025). *Adegraphics: An S4 lattice-based package for the representation of multivariate data*. <http://pbil.univ-lyon1.fr/ADE-4/>
- Duncan, O. D. (1961). A socioeconomic index for all occupations. In Jr. A. J. Reiss, P. K. H. O. D. Duncan, & C. C. North (Eds.), *Occupations and social status*. The Free Press.
- Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32(2), 407–499.
- Emerson, J. W., Green, W. A., Schloerke, B., Crowley, J., Cook, D., Hofmann, H., & Wickham, H. (2013). The generalized pairs plot. *Journal of Computational and Graphical Statistics*, 22(1), 79–91. <http://www.tandfonline.com/doi/ref/10.1080/10618600.2012.694762>
- Euler, L. (1758). Elementa doctrinae solidorum. *Novi Commentarii Academiae Scientiarum Petropolitanae*, 4, 109–140. <https://scholarlycommons.pacific.edu/euler-works/230/>
- Farquhar, A. B., & Farquhar, H. (1891). *Economic and industrial delusions: A discourse of the case for protection*. Putnam.
- Fienberg, S. E. (1971). Randomization and social affairs: The 1970 draft lottery. *Science*, 171, 255–261.
- Finn, J. D. (1967). *MULTIVARIANCE: Fortran program for univariate and multivariate analysis of variance and covariance*. School of Education, State University of New York at Buffalo.
- Fisher, R. A. (1923). Studies in crop variation. II. The manurial response of different potato varieties. *The Journal of Agricultural Science*, 13(2), 311–320. <https://hdl.handle.net/2440/15179>

- Fisher, R. A. (1925b). *Statistical methods for research workers*. Oliver & Boyd.
- Fisher, R. A. (1925a). *Statistical methods for research workers* (6th ed.). Oliver & Boyd.
- Fisher, R. A. (1936). 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>
- Fishkeller, M. A., Friedman, J. H., & Tukey, J. W. (1974). PRIM-9, an interactive multidimensional data display and analysis system. *Proceedings of the Pacific ACM Regional Conference*.
- Flury, B., & Riedwyl, H. (1988). *Multivariate statistics: A practical approach*. Chapman & Hall.
- Fox, J. (1987). Effect displays for generalized linear models. In C. C. Clogg (Ed.), *Sociological methodology, 1987* (pp. 347–361). Jossey-Bass.
- Fox, J. (2003). Effect displays in R for generalized linear models. *Journal of Statistical Software*, 8(15), 1–27.
- Fox, J. (2016). *Applied regression analysis and generalized linear models* (Third edition.). SAGE.
- Fox, J. (2020). *Regression diagnostics* (2nd ed.). SAGE Publications, Inc. <https://doi.org/10.4135/9781071878651>
- Fox, J. (2021). *A mathematical primer for social statistics* (2nd ed.). SAGE Publications, Inc. <https://doi.org/10.4135/9781071878835>
- Fox, J., & Monette, G. (1992). Generalized collinearity diagnostics. *Journal of the American Statistical Association*, 87(417), 178–183.
- Fox, J., & Weisberg, S. (2018a). *An R companion to applied regression* (Third). SAGE Publications. <https://books.google.ca/books?id=uPNrDwAAQBAJ>
- Fox, J., & Weisberg, S. (2018b). Visualizing fit and lack of fit in complex regression models with predictor effect plots and partial residuals. *Journal of Statistical Software*, 87(9). <https://doi.org/10.18637/jss.v087.i09>
- Fox, J., Weisberg, S., & Price, B. (2023). *Car: Companion to applied regression*. <https://CRAN.R-project.org/package=car>
- Fox, J., Weisberg, S., Price, B., Friendly, M., & Hong, J. (2025). *Effects: Effect displays for linear, generalized linear, and other models*. <https://cran.r-project.org/package=effects>
- Friedman, J., Hastie, T., Tibshirani, R., Narasimhan, B., Tay, K., Simon, N., & Yang, J. (2025). *Glmnet: Lasso and elastic-net regularized generalized linear models*. <https://glmnet.stanford.edu>
- Friendly, M. (1991). *SAS System for statistical graphics* (1st ed.). SAS Institute. http://www.sas.com/service/doc/pubcat/uspubcat/ind_files/56143.html
- Friendly, M. (1994). Mosaic displays for multi-way contingency tables. *Journal of the American Statistical Association*, 89, 190–200. <http://www.jstor.org/stable/2291215>
- Friendly, M. (1999). Extending mosaic displays: Marginal, conditional, and partial views of categorical data. *Journal of Computational and Graphical Statistics*, 8(3), 373–395. <http://datavis.ca/papers/drew/drew.pdf>
- Friendly, M. (2002). Corrrgrams: Exploratory displays for correlation matrices. *The American Statistician*, 56(4), 316–324. <https://doi.org/10.1198/000313002533>
- Friendly, M. (2007). HE plots for multivariate general linear models. *Journal of Computational and Graphical Statistics*, 16(2), 421–444. <https://doi.org/10.1198/106186007X208407>
- Friendly, M. (2008). The Golden Age of statistical graphics. *Statistical Science*, 23(4), 502–535. <https://doi.org/10.1214/08-STS268>
- Friendly, M. (2011). *Generalized ridge trace plots: Visualizing bias and precision with the genridge R package*. SCS Seminar.
- Friendly, M. (2013). The generalized ridge trace plot: Visualizing bias and precision. *Journal of Computational and Graphical Statistics*, 22(1), 50–68. <https://doi.org/10.1080/10618600.2012.681237>
- Friendly, M. (2022). The life and works of André-Michel Guerry, revisited. *Sociological Spectrum*, 42(4-6), 233–259. <https://doi.org/10.1080/02732173.2022.2078450>
- Friendly, M. (2024). *Genridge: Generalized ridge trace plots for ridge regression*. <https://github.com/friendly/genridge>
- Friendly, M. (2025a). *Mvinfluence: Influence measures and diagnostic plots for multivariate linear models*. <https://github.com/friendly/mvinfluence>
- Friendly, M. (2025b). *VcdExtra: Vcd extensions and additions*. <https://friendly.github.io/vcdExtra/>
- Friendly, M., & Fox, J. (2025). *Candisc: Visualizing generalized canonical discriminant and canonical correlation analysis*. <https://github.com/friendly/candisc/>

- Friendly, M., Fox, J., & Chalmers, P. (2024). *Matlib: Matrix functions for teaching and learning linear algebra and multivariate statistics*. <https://github.com/friendly/matlib>
- Friendly, M., & Kwan, E. (2003). Effect ordering for data displays. *Computational Statistics and Data Analysis*, 43(4), 509–539. [https://doi.org/10.1016/S0167-9473\(02\)00290-6](https://doi.org/10.1016/S0167-9473(02)00290-6)
- Friendly, M., & Kwan, E. (2009). Where's Waldo: Visualizing collinearity diagnostics. *The American Statistician*, 63(1), 56–65. <https://doi.org/10.1198/tast.2009.0012>
- Friendly, M., & Meyer, D. (2016). *Discrete data analysis with R: Visualization and modeling techniques for categorical and count data*. Chapman & Hall/CRC.
- Friendly, M., Monette, G., & Fox, J. (2013). Elliptical insights: Understanding statistical methods through elliptical geometry. *Statistical Science*, 28(1), 1–39. <https://doi.org/10.1214/12-STS402>
- Friendly, M., & Sigal, M. (2018). Visualizing tests for equality of covariance matrices. *The American Statistician*, 72(4), 144–155. <https://doi.org/10.1080/00031305.2018.1497537>
- Friendly, M., & Wainer, H. (2021). *A history of data visualization and graphic communication*. Harvard University Press. <https://doi.org/10.4159/9780674259034>
- Fuller, W. (2006). *Measurement error models* (2nd ed.). John Wiley & Sons.
- Funkhouser, H. G. (1937). Historical development of the graphical representation of statistical data. *Osiris*, 3(1), 269–405. <http://tinyurl.com/32ema9>
- Gabriel, K. R. (1971). The biplot graphic display of matrices with application to principal components analysis. *Biometrics*, 58(3), 453–467. <https://doi.org/10.2307/2334381>
- Gabriel, K. R. (1981). Biplot display of multivariate matrices for inspection of data and diagnosis. In V. Barnett (Ed.), *Interpreting multivariate data* (pp. 147–173). John Wiley; Sons.
- Galton, F. (1863). *Meteorographica, or methods of mapping the weather*. Macmillan. <http://www.mugu.com/galton/books/meteorographica/index.htm>
- Galton, F. (1886). Regression towards mediocrity in hereditary stature. *Journal of the Anthropological Institute*, 15, 246–263. <http://www.jstor.org/cgi-bin/jstor/viewitem/09595295/dm995266/99p0374f/0>
- Galton, F. (1889). *Natural inheritance*. Macmillan. <http://galton.org/books/natural-inheritance/pdf/galton-nat-inh-1up-clean.pdf>
- Gannett, H. (1898). *Statistical atlas of the united states, eleventh (1890) census*. U.S. Government Printing Office.
- Gastwirth, J. L., Gel, Y. R., & Miao, W. (2009). The impact of Levene's test of equality of variances on statistical theory and practice. *Statistical Science*, 24(3), 343–360. <https://doi.org/10.1214/09-STS301>
- Gayan De Silva. (2020). *Exploring the world of artificial neural networks -a beginner's overview*. <https://doi.org/10.13140/RG.2.2.14790.14406>
- Gelman, A., Hullman, J., & Kennedy, L. (2023). *Causal quartets: Different ways to attain the same average treatment effect*. http://www.stat.columbia.edu/~gelman/research/unpublished/causal_quartets.pdf
- Gittins, R. (1985). *Canonical analysis: A review with applications in ecology*. Springer-Verlag.
- Goeman, J., Meijer, R., Chaturvedi, N., & Lueder, M. (2022). *Penalized: L1 (lasso and fused lasso) and L2 (ridge) penalized estimation in GLMs and in the cox model*. <https://doi.org/10.32614/CRAN.package.penalized>
- Gorman, K. B., Williams, T. D., & Fraser, W. R. (2014). Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PLoS ONE*, 9(3), e90081. <https://doi.org/10.1371/journal.pone.0090081>
- Gower, J. C., & Hand, D. J. (1996). *Biplots*. Chapman & Hall.
- Gower, J. C., Lubbe, S. G., & Roux, N. J. L. (2011). *Understanding biplots*. Wiley. <http://books.google.ca/books?id=66gQCi5JOKYC>
- Grandjean, M. (2016). A social network analysis of Twitter: Mapping the digital humanities community. *Cogent Arts & Humanities*, 3(1), 1171458. <https://doi.org/10.1080/23311983.2016.1171458>
- Graybill, F. A. (1961). *An introduction to linear statistical models*. McGraw-Hill.
- Greenacre, M. (1984). *Theory and applications of correspondence analysis*. Academic Press.
- Greenacre, M. (2010). *Biplots in practice*. Fundación BBVA. <https://books.google.ca/books?id=dv4LrFP7U/EC>
- Guerry, A.-M. (1833). *Essai sur la statistique morale de la France*. Crochard.

- Hahsler, M., Buchta, C., & Hornik, K. (2024). *Seriation: Infrastructure for ordering objects using seriation*. <https://github.com/mhahsler/seriation>
- Haitovsky, Y. (1987). On multivariate ridge regression. *Biometrika*, 74(3), 563–570. <https://doi.org/10.1093/biomet/74.3.563>
- Harrell, F. E. (2015). *Regression modeling strategies: With applications to linear models, logistic and ordinal regression, and survival analysis*. Springer International Publishing. <https://books.google.ca/books?id=sQ90rgEACAAJ>
- Harrison, P. (2023). Langevitour: Smooth interactive touring of high dimensions, demonstrated with scRNA-seq data. *The R Journal*, 15(2), 206–219. <https://doi.org/10.32614/RJ-2023-046>
- Harrison, P. (2025). *Langevitour: Langevin tour*. <https://logarithmic.net/langevitour/>
- Hart, C., & Wang, E. (2022). *Detourr: Portable and performant tour animations*. <https://CRAN.R-project.org/package=detourr>
- Hartigan, J. A. (1975a). *Clustering algorithms*. John Wiley; Sons.
- Hartigan, J. A. (1975b). Printer graphics for clustering. *Journal of Statistical Computing and Simulation*, 4, 187–213.
- Hartley, H. O. (1950). The use of range in analysis of variance. *Biometrika*, 37(3–4), 271–280. <https://doi.org/10.1093/biomet/37.3-4.271>
- Harwell, M. R., Rubinstein, E. N., Hayes, W. S., & Olds, C. C. (1992). Summarizing monte carlo results in methodological research: The one- and two-factor fixed effects ANOVA cases. *Journal of Educational and Behavioral Statistics*, 17(4), 315–339. <https://doi.org/10.3102/10769986017004315>
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference and prediction* (2nd ed.). Springer. <http://www-stat.stanford.edu/~tibs/ElemStatLearn/>
- Healy, M. J. R. (1968). Multivariate normal plotting. *Journal of the Royal Statistical Society Series C*, 17(2), 157–161.
- Herschel, J. F. W. (1833). On the investigation of the orbits of revolving double stars: Being a supplement to a paper entitled "micrometrical measures of 364 double stars". *Memoirs of the Royal Astronomical Society*, 5, 171–222.
- Hoaglin, D. C., & Welsch, R. E. (1978). The hat matrix in regression and ANOVA. *The American Statistician*, 32(1), 17–22. <https://doi.org/10.1080/00031305.1978.10479237>
- Hocking, R. R. (2013). *Methods and applications of linear models: Regression and the analysis of variance*. Wiley. <https://books.google.ca/books?id=iq2J-1iS6HcC>
- Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12, 55–67.
- Hoerl, A. E., Kennard, R. W., & Baldwin, K. F. (1975). Ridge regression: Some simulations. *Communications in Statistics*, 4(2), 105–123. <https://doi.org/10.1080/03610927508827232>
- Hofmann, H., VanderPlas, S., & Ge, Y. (2022). *Ggpcp: Parallel coordinate plots in the ggplot2 framework*. <https://github.com/heike/ggpcp>
- Hofstadter, D. R. (1979). *Gödel, escher, bach: An eternal golden braid*. Basic Books.
- Højsgaard, S., Edwards, D., & Lauritzen, S. (2012). *Graphical models with R*. Springer Science & Business Media.
- Horst, A., Hill, A., & Gorman, K. (2022). *Palmerpenguins: Palmer archipelago (antarctica) penguin data*. <https://allisonhorst.github.io/palmerpenguins/>
- Hotelling, H. (1931). The generalization of Student's ratio. *The Annals of Mathematical Statistics*, 2(3), 360–378. <https://doi.org/10.1214/aoms/1177732979>
- Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4), 321. <https://doi.org/10.2307/2333955>
- Huang, F. L. (2019). MANOVA: A procedure whose time has passed? *Gifted Child Quarterly*, 64(1), 56–60. <https://doi.org/10.1177/0016986219887200>
- Huberty, C. J., & Morris, J. D. (1989). Multivariate analysis versus multiple univariate analyses. *Psychological Bulletin*, 105(2), 302–308. <https://doi.org/10.1037/0033-2909.105.2.302>
- Husson, F., Josse, J., Le, S., & Mazet, J. (2025). *FactoMineR: Multivariate exploratory data analysis and data mining*. <http://factominer.free.fr>

- Husson, F., Le, S., & Pagès, J. (2017). *Exploratory multivariate analysis by example using r*. Chapman & Hall. <https://doi.org/10.1201/b21874>
- IBM. (1965). *Proceedings of the IBM scientific computing symposium on statistics: Oct 21-23, 1963* (L. Robinson, Ed.). IBM. <https://www.amazon.com/Proceedings-Scientific-Computing-Symposium-Statistics/dp/B000GL5RLU>
- Inselberg, A. (1985). The plane with parallel coordinates. *The Visual Computer*, 1, 69–91.
- Isvoranu, A.-M., Epskamp, S., Waldorp, L. J., & Borsboom, D. (2022). *Network psychometrics with r: A guide for behavioral and social scientists*. Routledge. <https://doi.org/10.4324/9781003111238>
- Johnson, R., & Wichern, D. (1998). *Applied multivariate statistical analysis* (4th ed.). Prentice Hall.
- Kassambara, A., & Mundt, F. (2020). *Factoextra: Extract and visualize the results of multivariate data analyses*. <http://www.sthda.com/english/rpkgs/factoextra>
- Kastellec, J. P., & Leoni, E. L. (2007). Using graphs instead of tables in political science. *Perspectives on Politics*, 5(04), 755–771. <https://doi.org/10.1017/S1537592707072209>
- Korkmaz, S., Goksuluk, D., & Zararsiz, G. (2025). *MVN: Multivariate normality tests*. <https://selcukkorkmaz.github.io/mvn-tutorial/>
- Krijthe, J. (2023). *Rtsne: T-distributed stochastic neighbor embedding using a barnes-hut implementation*. <https://github.com/jkrijthe/Rtsne>
- Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1), 1–27. <https://doi.org/10.1007/bf02289565>
- Kwan, E., Lu, I. R. R., & Friendly, M. (2009). Tableplot: A new tool for assessing precise predictions. *Zeitschrift für Psychologie / Journal of Psychology*, 217(1), 38–48. <https://doi.org/10.1027/0044-3409.217.1.38>
- Larmarange, J. (2025). *Ggstats: Extension to ggplot2 for plotting stats*. <https://larmarange.github.io/ggstats/>
- Larsen, W. A., & McCleary, S. J. (1972). The use of partial residual plots in regression analysis. *Technometrics*, 14, 781–790.
- Lauritzen, S. L. (1996). *Graphical models*. Oxford University Press.
- Lawless, J. F., & Wang, P. (1976). A simulation study of ridge and other regression estimators. *Communications in Statistics*, 5, 307–323.
- Lee, E.-K., & Cook, D. (2009). A projection pursuit index for large p small n data. *Statistics and Computing*, 20(3), 381–392. <https://doi.org/10.1007/s11222-009-9131-1>
- Lee, S. (2021). *Liminal: Multivariate data visualization with tours and embeddings*. <https://CRAN.R-project.org/package=liminal>
- Levene, H. (1960). Robust tests for equality of variances. In I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, & H. B. Mann (Eds.), *Contributions to probability and statistics: Essays in honor of Harold Hotelling* (pp. 278–292). Stanford University Press.
- Lix, J. M., L. M. Keselman, & Keselman, H. J. (1996). Consequences of assumption violations revisited: A quantitative review of alternatives to the one-way analysis of variance F test. *Review of Educational Research*, 66(4), 579–619. <https://doi.org/10.3102/00346543066004579>
- Longley, J. W. (1967). An appraisal of least squares programs for the electronic computer from the point of view of the user. *Journal of the American Statistical Association*, 62, 819–841. <https://doi.org/https://www.tandfonline.com/doi/abs/10.1080/01621459.1967.10500896>
- Lüdecke, D. (2025). *Ggeffects: Create tidy data frames of marginal effects for ggplot from model outputs*. <https://strengejache.github.io/ggeffects/>
- Lüdecke, D., Ben-Shachar, M. S., Patil, I., Waggoner, P., & Makowski, D. (2021). performance: An R package for assessment, comparison and testing of statistical models. *Journal of Open Source Software*, 6(60), 3139. <https://doi.org/10.21105/joss.03139>
- Lüdecke, D., Ben-Shachar, M. S., Patil, I., Wiernik, B. M., & Makowski, D. (2022). Easystats: Framework for easy statistical modeling, visualization, and reporting. In CRAN. <https://easystats.github.io/easystats/>
- Maaten, L. van der, & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605. <http://www.jmlr.org/papers/v9/vandermaaten08a.html>
- Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3), 519–530. <https://doi.org/http://dx.doi.org/10.2307/2334770>
- Mardia, K. V. (1974). Applications of some measures of multivariate skewness and kurtosis in testing

- normality and robustness studies. *Sankhya: The Indian Journal of Statistics, Series B*, 36(2), 115–128. <http://www.jstor.org/stable/25051892>
- Marquardt, D. W. (1970). Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12, 591–612.
- Martí, R., & Laguna, M. (2003). Heuristics and meta-heuristics for 2-layer straight line crossing minimization. *Discrete Applied Mathematics*, 127(3), 665–678.
- Matejka, J., & Fitzmaurice, G. (2017, May). Same stats, different graphs. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. <https://doi.org/10.1145/3025453.3025912>
- Matloff, N. (2011). *The art of R programming: A tour of statistical software design*. No Starch Press.
- McDonald, G. C. (2009). Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1), 93–100. <https://doi.org/10.1002/wics.14>
- McGowan, L. D., Gerke, T., & Barrett, M. (2023). Causal inference is not just a statistics problem. *Journal of Statistics and Data Science Education*, 1–9. <https://doi.org/10.1080/26939169.2023.2276446>
- Meyer, D., Zeileis, A., Hornik, K., & Friendly, M. (2024). *Vcd: Visualizing categorical data*. <https://doi.org/10.32614/CRAN.package.vcd>
- Meyers, L. S., Gamst, G., & Guarino, A. J. (2006). *Applied multivariate research: Design and interpretation*. SAGE Publications.
- Monette, G. (1990). Geometry of multiple regression and interactive 3-D graphics. In J. Fox & S. Long (Eds.), *Modern methods of data analysis* (pp. 209–256). SAGE Publications.
- O'Brien, P. C. (1992). Robust procedures for testing equality of covariance matrices. *Biometrics*, 48(3), 819–827. <http://www.jstor.org/stable/2532347>
- Oksanen, J., Simpson, G. L., Blanchet, F. G., Kindt, R., Legendre, P., Minchin, P. R., O'Hara, R. B., Solymos, P., Stevens, M. H. H., Szoezs, E., Wagner, H., Barbour, M., Bedward, M., Bolker, B., Borcard, D., Borman, T., Carvalho, G., Chirico, M., De Caceres, M., ... Weedon, J. (2025). *Vegan: Community ecology package*. <https://vegandevs.github.io/vegan/>
- Otto, J., & Kahle, D. (2023). *Ggdensity: Interpretable bivariate density visualization with ggplot2*. <https://jamesotto852.github.io/ggdensity/>
- Pearson, K. (1896). Contributions to the mathematical theory of evolution—III, regression, heredity and panmixia. *Philosophical Transactions of the Royal Society of London*, 187, 253–318.
- Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 6(2), 559–572.
- Pearson, K. (1903). I. Mathematical contributions to the theory of evolution. —XI. On the influence of natural selection on the variability and correlation of organs. *Philosophical Transactions of the Royal Society of London*, 200(321–330), 1–66. <https://doi.org/10.1098/rsta.1903.0001>
- Pedersen, T. L., & Robinson, D. (2025). *Ganimate: A grammar of animated graphics*. <https://ganimate.com>
- Pineo, P. O., & Porter, J. (1967). Occupational prestige in canada*. *Canadian Review of Sociology*, 4(1), 24–40. <https://doi.org/10.1111/j.1755-618X.1967.tb00472.x>
- Pineo, P. O., & Porter, J. (2008). Occupational prestige in canada. *Canadian Review of Sociology*, 4(1), 24–40. <https://doi.org/10.1111/j.1755-618x.1967.tb00472.x>
- Playfair, W. (1786). *Commercial and political atlas: Representing, by copper-plate charts, the progress of the commerce, revenues, expenditure, and debts of england, during the whole of the eighteenth century*. Debrett; Robinson;; Sewell. <http://ucpj.uchicago.edu/Isis/journal/demo/v000n000/000000/000000.fg4.html>
- Playfair, W. (1801). *Statistical breviary; shewing, on a principle entirely new, the resources of every state and kingdom in Europe*. Wallis.
- Reaven, G. M., & Miller, R. G. (1968). Study of the relationship between glucose and insulin responses to an oral glucose load in man. *Diabetes*, 17(9), 560–569. <https://doi.org/10.2337/diab.17.9.560>
- Reaven, G. M., & Miller, R. G. (1979). An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia*, 16, 17–24.
- Robinaugh, D. J., Hoekstra, R. H. A., Toner, E. R., & Borsboom, D. (2019). The network approach to psychopathology: A review of the literature 2008–2018 and an agenda for future research. *Psychological Medicine*, 50(3), 353–366. <https://doi.org/10.1017/s0033291719003404>
- Rogan, J. C., & Keselman, H. J. (1977). Is the ANOVA f-test robust to variance heterogeneity when sample

- sizes are equal?: An investigation via a coefficient of variation. *American Educational Research Journal*, 14(4), 493–498. <https://doi.org/10.3102/00028312014004493>
- Rousseeuw, Peter J., Ruits, I., & Tukey, J. W. (1999). The bagplot: A bivariate boxplot. *The American Statistician*, 53(4), 382–387.
- Rousseeuw, Peter J., Van Aelst, S., Van Driessen, K., & Gulló, J. A. (2004). Robust multivariate regression. *Technometrics*, 46(3), 293–305. <https://doi.org/10.1198/004017004000000329>
- Rousseeuw, Peter J., & Yohai, V. J. (1984). Robust regression by means of S-estimators. In J. Franke, W. Härdle, & D. Martin (Eds.), *Robust and nonlinear time series analysis* (Vol. 26, pp. 256–272). Springer.
- Sarkar, D. (2025). *Lattice: Trellis graphics for r*. <https://lattice.r-forge.r-project.org/>
- Scheffé, H. A. (1960). *The analysis of variance*. Wiley.
- Schloerke, B., Cook, D., Larmarange, J., Briatte, F., Marbach, M., Thoen, E., Elberg, A., & Crowley, J. (2025). *GGally: Extension to ggplot2*. <https://ggobi.github.io/ggally/>
- Scott, D. W. (1992). *Multivariate density estimation: Theory, practice, and visualization*. Wiley.
- Searle, S. R., Speed, F. M., & Milliken, G. A. (1980). Population marginal means in the linear model: An alternative to least squares means. *The American Statistician*, 34(4), 216–221.
- Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3–4), 591–611. <https://doi.org/10.1093/biomet/52.3-4.591>
- Shepard, R. N. (1962a). The analysis of proximities: Multidimensional scaling with an unknown distance function. i. *Psychometrika*, 27(2), 125–140. <https://doi.org/10.1007/bf02289630>
- Shepard, R. N. (1962b). The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3), 219–246. <https://doi.org/10.1007/bf02289621>
- Shepard, R. N., Romney, A. K., Nerlove, S. B., & Board, M. S. S. (1972a). *Multidimensional scaling: theory and applications in the behavioral sciences: Vols. II. Applications*. Seminar Press. <https://books.google.ca/books?id=PpFAAQAAIAAJ>
- Shepard, R. N., Romney, A. K., Nerlove, S. B., & Board, M. S. S. (1972b). *Multidimensional scaling: Theory and applications in the behavioral sciences: Vols. I. Theory*. Seminar Press. <https://books.google.ca/books?id=pJRAAQAAIAAJ>
- Shoben, E. J. (1983). Applications of multidimensional scaling in cognitive psychology. *Applied Psychological Measurement*, 7(4), 473–490. <https://doi.org/10.1177/014662168300700406>
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman & Hall.
- Simpson, E. H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society, Series B*, 30, 238–241.
- 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., Lang, D. T., Buja, A., & Cook, D. (2003). GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4), 423–444. [https://doi.org/10.1016/s0167-9473\(02\)00286-4](https://doi.org/10.1016/s0167-9473(02)00286-4)
- Teator, P. (2011). *R cookbook*. O'Reilly Media.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B: Methodological*, 58, 267–288.
- Tiku, M. L., & Balakrishnan, N. (1984). Testing equality of population variances the robust way. *Communications in Statistics - Theory and Methods*, 13(17), 2143–2159. <https://doi.org/10.1080/03610928408828818>
- Timm, N. H. (1975). *Multivariate analysis with applications in education and psychology*. Wadsworth (Brooks/Cole).
- Torgerson, W. S. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4), 401–419. <https://doi.org/10.1007/bf02288916>
- Tufte, E. R. (1983). *The visual display of quantitative information*. Graphics Press.
- Tukey, J. W. (1959). A quick, compact, two sample test to Duckworth's specifications. *Technometrics*, 1, 31–48. <https://doi.org/10.2307/1266308>
- Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 71–86. <https://doi.org/10.1162/jocn.1991.3.1.71>
- VanderPlas, S., Ge, Y., Unwin, A., & Hofmann, H. (2023). Penguins go parallel: A grammar of graphics

- framework for generalized parallel coordinate plots. *Journal of Computational and Graphical Statistics*, 1–16. <https://doi.org/10.1080/10618600.2023.2195462>
- Velleman, P. F., & Welsh, R. E. (1981). Efficient computing of regression diagnostics. *The American Statistician*, 35(4), 234–242.
- Vinod, H. D. (1978). A survey of ridge regression and related techniques for improvements over ordinary least squares. *The Review of Economics and Statistics*, 60(1), 121–131. <http://www.jstor.org/stable/1924340>
- Vu, V. Q., & Friendly, M. (2024). *Ggbiplot: A grammar of graphics implementation of biplots*. <https://github.com/friendly/ggbiplot>
- Waddell, A., & Oldford, R. W. (2023). *Loon: Interactive statistical data visualization*. <https://CRAN.R-project.org/package=loon>
- Warne, F. T. (2014). A primer on multivariate analysis of variance(MANOVA) for behavioral scientists. *Practical Assessment, Research & Evaluation*, 19(1). <https://scholarworks.umass.edu/pare/vol19/iss1/17/>
- Wegman, E. J. (1990). Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411), 664–675.
- Wei, T., & Simko, V. (2024). *Corrplot: Visualization of a correlation matrix*. <https://github.com/taiyun/corrplot>
- Welch, B. L. (1947). The generalization of "student's" problem when several different population variances are involved. *Biometrika*, 34(1–2), 28–35. <https://doi.org/10.1093/biomet/34.1-2.28>
- West, D. B. (2001). *Introduction to graph theory*. Prentice hall.
- Whittaker, J. (1990). *Graphical models in applied multivariate statistics*. John Wiley; Sons.
- Wickham, H. (2014). *Advanced R*. Chapman and Hall/CRC.
- 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. (2025). *Tourrr: Tour methods for multivariate data visualisation*. <https://github.com/ggobi/tourrr>
- Wickham, H., Cook, D., Hofmann, H., & Buja, A. (2011). Tourrr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2). <https://doi.org/10.18637/jss.v040.i02>
- Wilkinson, G. N., & Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Applied Statistics*, 22(3), 392. <https://doi.org/10.2307/2346786>
- Winer, B. J. (1962). *Statistical principles in experimental design*. McGraw-Hill.
- Wood, S. N. (2006). *Generalized additive models: An introduction with r*. Chapman; Hall/CRC Press.
- Wright, K. (2021). *Corrogram: Plot a correlogram*. <https://kwstat.github.io/corrogram/>
- Xie, Y. (2021). *Animation: A gallery of animations in statistics and utilities to create animations*. <https://yihui.org/animation/>
- Xu, Z., & Oldford, R. W. (2021). *Loon.tour: Tour in 'loon'*. <https://cran.r-project.org/package=loon.tour>
- Yee, T. W. (2015). *Vector generalized linear and additive models: With an implementation in r*. Springer.
- Yee, T. W. (2025). *VGAM: Vector generalized linear and additive models*. <https://CRAN.R-project.org/package=VGAM>
- Yohai, V. J. (1987). High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics*, 15(2), 642–656.
- Zhang, J., & Boos, D. D. (1992). Bootstrap critical values for testing homogeneity of covariance matrices. *Journal of the American Statistical Association*, 87(418), 425–429. <http://www.jstor.org/stable/2290273>

Index

- 1970 draft lottery, 13–16
- AddHealth data, 309
- adegraphics package, 123
- animation package, xii
- anscombe data, 8
- Anscombe’s quartet, 8
- bagplot, 51–52
- banknote data, 281
- bar chart, 7
- bfi data, 94
- biplots
- collinearity, 245–248
- bivariate density plot, 53
- Canadian occupational prestige, 182–184
- candisc package, 352
- canonical correlation, 5
- car package, 30, 41, 177, 182, 397
- codingMatrices package, 169
- collinearity biplots, 245–248
- condition indices, 244
- corrgram, 69–73
- corrgram package, 69
- corrplot package, 69, 70
- data ellipse, 5, 9, 26, 34–50
- datasaurus dozen, 10
- datasauRus package, 10
- datasets, 382, 383
 - AddHealth, 309
 - anscombe, 8
 - banknote, 281
 - bfi, 94
 - Davis, 11
 - Diabetes, 132
 - dogfood, 292, 340
 - Draft1970, 14
 - longley, 255
 - mathscore, 273
 - NLSY, 315, 405
 - Parenting, 303
 - peng, 47
 - penguins, 47
 - pollen, xii
- Prestige, xv, 41, 182
- Rohwer, 331
- Salaries, 26, 101
- schooldata, 321
- schoolsites, 321
- workers, 109
- datasets package, 47
- Davis data, 11
- depth, 51
- Diabetes data, 132
- diagnostic plots, 181–182
- dogfood data, 292, 340
- Draft1970 data, 14
- Duncan occupational prestige, 178–181
- effects package, 203
- factoextra package, 123
- FactoMineR package, 113, 123
- genridge package, 235, 255
- GGally package, 76, 322
- ggridge package, 141
- ggbiplot package, 123
- ggdensity package, 53
- ggeffects package, 203
- ggdada package, 51
- ggpcp package, 80
- ggplot2 package, xv, 27, 30
- ggstats package, 191
- glmnet package, 255
- gt package, 7
- HE plot, 5
- heplots package, 318, 339, 351, 378, 391
- high density region, 53
- igraph package, 95
- influence, 10, 12
- ipsatized scores, 237
- lack of fit, 10
- lattice package, xv, 30, 123, 203
- line graph, 7
- loess, 16
- longley data, 255
- loon package, 94

- M-estimators, 409
 Mahalanobis distance, 35
 marginaleffects package, 203
 MASS package, 255, 280
 mathscore data, 273
 matlib package, xv, 111
 metamer package, 10
 modelsummary package, 186
 mvinfluence package, 328, 398, 399
 MVN package, 291, 325
 NLSY data, 315, 405
 non-parametric
 bivariate density plot, 53
 ordering
 factor levels, 183
 variables, 84
 outliers, 10, 52, 248
 packages
 adegraphics, 123
 animation, xii
 candisc, 352
 car, 30, 41, 177, 182, 397
 codingMatrices, 169
 corrgram, 69
 corrplot, 69, 70
 datasauRus, 10
 datasets, 47
 effects, 203
 factoextra, 123
 FactoMineR, 113, 123
 genridge, 235, 255
 GGally, 76, 322
 ggridge, 141
 ggbiplot, 123
 ggdensity, 53
 ggeffects, 203
 gggda, 51
 ggpcp, 80
 ggplot2, xv, 27, 30
 ggstats, 191
 glmnet, 255
 gt, 7
 heplots, 318, 339, 351, 378, 391
 igraph, 95
 lattice, xv, 30, 123, 203
 loon, 94
 marginaleffects, 203
 MASS, 255, 280
 matlib, xv, 111
 metamer, 10
 modelsummary, 186
 mvinfluence, 328, 398, 399
 MVN, 291, 325
 palmerpenguins, 47
 penalized, 255
 rgl, 132
 rgl, xii
 Rtsne, 139
 seriation, 142
 stats, 397
 tinytable, 7
 tourr, 90, 93
 vcd, 74
 vcdExtra, 74
 vegan, 136
 VGAM, 289
 VisCollin, 235
 palmerpenguins package, 47
 parallel coordinate plots, 79–84
 Parenting data, 303
 penalized package, 255
 peng data, 47, 382–383
 penguins data, 47
 pie chart, 7
 pollen data, xii
 precision, 28
 Prestige data, xv, 41, 182
 principal components, 39
 projection, 5
 pure error, 9
 quartet
 Anscombe, 8, 25
 causal, 10
 Rashamon, 10
 regression, 181
 quartets
 dogfood, 340
 regression quartet, 181
 rgl package, xii
 rgl package, 132
 robust estimation, 409–411
 Rohwer data, 331
 Rtsne package, 139
 Salaries data, 26, 101
 scatterplot, 7
 scatterplot matrix, 25, 63–69
 generalized, 74–78
 schooldata data, 321
 schoolsites data, 321
 seriation package, 142
 Simpson’s paradox, 54–57, 197
 smoother, 26–30

loess, 16

non-parametric, 28

stats package, 397

stratifier, 26

Structural equation model, 4

tableplots, 245

tinytable package, 7

tourr package, 90, 93

Tukey, John, 25

uncertainty, 28

vcd package, 74

vcdExtra package, 74

vegan package, 136

VGAM package, 289

VisCollin package, 235

visual thinking, 7

visual thinning, 50, 67–69

workers data, 109

Visualization of Multivariate Data and Models in R

A Romance in Many Dimensions

Michael Friendly



CRC Press
Taylor & Francis Group

Here is where the dedication goes ...

Table of contents

Preface	ix
ONE, TWO, MANY	ix
Flatland	x
EUREKA!	xii
Multivariate scientific discoveries	xii
What I assume	xv
Conventions used in this book	xv
I Orienting Ideas	1
1 Introduction	3
1.1 Multivariate vs. multivariable methods	3
1.2 Why use a multivariate design	3
1.3 Linear models: Univariate to multivariate	4
1.4 Visualization is harder	4
1.5 Problems in understanding and communicating MLM results	5
2 Getting Started	7
2.1 Why plot your data?	7
2.1.1 Anscombe's Quartet	7
2.1.2 One lousy point can ruin your day	10
2.1.3 Shaken, not stirred: The 1970 Draft Lottery	13
2.2 Plots for data analysis	20
2.2.1 Model plots	20
2.2.2 Diagnostic plots	21
2.2.3 Principles of graphical display	21
2.3 What have we learned?	21
II Exploratory Methods	23
3 Plots of Multivariate Data	25
3.1 Bivariate summaries	26
3.1.1 Smoothers	27
3.1.2 Stratifiers	30
3.1.3 Conditioning	33
3.2 Data Ellipses	34
3.2.1 Ellipse properties	38
3.2.2 R functions for data ellipses	41
3.2.3 Example: Penguins data	47
3.2.4 Visual thinning	50
3.3 Bagplots	51
3.4 Non-parametric bivariate density plots	53
3.5 Simpson's paradox: marginal and conditional relationships	54
3.6 (a) Ignoring species	55
3.7 (b) By species	56

3.8 (c) Within species	56
3.9 Multivariate normality and outliers	57
3.9.1 Galton data	57
3.9.2 Penguin data	60
3.10 Scatterplot matrices	63
3.10.1 Visual thinning	67
3.11 Corrrgrams	69
3.12 Generalized pairs plots	74
3.13 Parallel coordinate plots	79
3.14 Animated tours	84
3.14.1 Projections	84
3.14.2 Touring methods	89
3.15 Network diagrams	94
3.15.1 Crime data	96
3.15.2 Partial correlations	97
3.15.3 Visualizing partial correlations	98
3.16 What have we learned?	99
3.17 Exercises	101
4 Dimension Reduction	103
4.1 <i>Flatland and Spaceland</i>	103
4.1.1 Multivariate juicers	103
4.2 Principal components analysis (PCA)	105
4.2.1 PCA by springs	106
4.2.2 Mathematics and geometry of PCA	108
4.2.3 Finding principal components	113
4.2.4 Visualizing variance proportions: screeplots	115
4.2.5 Visualizing PCA scores and variable vectors	116
4.3 Biplots	121
4.3.1 Constructing a biplot	122
4.3.2 Biplots in R	123
4.3.3 Example: Crime data	123
4.3.4 Biplot contributions and quality	126
4.3.5 Supplementary variables	128
4.3.6 Example: Diabetes data	132
4.4 Nonlinear dimension reduction	134
4.4.1 Multidimensional scaling	135
4.4.2 t-SNE	138
4.5 Application: Variable ordering for data displays	142
4.6 Application: Eigenfaces	144
4.7 Elliptical insights: Outlier detection	151
4.7.1 Example: Penguin data	151
4.8 What have we learned?	153
III Univariate Linear Models	159
5 Overview of Linear models	161
5.1 The General Linear Model	163
5.1.1 Model formulas	164
5.1.2 Model matrices	168
5.1.3 Coding factors and contrasts	169
5.2 What have we learned?	176
6 Plots for univariate response models	177

6.1	The “regression quartet”	178
6.1.1	Example: Duncan’s occupational prestige	178
6.1.2	Diagnostic plots	181
6.1.3	Example: Canadian occupational prestige	182
6.2	Other Model plots	184
6.3	Coefficient displays	185
6.3.1	Displaying coefficients	186
6.3.2	Visualizing coefficients	187
6.3.3	More useful coefficient plots	189
6.4	Added-variable and related plots	192
6.4.1	Properties of AV plots	195
6.4.2	Marginal - conditional plots	196
6.4.3	Prestige data	198
6.4.4	Component + Residual plots	198
6.5	Effect displays	202
6.5.1	Prestige data	204
6.6	Outliers, leverage and influence	208
6.6.1	The leverage-influence quartet	209
6.6.2	Influence plots	216
6.6.3	Duncan data	216
6.6.4	Influence in added-variable plots	217
6.7	What have we learned?	220
7	Topics in Linear Models	221
7.1	Ellipsoids in data space and β space	221
7.1.1	Coffee, stress and heart disease	223
7.2	Measurement error	227
7.2.1	OLS is BLUE	227
7.2.2	Errors in predictors	227
7.2.3	Coffee data: β space	231
7.3	What have we learned?	232
8	Collinearity & Ridge Regression	235
8.1	What is collinearity?	235
8.1.1	Visualizing collinearity	237
8.1.2	Data space and β space	238
8.2	Measuring collinearity	241
8.2.1	Variance inflation factors	241
8.2.2	Collinearity diagnostics	243
8.3	Tableplots	245
8.4	Collinearity biplots	245
8.5	Remedies for collinearity: What can I do?	248
8.6	Ridge regression	252
8.6.1	Properties of ridge regression	252
8.6.2	The <code>genridge</code> package	255
8.7	Univariate ridge trace plots	255
8.8	Bivariate ridge trace plots	258
8.8.1	Visualizing the bias-variance tradeoff	259
8.9	Low-rank views	262
8.9.1	Biplot view	266
8.10	What have we learned?	267
IV	Multivariate Linear Models	269

9 Hotelling's T^2	271
9.1 T^2 as a generalized t -test	271
9.2 T^2 properties	272
Example: Maths score data	273
9.3 HE plot and discriminant axis	277
9.3.1 <code>heplot()</code>	277
9.4 Discriminant analysis	280
9.5 More variables	281
9.5.1 Biplots	284
9.5.2 Testing mean differences	284
9.6 Variance accounted for: Eta square (η^2)	286
9.7 What we've learned	287
9.8 Exercises	287
10 Multivariate Linear Models	289
10.1 Structure of the MLM	290
10.1.1 Assumptions	291
10.2 Fitting the model	292
10.2.1 Example: Dog food data	292
10.2.2 Sums of squares	294
10.2.3 How big is SS_H compared to SS_E ?	296
10.3 Multivariate test statistics	298
10.3.1 Testing contrasts and linear hypotheses	298
10.4 ANOVA → MANOVA	301
10.4.1 Example: Father parenting data	302
10.4.2 Ordered factors	309
10.4.3 Example: Adolescent mental health	309
10.4.4 Factorial MANOVA	314
10.5 MRA → MMRA	314
10.5.1 Example: NLSY data	315
10.5.2 Example: School data	321
10.6 Model diagnostics for MLMs	325
10.6.1 Multivariate normality of residuals	325
10.6.2 Distance plot	327
10.6.3 Multivariate influence	328
10.7 ANCOVA → MANCOVA	330
10.7.1 Example: Paired-associate tasks and academic performance	331
10.8 What we've learned	337
11 Visualizing Multivariate Models	339
11.1 HE plot framework	340
11.2 HE plot construction	341
11.2.1 MANOVA model	345
11.3 HE plots	347
11.4 Significance scaling	348
11.5 Visualizing contrasts and linear hypotheses	349
11.6 HE plot matrices	351
11.7 Low-D views: Canonical analysis	351
11.7.1 Coeficients	353
11.7.2 Canonical scores plot	354
11.7.3 Canonical HE plot	354
11.8 Factorial MANOVA	356
11.9 Quantitative predictors: MMRA	361
11.10 Canonical correlation analysis	364

11.11 MANCOVA models	370
11.12 What have we learned?	374
12 Visualizing Equality of Covariance Matrices	377
12.1 Homogeneity of Variance in Univariate ANOVA	378
12.2 Visualizing Levene's test	380
12.3 Homogeneity of variance in MANOVA	381
12.4 Box's \mathcal{M} test	385
12.5 Visualizing heterogeneity	386
12.6 Visualizing Box's \mathcal{M}	387
12.7 Low-rank views	387
12.7.1 Small dimensions can matter	389
12.8 Other measures of heterogeneity	389
12.9 Multivariate analog of Levine's test	392
12.9.1 Canonical discriminant analysis	392
12.10 What Have We Learned?	394
13 Multivariate Influence and Robust Estimation	397
13.1 Multivariate influence	398
13.1.1 Notation	398
13.1.2 Hat values and residuals	398
13.1.3 Cook's distance	398
13.1.4 Leverage and residual components	399
13.2 The Mysterious Case 9	399
13.2.1 Cook's D	401
13.2.2 DFBETAS	403
13.3 Example: NLSY data	405
13.4 Example: Penguin data	406
13.5 Robust Estimation	409
13.6 Example: Penguin data	409
14 Case studies	413
14.1 Fitting the MLM	415
14.1.1 HE plot	416
14.1.2 Canonical space	417
14.2 Social cognitive measures	420
14.2.1 Model checking	421
14.2.2 Canonical HE plot	423
V End matter	427
Colophon	429
Package versions	429
References	433

Preface

This book is about graphical methods developed recently for multivariate data, and their uses in understanding relationships when there are several aspects to be considered together. Data visualization methods for statistical analysis are well-developed for simple linear models with a single outcome variable. However, with applied research in the sciences, social and behavioral in particular, it is often the case that the phenomena of interest (e.g., depression, job satisfaction, academic achievement, childhood ADHD disorders, etc.) can be measured in several different ways or related aspects. Understanding how these different aspects are *related* contributes to our knowledge of the general phenomenon.

For example, if academic achievement can be measured for adolescents by reading, mathematics, science and history scores, how do predictors such as parental encouragement, school environment and socioeconomic status affect all these outcomes? In a similar way? In different ways? In such cases, much more can be understood from a multivariate approach that considers the correlations among the outcomes. Yet, sadly, researchers typically examine the outcomes one by one which often only tells part of the data story.

However, to do this it is useful to set the stage for multivariate thinking, with a grand scheme for statistics and data visualization, a parable, and an example of multivariate discovery.

ONE, TWO, MANY

There is an old and helpful idea I learned from John Hartigan in my graduate days at Princeton:

In statistics and data visualization *all* methods can be classified by the number of dimensions contemplated, on a scale of **ONE, TWO, MANY**.

By this, he meant that, at a global level, all data, statistical summaries, and graphical displays could be classified as:

- **univariate**: a single variable, considered in isolation (age, COVID cases, pizzas ordered). Univariate numerical summaries are means, medians, measures of variability, and so forth. Univariate displays include dot plots, boxplots, histograms and density estimates.
- **bivariate**: two variables, considered jointly. Numerical summaries include correlations, covariances and two-way tables of frequencies or measures of association for categorical variables. Bivariate displays include scatterplots and mosaic plots.
- **multivariate**: three or more variables, considered jointly. Numerical summaries include correlation and covariance matrices, consisting of all pairwise values, but also derived measures from the analysis of these matrices (eigenvalues, eigenvectors). Graphical displays of multivariate data can sometimes be shown in 3D, but often involve multiple views of the data projected into 2D plots.

As a quasi-numerical scale, I refer to these as **1D**, **2D** and **nD**. This admits the possibility of half-integer cases, such as **1.5D**, where the main focus is on a single variable, but that is classified by a simple factor (e.g., gender), or **2.5D** where a 2D scatterplot can show other variables using color, shape or other visual attributes. His point in this classification was that once you've reached three variables, all higher dimensions involve similar summaries and data displays.

Univariate and bivariate methods and displays are well-known. This book is about how these ideas can be

extended to an n -dimensional world. Three-dimensional data displays are now fairly easy to produce, even if they are sometimes difficult to understand. But how can we even think about four or more dimensions? The difficulty can be appreciated by considering the tale of *Flatland*.

Flatland

To comport oneself with perfect propriety in Polygonal society, one ought to be a Polygon oneself. —
Edwin A. Abbott, *Flatland*

In 1884, an English schoolmaster, Edwin Abbott Abbott, shook the world of Victorian culture with a slim volume, *Flatland: A Romance of Many Dimensions* (Abbott, 1884). He described a two-dimensional world, *Flatland*, inhabited entirely by geometric figures in the plane. His purpose was satirical, to poke fun at the social and gender class system at the time: Women were mere line segments, while men were represented as polygons with varying numbers of sides—a triangle was a working man, but acute isosceles were soldiers or criminals of very small angle; gentlemen and professionals had more sides. Abbott published this under the pseudonym, “A Square”, suggesting his place in the hierarchy.

True, said the Sphere; it appears to you a Plane, because you are not accustomed to light and shade and perspective; just as in Flatland a Hexagon would appear a Straight Line to one who has not the Art of Sight Recognition. But in reality it is a Solid, as you shall learn by the sense of Feeling. —
Edwin A. Abbott, *Flatland*

But how did it *feel* to be a member of a flatland society? How could a point (a newborn child?) understand a line (a woman)? How does a Triangle “see” a Hexagon or even a infinitely-sided Circle? Abbott introduces the very idea of different dimensions of existence through dreams and visions:

- A Square dreams of visiting a one-dimensional *Lineland* where men appear as lines, and women are merely “illustrious points”, but the inhabitants can only see the Square as lines.
- In a vision, the Square is visited by a Sphere, to illustrate what a 2D Flatlander could understand from a 3D sphere (Figure 1) that passes through the plane he inhabits. It is a large circle when seen at the moment of its’ greatest extent. As the Sphere rises, it becomes progressively smaller, until it becomes a point, and then vanishes.

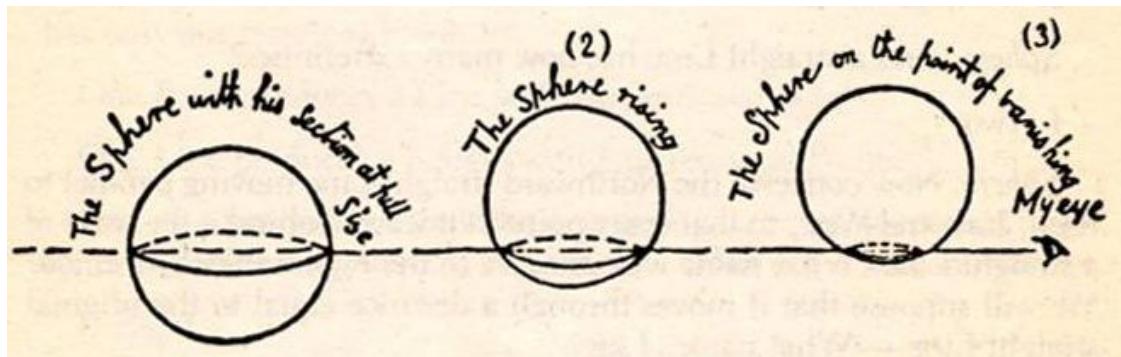


Figure 1: A 2D Flatlander seeing a sphere as it passes through Flatland. The line, labeled ‘My Eye’ indicates what the Flatlander would see. Source: Abbott (1884)

Abbott goes on to state what could be considered as a demonstration (or proof) by induction of the difficulties of seeing in 1, 2, 3 dimensions, and how the idea motion over time (one more dimension) could allow citizens of any 1D, 2D, 3D world to contemplate one more dimension.

In One Dimensions, did not a moving Point produce a Line with two terminal points? In two

Dimensions, did not a moving Line produce a Square with four terminal points? In Three Dimensions, did not a moving Square produce - did not the eyes of mine behold it - that blessed being, a Cube, with eight terminal points? And in Four Dimensions, shall not a moving Cube - alas, for Analogy, and alas for the Progress of Truth if it be not so - shall not, I say the motion of a divine Cube result in a still more divine organization with sixteen terminal points? — Edwin A. Abbott

For Abbot, the way for a citizen of any world to imagine one more dimension was to consider how a higher-dimensional object would change over time.¹ A line moved over time could produce a rectangle as shown in Figure 2; that rectangle moving in another direction over time would produce a 3D figure, and so forth.

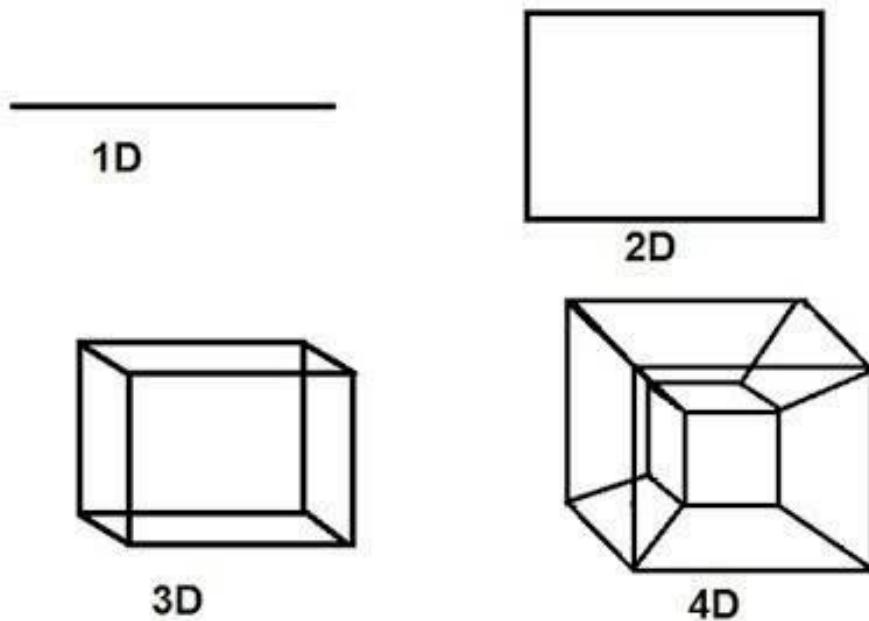


Figure 2: Geometrical objects in 1 to 4 dimensions. One more dimension can be thought of as the trace of movement over time.

But wait! Where does that 4D thing (a *tesseract*) come from? To really see a tesseract it helps to view it in an animation over time ([?@fig-tesseract](#)). But like the Square, contemplating 3D from a 2D world, it takes some imagination.

Yet the deep mathematics of more than three dimensions only emerged in the 19th century. In Newtonian mechanics, space and time were always considered independent of each other. Our familiar three-dimensional space, of length, width, and height had formed the backbone of Euclidean geometry for millenia. However, the idea that space and time are indeed interwoven was first proposed by German mathematician Hermann Minkowski (1864–1909) in 1908. This was a powerful idea. It bore fruit when Albert Einstein revolutionized the Newtonian conceptions of gravity in 1915 when he presented a theory of general relativity which was based primarily on the fact that mass and energy warp the fabric of four-dimensional spacetime.

The parable of *Flatland* can provide inspiration for statistical thinking and data visualization. Once we go beyond bivariate statistics and 2D plots, we are in a multivariate world of possibly MANY dimensions. It takes only some imagination and suitable methods to get there.

¹In his famous TV series, *Cosmos*, Carl Sagan provides [an intriguing video presentation](#) Flatland and the 4th dimension. However, as far back as 1754 ([Cajori, 1926](#)), the idea of adding a fourth dimension appears in Jean le Rond d'Alembert's "Dimensions", and one realization of a four-dimensional object is a *tesseract*, shown in Figure 2.

Like Abbott's *Flatland*, this book is a romance, in many dimensions, of what we can learn from modern methods of data visualization.

EUREKA!

Even modest sized multivariate data can have secrets that can be revealed in the right view. As an example, David Coleman at RCA Laboratories in Princeton, N.J. generated a dataset of five (fictitious) measurements of grains of pollen for the 1986 Data Exposition at the Joint statistical Meetings. The first three variables are the lengths of geometric features 3848 observed sampled pollen grains – in the x, y, and z dimensions: a `ridge` along x, a `nub` in the y direction, and a `crack` in along the z dimension. The fourth variable is pollen grain `weight`, and the fifth is `density`. The challenge was to “find something interesting” in this dataset, now available as `pollen`.

Those who solved the puzzle were able to find an orientation of this 5-dimensional dataset, such that zooming in revealed a magic word, “EUREKA” spelled in points, as in the following figure.

The path to finding the hidden word can be seen better in a 3D animation. The online version of the book uses The `rgl` package ([Adler & Murdoch, 2023](#)) create a 3D scatterplot of the first three variables. Then the `animation` package ([Xie, 2021](#)) is used to record a sequence of images, adjusting the `rgl::par3d(zoom)` value.

Multivariate scientific discoveries

Lest this example seem contrived (which it admittedly is), multivariate visualization has played an important role in quite a few scientific discoveries. Among these, Francis Galton's ([1863](#)) discovery of the anti-cyclonic pattern of wind direction in relation to barometric pressure from many weather measures recorded systematically across all weather stations, lighthouses and observatories in Europe in December 1861 stands out as the best example of a scientific discovery achieved almost entirely through graphical means— something that was totally unexpected, and purely the product of his use of remarkably novel high-dimensional graphs ([Friendly & Wainer, 2021, pp. 170–173](#)).

A more recent example is the discovery of two general classes in the development of Type 2 diabetes by Reaven & Miller ([1979](#)), using PRIM-9 ([Fishkeller et al., 1974](#)), the first computer system for high-dimensional visualization². In an earlier study Reaven & Miller ([1968](#)) examined the relation between blood glucose levels and the production of insulin in normal subjects and in patients with varying degrees of hyperglycemia (elevated blood sugar level). They found a peculiar ‘horse shoe’ shape in this relation (shown in Figure 4), about which they could only speculate: perhaps individuals with the best glucose tolerance also had the lowest levels of insulin as a response to an oral dose of glucose; perhaps those with low glucose response could secrete higher levels of insulin; perhaps those who were low on both glucose and insulin responses followed some other mechanism. In 2D plots, this was a mystery.

```
data(Diabetes, package="heplots")
plot(instest ~ glutest, data=Diabetes,
      pch=16,
      cex.lab=1.25,
      xlab="Glucose response",
      ylab="Insulin response")
```

An answer to their questions came ten years later, when they were able to visualize similar but new data

²PRIM-9 is an acronym for Picturing, Rotation, Isolation and Masking in up to 9 dimensions. These operations are fundamental to interactive and dynamic data visualization.

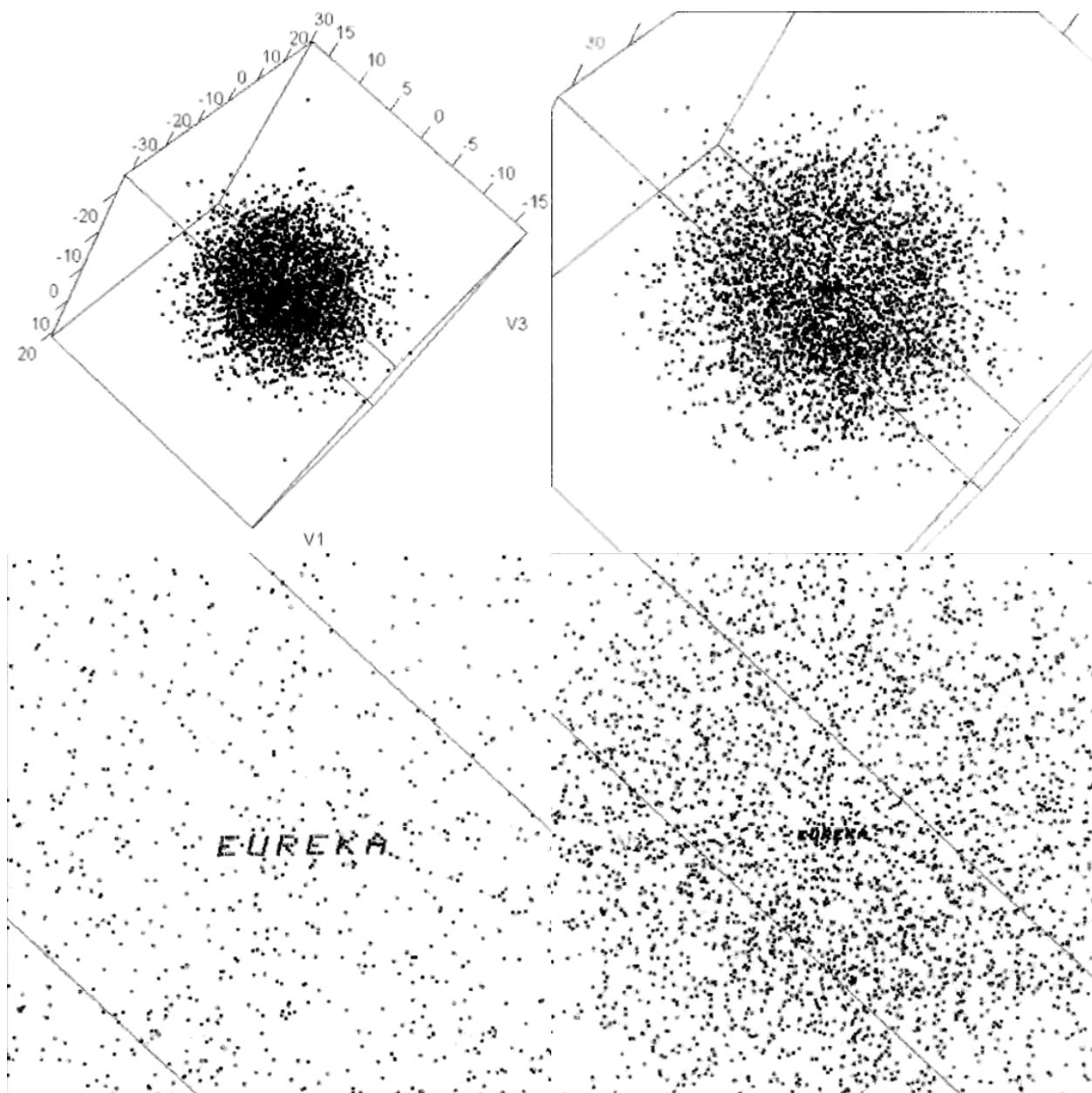


Figure 3: Four views of the pollen data, zooming in, clockwise from the upper left to discover the word “EUREKA”.

in 3D using the PRIM-9 system. In a carefully controlled study, they also measured “steady state plasma glucose” (SSPG), a measure of the efficiency of use of insulin in the body, where large values mean insulin resistance, as well as other variables. PRIM-9 allowed them to explore various sets of three variables, and, more importantly, to rotate a given plot in three dimensions to search for interesting features. One plot that stood out concerned the relation between plasma glucose response, plasma insulin response and SSPG response, shown in Figure 5.

From this graphical insight, they were able to classify the participants into three groups, based on clinical levels of glucose and insulin. The people in the wing on the left in Figure 5 were considered to have overt diabetes, the most advanced form, characterized by elevated fasting blood glucose concentration and classical

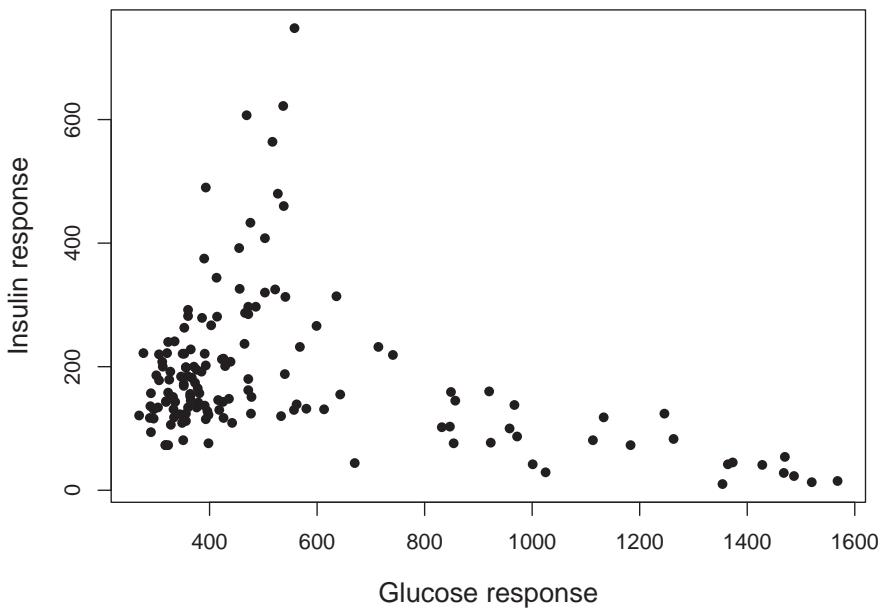


Figure 4: Reproduction of a graph similar to that from Reaven & Miller (1968) on the relationship between glucose and insulin response to being given an oral dose of glucose.

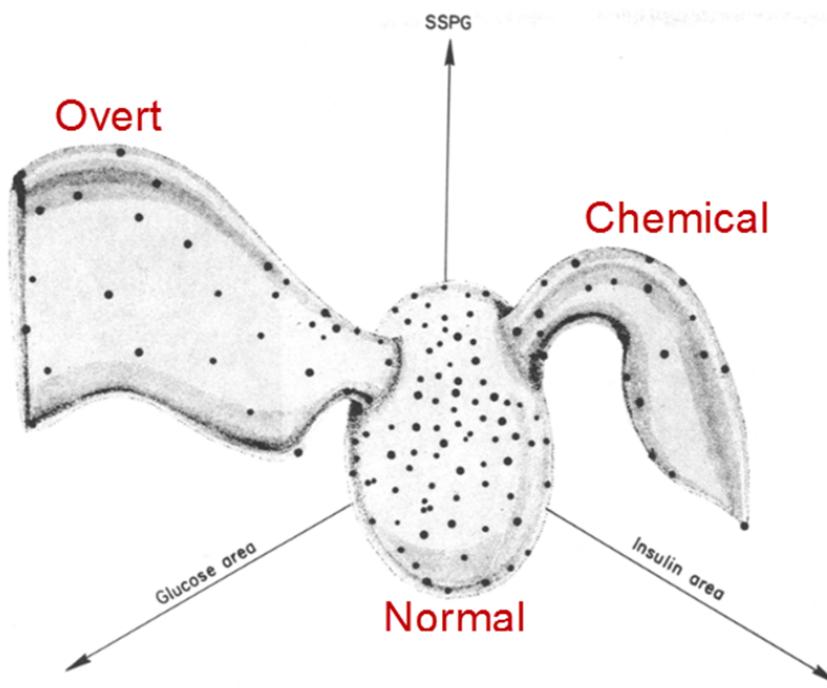


Figure 5: Artist's rendition of data from Reaven & Miller (1979) as seen in three dimensions using the PRIM-9 system. Labels for the clusters have been added, identifying the three groups of patients. *Source:* Reaven & Miller (1979).

diabetic symptoms. Those in the right wing were classified as latent or chemical diabetics, with no symptoms of diabetes but demonstrable abnormality of oral or intravenous glucose tolerance. Those in the central blob were classified as normal.

Previous thinking was that Type 2 diabetes (when the body cannot make *enough* insulin, as opposed to Type I, an autoimmune condition where the pancreatic cells have been destroyed) progressed from the chemical stage to an overt one in a smooth transition. However, it was clear from Figure 5 that the only “path” from one to the other lead through the cluster of normal patients near the origin, so that explanation must be wrong. Instead, this suggested that the chemical and overt diabetics were distinct classes. Indeed, longitudinal studies showed that patients classified as chemical diabetics rarely developed the overt form. The understanding of the etiology of Type 2 diabetes was altered dramatically by the power of high-D interactive graphics.

What I assume

It is assumed that the reader has a background in applied *intermediate* statistics including material on univariate linear models including analysis of variance (ANOVA) and multiple regression. This means you should be familiar with ... **TODO:** Complete this required background

There will also be some mathematics in the book where words and diagrams are not enough. The mathematical level will be intermediate, mostly consisting of simple algebra. No derivations, proofs, theorems here! For multivariate methods, it will be useful to express ideas using matrix notation to simplify presentation. The single symbol I'm using math to express ideas, and all you will need is a reading-level of understanding. For this, the first chapter of Fox (2021), *A mathematical primer for social statistics*, is excellent. If you want to learn something of using matrix algebra for data analysis and statistics, I recommend our package **matlib** (Friendly et al., 2024) .

I also assume the reader to have at least a basic familiarity with R. While R fundamentals are outside the scope of the book, I believe that this language provides a rich set of resources, far beyond that offered by other statistical software packages, and is well worth learning.

For those not familiar with R, I recommend Matloff (2011), Wickham (2014), and Cotton (2013) for introductions to programming in the language. Fox & Weisberg (2018a) and Teator (2011) are great for learning about how to conduct basic statistical analyses in R. **TODO:** Revise this list.

TODO: Add stuff on general books about graphics

Conventions used in this book

The following typographic conventions are used in this book:

- *italic* : indicates terms or phrases to be *emphasized* or defined in the text; **bold** : is used for terms to be given **strong emphasis**, particularly for their first mention.
- Package names are printed in **bold** and colored **brown**, for example **ggplot2** , **lattice** and the **matlib** . These uses generate citations like **ggplot2** (Wickham et al., 2023) . They are automatically indexed, individually and under a “Packages” heading.
- Datasets are rendered as their name in monospaced font, like **Prestige** or indicating the package from which they come, as in **carData::Prestige** . These too are automatically indexed.
- A monospaced **typewriter** font is used in the text to refer to variable and function names, like **education** and **plot()**; also for R statement elements and keywords as well.
- R code in program listings and output is presented in a monospaced (**typewriter**) font, **fira mono**

- For R functions in packages, I use the notation `package::function()`, for example: `car::Anova()`, to identify the package where those functions are defined.

Part I

Orienting Ideas

1

Introduction

This material may or may not survive; it was taken from an earlier article.

1.1 Multivariate vs. multivariable methods

multivariate \neq multivariable

In this era of multivitamins, multitools, multifactor authentication and even the multiverse, it is well to understand the distinction between *multivariate* and *multivariable* methods as these terms are generally used and as I use them here in relation to statistical methods and data visualization. The distinction is simple:

- **Multivariate methods** for linear models such as multivariate regression have more than one dependent, response or outcome variable. Other multivariate methods such as principal components analysis or factor analysis treat all variables on an equal footing.
 - **Multivariable methods** have a single dependent variable and more than one independent variables or covariates.
-

1.2 Why use a multivariate design

A particular research outcome (e.g., depression, neuro-cognitive functioning, academic achievement, self-concept, attention deficit hyperactivity disorders) might take on a multivariate form if it has several observed measurement scales or related aspects by which it is quantified, or if there are multiple theoretically distinct outcomes that should be assessed in conjunction with each other (e.g., using depression, generalized anxiety, and stress inventories to model overall happiness). In this situation, the primary concern of the researcher is to ascertain the impact of potential predictors on two or more response variables simultaneously.

For example, if academic achievement is measured for adolescents by their reading, mathematics, science, and history scores, the following questions are of interest:

- Do predictors such as parent encouragement, socioeconomic status and school environmental variables affect *all* of these outcomes?
- Do they affect them in the *same* or *different* ways?
- How many different aspects of academic achievement can be distinguished in the predictors? Equivalently, is academic achievement *unidimensional* or *multidimensional* in relation to the predictors?

Similarly, if psychiatric patients in various diagnostic categories are measured on a battery of tests related to social skills and cognitive functioning, we might want to know:

- Which measures best discriminate among the diagnostic groups?
- Which measures are most predictive of positive outcomes?

- Further, how are the *relationships* between the outcomes affected by the predictors?

Such questions obviously concern more than just the separate univariate relations of each response to the predictors. Equally, or perhaps more importantly, are questions of how the response variables are predicted *jointly*.

i SEM

Structural equation modeling (SEM) offers another route to explore and analyze the relationships among multiple predictors and multiple responses. They have the advantage of being able to test potentially complex systems of linear equations in very flexible ways; however, these methods are often far removed from data analysis *per se* and except for path diagrams offer little in the way of visualization methods to aid in understanding and communicating the results. The graphical methods we describe here can also be useful in a SEM context.

1.3 Linear models: Univariate to multivariate

For classical linear models for ANOVA and regression, the step from a univariate model for a single response, y , to a multivariate one for a collection of p responses, \mathbf{y} is conceptually very easy. That's because the univariate model,

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q + \epsilon_i,$$

or, in matrix terms,

$$\mathbf{y} = \mathbf{X} \boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \text{with } \mathbf{u} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}),$$

generalizes directly to an analogous multivariate linear model (MLM),

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_p] = \mathbf{X} \mathbf{B} + \boldsymbol{\varepsilon} \quad \text{with } \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

for multiple responses (as will be discussed in detail). The design matrix, \mathbf{X} remains the same, and the vector $\boldsymbol{\beta}$ of coefficients becomes a matrix \mathbf{B} , with one column for each of the p outcome variables.

Happily as well, hypothesis tests for the MLM are also straight-forward generalizations of the familiar F and t -tests for univariate response models. Moreover, there is a rich geometry underlying these generalizations which we can exploit for understanding and visualization.

1.4 Visualization is harder

However, with two or more response variables, visualizations for multivariate models are not as simple as they are for their univariate counterparts for understanding the effects of predictors, model parameters, or model diagnostics. Consequently, the results of such studies are often explored and discussed solely in terms of coefficients and significance, and visualizations of the relationships are only provided for one response variable at a time, if at all. This tradition can mask important nuances, and lead researchers to draw erroneous conclusions.

The aim of this book is to describe and illustrate some central methods that we have developed over the

last ten years that aid in the understanding and communication of the results of multivariate linear models (Friendly, 2007; Friendly & Meyer, 2016). These methods rely on *data ellipsoids* as simple, minimally sufficient visualizations of variance that can be shown in 2D and 3D plots. As will be demonstrated, the *Hypothesis-Error (HE) plot* framework applies this idea to the results of multivariate tests of linear hypotheses.

Further, in the case where there are more than just a few outcome variables, the important nectar of their relationships to predictors can often be distilled in a multivariate juicer—a **projection** of the multivariate relationships to the predictors in the low-D space that captures most of the flavor. This idea can be applied using *canonical correlation plots* and with *canonical discriminant HE plots*.

1.5 Problems in understanding and communicating MLM results

In my consulting practice within the Statistical Consulting Service at York University, I see hundreds of clients each year ranging from advanced undergraduate thesis students, to graduate students and faculty from a variety of fields. Over the last two decades, and across each of these groups, I have noticed an increasing desire to utilize multivariate methods. As researchers are exposed to the utility and power of multivariate tests, they see them as an appealing alternative to running many univariate ANOVAs or multiple regressions for each response variable separately.

However, multivariate analyses are more complicated than such approaches, especially when it comes to understanding and communicating results. Output is typically voluminous, and researchers will often get lost in the numbers. While software (SPSS, SAS and R) make tabular summary displays easy, these often obscure the findings that researchers are most interested in. The most common analytic oversights that we have observed are:

- **Atomistic data screening:** Researchers have mostly learned the assumptions (the Holy Trinity of normality, constant variance and independence) of univariate linear models, but then apply *univariate* tests (e.g., Shapiro-Wilk) and diagnostic plots (normal QQ plots) to every predictor and every response.
- **Bonferroni everywhere:** Faced with the task of reporting the results for multiple response measures and a collection of predictors for each, a common tendency is to run (and sometimes report) each of the separate univariate response models and then apply a correction for multiple testing. Not only is this confusing and awkward to report, but it is largely unnecessary because the multivariate tests provide protection for multiple testing.
- **Reverting to univariate visualizations:** To display results, SPSS and SAS make some visualization methods available through menu choices or syntax, but usually these are the wrong (or at least unhelpful) choices, in that they generate separate univariate graphs for the individual responses.

This book to discusses a few essential procedures for multivariate linear models, how their interpretation can be aided through the use of well-crafted (though novel) visualizations, and provides replicable sample code in R to showcase their use in applied behavioral research. A later section [ref?] provides some practical guidelines for analyzing, visualizing and reporting such models to help avoid these and other problems.

Package summary:

1 packages used here: knitr

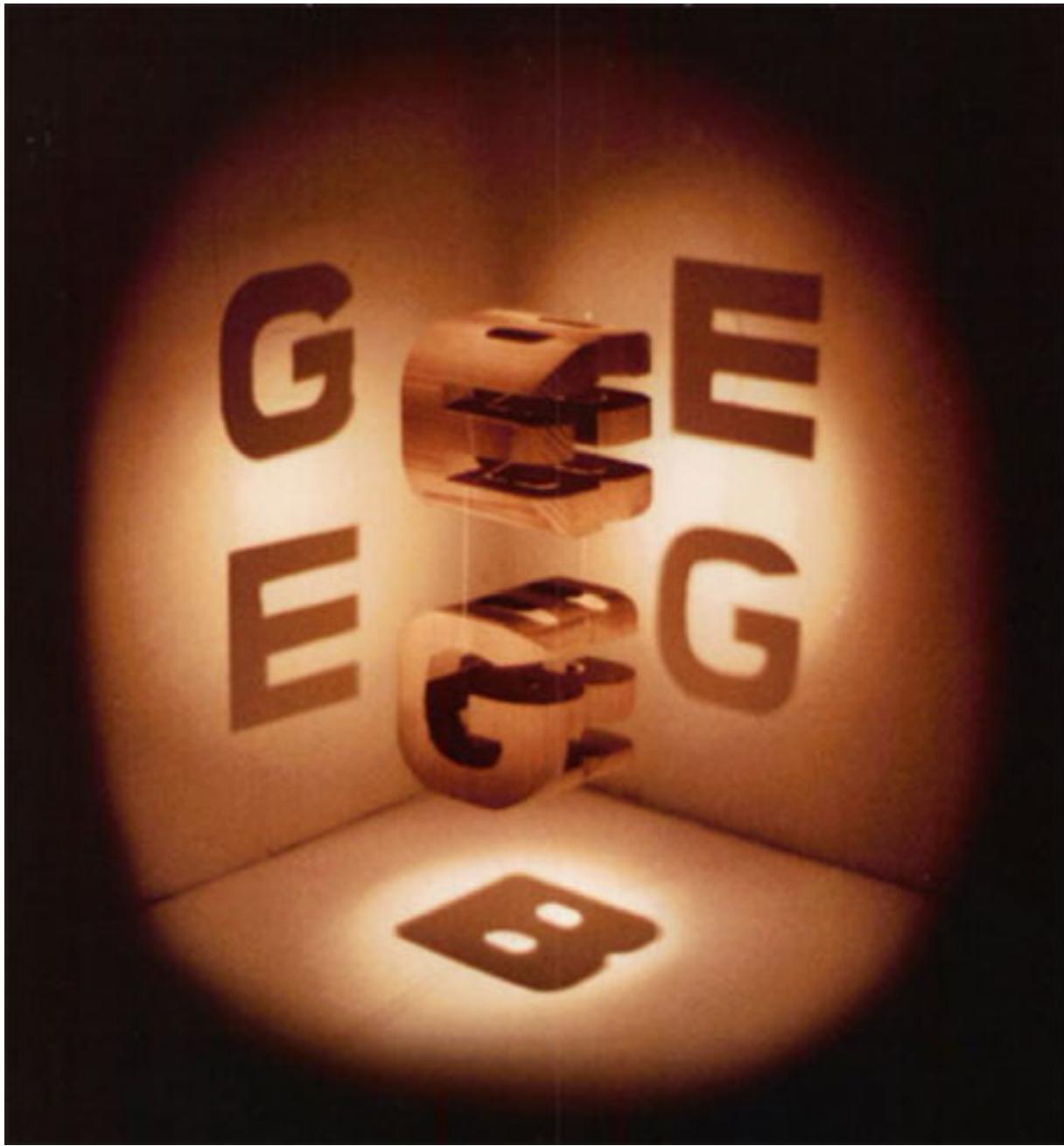


Figure 1.1: Projection: The cover image from Hofstadter's *Gödel, Bach and Escher* illustrates projection of 3D solids onto each 2D plane.

2

Getting Started

2.1 Why plot your data?

Getting information from a table is like extracting sunlight from a cucumber. Farquhar & Farquhar (1891)

At the time the Farquhar brothers wrote this pithy aphorism, graphical methods for understanding data had advanced considerably, but were not universally practiced, prompting their complaint. Most data tables we see are designed for looking something up like the number of measles cases in Ontario compared to other provinces. They are not usually designed to reveal *patterns, trends or anomalies*, although this can be accomplished with modern table generating software such as the `tinytable` or `gt` packages.¹

The main graphic forms we use today—the pie chart, line graphs and bar—were invented by William Playfair around 1800 (Playfair, 1786, 1801). The scatterplot arrived shortly after (Herschel, 1833) and thematic maps showing the spatial distributions of social variables (crime, suicides, literacy) were used for the first time to reason about important societal questions (Guerry, 1833) such as “is increased education associated with lower rates of crime?”

In the last half of the 18th Century, the idea of correlation was developed (Galton, 1886; Pearson, 1896) and the period, roughly 1860–1890, dubbed the “Golden Age of Graphics (Funkhouser, 1937) became the richest period of innovation and beauty in the entire history of data visualization. During this time there was an incredible development of *visual thinking*, represented by the work of Charles Joseph Minard, advances in the role of visualization within scientific discovery, as illustrated through Francis Galton, and graphical excellence, embodied in state statistical atlases produced in France and elsewhere. See Friendly (2008); Friendly & Wainer (2021) for this history. .

This chapter introduces the importance of graphing data through three nearly classic stories with the following themes:

- **summary statistics are not enough:** Anscombe’s Quartet demonstrates datasets that are indistinguishable by numerical summary statistics (mean, standard deviation, correlation), but whose relationships are vastly different.
- **one lousy point can ruin your day:** A researcher is mystified by a difference between a correlation for men and women until she plots the data.
- **finding the signal in noise:** The story of the US 1970 Draft Lottery shows how a weak, but reliable signal, reflecting bias in a process can be revealed by graphical enhancement and summarization.

2.1.1 Anscombe’s Quartet

In 1973, Francis Anscombe (Anscombe, 1973) famously constructed a set of four datasets illustrate the importance of plotting the graphs before analyzing and model building, and the effect of unusual observations

¹For example, a cell in a table can be used to show a “sparklines” (Tufte (1983)), tiny versions of a line graph or bar chart. As well, table rows and/or columns can be sorted to show trends or background colors can be used to show unusual values.

on fitted models. Now known as *Anscombe's Quartet*, these datasets had identical statistical properties: the same means, standard deviations, correlations and regression lines.

His purpose was to debunk three notions that had been prevalent at the time:

- Numerical calculations are exact, but graphs are coarse and limited by perception and resolution;
- For any particular kind of statistical data there is just one set of calculations constituting a correct statistical analysis;
- Performing intricate calculations is virtuous, whereas actually looking at the data is cheating.

The dataset `datasets::anscombe` has 11 observations, recorded in wide format, with variables `x1:x4` and `y1:y4`.

```
data(anscombe)
head(anscombe)
#>   x1 x2 x3 x4   y1   y2   y3   y4
#> 1 10 10 10   8 8.04 9.14 7.46 6.58
#> 2  8   8   8   8 6.95 8.14 6.77 5.76
#> 3 13 13 13   8 7.58 8.74 12.74 7.71
#> 4  9   9   9   8 8.81 8.77 7.11 8.84
#> 5 11 11 11   8 8.33 9.26 7.81 8.47
#> 6 14 14 14   8 9.96 8.10 8.84 7.04
```

The following code transforms this data to long format and calculates some summary statistics for each dataset.

```
anscombe_long <- anscombe |>
  pivot_longer(everything(),
    names_to = c(".value", "dataset"),
    names_pattern = "(.)(.)"
  ) |>
  arrange(dataset)

anscombe_long |>
  group_by(dataset) |>
  summarise(xbar      = mean(x),
            ybar      = mean(y),
            r         = cor(x, y),
            intercept = coef(lm(y ~ x))[1],
            slope     = coef(lm(y ~ x))[2]
  )
#> # A tibble: 4 x 6
#>   dataset xbar  ybar      r intercept slope
#>   <chr>   <dbl> <dbl>  <dbl>     <dbl> <dbl>
#> 1 1       9    7.50  0.816    3.00  0.500
#> 2 2       9    7.50  0.816    3.00  0.5
#> 3 3       9    7.5  0.816    3.00  0.500
#> 4 4       9    7.50  0.817    3.00  0.500
```

As we can see, all four datasets have nearly identical univariate and bivariate statistical measures. You can only see how they differ in graphs, which show their true natures to be vastly different.

Figure 2.1 is an enhanced version of Anscombe's plot of these data, adding helpful annotations to show visually the underlying statistical summaries.

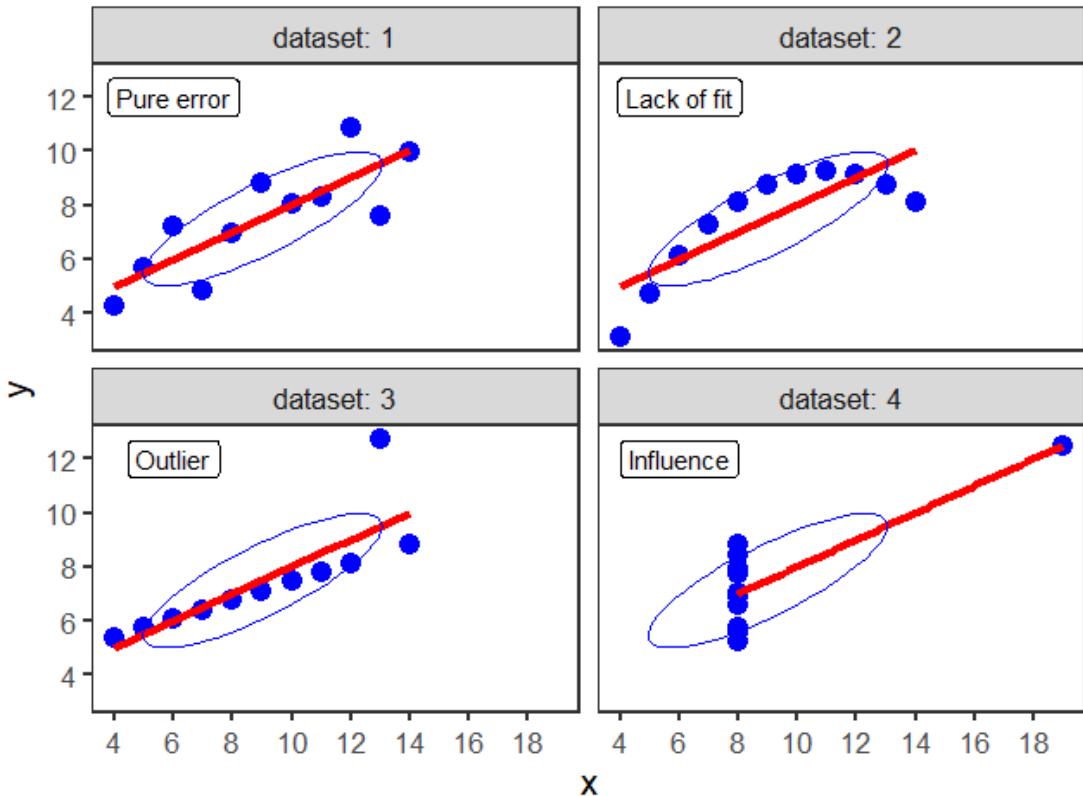


Figure 2.1: Scatterplots of Anscombe's Quartet. Each plot shows the fitted regression line and a 68% data ellipse representing the correlation between x and y .

This figure is produced as follows, using a single call to `ggplot()`, faceted by `dataset`. As we will see later (Section 3.2), the data ellipse (produced by `stat_ellipse()`) reflects the correlation between the variables.

```
desc <- tibble(
  dataset = 1:4,
  label = c("Pure error", "Lack of fit", "Outlier", "Influence")
)

ggplot(anscombe_long, aes(x = x, y = y)) +
  geom_point(color = "blue", size = 4) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE,
             color = "red", linewidth = 1.5) +
  scale_x_continuous(breaks = seq(0,20,2)) +
  scale_y_continuous(breaks = seq(0,12,2)) +
  stat_ellipse(level = 0.5, color=col, type="norm") +
  geom_label(data=desc, aes(label = label), x=6, y=12) +
  facet_wrap(~dataset, labeller = label_both)
```

The subplots are labeled with the statistical idea they reflect:

- dataset 1: **Pure error.** This is the typical case with well-behaved data. Variation of the points around the line reflect only measurement error or unreliability in the response, y .
- dataset 2: **Lack of fit.** The data is clearly curvilinear, and would be very well described by a quadratic,

`y ~ poly(x, 2)`. This violates the assumption of linear regression that the fitted model has the correct form.

- dataset 3: **Outlier**. One point, second from the right, has a very large residual. Because this point is near the extreme of x , it pulls the regression line towards it, as you can see by imagining a line through the remaining points.
- dataset 4: **Influence**. All but one of the points have the same x value. The one unusual point has sufficient influence to force the regression line to fit it **exactly**.

One moral from this example:

Linear regression only “sees” a line. It does its’ best when the data are really linear. Because the line is fit by least squares, it pulls the line toward discrepant points to minimize the sum of squared residuals.

i Datasaurus Dozen

The method Anscombe used to compose his quartet is unknown, but it turns out that there is a method to construct a wider collection of datasets with identical statistical properties. After all, in a bivariate dataset with n observations, the correlation has $(n - 2)$ degrees of freedom, so it is possible to choose $n - 2$ of the (x, y) pairs to yield any given value. As it happens, it is also possible to create any number of datasets with the same means, standard deviations and correlations with nearly any shape you like — even a dinosaur!

The *Datasaurus Dozen* was first publicized by Alberto Cairo in a [blog post](#) and are available in the [datasauRus](#) package ([Davies et al., 2022](#)) . As shown in `?@fig-datasaurus`, the sets include a star, cross, circle, bullseye, horizontal and vertical lines, and, of course the “dino”. The method ([Matejka & Fitzmaurice, 2017](#)) uses *simulated annealing*, an iterative process that perturbs the points in a scatterplot, moving them towards a given shape while keeping the statistical summaries close to the fixed target value.

The [datasauRus](#) package just contains the datasets, but a general method, called *statistical metamers*, for producing such datasets has been described by [Elio Campitelli](#) and implemented in the [metamer](#) package.

i Quartets

The essential idea of a statistical “quartet” is to illustrate four quite different datasets or circumstances that seem superficially the same, but yet are paradoxically very different when you look behind the scenes. For example, in the context of causal analysis Gelman et al. ([2023](#)), illustrated sets of four graphs, within each of which all four represent the same average (latent) causal effect but with much different patterns of individual effects; McGowan et al. ([2023](#)) provide another illustration with four seemingly identical data sets each generated by a different causal mechanism. As an example of machine learning models, Biecek et al. ([2023](#)), introduced the “Rashamon Quartet”, a synthetic dataset for which four models from different classes (linear model, regression tree, random forest, neural network) have practically identical predictive performance. In all cases, the paradox is solved when their visualization reveals the distinct ways of understanding structure in the data. The [quartets](#) package contains these and other variations on this theme.

2.1.2 One lousy point can ruin your day

In the mid 1980s, a consulting client had a strange problem.² She was conducting a study of the relation between body image and weight preoccupation in exercising and non-exercising people ([Davis, 1990](#)). As part of the design, the researcher wanted to know if self-reported weight could be taken as a reliable indicator

²This story is told apocryphally. The consulting client actually did plot the data, but needed help making better graphs.

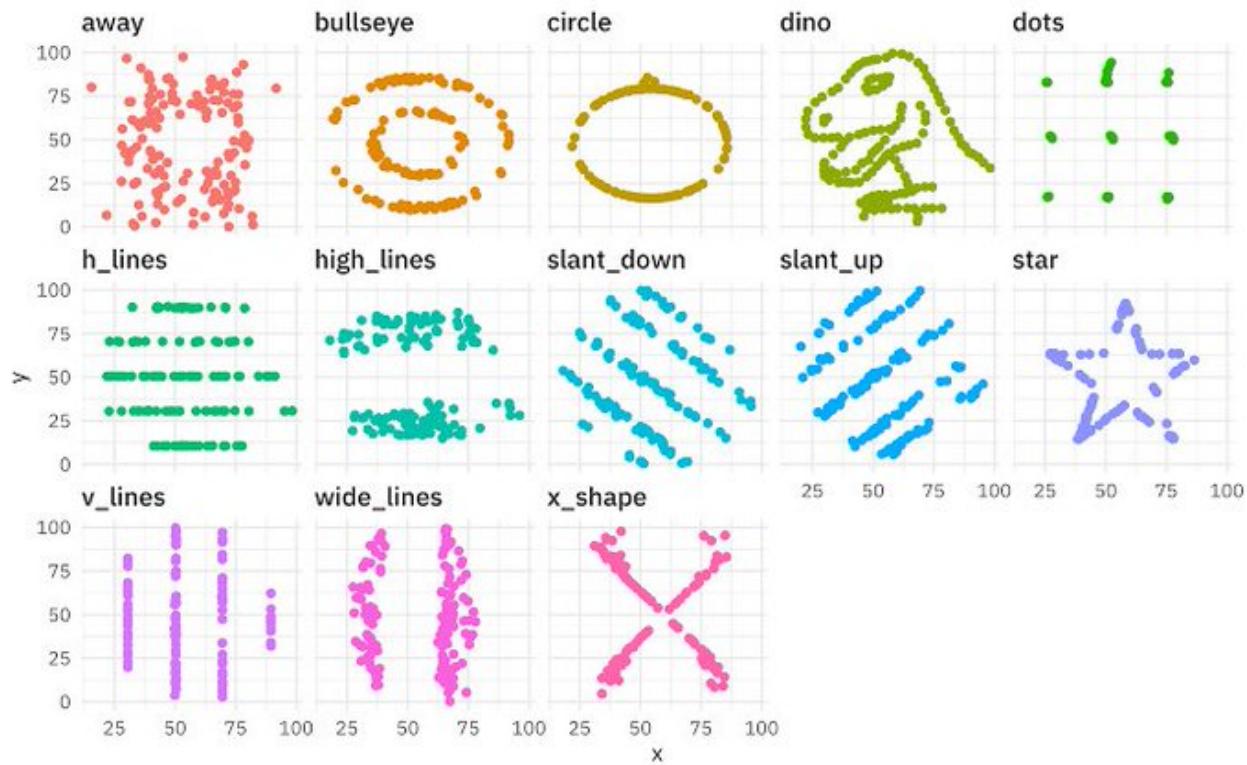


Figure 2.2: Plots of the Dinosaur Dozen datasets. Source: [Selçuk Korkmaz on X](#)

of true weight measured on a scale. It was expected that the correlations between reported and measured weight should be close to 1.0, and the slope of the regression lines for men and women should also be close to 1.0. The dataset is `carData::Davis`.

She was therefore very surprised to see the following numerical results: For men, the correlation was nearly perfect, but not so for women.

```
data(Davis, package="carData")
Davis <- Davis |>
  drop_na()          # drop missing cases
Davis |>
  group_by(sex) |>
  select(sex, weight, repwt) |>
  summarise(r = cor(weight, repwt))
#> # A tibble: 2 x 2
#>   sex      r
#>   <fct> <dbl>
#> 1 F      0.501
#> 2 M      0.979
```

Similarly, the regression lines showed the expected slope for men, but that for women was only 0.26.

```
Davis |>
  nest(data = -sex) |>
  mutate(model = map(data, ~ lm(repwt ~ weight, data = .)),
```

```

tidied = map(model, tidy)) |>
unnest(tidied) |>
filter(term == "weight") |>
select(sex, term, estimate, std.error)
#> # A tibble: 2 x 4
#>   sex    term   estimate std.error
#>   <fct> <chr>     <dbl>     <dbl>
#> 1 M      weight     0.990    0.0229
#> 2 F      weight     0.262    0.0459

```

“What could be wrong here?”, the client asked. The consultant replied with the obvious question:

Did you plot your data?

The answer turned out to be one discrepant point, a female, whose measured weight was 166 kg (366 lbs!). This single point exerted so much influence that it pulled the fitted regression line down to a slope of only 0.26.

```

# shorthand to position legend inside the figure
legend_inside <- function(position) {
  theme(legend.position = "inside",
        legend.position.inside = position)
}

Davis |>
  ggplot(aes(x = weight, y = repwt,
             color = sex, shape = sex, linetype = sex)) +
  geom_point(size = ifelse(Davis$weight==166, 6, 2)) +
  geom_smooth(method = "lm", formula = y~x, se = FALSE) +
  labs(x = "Measured weight (kg)",
       y = "Reported weight (kg)") +
  scale_linetype_manual(values = c(F = "longdash",
                                    M = "solid")) +
  legend_inside(c(.8, .8))

```

In this example, it was arguable that x and y axes should be reversed, to determine how well measured weight can be predicted from reported weight. In `ggplot` this can easily be done by reversing the `x` and `y` aesthetics.

```

Davis |>
  ggplot(aes(y = weight, x = repwt, color = sex, shape=sex)) +
  geom_point(size = ifelse(Davis$weight==166, 6, 2)) +
  labs(y = "Measured weight (kg)",
       x = "Reported weight (kg)") +
  geom_smooth(method = "lm", formula = y~x, se = FALSE) +
  legend_inside(c(.8, .8))

```

In Figure 2.4, this discrepant observation again stands out like a sore thumb, but it makes very little difference in the fitted line for females. The reason is that this point is well within the range of the x variable (`repwt`). To impact the slope of the regression line, an observation must be unusual in *both* x and y . We take up the topic of how to detect influential observations and what to do about them in Chapter 6.

The value of such plots is not only that they can reveal possible problems with an analysis, but also help identify their reasons and suggest corrective action. What went wrong here? Examination of the original

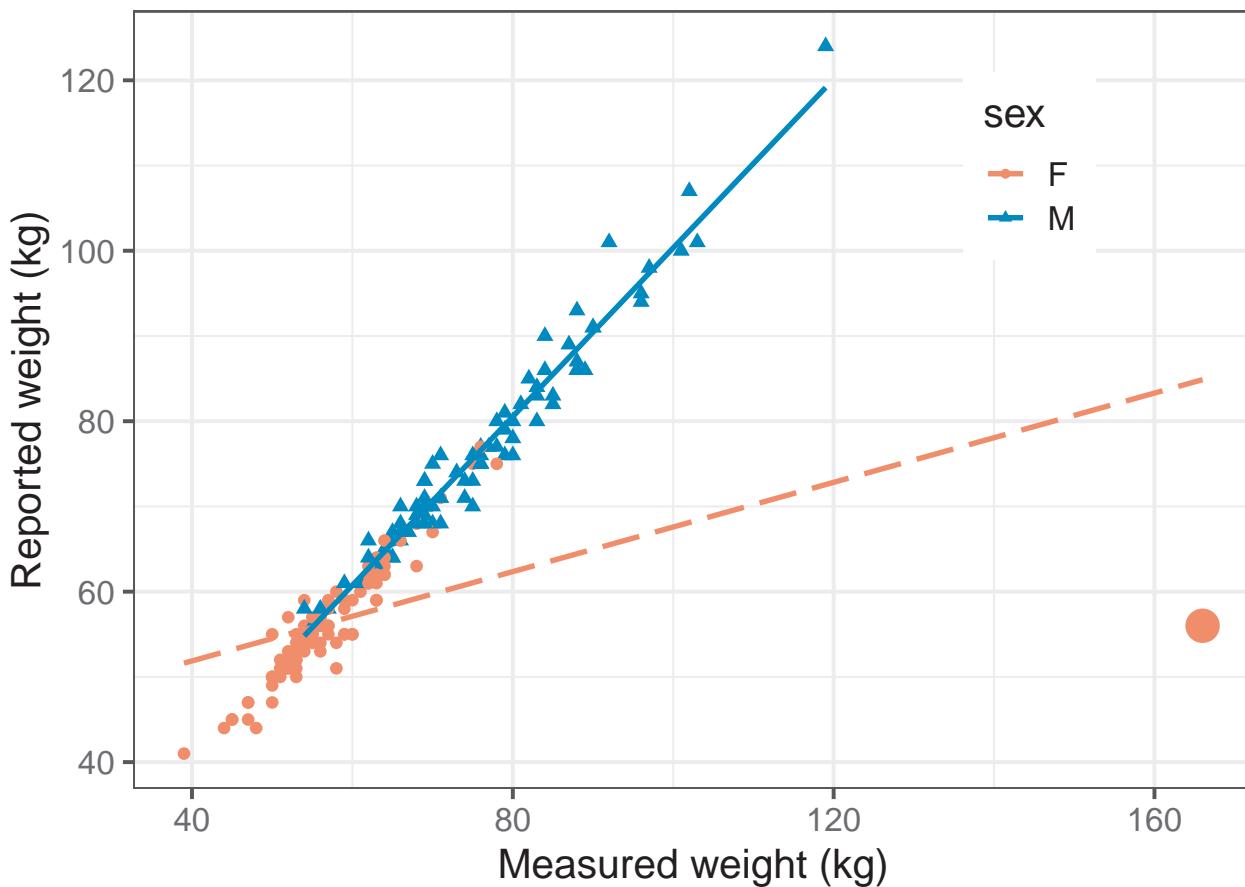


Figure 2.3: Regression for Davis' data on reported weight and measures weight for men and women. Separate regression lines, predicting reported weight from measured weight are shown for males and females. One highly unusual point is highlighted.

data showed that this person switched the values, recording her reported weight in the box for measured weight and vice versa.

2.1.3 Shaken, not stirred: The 1970 Draft Lottery

Although we often hear that data speak for themselves, their voices can be soft and sly.—Frederick Mosteller (1983), *Beginning Statistics with Data Analysis*, p. 234.

The power of graphics is particularly evident when data contains a weak signal embedded in a field of noise. To the casual glance, there may seem to be nothing going on, but the signal can be made apparent in an incisive graph.

A dramatic example of this occurred in 1969 when the U.S. military conducted a lottery, the first since World War II, to determine which young men would be called up to serve in the Vietnam War for 1970. The U.S. Selective Service had devised a system to rank eligible men according to a random drawing of their birthdays. There were 366 blue plastic capsules containing birth dates placed in a transparent glass container and drawn by hand to assign ranked order-of-call numbers to all men within the 18-26 age range.

In an attempt to make the selection process also transparent, the proceeding was covered on radio, TV and film and the dates posted in order on a large display board. The first capsule—drawn by Congressman Alexander Pirnie (R-NY) of the House Armed Services Committee—contained the date September 14, so all men born on September 14 in any year between 1944 and 1950 were assigned lottery number 1, and would be

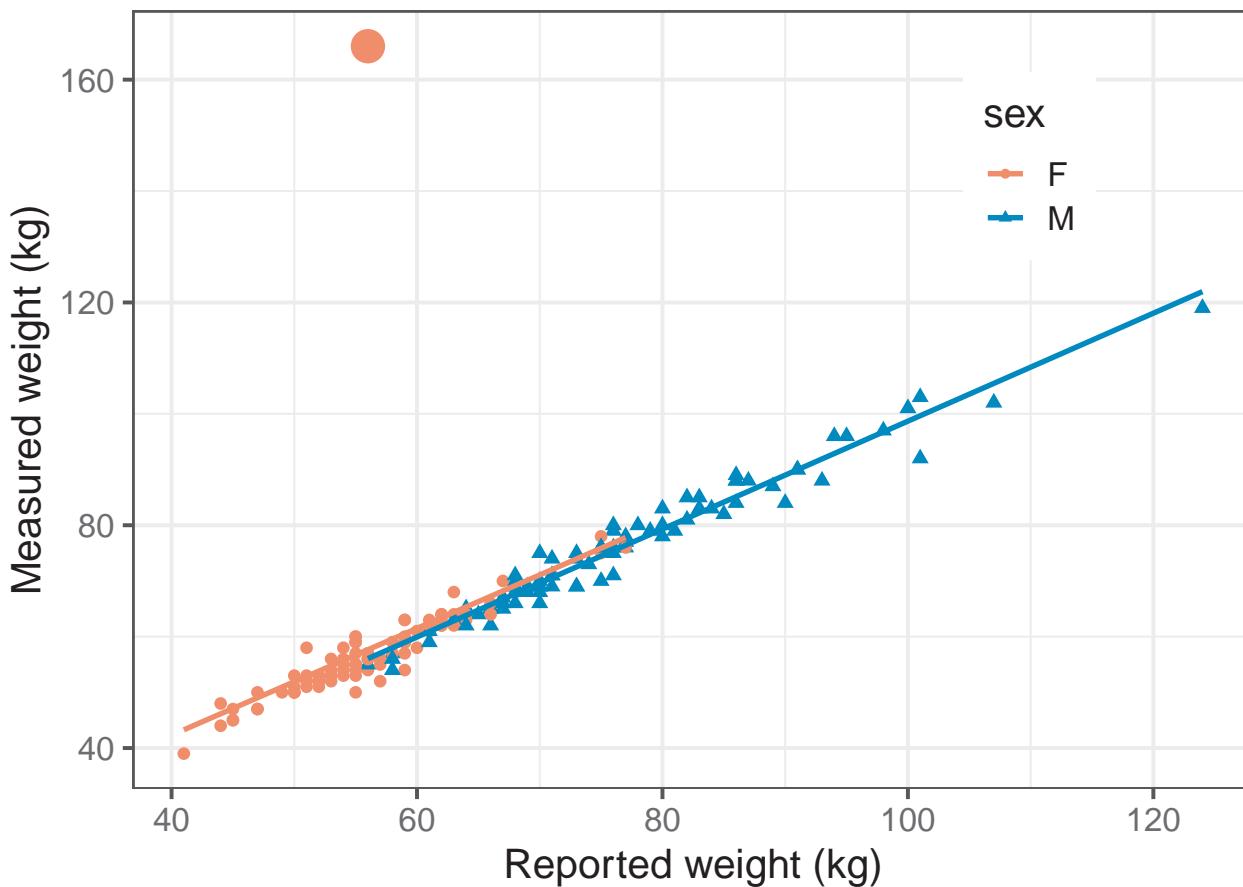


Figure 2.4: Regression for Davis' data on reported weight and measures weight for men and women. Separate regression lines, predicting measured weight from reported weight are shown for males and females. The highly unusual point no longer has an effect on the fitted lines.

drafted first. April 24 was drawn next, then December 30, February 14, and so on until June 8, selected last. At the time of the drawing, US officials stated that those with birthdays drawn in the first third would almost certainly be drafted, while those in the last third would probably avoid the draft (Fienberg, 1971).

I watched this unfold with considerable interest because I was eligible for the Draft that year. I was dismayed when my birthday, May 7, came up ranked 35. **Ugh!** Could some data analysis and graphics get me out of my funk?

The data, from the official [Selective Service listing](#) are contained in the dataset `vcdExtra::Draft1970`, ordered by `Month` and `birthdate` (`Day`), with `Rank` as the order in which the birthdates were drawn.

```
library(ggplot2)
library(dplyr)
data(Draft1970, package = "vcdExtra")
dplyr::glimpse(Draft1970)
#> Rows: 366
#> Columns: 3
#> $ Day    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
#> $ Rank   <int> 305, 159, 251, 215, 101, 224, 306, 199, 194, 325, 32~
#> $ Month  <ord> Jan, Jan, Jan, Jan, Jan, Jan, Jan, Jan, Ja~
```



Figure 2.5: Congressman Alexander Pirnie (R-NY) drawing the first capsule for the Selective Service draft, Dec 1, 1969. Source: <https://bit.ly/45c23sB>

A basic scatterplot, slightly prettified, is shown in Figure 2.6. The points are colored by month, and month labels are shown at the bottom.

```
# make markers for months at their mid points
months <- data.frame(
  month =unique(Draft1970$Month),
  mid = seq(15, 365-15, by = 30))

ggplot2:: theme_set(theme_bw(base_size = 16))
gg <- ggplot(Draft1970, aes(x = Day, y = Rank)) +
  geom_point(size = 2.5, shape = 21,
             alpha = 0.3,
             color = "black",
             aes(fill=Month)
  ) +
  scale_fill_manual(values = rainbow(12)) +
  geom_text(data=months, aes(x=mid, y=0, label=month), nudge_x = 5) +
  geom_smooth(method = "lm", formula = y ~ 1,
              col = "black", fill="grey", linetype = "dashed", alpha=0.6) +
  labs(x = "Day of the year",
       y = "Lottery rank") +
  theme(legend.position = "none")
gg
```

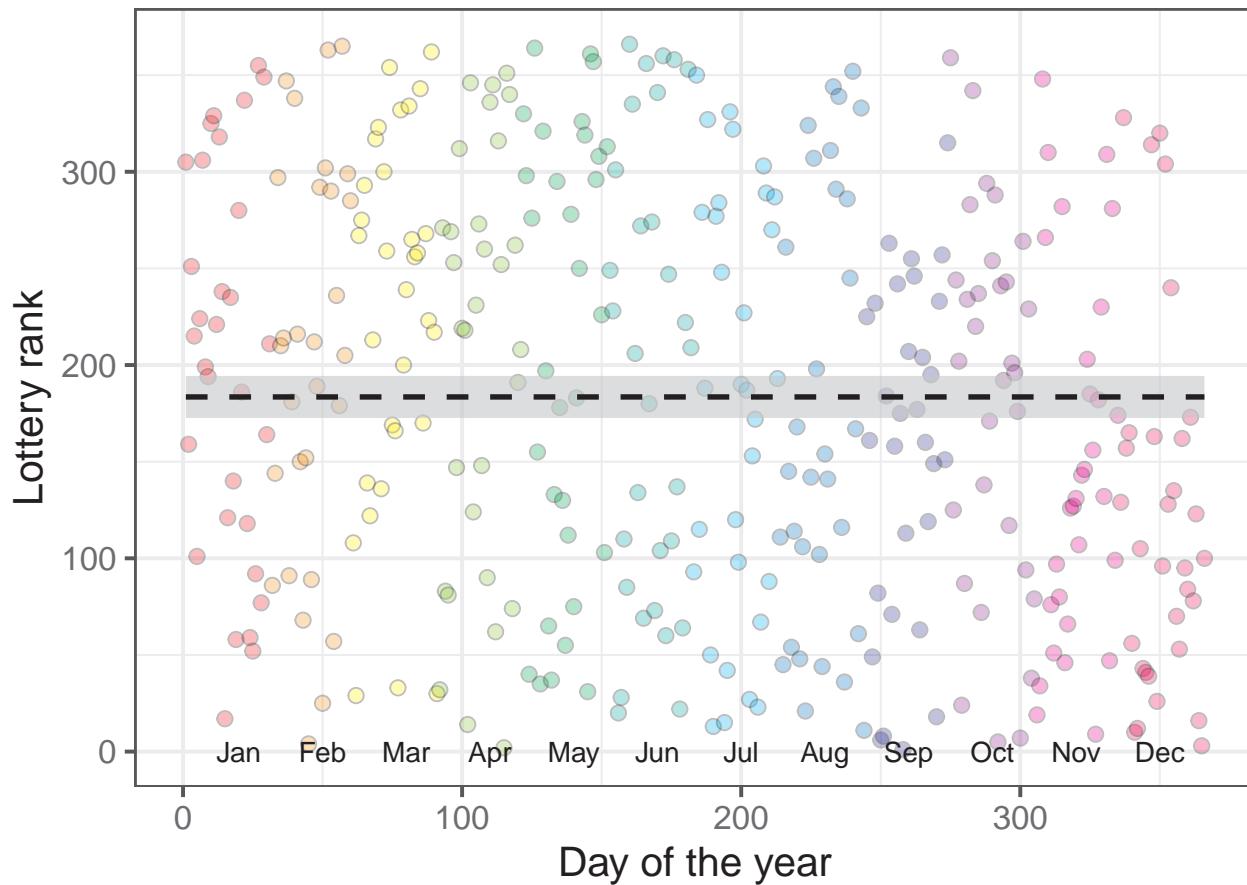


Figure 2.6: Basic scatterplot of 1970 Draft Lottery data plotting rank order of selection against birthdates in the year. Points are colored by month. The horizontal line is at the average rank.

The ranks do seem to be essentially random. Is there any reason to suspect a flaw in the selection process, as I firmly hoped at the time?

If you stare at the graph in Figure 2.6 long enough, you just can make out a sparsity of points in the upper right corner and also in the lower left corner compared to the opposite corners. But probably not until I told you.

Visual smoothers

Fitting a linear regression line or a smoothed (loess) curve can bring out the signal lurking in the background of a field of nearly random points. Figure 2.7 shows a definite trend to lower ranks for birthdays toward the end of the year. Those born earlier in the year were more likely to be given lower ranks, calling them up sooner for the draft.

```
ggplot(Draft1970, aes(x = Day, y = Rank)) +
  geom_point(size = 2.5, shape = 21,
             alpha = 0.3,
             color = "black",
             aes(fill=Month)) +
  scale_fill_manual(values = rainbow(12)) +
  geom_smooth(method = "lm", formula = y~1,
```

```

    se = FALSE,
    col = "black", fill="grey", linetype = "dashed", alpha=0.6) +
geom_smooth(method = "loess", formula = y~x,
            color = "blue", se = FALSE,
            alpha=0.25) +
geom_smooth(method = "lm", formula = y~x,
            color = "darkgreen",
            fill = "darkgreen",
            alpha=0.25) +
geom_text(data=months, aes(x=mid, y=0, label=month), nudge_x = 5) +
labs(x = "Day of the year",
     y = "Lottery rank") +
theme(legend.position = "none")

```

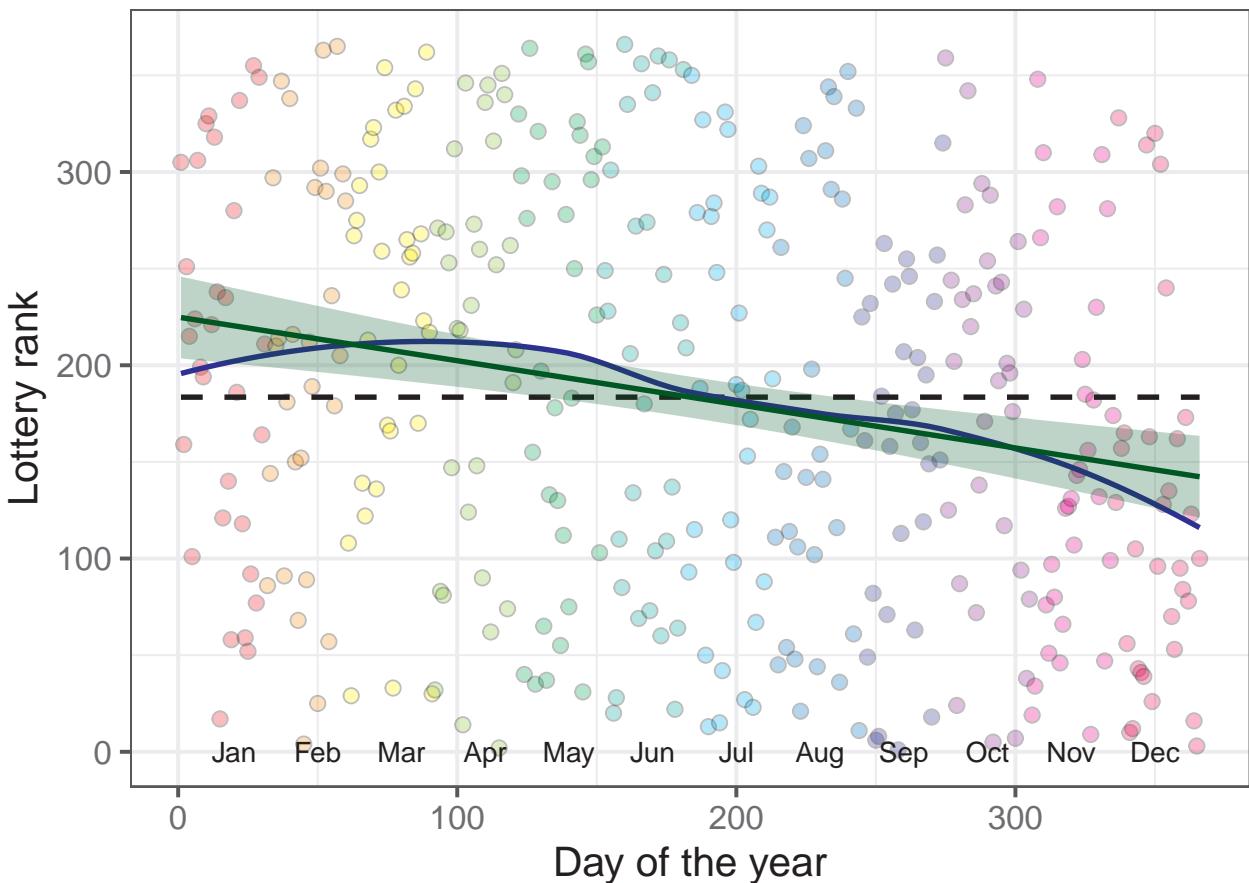


Figure 2.7: Enhanced scatterplot of 1970 Draft Lottery data adding a linear regression line and loess smooth.

Is this a real effect? Even though the points seem to be random over the year, linear regression of Rank on Day shows a highly significant negative effect even though the correlation³ is small ($r = -0.226$). The slope, -0.226 , means that for each additional day in the year the lottery rank decreases about $1/4$ toward the front of the draft line; that's nearly 7 ranks per month.

³Because both days of the year and rank in the lottery are the integers, 1 to 366, the Pearson correlation and Spearman rank order correlation are identical.

```

draft.mod <- lm(Rank ~ Day, data=Draft1970)
with(Draft1970, cor(Day, Rank))
#> [1] -0.226
coef(draft.mod)
#> (Intercept)      Day
#>     224.913    -0.226

```

So, smoothing the data, using either the linear regression line or a nonparametric smoother is one important technique for seeing a weak signal in a noisy background.

Statistical summaries

Another way to enhance the signal-to-noise ratio of a graph is to plot summaries of the messy data points. For example, you might make boxplots of the ranks by month, or calculate and plot the mean or median rank by month and plot those together with some indication of variability within month.

Figure 2.8 plots the average Rank for each month with error bars showing the mean ± 1 standard errors against the average Day. The message of rank decreasing nearly linearly with month is now more dramatic. The correlation between the means is $r = -0.867$; the slope is -0.231 , similar to what we found for the raw data.

```

means <- Draft1970 |>
  group_by(Month) |>
  summarize(Day = mean(Day),
            se = sd(Rank / sqrt(n())),
            Rank = mean(Rank))

ggplot(aes(x = Day, y = Rank), data=means) +
  geom_point(size = 4) +
  geom_smooth(method = "lm", formula = y~x,
              color = "blue", fill = "blue", alpha = 0.1) +
  geom_errorbar(aes(ymin = Rank-se, ymax = Rank+se),
                width = 8, linewidth = 1.3) +
  geom_text(data=months, aes(x=mid, y=0, label=month), nudge_x = 5) +
  ylim(100, 250) +
  labs(x = "Average day of the year",
       y = "Average lottery rank")

```

The visual impression of a linearly decreasing trend in lottery rank is much stronger in Figure 2.8 than in Figure 2.7 for two reasons:

- Replacing the data points with their means strengthens the signal in relation to noise, which is essentially eliminated by plotting means and error bars rather than the raw data.
- The narrower vertical range (100–250) in the plot of means makes the slope of the line appear steeper. (However, the slope of the means, $b = -0.231$ is nearly the same as that for the data points.) The narrower range also makes deviations from the regression line more noticeable.

What happened here?

Previous lotteries carried out by drawing capsules from a container had occasionally suffered the embarrassment that an empty capsule was selected because of vigorous mixing (Fienberg, 1971). So for the 1970 lottery, the birthdate capsules were put in cardboard boxes, one for each month and these were carefully emptied into the glass container in order of month: Jan., Feb., ... Dec., gently shaken in atop the pile already there. All might have been well had the persons drawing the capsules put their hand in truly randomly, but generally

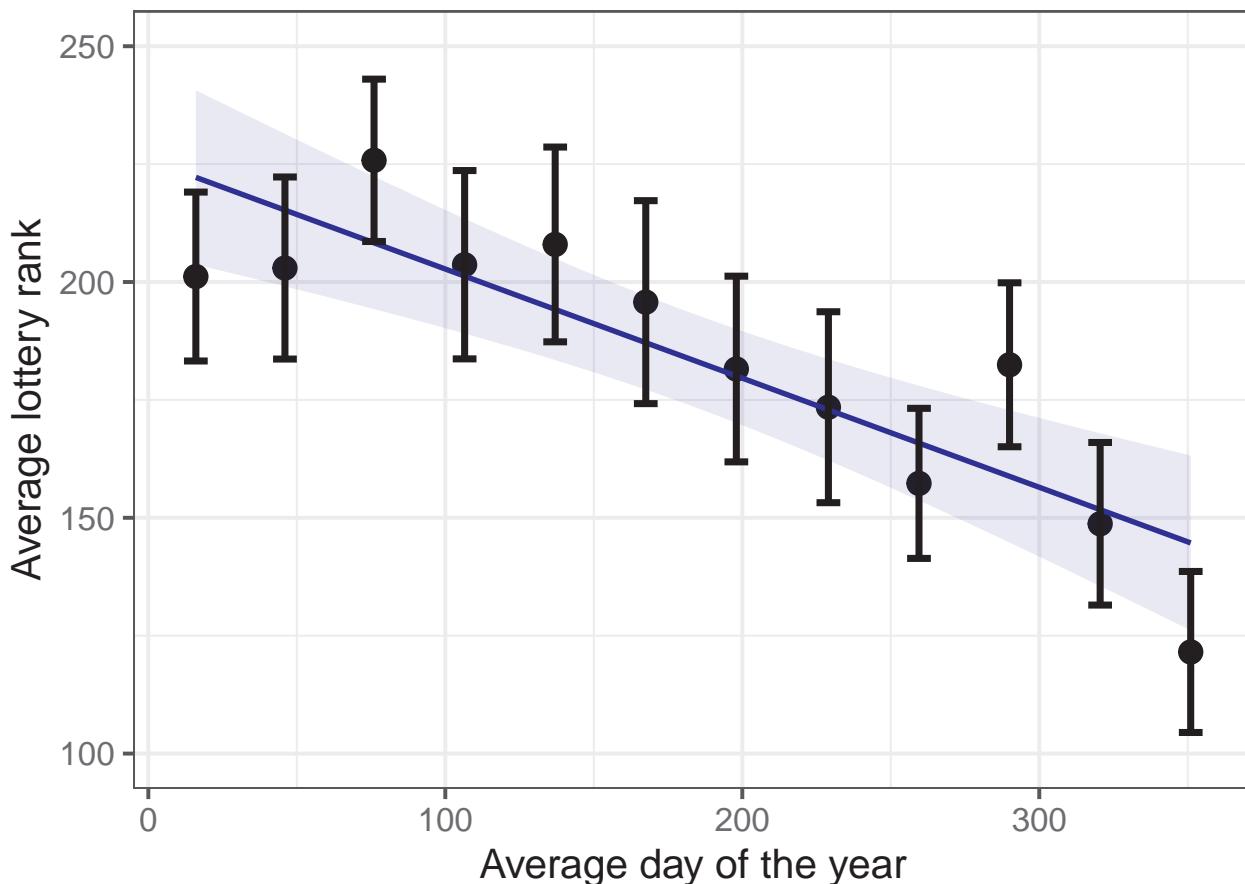


Figure 2.8: Plot of the average rank per month with ± 1 standard error bars. The line shows the least squares regression line, treating months as equally spaced.

they picked from toward the top of the container. Consequently, those born later in the year had a greater chance of being picked earlier.

There was considerable criticism of this procedure once the flaw had been revealed by analyses such as described here. In the following year, the Selective Service called upon the National Bureau of Standards to devise a better procedure. In 1971 they used two drums, one with the dates of the year and another with the rank numbers 1–366. As a date capsule was drawn randomly from the first drum, another from the numbers drum was picked simultaneously, giving a doubly-randomized sequence.

Of course, if they had R, the entire process could have been done using `sample()`:

```
set.seed(42)
date = seq(as.Date("1971-01-01"), as.Date("1971-12-31"), by="+1 day")
rank = sample(seq_along(date))
draft1971 <- data.frame(date, rank)

head(draft1971, 4)
#>      date rank
#> 1 1971-01-01    49
#> 2 1971-01-02   321
#> 3 1971-01-03   153
```

```
#> 4 1971-01-04 74
tail(draft1971, 4)
#>           date rank
#> 362 1971-12-28 247
#> 363 1971-12-29   8
#> 364 1971-12-30 333
#> 365 1971-12-31 132
```

And, what would have happened to me and all others born on a May 7th, if they did it this way? My lottery rank would have 274!⁴

```
me <- as.Date("1971-05-07")
draft1971[draft1971$date == me,]
#>           date rank
#> 127 1971-05-07 274
```

2.2 Plots for data analysis

Visualization methods take an enormous variety of forms, so it is useful to distinguish several broad categories according to their *use in data analysis*:

- **data plots** : primarily plot the raw data, often with annotations to aid interpretation (regression lines and smooths, data ellipses, marginal distributions). A survey of these methods is presented in Section 3.1.
- **reconnaissance plots** : with more than a few variables, reconnaissance plots provide a high-level, bird's-eye overview of the data, allowing you to see patterns that might not be visible in a set of separate plots. Some examples are scatterplot matrices (Section 3.10) showing all bivariate plots of variables in a dataset; correlation diagrams (Section 3.11), using visual glyphs to represent the correlations between all pairs of variables and “trellis” or faceted plots that show how a focal relation of one or more variables differs across values of other variables.
- **model plots** : plot the results of a fitted model, such as a regression line or curve to show uncertainty, or a regression surface in 3D, or a plot of coefficients in model together with confidence intervals. Other model plots try to take into account that a fitted model may involve more variables than can be shown in a static 2D plot. Some examples of these are added variable plots (Section 6.4), and marginal effect plots (Section 6.5), both of which attempt to show the *net* relation of two focal variables, controlling or adjusting for other variables in a model.
- **diagnostic plots** : indicating potential problems with the fitted model. These include residual plots, influence plots, plots for testing homogeneity of variance and so forth, illustrated in Section 6.1.2.
- **dimension reduction plots** : plot representations of the data into a space of fewer dimensions than the number of variables in the dataset. Simple examples include principal components analysis (PCA) and the related biplots, and multidimensional scaling (MDS) methods. This is the topic of Chapter 4, but this powerful idea runs through the rest of the book.

2.2.1 Model plots

Model plots show the fitted or predicted values from a statistical model and provide visual summaries...

⁴A personal note: I escaped being drafted, but I moved to Canada in 1971. Looking back today, it's one of the best decisions I ever made.

2.2.2 Diagnostic plots

2.2.3 Principles of graphical display

[This could be a separate chapter]

- Criteria for assessing graphs: communication goals
- Effective data display:
 - Make the data stand out
 - Make graphical comparison easy
 - Effect ordering: For variables and unordered factors, arrange them according to the effects to be seen
- Visual thinning: As the data becomes more complex, focus more on impactful summaries

->

2.3 What have we learned?

This chapter demonstrates why visualization isn't just a "nice-to-have" feature in data analysis—it's absolutely essential. Through compelling historical examples and modern techniques, we've discovered some fundamental idea that every data analyst should embrace:

- **Summary statistics can lie (beautifully):** Anscombe's Quartet reveals that datasets can have identical means, correlations, and regression coefficients yet tell completely different stories. The quartet's four plots—pure error, lack of fit, outliers, and influence—show that numerical summaries without visualization can lead us wildly astray. Modern extensions like the Datasaurus Dozen prove this isn't just a quirky historical example—you can literally hide a dinosaur in your data while maintaining identical statistical properties! Talk about the dinosaur in the room!
- **One rogue data point can hijack your entire analysis:** Plotting the raw data facilitates critical engagement with our statistical models. The Davis weight study demonstrates how a single influential observation (one participant who accidentally switched their reported and measured weights) can completely distort relationships between variables. What appeared to be a puzzling gender difference in the reliability of self-reported weight vanished once the data was plotted and the outlier revealed itself as a simple recording error.
- **Meaning becomes more apparent through thoughtful visualizations of our well-considered models:** Statistical modeling helps guide our attention in what might otherwise be a chaotic plot of raw data. The 1970 Draft Lottery story shows how graphics can reveal systematic bias hiding in apparently random data. While individual lottery numbers seemed random, smoothing techniques and summary plots exposed a clear pattern—later birthdays were systematically favored due to poor mixing of the lottery capsules. Sometimes the most important patterns are the ones that whisper rather than shout.
- **Different plot types serve different purposes:** The chapter introduces a taxonomy of visualization goals that helps us choose the right tool for each analytical task. Data plots show raw observations, reconnaissance plots provide bird's-eye overviews of complex datasets, model plots reveal fitted relationships, diagnostic plots expose model problems, and dimension reduction plots tame high-dimensional complexity. Each serves a distinct role in the analytical process.
- **Visual enhancement amplifies signal over noise:** Whether through smoothing lines, statistical summaries, or careful use of color and annotation, the chapter shows how thoughtful visual design can make weak patterns stand out dramatically. The Draft Lottery analysis becomes far more convincing when we plot monthly averages rather than individual data points, transforming a subtle correlation into an unmistakable trend.

The overarching message is clear: in an era where we can compute any statistic imaginable, the humble graph

remains our most powerful tool for understanding what our data are really trying to tell us. As the Farquhar brothers noted over a century ago, getting information from tables alone is like extracting sunlight from a cucumber—possible in theory, but why make it so hard on yourself?

Part II

Exploratory Methods

3

Plots of Multivariate Data

There is no excuse for failing to plot and look.

The greatest value of a picture is when it forces us to notice what we never expected to see. — John W. Tukey, *Exploratory Data Analysis*, 1977

These quotes from John Tukey remind us that data analysis should nearly always start with graphs to help us understand the main features of our data. It is important to understand the general *patterns* and *trends*: Are relationships increasing or decreasing? Are they approximately linear or non-linear? But it is also important to spot *anomalies*: “unusual” observations, groups of points that seem to differ from the rest, and so forth. As we saw with Anscombe’s quartet (Section 2.1.1) numerical summaries hide features that are immediately apparent in a plot.

This chapter introduces a toolbox of basic graphical methods for visualizing multivariate datasets. It starts with some simple techniques to enhance the basic scatterplot with graphical *annotations* such as fitted lines, curves and data ellipses to *summarize* the relation between two variables.

To visualize more than two variables, we can view all pairs of variables in a scatterplot matrix or shift gears entirely to show multiple variables along a set of parallel axes. As the number of variables increases, we may need to suppress details with stronger summaries for a high-level reconnaissance of our data terrain, as we do by zooming out on a map. For example, we can simply remove the data points or make them nearly transparent to focus on the visual summaries provided by fitted lines or other graphical summaries.

Packages

In this chapter I use the following packages. Load them now:

```
library(car)
library(ggplot2)
library(dplyr)
library(tidyr)
library(corrplot)
library(corrgram)
library(GGally)
library(ggdensity)
library(patchwork)
library(ggpcp)
library(tourr)
library(hplots)
library(gggda)
```

3.1 Bivariate summaries

The basic scatterplot is the workhorse of multivariate data visualization, showing how one variable, y , often an outcome to be explained by or varies with another, x . It is a building block for many useful techniques, so it is helpful to understand how it can be used as a tool for thinking in a wider, multivariate context.

The essential idea is that we can start with a simple version of the scatterplot and add annotations to show interesting features more clearly. We consider the following here:

- **Smoothers:** Showing overall trends, perhaps in several forms, as visual summaries such as fitted regression lines or curves and nonparametric smoothers.
- **Stratifiers:** Using color, shape or other features to identify subgroups; more generally, *conditioning* on other variables in multi-panel displays;
- **Data ellipses:** A compact 2D visual summary of bivariate linear relations and uncertainty assuming normality; more generally, contour plots of bivariate density.

Example 3.1. Academic salaries

Let's start with data on the academic salaries of faculty members collected at a U.S. college for the purpose of assessing salary differences between male and female faculty members, and perhaps address anomalies in compensation. The dataset `carData::Salaries` gives data on nine-month salaries and other variables for 397 faculty members in the 2008-2009 academic year.

```
data(Salaries, package = "carData")
str(Salaries)
#> 'data.frame': 397 obs. of 6 variables:
#> $ rank      : Factor w/ 3 levels "AsstProf","AssocProf",...: 3 3 1 3 3 2 3 3 3 ...
#> $ discipline : Factor w/ 2 levels "A","B": 2 2 2 2 2 2 2 2 2 ...
#> $ yrs.since.phd: int 19 20 4 45 40 6 30 45 21 18 ...
#> $ yrs.service : int 18 16 3 39 41 6 23 45 20 18 ...
#> $ sex        : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 1 ...
#> $ salary     : int 139750 173200 79750 115000 141500 97000 175000 147765 119250 129000 ...
```

The most obvious, but perhaps naive, predictor of `salary` is `years.since.phd`. For simplicity, I'll refer to this as years of "experience." Before looking at differences between males and females, we would want consider faculty `rank` (related also to `yrs.service`) and `discipline`, recorded here as "A" ("theoretical" departments) or "B" ("applied" departments). But, for a basic plot, we will ignore these for now to focus on what can be learned from plot annotations.

```
library(ggplot2)
gg1 <- ggplot(Salaries,
  aes(x = yrs.since.phd, y = salary)) +
  geom_jitter(size = 2) +
  scale_y_continuous(labels = scales::dollar_format(
    prefix = "$", scale = 0.001, suffix = "K")) +
  labs(x = "Years since PhD",
       y = "Salary")

gg1 + geom_rug(position = "jitter", alpha = 1/4)
```

There is quite a lot we can see "just by looking" at this simple plot, but the main things are:

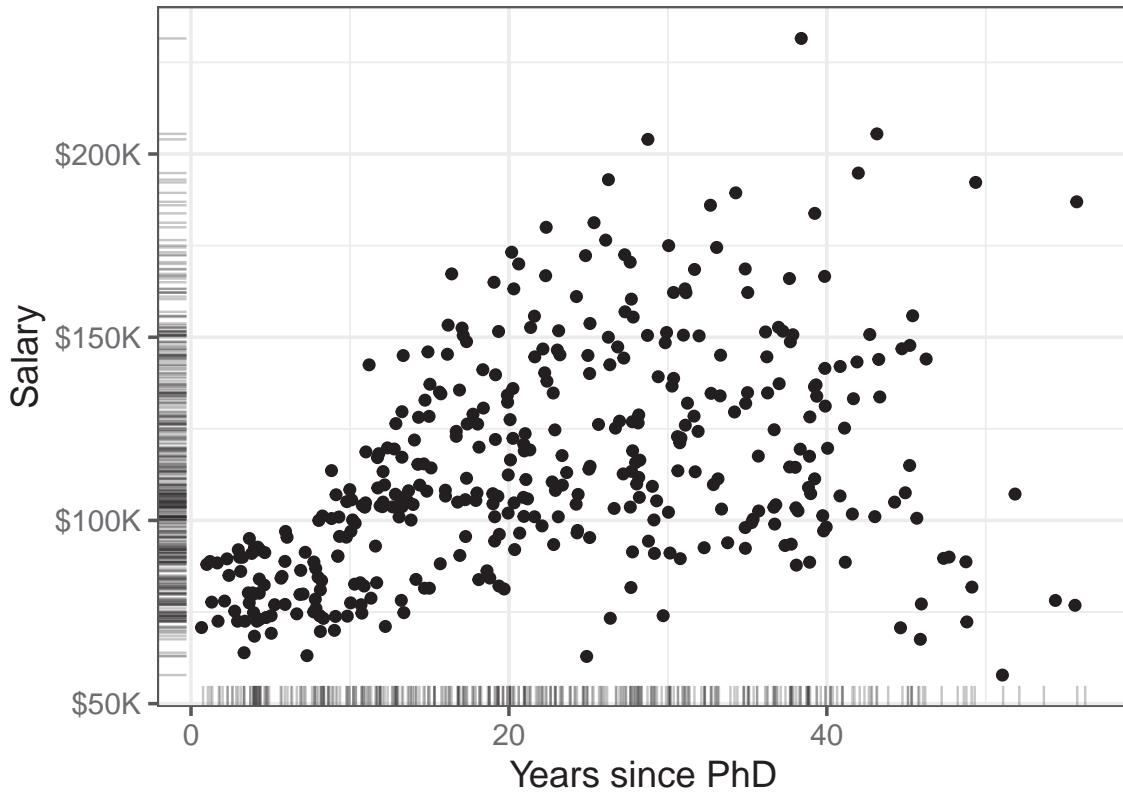


Figure 3.1: Naive scatterplot of Salary vs. years since PhD, ignoring other variables, and without graphical annotations.

- Salary increases generally from 0 - 40 years since the PhD, but then maybe begins to drop off (partial retirement?);
- Variability in salary increases among those with the same experience, a “fan-shaped” pattern that signals a violation of homogeneity of variance in simple regression;
- Data beyond 50 years is thin, but there are some quite low salaries there. Adding rug plots to the X and Y axes is a simple but effective way to show the marginal distributions of the observations. Jitter and transparency helps to avoid overplotting due to discrete values.

3.1.1 Smoothers

Smoothers are among the most useful graphical annotations you can add to such plots, giving a visual summary of how y changes with x . The most common smoother is a line showing the linear regression for y given x , expressed in math notation as $\mathbb{E}(y|x) = b_0 + b_1x$. If there is doubt that a linear relation is an adequate summary, you can try a quadratic or other polynomial smoothers.

In `ggplot2`, these are easily added to a plot using `geom_smooth()` with `method = "lm"`, and a model `formula`, which (by default) is `y ~ x` for a linear relation or `y ~ poly(x, k)` for a polynomial of degree k .

```
gg1 +
  geom_smooth(method = "lm", formula = "y ~ x",
              color = "red", fill= "pink",
              linewidth = 2) +
  geom_smooth(method = "lm", formula = "y ~ poly(x,2)",
```

```
color = "darkgreen", fill = "lightgreen",
linewidth = 2)
```

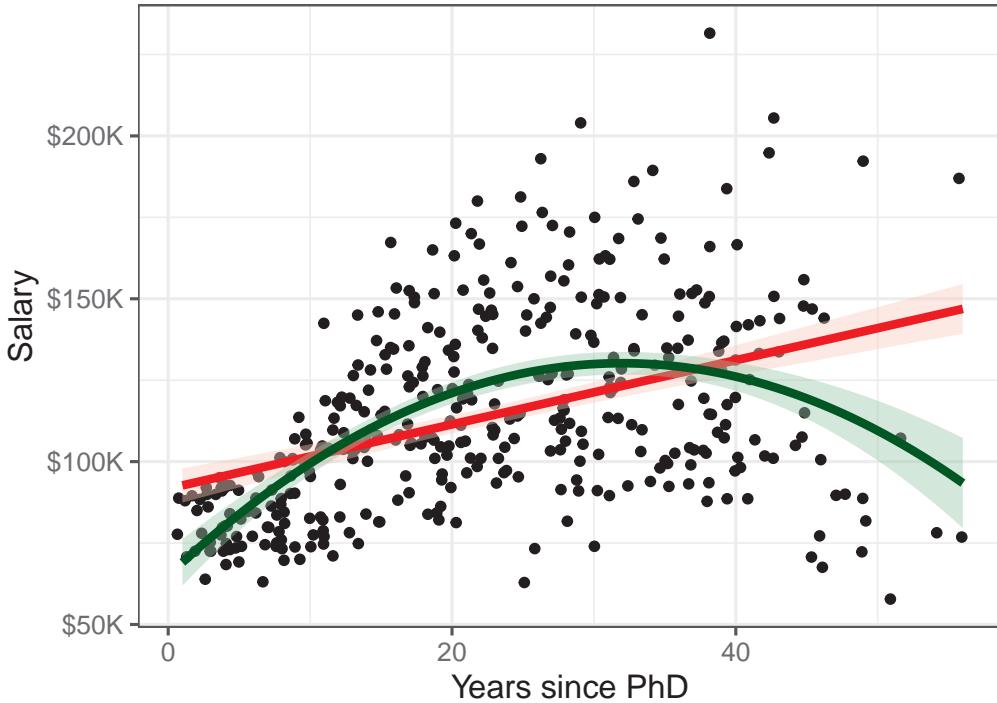


Figure 3.2: Scatterplot of Salary vs. years since PhD, showing linear and quadratic smooths with 95% confidence bands.

This serves to highlight some of our impressions from the basic scatterplot shown in Figure 3.1, making them more apparent. And that's precisely the point: the regression smoother draws attention to a possible pattern that we can consider as a visual summary of the data. You can think of this as showing what a linear (or quadratic) regression “sees” in the data. Statistical tests can help you decide if there is more evidence for a quadratic fit compared to the simpler linear relation.

It is useful to also show some indication of *uncertainty* (or inversely, *precision*) associated with the predicted values. Both the linear and quadratic trends are shown in Figure 3.2 with 95% pointwise confidence bands.¹ These are necessarily narrower in the center of the range of x where there is typically more data; they get wider toward the highest values of experience where the data are thinner.

Non-parametric smoothers

The most generally useful idea is a smoother that tracks an average value, $\mathbb{E}(y|x)$, of y as x varies across its range *without* assuming any particular functional form, and so avoiding the necessity to choose among $y \sim \text{poly}(x, 1)$, or $y \sim \text{poly}(x, 2)$, or $y \sim \text{poly}(x, 3)$, etc.

¹Confidence bands allow us to visualize the uncertainty around a fitted regression curve, which can be of two types: *pointwise intervals* or *simultaneous intervals*. The default setting in ‘`ggplot2::geom_smooth()`’ calculates pointwise intervals (using `stats::predict.lm(..., interval="confidence")`) at a confidence level $1 - \alpha$ for the predicted response at *each value* x_i of a predictor, and have the frequentist interpretation that over repeated sampling only 100α of the predictions at x_i will be outside that interval. In contrast, simultaneous intervals are calculated so that $1 - \alpha$ is the probability that *all of them* cover their corresponding true values simultaneously. These are necessarily wider than pointwise intervals. Commonly used methods for constructing simultaneous confidence bands in regression are the Bonferroni and Scheffé methods, which control the family-wise error rate over all values of x_i . See for precise definitions of these terms. These are different from a *prediction band*, which is used to represent the uncertainty about the value of a **new** data-point on the curve, but subject to the additional variance reflected in one observation.

Non-parametric smoothers attempt to estimate $\mathbb{E}(y|x) = f(x)$ where $f(x)$ is some smooth function. These typically use a collection of weighted *local regressions* for each x_i within a window centered at that value. In the method called *lowess* or *loess* (Cleveland, 1979; Cleveland & Devlin, 1988), a weight function is applied, giving greatest weight to x_i and a weight of 0 outside a window containing a certain fraction, s , called *span*, of the nearest neighbors of x_i . The fraction, s , is usually within the range $1/3 \leq s \leq 2/3$, and it determines the smoothness of the resulting curve; smaller values produce a wigglier curve and larger values giving a smoother fit (an optimal span can be determined by k -fold cross-validation to minimize a measure of overall error of approximation).

Non-parametric regression is a broad topic; see Fox (2016), Ch. 18 for a more general treatment including smoothing splines, and Wood (2006) for generalized additive models, fit using `method = "gam"` in `ggplot2`, which is the default when the largest group has more than 1,000 observations.

Figure 3.3 shows the addition of a loess smooth to the plot in Figure 3.2, suppressing the confidence band for the linear regression. The loess fit is nearly coincident with the quadratic fit, but has a slightly wider confidence band.

```
gg1 +
  geom_smooth(method = "loess", formula = "y ~ x",
              color = "blue", fill = scales::muted("blue"),
              linewidth = 2) +
  geom_smooth(method = "lm", formula = "y ~ x", se = FALSE,
              color = "red",
              linewidth = 2) +
  geom_smooth(method = "lm", formula = "y ~ poly(x,2)",
              color = "darkgreen", fill = "lightgreen",
              linewidth = 2)
```

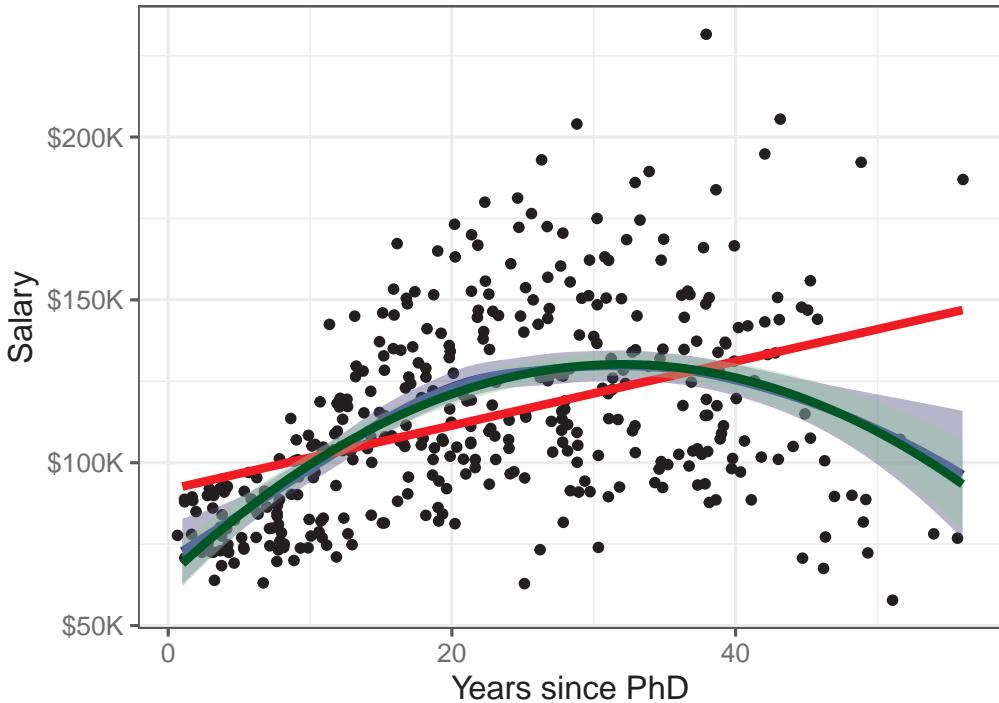


Figure 3.3: Scatterplot of Salary vs. years since PhD, adding the loess smooth. The loess smooth curve and confidence band in green is nearly indistinguishable from a quadratic fit in blue.

But now comes an important question: is it reasonable that academic salary should increase up to about 40 years since the PhD degree and then decline? The predicted salary for someone still working 50 years after earning their degree is about the same as a person at 15 years. What else is going on here?

3.1.2 Stratifiers

Very often, we have a main relationship of interest, but various groups in the data are identified by discrete factors (like faculty `rank` and `sex`, their type of `discipline` here), or there are quantitative predictors for which the main relation might vary. In the language of statistical models such effects are *interaction* terms, as in $y \sim \text{group} + x + \text{group}:x$, where the term `group:x` fits a different slope for each group and the grouping variable is often called a *moderator* variable. Common moderator variables are ethnicity, health status, social class and level of education. Moderators can also be continuous variables as in $y \sim x_1 + x_2 + x_1:x_2$.

I call these *stratifiers*, recognizing that we should consider breaking down the overall relation to see whether and how it changes over such “other” variables. Such variables are most often factors, but we can cut a continuous variable into ranges (*shingles*) and do the same graphically. There are two general stratifying graphical techniques:

- **Grouping:** Identify subgroups in the data by assigning different visual attributes, such as color, shape, line style, etc. within a single plot. This is quite natural for factors; quantitative predictors can be accommodated by cutting their range into ordered intervals. Grouping has the advantage that the levels of a grouping variable can be shown within the same plot, facilitating direct comparison.
- **Conditioning:** Showing subgroups in different plot panels. This has the advantages that relations for the individual groups more easily discerned and one can easily stratify by two (or more) other variables jointly, but visual comparison is more difficult because the eye must scan from one panel to another.

i History Corner

Recognition of the roles of visual grouping by factors within a panel and conditioning in multi-panel displays was an important advance in the development of modern statistical graphics. It began at A.T.&T. Bell Labs in Murray Hill, NJ in conjunction with the **S** language, the mother of R.

Conditioning displays (originally called *coplots* (Chambers & Hastie, 1991)) are simply a collection of 1D, 2D or 3D plots separate panels for subsets of the data broken down by one or more factors, or, for quantitative variables, subdivided into a factor with several overlapping intervals (*shingles*). The first implementation was in *Trellis* plots (Becker et al., 1996; Cleveland, 1985).

Trellis displays were extended in the **lattice** package (Sarkar, 2025), which offered:

- A **graphing syntax** similar to that used in statistical model formulas: $y \sim x | g$ conditions the data by the levels of `g`, with `|` read as “given”; two or more conditioning are specified as $y \sim x | g_1 + g_2 + \dots$, with `+` read as “and”.
- **Panel functions** define what is plotted in a given panel. `panel.xyplot()` is the default for scatterplots, plotting points, but you can add `panel.lmline()` for regression lines, `latticeExtra::panel.smoother()` for loess smooths and a wide variety of others.

The **car** package (Fox et al., 2023) supports this graphing syntax in many of its functions. **ggplot2** does not; it uses aesthetics (`aes()`), which map variables in the data to visual characteristics in displays.

The most obvious variable that affects academic salary is `rank`, because faculty typically get an increase in salary with a promotion that carries through in their future salary. What can we see if we group by `rank` and fit a separate smoothed curve for each?

In **ggplot2** thinking, grouping is accomplished simply by adding an aesthetic, such as `color = rank`. What happens then is that points, lines, smooths and other `geom_*`() inherit the feature that they are differentiated by color. In the case of `geom_smooth()`, we get a separate fit for each subset of the data, according to `rank`.

```
# make some re-useable pieces to avoid repetitions
scale_salary <- scale_y_continuous(
  labels = scales::dollar_format(prefix="$",
                                 scale = 0.001,
                                 suffix = "K"))

# position the legend inside the plot
legend_pos <- theme(legend.position = "inside",
                     legend.position.inside = c(.1, 0.95),
                     legend.justification = c(0, 1))

ggplot(Salaries,
       aes(x = yrs.since.phd, y = salary,
           color = rank, shape = rank)) +
  geom_point() +
  scale_salary +
  labs(x = "Years since PhD",
       y = "Salary") +
  geom_smooth(aes(fill = rank),
              method = "loess", formula = "y ~ x",
              linewidth = 2) +
  legend_pos
```

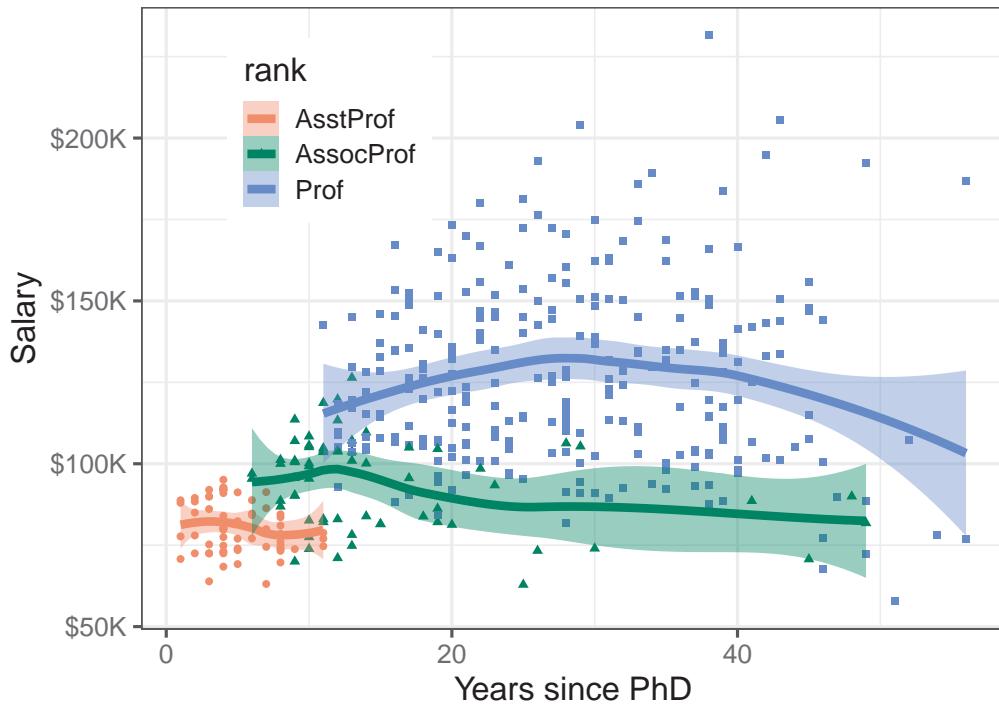


Figure 3.4: Scatterplot of Salary vs. years since PhD, grouped by rank.

Well, there is a different story here. Salaries generally occupy separate vertical levels, increasing with academic rank. The horizontal extents of the smoothed curves show their ranges. Within each rank there is some initial increase after promotion, and then some tendency to decline with increasing years. But, by and large, years since the PhD doesn't make as much difference once we've taken academic rank into account.

What about the `discipline` which is classified, perhaps peculiarly, as “theoretical” vs. “applied”? The values are just “A” and “B”, so I map these to more meaningful labels before making the plot.

```
Salaries <- Salaries |>
  mutate(discipline =
    factor(discipline,
           labels = c("A: Theoretical", "B: Applied")))

Salaries |>
  ggplot(aes(x = yrs.since.phd, y = salary, color = discipline)) +
  geom_point() +
  scale_salary +
  geom_smooth(aes(fill = discipline),
              method = "loess", formula = "y ~ x",
              linewidth = 2) +
  labs(x = "Years since PhD",
       y = "Salary") +
  legend_pos
```

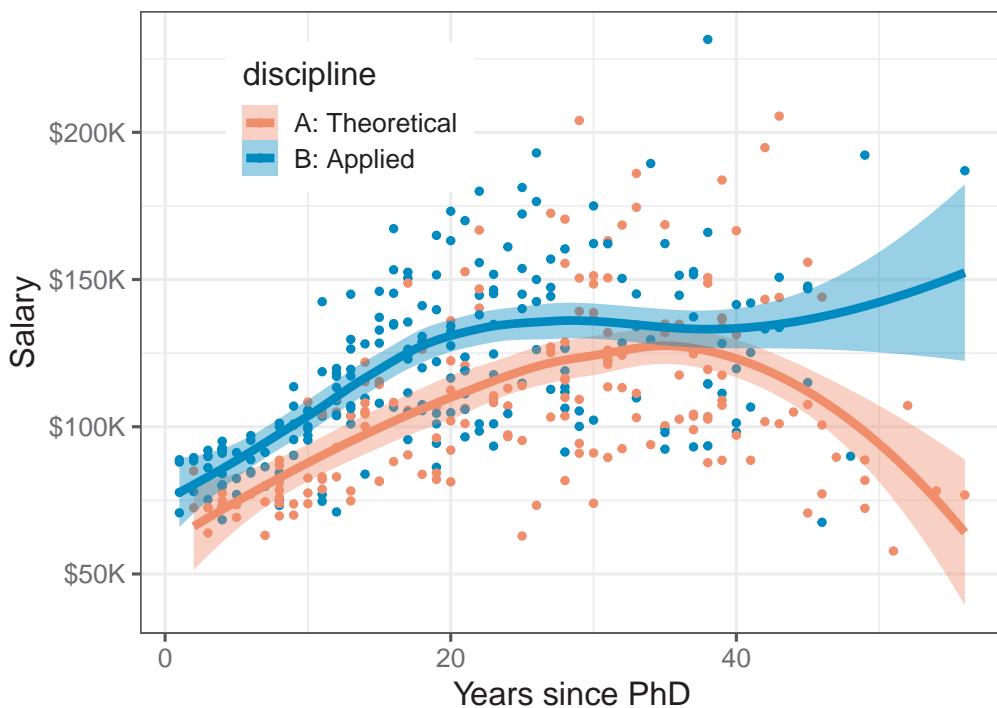


Figure 3.5: Scatterplot of Salary vs. years since PhD, grouped by discipline.

The story in Figure 3.5 is again different. Faculty in applied disciplines on average earn about 10,000\$ more per year on average than their theoretical colleagues.

```
Salaries |>
  group_by(discipline) |>
  summarize(mean = mean(salary))
#> # A tibble: 2 x 2
#>   discipline      mean
```

```
#> <fct>      <dbl>
#> 1 A: Theoretical 108548.
#> 2 B: Applied     118029.
```

For both groups, there is an approximately linear relation up to about 30–40 years, but the smoothed curves then diverge into the region where the data is thinner.

This result is more surprising than differences among faculty ranks. The effect of annotation with smoothed curves as visual summaries is apparent, and provides a stimulus to think about *why* these differences (if they are real) exist between theoretical and applied professors, and maybe *should* theoreticians be paid more!

3.1.3 Conditioning

The previous plots use grouping by color to plot the data for different subsets inside the same plot window, making comparison among groups easier, because they can be directly compared along a common vertical scale ². This gets messy, however, when there are more than just a few levels, or worse—when there are two (or more) variables for which we want to show separate effects. In such cases, we can plot separate panels using the `ggplot2` concept of *faceting*. There are two options: `facet_wrap()` takes one or more conditioning variables and produces a ribbon of plots for each combination of levels; `facet_grid(row ~ col)` takes two or more conditioning variables and arranges the plots in a 2D array identified by the `row` and `col` variables.

Let's look at salary broken down by the combinations of discipline and rank. Here, I chose to stratify using color by rank within each of panels faceting by discipline. Because there is more going on in this plot, a linear smooth is used to represent the trend.

```
Salaries |>
  ggplot(aes(x = yrs.since.phd, y = salary,
             color = rank, shape = rank)) +
  geom_point() +
  scale_salary +
  labs(x = "Years since PhD",
       y = "Salary") +
  geom_smooth(aes(fill = rank),
              method = "lm", formula = "y ~ x",
              linewidth = 2, alpha = 1/4) +
  facet_wrap(~ discipline) +
  legend_pos
```

Once both of these factors are taken into account, there does not seem to be much impact of years of service. Salaries in theoretical disciplines are noticeably greater than those in applied disciplines at all ranks, and there are even greater differences among ranks.

Finally, to shed light on the question that motivated this example— are there anomalous differences in salary for men and women— we can look at differences in salary according to sex, when discipline and rank are taken into account. To do this graphically, condition by both variables, but use `facet_grid(discipline ~ rank)` to arrange their combinations in a grid whose rows are the levels of `discipline` and columns are those of `rank`. I want to make the comparison of males and females most direct, so I use `color = sex` to stratify the panels. The smoothed regression lines and error bands are calculated separately for each combination of discipline, rank and sex.

²The classic study by Cleveland & McGill (1984);Cleveland & McGill (1985) shows that judgements of magnitude along a common scale are more accurate than those along separate, aligned scales.

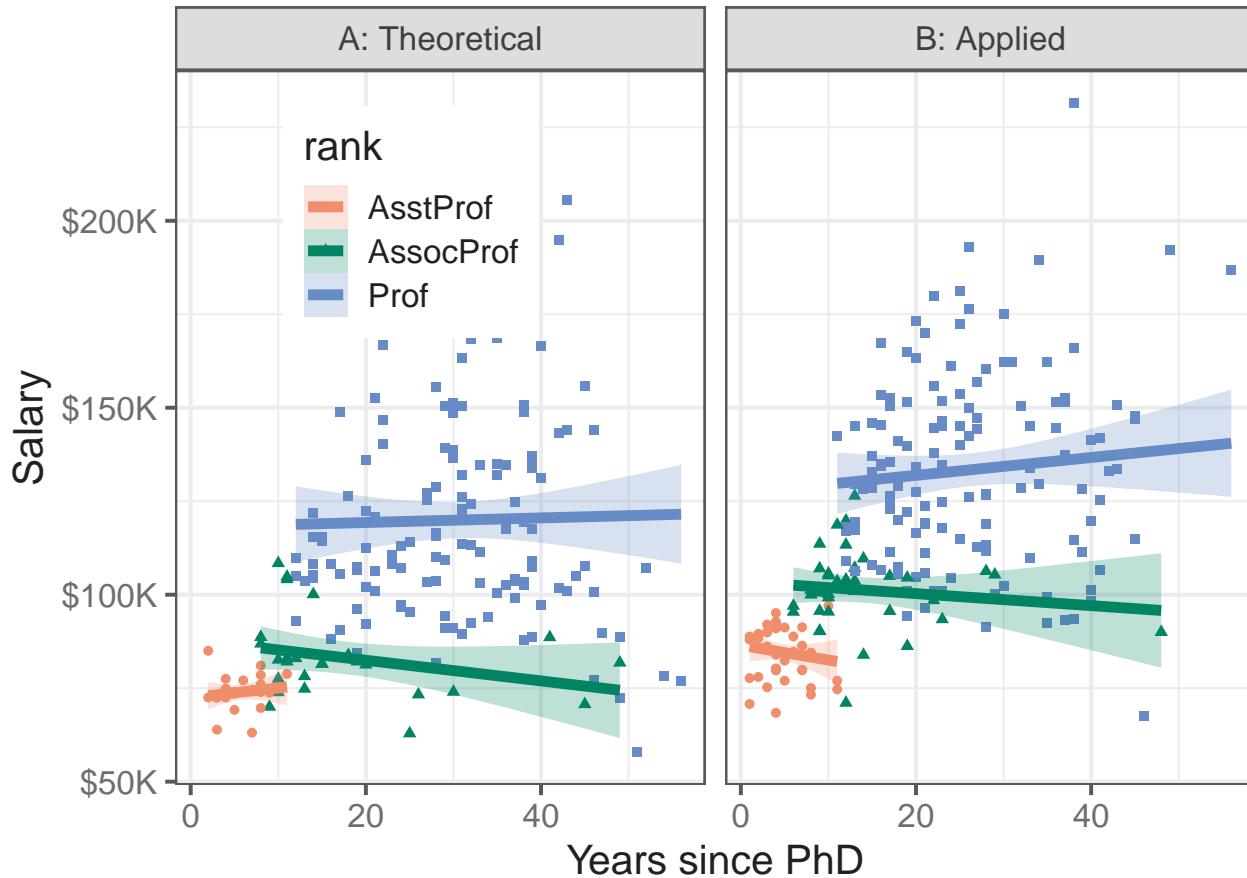


Figure 3.6: Scatterplot of Salary vs. years since PhD, grouped by rank, with separate panels for discipline.

```
Salaries |>
  ggplot(aes(x = yrs.since.phd, y = salary, color = sex)) +
  geom_point() +
  scale_salary +
  labs(x = "Years since PhD",
       y = "Salary") +
  geom_smooth(aes(fill = sex),
              method = "lm", formula = "y ~ x",
              linewidth = 2, alpha = 1/4) +
  facet_grid(discipline ~ rank) +
  theme_bw(base_size = 14) +
  legend_pos
```

3.2 Data Ellipses

The *data ellipse* (Monette, 1990), or *concentration ellipse* (Dempster, 1969) is a remarkably simple and effective display for viewing and understanding bivariate relationships in multivariate data. The data ellipse

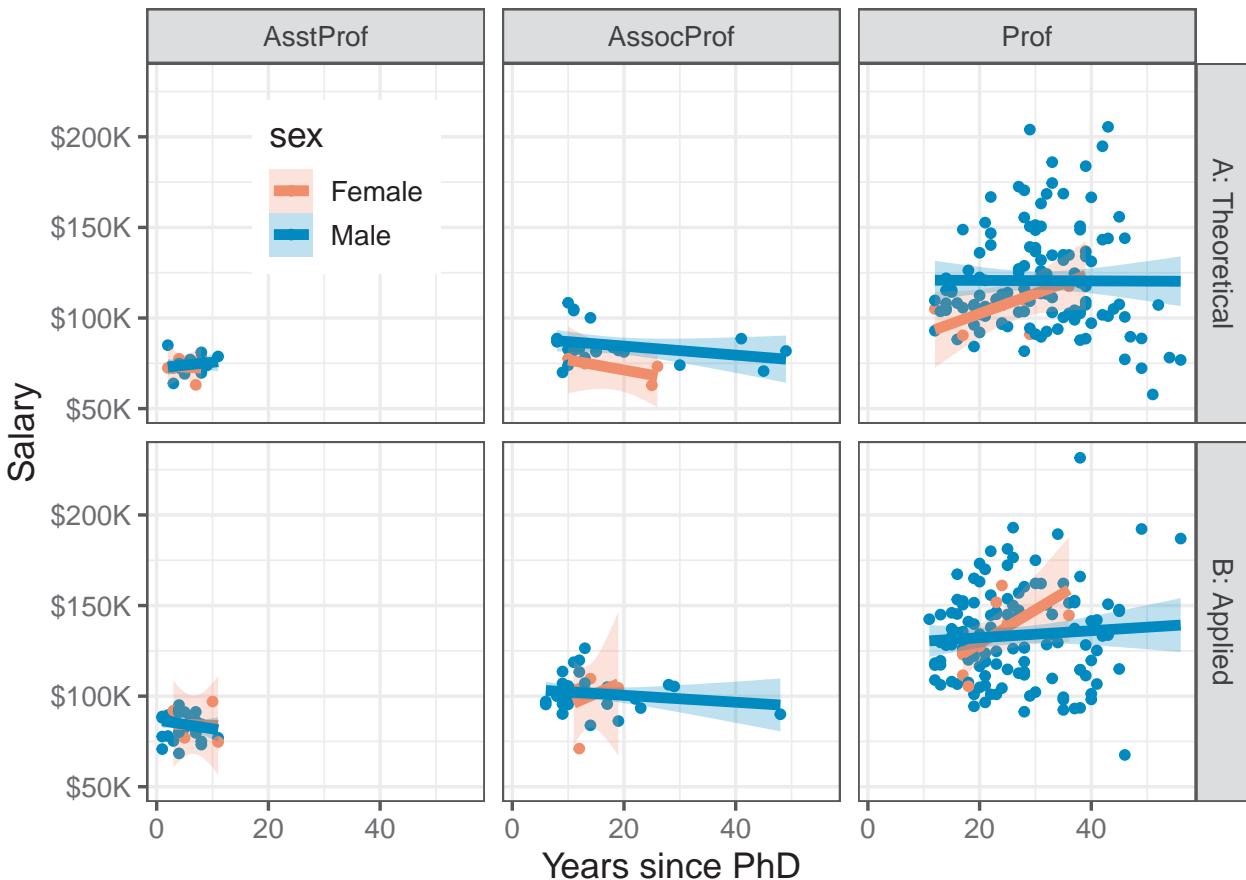


Figure 3.7: Scatterplot of Salary vs. years since PhD, grouped by sex, faceted by discipline and rank.

is typically used to add a visual summary to a scatterplot, that shows all together the means, standard deviations, correlation, and slope of the regression line for two variables, perhaps stratified by another variable.

Under the classical assumption that the data are bivariate normally distributed, the data ellipse is also a **sufficient** visual summary, in the sense that it captures **all** relevant features of the data. See Friendly et al. (2013) for a complete discussion of the role of ellipsoids in statistical data visualization.

The data ellipse is based on the idea that in a bivariate normal distribution, the contours of equal probability form a series of concentric ellipses. If the variables were uncorrelated and had the same variances, these would be circles, and Euclidean distance would measure the distance of each observation from the mean. When the variables are correlated, a different measure, *Mahalanobis distance* is the proper measure of how far a point is from the mean, taking the correlation into account.

To illustrate, Figure 3.8 shows a scatterplot with labels for two points, “A” and “B”. Which is further from the mean, “X”? A contour of constant Euclidean distance, shown by the red dashed circle, ignores the apparent negative correlation, so point “A” is further. The blue ellipse for Mahalanobis distance takes the correlation into account, so point “B” has a greater distance from the mean.

Mathematically, Euclidean (squared) distance for p variables, $j = 1, 2, \dots, p$, is just a generalization of the square of a univariate standardized (z) score, $z^2 = [(y - \bar{y})/s]^2$,

$$D_E^2(\mathbf{y}) = \sum_j^p z_j^2 = \mathbf{z}^\top \mathbf{z} = (\mathbf{y} - \bar{\mathbf{y}})^\top \text{diag}(\mathbf{S})^{-1}(\mathbf{y} - \bar{\mathbf{y}}),$$

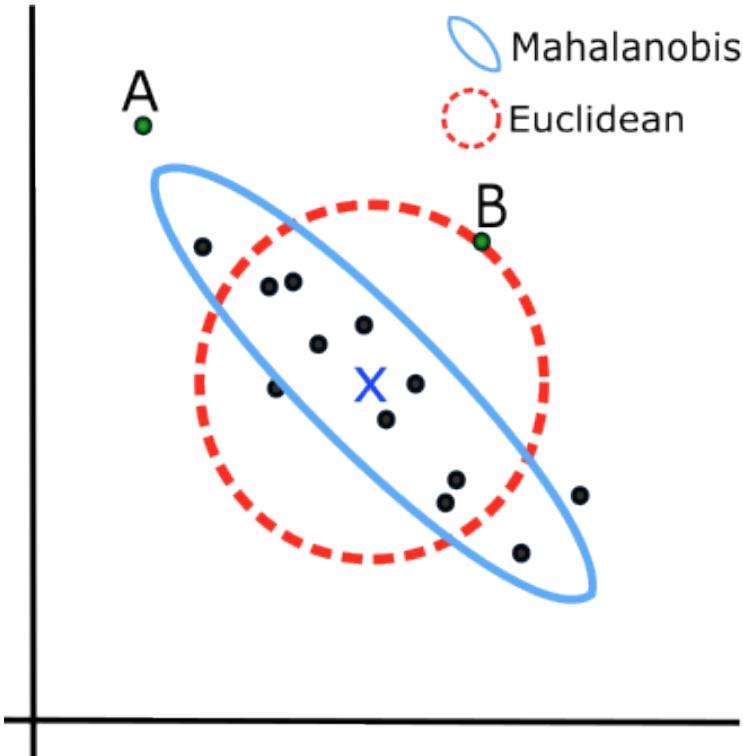


Figure 3.8: 2D data with curves of constant distance from the centroid. The blue solid ellipse shows a contour of constant Mahalanobis distance, taking the correlation into account; the dashed red circle is a contour of equal Euclidean distance. Given the data points, Which of the points **A** and **B** is further from the mean ($\bar{\mathbf{y}}$)? *Source:* Re-drawn from Ou Zhang

where \mathbf{S} is the sample variance-covariance matrix, $\mathbf{S} = (n - 1)^{-1} \sum_{i=1}^n (\mathbf{y}_i - \bar{\mathbf{y}})^T (\mathbf{y}_i - \bar{\mathbf{y}})$.

Mahalanobis distance takes the correlations into account simply by using the covariances as well as the variances,

$$D_M^2(\mathbf{y}) = (\mathbf{y} - \bar{\mathbf{y}})^T S^{-1} (\mathbf{y} - \bar{\mathbf{y}}) . \quad (3.1)$$

In Equation 3.1, the inverse S^{-1} serves to “divide” the matrix $(\mathbf{y} - \bar{\mathbf{y}})^T (\mathbf{y} - \bar{\mathbf{y}})$ of squared distances by the variances (and covariances) of the variables, as in the univariate case.

For p variables, the data *ellipsoid* \mathcal{E}_c of size c is a p -dimensional ellipse, defined as the set of points $\mathbf{y} = (y_1, y_2, \dots, y_p)$ whose squared Mahalanobis distance, $D_M^2(\mathbf{y})$ is less than or equal to c^2 ,

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{S}) := \{D_M^2(\mathbf{y}) \leq c^2\} .$$

A computational definition recognizes that the boundary of the ellipsoid can be found by transforming a unit sphere \mathcal{P} centered at the origin, $\mathcal{P} := \{\mathbf{x}^T \mathbf{x} = 1\}$, by $\mathbf{S}^{1/2}$ and then shifting that to centroid of the data,

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{S}) = \bar{\mathbf{y}} \oplus \mathbf{S}^{1/2} \mathcal{P} ,$$

where $\mathbf{S}^{1/2}$ represents a rotation and scaling and the notation \oplus represents translation to a new centroid, $\bar{\mathbf{y}}$ here. The matrix $\mathbf{S}^{1/2}$ is commonly computed as the Cholesky factor of \mathbf{S} . Slightly abusing notation and taking the unit sphere \mathcal{P} as given (like an identity matrix \mathbf{I}), we can write the data ellipsoid as simply:

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{S}) = \bar{\mathbf{y}} \oplus c\sqrt{\mathbf{S}} . \quad (3.2)$$

When \mathbf{y} is (at least approximately) bivariate normal, $D_M^2(\mathbf{y})$ has a large-sample χ^2 distribution (χ^2 with 2 df), so

- $c^2 = \chi_2^2(0.5) = 1.39$ gives a data ellipse covering 50% of the data points, a bivariate analog of the central box of a boxplot.
- $c^2 = \chi_2^2(0.68) = 2.28$ gives a “1 standard deviation bivariate ellipse,” an analog of the standard interval $\bar{y} \pm 1s$,
- $c^2 = \chi_2^2(0.95) = 5.99 \approx 6$ gives a data ellipse of 95% coverage.

In not-large samples, the radius c of the ellipsoid is better approximated by a multiple of a $F_{p,n-p}$ distribution, becoming $c = \sqrt{2F_{2,n-2}^{1-\alpha}}$ in the bivariate case ($p = 2$) for coverage $1 - \alpha$.

These are illustrated in Figure 3.9 with ellipses of 50%, 68% and 95% coverage for a the matrix \mathbf{S} defined below and $\bar{\mathbf{y}} = \mathbf{0}$. Here, the variance of \mathbf{y}_1 is twice that of \mathbf{y}_2 and the correlation works out to $r(y_1, y_2) = 0.35$.

```
ybar <- c(0, 0)
S <- matrix(c(1, .5, .5, 2), 2, 2)
rownames(S) <- colnames(S) <- c("y1", "y2")
S
#>      y1   y2
#> y1 1.0 0.5
#> y2 0.5 2.0
```

Statistical ellipses are conveniently drawn using `car::ellipse()`. `heplots::ellipse.label()` provides flexible ways to add labels to ellipses at various locations around the ellipse shown in Figure 3.9. These are called repeatedly to overlay the three ellipses.

```
levels <- c(0.50, 0.68, 0.95)
c <- qchisq(levels, df = 2) |> round(2) |> print()
#> [1] 1.39 2.28 5.99

# labels for ellipses, using plotmath
lab1 <- bquote(paste("c =", chi[2]^2, "(", .(levels[1]), ") =", .(c[1])))
lab2 <- bquote(paste("c =", chi[2]^2, "(", .(levels[2]), ") =", .(c[2])))
lab3 <- bquote(paste("c =", chi[2]^2, "(", .(levels[3]), ") =", .(c[3])))

e1 <- ellipse(ybar, S, radius=qchisq(levels[1], 2),
               col = "blue", fill=TRUE, fill.alpha = 0.5,
               add=FALSE,
               xlim=c(-8, 8), ylim=c(-9.5, 9.5),
               asp=1, grid = FALSE,
               xlab = expression(y[1]),
               ylab = expression(y[2]),
               cex.lab = 1.5)
label.ellipse(e1, label = lab1, label.pos = "S",
              cex = 1.2)

e2 <- ellipse(ybar, S, radius=qchisq(levels[2], 2),
               col="blue", fill=TRUE, fill.alpha=0.3)
label.ellipse(e2, label = lab2, label.pos = "N",
              cex = 1.2)
```

```
e3 <- ellipse(ybar, S, radius=qchisq(levels[3], 2),
               col="blue", fill=TRUE, fill.alpha=0.1)
label.ellipse(e3, label = lab3, label.pos = "N",
              cex = 1.2)
```

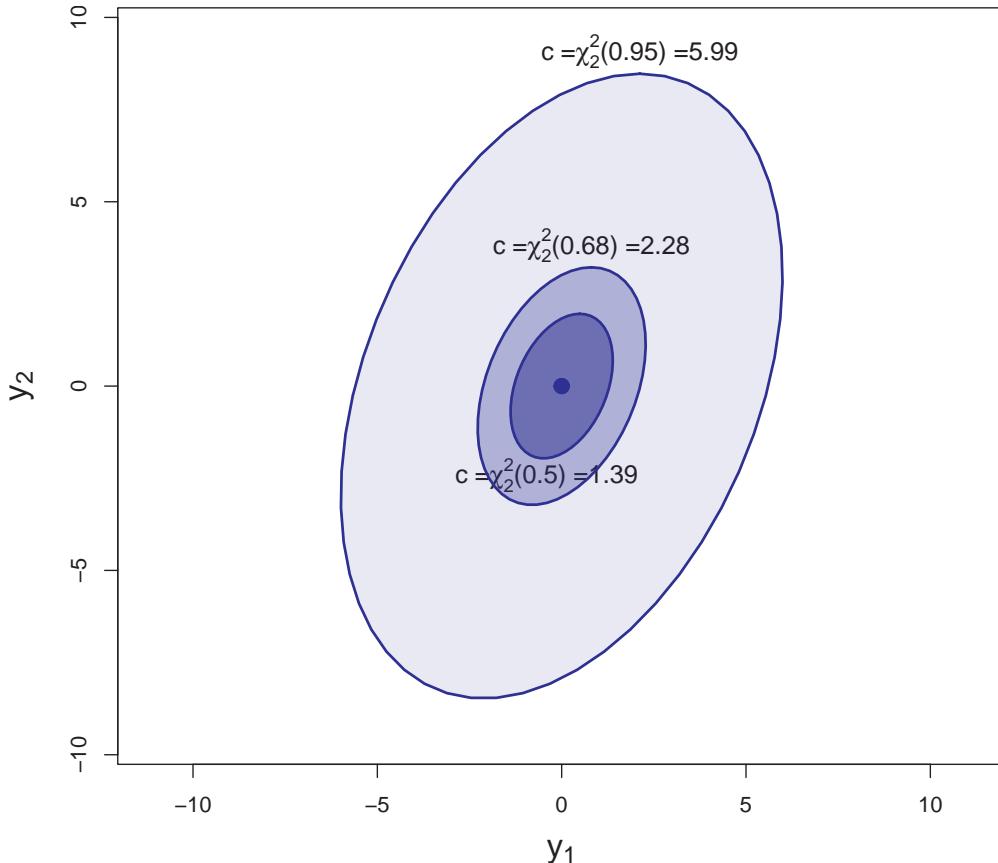


Figure 3.9: Data ellipses of 50%, 68% and 95% coverage when the means are $\bar{\mathbf{y}} = \mathbf{0}$ and the variance-covariance matrix is \mathbf{S} .

As always, graphic details matter. Figure 3.9 uses `asp = 1` so that units in the plot are the same for y_1 and y_2 , and we can see the greater variance for y_2 as well as the correlation. Facilities of `grDevices::plotmath()` are used to provide mathematical annotations in the plot.

3.2.1 Ellipse properties

The essential ideas of correlation and regression and their relation to ellipses go back to Galton (1886). Galton's goal was to predict (or explain) how a heritable trait, Y , (e.g., height) of children was related to that of their parents, X . He made a semi-graphic table of the frequencies of 928 observations of the average height of father and mother versus the height of their child, shown in Figure 3.10. (Today, we would put child height on the y axis, but Galton was working from the table, so he organized it with parent height as the rows.) He then drew smoothed contour lines of equal frequencies and had the wonderful visual insight that these formed concentric shapes that were tolerably close to ellipses.

He then calculated summaries, $\text{Ave}(Y|X)$, and, for symmetry, $\text{Ave}(X|Y)$, and plotted these as lines of means on his diagram. Lo and behold, he had a second visual insight: the lines of means of $(Y|X)$ and $(X|Y)$ corresponded approximately to the loci of horizontal and vertical tangents to the concentric ellipses. To

complete the picture, he added lines showing the geometric major and minor axes of the family of ellipses (which turned out to be the principal components) with the result shown in Figure 3.10.

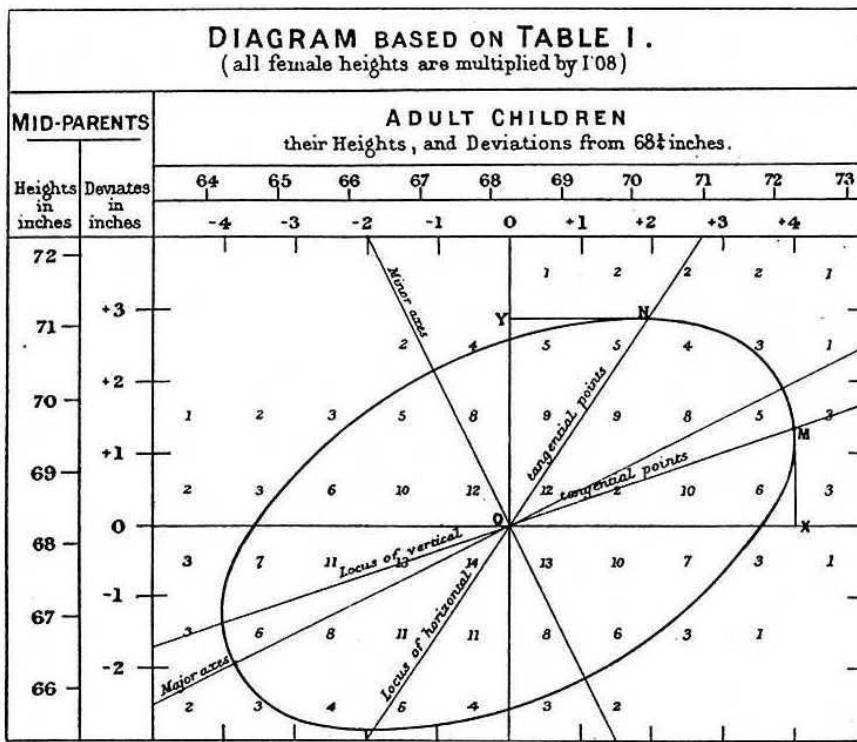


Figure 3.10: Galton's 1886 diagram, showing the relationship of height of children to the average of their parents' height. The diagram is essentially an overlay of a geometrical interpretation on a bivariate grouped frequency distribution, shown as numbers.

For two variables, x and y , the remarkable properties of the data ellipse are illustrated in Figure 3.11, a modern reconstruction of Galton's diagram.

```

data(Galton, package = "HistData")
sunflowerplot(parent ~ child, data=Galton,
  xlim=c(61,75),
  ylim=c(61,75),
  seg.col="black",
  xlab="Child height",
  ylab="Mid Parent height")

y.x <- lm(parent ~ child, data=Galton)      # regression of y on x
abline(y.x, lwd=2)
x.y <- lm(child ~ parent, data=Galton)      # regression of x on y
cc <- coef(x.y)
abline(-cc[1]/cc[2], 1/cc[2], lwd=2, col="gray")

with(Galton,
  car::dataEllipse(child, parent,
    plot.points=FALSE,
    levels=c(0.40, 0.68, 0.95),
    )
  )

```

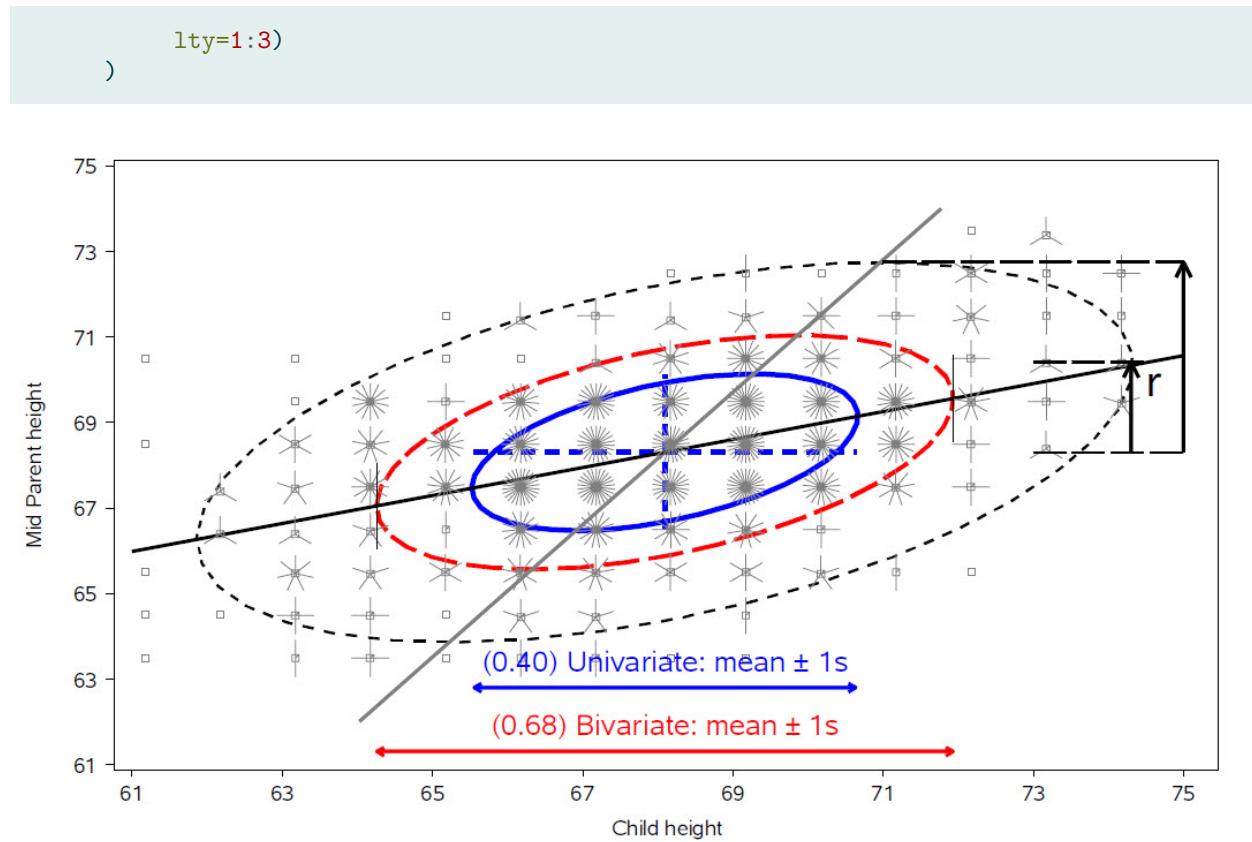


Figure 3.11: Sunflower plot of Galton’s data on heights of parents and their children (in.), with 40%, 68% and 95% data ellipses and the regression lines of y on x (black) and x on y (grey).

- The ellipses have the mean vector (\bar{x}, \bar{y}) as their center.
- The lengths of arms of the blue dashed central cross show the standard deviations of the variables, which correspond to the shadows of the ellipse covering 40% of the data. These are the bivariate analogs of the standard intervals $\bar{x} \pm 1s_x$ and $\bar{y} \pm 1s_y$.
- More generally, shadows (projections) on the coordinate axes, or any linear combination of them, give any standard interval, $\bar{x} \pm ks_x$ and $\bar{y} \pm ks_y$. Those with $k = 1, 1.5, 2.45$, have bivariate coverage 40%, 68% and 95% respectively, corresponding to these quantiles of the χ^2 distribution with 2 degrees of freedom, i.e., $\chi^2_{(0.40)} \approx 1^2$, $\chi^2_{(0.68)} \approx 1.5^2$, and $\chi^2_{(0.95)} \approx 2.45$. The shadows of the 68% ellipse are the bivariate analog of a univariate $\bar{x} \pm 1s_x$ interval.
- The regression line predicting y from x goes through the points where the ellipses have vertical tangents. The other regression line, predicting x from y goes through the points of horizontal tangency.
- The correlation $r(x, y)$ is the ratio of the vertical segment from the mean of y to the regression line to the vertical segment going to the top of the ellipse as shown at the right of the figure. It is $r = 0.46$ in this example.
- The residual standard deviation, $s_e = \sqrt{MSE} = \sqrt{\sum(y - \bar{y})^2/n - 2}$, is the half-length of the ellipse at the mean \bar{x} .

Because Galton’s values of `parent` and `child` height were recorded in class intervals of 1 in., they are shown as sunflower symbols in Figure 3.11, with multiple ‘petals’ reflecting the number of observations at each location. This plot (except for annotations) is constructed using `sunflowerplot()` and `car::dataEllipse()` for the ellipses.

```

data(Galton, package = "HistData")

sunflowerplot(parent ~ child, data=Galton,
  xlim=c(61,75),
  ylim=c(61,75),
  seg.col="black",
  xlab="Child height",
  ylab="Mid Parent height")

y.x <- lm(parent ~ child, data=Galton)      # regression of y on x
abline(y.x, lwd=2)
x.y <- lm(child ~ parent, data=Galton)      # regression of x on y
cc <- coef(x.y)
abline(-cc[1]/cc[2], 1/cc[2], lwd=2, col="gray")

with(Galton,
  car::dataEllipse(child, parent,
    plot.points=FALSE,
    levels=c(0.40, 0.68, 0.95),
    lty=1:3)
)

```

Finally, as Galton noted in his diagram, the principal major and minor axes of the ellipse have important statistical properties. Pearson (1901) would later show that their directions are determined by the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots$ of the covariance matrix \mathbf{S} and their radii by the square roots, $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots$ of the corresponding eigenvalues.

3.2.2 R functions for data ellipses

A number of packages provide functions for drawing data ellipses in a scatterplot, with various features.

- `car::scatterplot()`: uses base R graphics to draw 2D scatterplots, with a wide variety of plot enhancements including linear and non-parametric smoothers (loess, gam), a formula method, e.g., `y ~ x | group`, and marking points and lines using symbol shape, color, etc. Importantly, the `car` package generally allows automatic identification of “noteworthy” points by their labels in the plot using a variety of methods. For example, `method = "mahal"` labels cases with the most extreme Mahalanobis distances; `method = "r"` selects points according to their value of `abs(y)`, which is appropriate in residual plots.
- `car::dataEllipse()`: plots classical or robust data ellipses for one or more groups, with the same facilities for point identification. The robust version (`robust=TRUE`) uses the multivariate t distribution (using `MASS::cov.trob()`) rather than the Gaussian,
- `heplots::covEllipses()`: draws classical or robust data ellipses for one or more groups in a one-way design and optionally for the pooled total sample, where the focus is on homogeneity of within-group covariance matrices.
- `ggplot2::stat_ellipse()`: uses the calculation methods of `car::dataEllipse()` to add unfilled (`geom = "path"`) or filled (`geom = polygon`) data ellipses in a `ggplot` scatterplot, using inherited aesthetics.

Example 3.2. Canadian occupational prestige

These examples use the data on the prestige of 102 occupational categories and other measures from the 1971 Canadian Census, recorded in `Prestige`.³ Our interest is in understanding how `prestige` (the Pineo & Porter (2008) prestige score for an occupational category, derived from a social survey) is related to census

³The dataset was collected by Bernard Blishen, William Carroll and Catherine Moore, but apparently unpublished. A version updated to the 1981 census is described in Blishen et al. (1987).

measures of the average education, income, percent women of incumbents in those occupations. Occupation type is a factor with levels "bc" (blue collar), "wc" (white collar) and "prof" (professional).

```
data(Prestige, package="carData")
# `type` is really an ordered factor. Make it so.
Prestige$type <- ordered(Prestige$type,
                           levels=c("bc", "wc", "prof"))
str(Prestige)
#> 'data.frame': 102 obs. of 6 variables:
#>   $ education: num 13.1 12.3 12.8 11.4 14.6 ...
#>   $ income    : int 12351 25879 9271 8865 8403 11030 8258 14163 11377 11023 ...
#>   $ women     : num 11.16 4.02 15.7 9.11 11.68 ...
#>   $ prestige   : num 68.8 69.1 63.4 56.8 73.5 77.6 72.6 78.1 73.1 68.8 ...
#>   $ census    : int 1113 1130 1171 1175 2111 2113 2133 2141 2143 2153 ...
#>   $ type      : Ord.factor w/ 3 levels "bc" <"wc" <"prof": 3 3 3 3 3 3 3 3 3 ...
```

I first illustrate the relation between `income` and `prestige` in Figure 3.12 using `car::scatterplot()` with many of its bells and whistles, including marginal boxplots for the variables, the linear regression line, loess smooth and the 68% data ellipse.

```
scatterplot(prestige ~ income, data=Prestige,
            pch = 16, cex.lab = 1.25,
            regLine = list(col = "red", lwd=3),
            smooth = list(smoother=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "darkgreen",
                          col.var = "darkgreen"),
            ellipse = list(levels = 0.68),
            id = list(n=4, method = "mahal", col="black", cex=1.2))
#> general.managers      lawyers      ministers      physicians
#>                 2             17            20            24
```

There is a lot that can be seen here:

- `income` is positively skewed, as is often the case.
- The loess smooth, on the scale of income, shows `prestige` increasing up to \$15,000 (these are 1971 incomes), and then leveling off.
- The data ellipse, centered at the means encloses approximately 68% of the data points. It adds visual information about the correlation and precision of the linear regression; but here, the non-linear trend for higher incomes strongly suggests a different approach.
- The four points identified by their labels are those with the largest Mahalanobis distances. `scatterplot()` prints their labels to the console.

Figure 3.13 shows a similar plot for education, which from the boxplot appears to be reasonably symmetric. The smoothed curve is quite close to the linear regression, according to which `prestige ~ education`, `data=Prestige`)`["education"]` = 5.361 with each year of education.

```
scatterplot(prestige ~ education, data=Prestige,
            pch = 16, cex.lab = 1.25,
            regLine = list(col = "red", lwd=3),
            smooth = list(smoother=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "darkgreen",
```

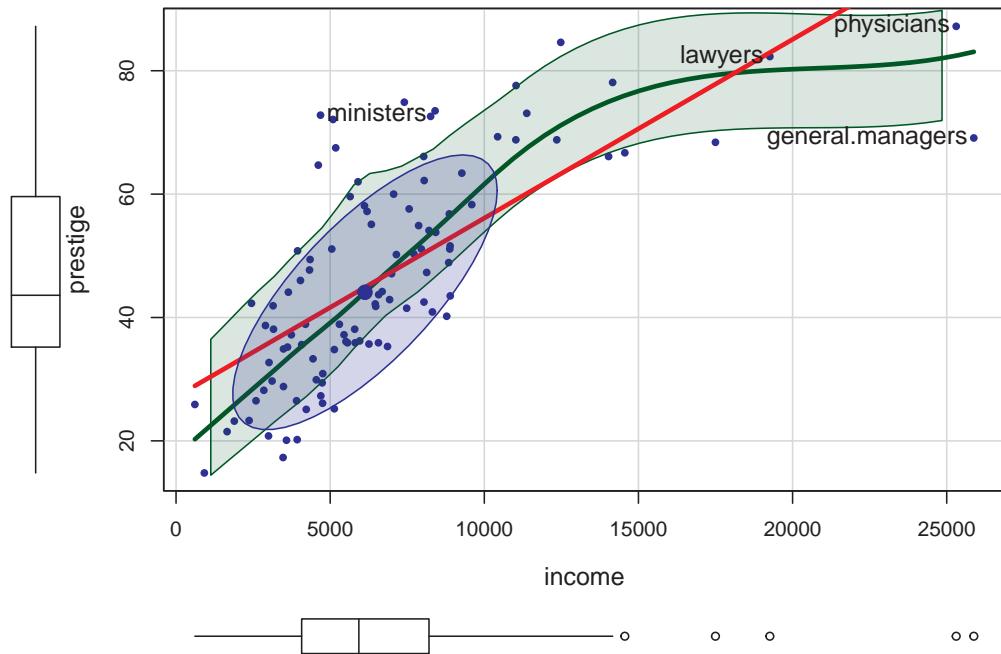


Figure 3.12: Scatterplot of prestige vs. income, showing the linear regression line (red), the loess smooth with a confidence envelope (darkgreen) and a 68% data ellipse. Points with the 4 largest D^2 values are labeled.

```

col.var = "darkgreen",
ellipse = list(levels = 0.68),
id = list(n=4, method = "mahal", col="black", cex=1.2))
#> physicians file.clerks    newsboys    farmers
#>        24          41          53          67

```

In this plot, farmers, newsboys, file.clerks and physicians are identified as noteworthy, for being furthest from the mean by Mahalanobis distance. In relation to their typical level of education, these are mostly understandable, but it is nice that farmers are rated of higher prestige than their level of education would predict.

Note that the `method` argument for point identification can take a vector of case numbers indicating the points to be labeled. So, to label the observations with large absolute standardized residuals in the linear model `m`, you can use `method = which(abs(rstandard(m)) > 2)`.

```

m <- lm(prestige ~ education, data=Prestige)
scatterplot(prestige ~ education, data=Prestige,
            pch = 16, cex.lab = 1.25,
            boxplots = FALSE,
            regLine = list(col = "red", lwd=3),
            smooth = list(smoother=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "black",
                          col.var = "darkgreen"),
            ellipse = list(levels = 0.68),
            id = list(n=4, method = which(abs(rstandard(m))>2),
                      col="black", cex=1.2)) |> invisible()

```

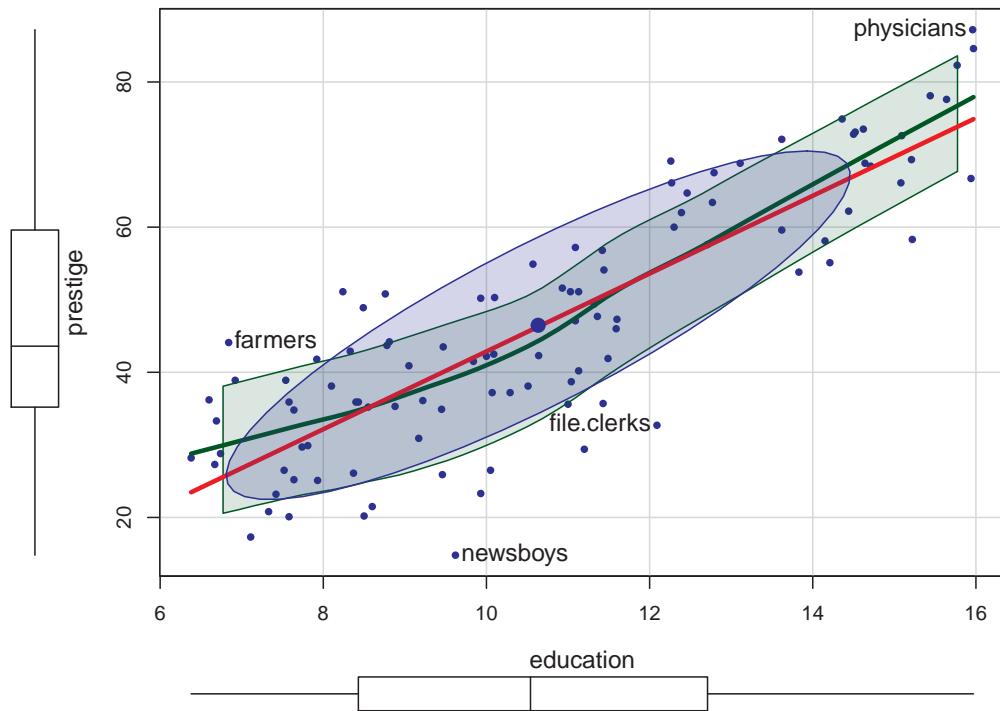


Figure 3.13: Scatterplot of prestige vs. education, showing the linear regression line (red), the loess smooth with a confidence envelope (darkgreen) and a 68% data ellipse.

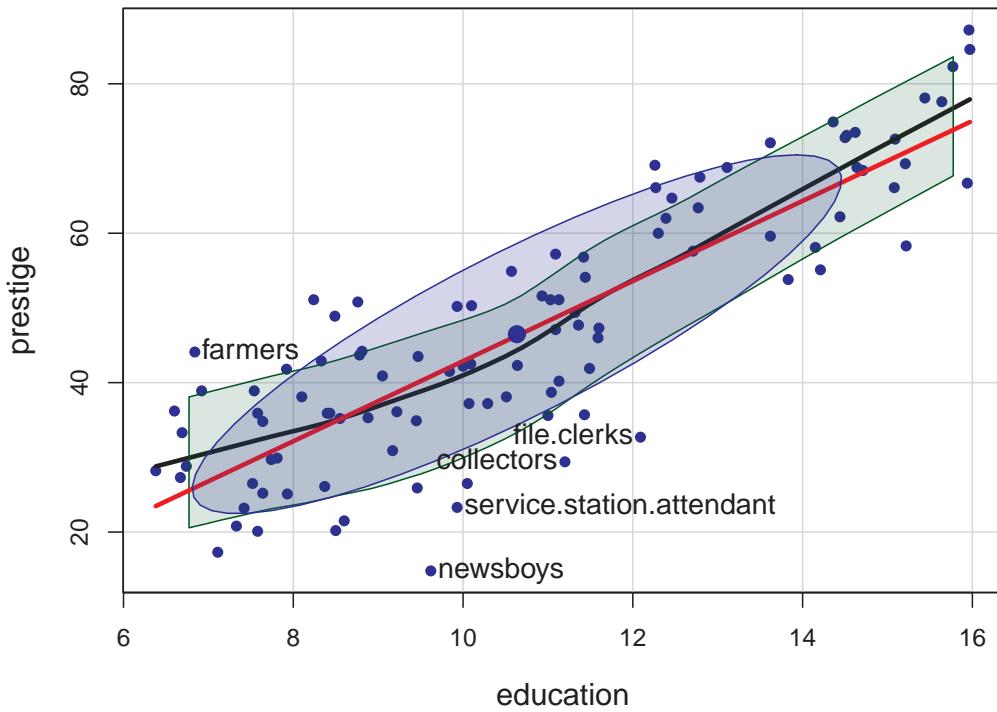


Figure 3.14: Scatterplot of prestige vs. education, labeling points whose absolute standardized residual is > 2 .

3.2.2.1 Plotting on a log scale

A typical remedy for the non-linear relationship of income to prestige is to plot income on a log scale. This usually makes sense, and expresses a belief that a **multiple** of or **percentage increase** in income has a constant impact on prestige, as opposed to the **additive** interpretation for income itself.

For example, the slope of the linear regression line in Figure 3.12 is given by `coef(lm(prestige ~ income, data=Prestige))["income"] = 0.003`. Multiplying this by 1000 says that a \$1000 increase in `income` is associated with an average increase of `prestige` of 2.9.

In the plot below, `scatterplot(..., log = "x")` re-scales the x-axis to the $\log_e()$ scale. The slope, `coef(lm(prestige ~ log(income), data=Prestige))["log(income)"] = 21.556` says that a 1% increase in salary is associated with an average change of 21.55 / 100 in prestige.

```
scatterplot(prestige ~ income, data=Prestige,
            log = "x",
            pch = 16, cex.lab = 1.25,
            regLine = list(col = "red", lwd=3),
            smooth = list(smooth=loessLine,
                          lty.smooth = 1, lwd.smooth=3,
                          col.smooth = "darkgreen", col.var = "darkgreen"),
            ellipse = list(levels = 0.68),
            id = list(n=4, method = "mahal", col="black", cex=1.2))
#> general.managers      ministers      newsboys      babysitters
#>                      2             20            53            63
```

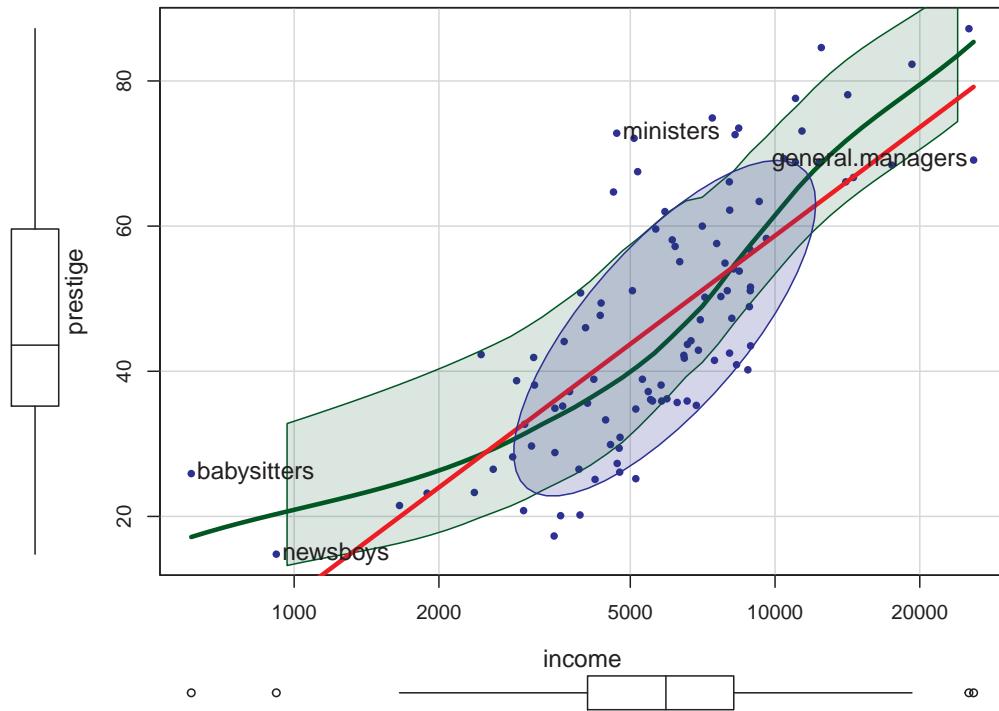


Figure 3.15: Scatterplot of prestige vs. $\log(\text{income})$.

The smoothed curve in Figure 3.15 exhibits a slight tendency to bend upwards, but a linear relation is a reasonable approximation.

3.2.2.2 Stratifying

Before going further, it is instructive to ask what we could see in the relationship between income and prestige if we stratified by type of occupation, fitting separate regressions and smooths for blue collar, white collar and professional incumbents in these occupations.

The formula `prestige ~ income | type` (read: income *given* type) is a natural way to specify grouping by `type`; separate linear regressions and smooths are calculated for each group, applying the color and point shapes specified by the `col` and `pch` arguments.

```
scatterplot(prestige ~ income | type, data=Prestige,
            col = c("blue", "red", "darkgreen"),
            pch = 15:17,
            grid = FALSE,
            legend = list(coords="bottomright"),
            regLine = list(lwd=3),
            smooth=list(smoothers=loessLine,
                        var=FALSE, lwd.smooth=2, lty.smooth=1))
```

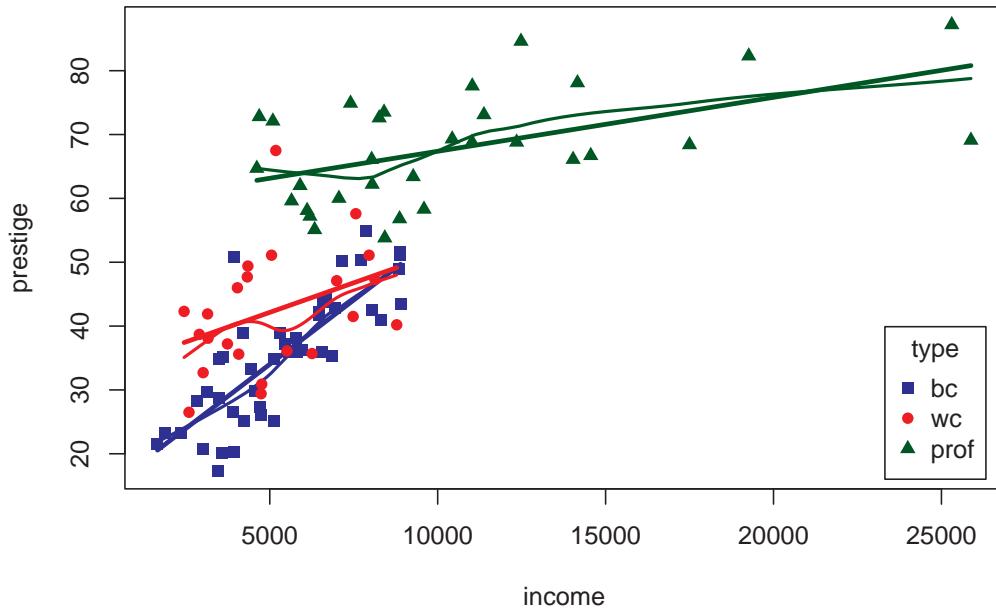


Figure 3.16: Scatterplot of prestige vs. income, stratified by occupational type. This implies a different interpretation, where occupation type is a moderator variable.

This visual analysis offers a different interpretation of the dependence of prestige on income, which appeared to be non-linear when occupation type was ignored. Instead, Figure 3.16 suggests an *interaction* of income by type. In a model formula this would be expressed as one of:

```
lm(prestige ~ income + type + income:type, data = Prestige)
lm(prestige ~ income * type, data = Prestige)
```

These models signify that there are different slopes (and intercepts) for the three occupational types. In this interpretation, `type` is a moderator variable, with a different story. The slopes of the fitted lines suggest that among blue collar workers, prestige increases sharply with their income. For white collar and professional workers, there is still an increasing relation of prestige with income, but the effect of income (slope) diminishes with higher occupational category. A different plot entails a different story.

3.2.3 Example: Penguins data

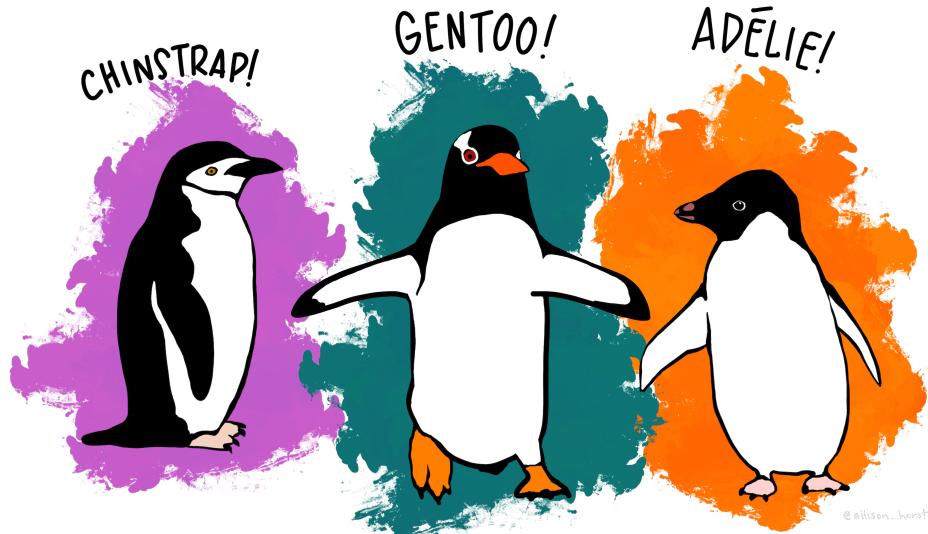


Figure 3.17: Penguin species observed in the Palmer Archipelago. This is a cartoon, but it illustrates some features of penguin body size measurements, and the colors typically used for species. Image: Allison Horst

The `penguins` dataset from the [palmerpenguins](#) package (Horst et al., 2022) provides further instructive examples of plots and analyses of multivariate data. The data consists of measurements of body size (flipper length, body mass, bill length and depth) of 344 penguins collected at the [Palmer Research Station](#) in Antarctica.

There were three different species of penguins (Adélie, Chinstrap & Gentoo) collected from 3 islands in the Palmer Archipelago between 2007–2009 (Gorman et al., 2014). The purpose was to examine differences in size or appearance of these species, particularly differences among the sexes (sexual dimorphism) in relation to foraging and habitat.

Here, I use a slightly altered version of the dataset, `heplots::peng`, constructed by renaming variables to remove the units, making factors of character variables and deleting a few cases with missing data.⁴

```
data(penguins, package = "palmerpenguins")
peng <- penguins |>
  rename(
    bill_length = bill_length_mm,
    bill_depth = bill_depth_mm,
    flipper_length = flipper_length_mm,
    body_mass = body_mass_g
  ) |>
  mutate(species = as.factor(species),
         island = as.factor(island),
         sex = as.factor(substr(sex,1,1))) |>
  tidyr::drop_na()

str(peng)
```

⁴In R 4.5.0, a revised version of the `palmerpenguins` dataset, named `penguins` was added to the base R `datasets`. This corrected a few of the problems that led me to create `heplots::peng`, but still retains the cases with missing data.

```
#> tibble [333 x 8] (S3: tbl_df/tbl/data.frame)
#> $ species      : Factor w/ 3 levels "Adelie","Chinstrap",...: 1 1 1 1 1 1 1 1 1 ...
#> $ island       : Factor w/ 3 levels "Biscoe","Dream",...: 3 3 3 3 3 3 3 3 3 ...
#> $ bill_length  : num [1:333] 39.1 39.5 40.3 36.7 39.3 38.9 39.2 41.1 38.6 34.6 ...
#> $ bill_depth   : num [1:333] 18.7 17.4 18 19.3 20.6 17.8 19.6 17.6 21.2 21.1 ...
#> $ flipper_length: int [1:333] 181 186 195 193 190 181 195 182 191 198 ...
#> $ body_mass    : int [1:333] 3750 3800 3250 3450 3650 3625 4675 3200 3800 4400 ...
#> $ sex          : Factor w/ 2 levels "f","m": 2 1 1 1 2 1 2 1 2 2 ...
#> $ year         : int [1:333] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

There are quite a few variables to choose for illustrating data ellipses in scatterplots. Here I focus on the measures of their bills, `bill_length` and `bill_depth` (indicating curvature) and show how to use `ggplot2` for these plots.

I'll be using the penguins data quite a lot, so it is useful to set up custom colors like those used in Figure 3.17, and shown in Figure 3.18 with their color codes. These are shades of:

- Adelie: orange,
- Chinstrap: purple, and
- Gentoo: green.

Adelie	Chinstrap	Gentoo
#FDBF6F	#CAB2D6	#B2DF8A
#F89D38	#9A78B8	#73C05B
#F37A00	#6A3D9A	#33a02c

Figure 3.18: Color palettes used for penguin species.

To use these in `ggplot2` I define a function `peng.colors()` that allows shades of light, medium and dark and then functions `scale_*_penguins()` for color and fill.

```
peng.colors <- function(shade=c("medium", "light", "dark")) {
  shade = match.arg(shade)
  #           light     medium     dark
  oranges <- c("#FDBF6F", "#F89D38", "#F37A00") # Adelie
  purples <- c("#CAB2D6", "#9A78B8", "#6A3D9A") # Chinstrap
  greens <- c("#B2DF8A", "#73C05B", "#33a02c") # Gentoo
```

```

cols.vec <- c(oranges, purples, greens)
cols.mat <-
  matrix(cols.vec, 3, 3,
         byrow = TRUE,
         dimnames = list(species = c("Adelie", "Chinstrap", "Gentoo"),
                         shade = c("light", "medium", "dark")))
# get shaded colors
cols.mat[, shade ]
}

# define color and fill scales
scale_fill_penguins <- function(shade=c("medium", "light", "dark"), ...){
  shade = match.arg(shade)
  ggplot2::discrete_scale(
    "fill", "penguins",
    scales::manual_pal(values = peng.colors(shade)), ...)
}

scale_colour_penguins <- function(shade=c("medium", "light", "dark"), ...){
  shade = match.arg(shade)
  ggplot2::discrete_scale(
    "colour", "penguins",
    scales::manual_pal(values = peng.colors(shade)), ...)
}
scale_color_penguins <- scale_colour_penguins

```

This is used to define a `theme_penguins()` function that I use to simply change the color and fill scales for plots below.

```

theme_penguins <- function(shade=c("medium", "light", "dark"), ...) {
  shade = match.arg(shade)
  list(scale_color_penguins(shade=shade),
       scale_fill_penguins(shade=shade)
     )
}

```

An initial plot using `ggplot2` shown in Figure 3.19 uses color and point shape to distinguish the three penguin species. I annotate the plot of points using the linear regression lines, loess smooths to check for non-linearity and 95% data ellipses to show precision of the linear relation.

```

ggplot(peng,
       aes(x = bill_length, y = bill_depth,
           color = species, shape = species, fill=species)) +
  geom_point(size=2) +
  geom_smooth(method = "lm", formula = y ~ x,
              se=FALSE, linewidth=2) +
  geom_smooth(method = "loess", formula = y ~ x,
              linewidth = 1.5, se = FALSE, alpha=0.1) +
  stat_ellipse(geom = "polygon", level = 0.95, alpha = 0.2) +
  theme_penguins("dark") +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.15))

```

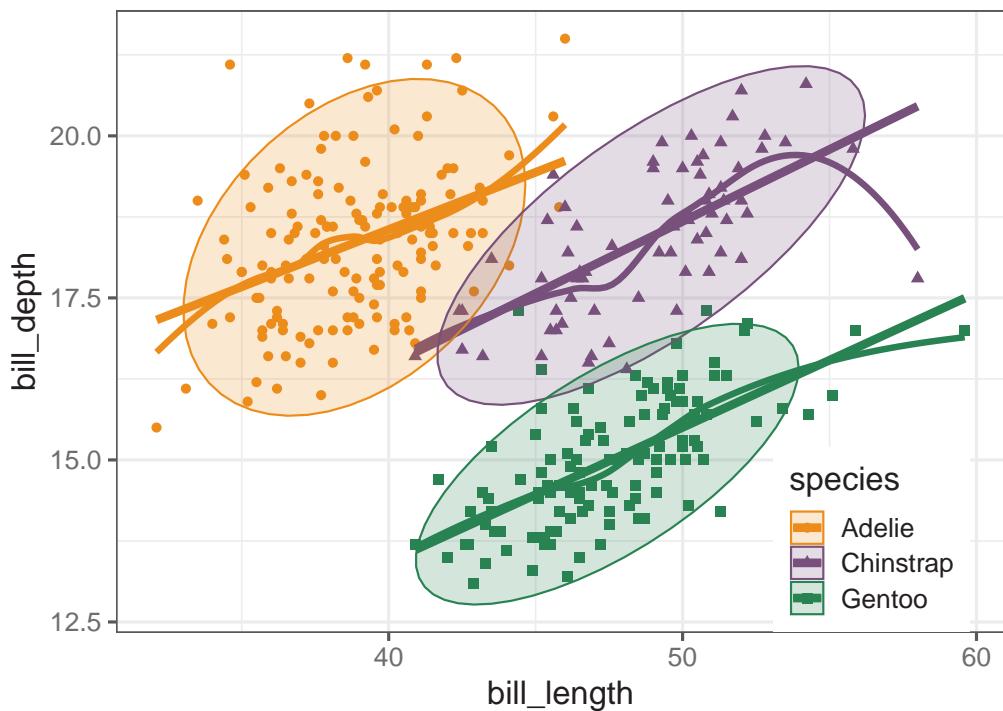


Figure 3.19: Penguin bill length and bill depth according to species.

3.2.4 Visual thinning

Overall, the three species occupy different regions of this 2D space and for each species the relation between bill length and depth appears reasonably linear. Given this, we can suppress plotting the data points to get a visual summary of the data using the fitted regression lines and data ellipses, as shown in Figure 3.20.

This idea, of **visual thinning** a graph to focus on what should be seen, becomes increasingly useful as the data becomes more complex. The `ggplot2` framework encourages this, because we can think of various components as layers, to be included or not. In Figure 3.20 I chose to include only the regression line and add data ellipses of 40%, 68% and 95% coverage to highlight the increasing bivariate density around the group means.

```
ggplot(peng,
       aes(x = bill_length, y = bill_depth,
           color = species, shape = species, fill=species)) +
  geom_smooth(method = "lm", se=FALSE, linewidth=2) +
  stat_ellipse(geom = "polygon", level = 0.95, alpha = 0.2) +
  stat_ellipse(geom = "polygon", level = 0.68, alpha = 0.2) +
  stat_ellipse(geom = "polygon", level = 0.40, alpha = 0.2) +
  theme_penguins("dark") +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.15))
```

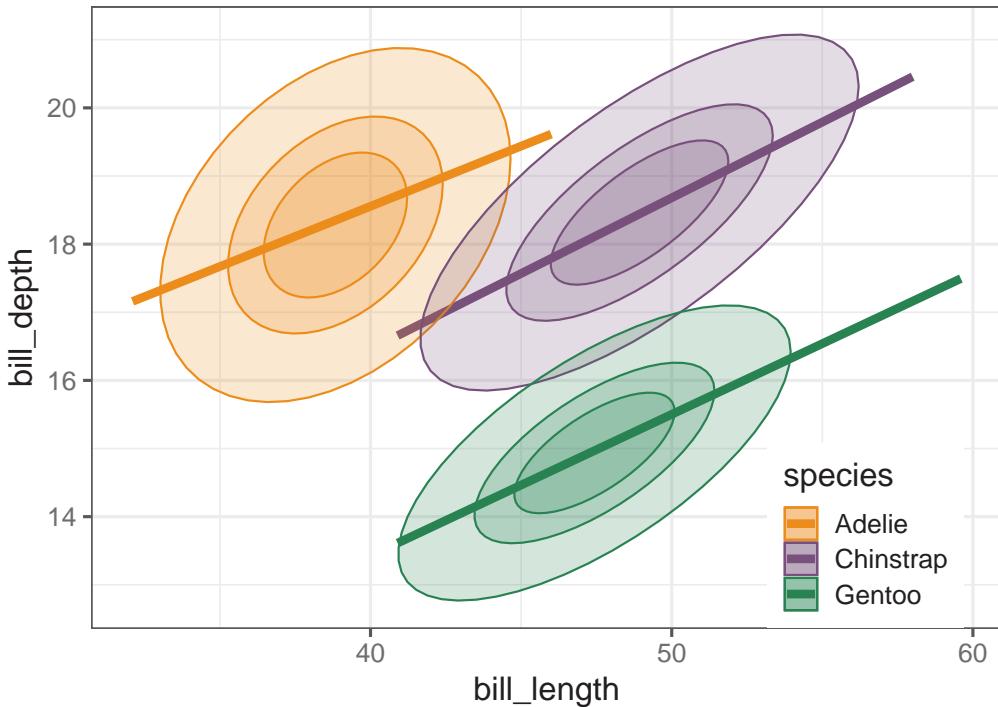


Figure 3.20: Visual thinning: Suppressing the data points gives a visual summary of the relation between bill length and bill depth using the regression line and data ellipses.

3.3 Bagplots

If you are concerned about the assumption of bivariate normality entailed by the data ellipse, a very nice non-parametric and robust alternative is a 2D generalization of a boxplot called a **bagplot**, introduced by Peter J. Rousseeuw et al. (1999). The idea is very simple. The bagplot consists of three nested polygons, called the “bag”, the “fence”, and the “loop”:

- **bag:** The central 50% box of the boxplot is replaced by a polygon called the “bag”, constructed on the basis of *depth* of points from the bivariate depth median point. Depth generalizes the univariate concept of rank, but counted from the medians point outward in any direction. The bag contains at most 50% of the most central data points.
- **fence:** The univariate fences are replaced by a “fence” polygon, by expanding the bag outward by a factor (`coef`), usually 3, from the depth median.
- **loop:** Points beyond the fence (the “loop”) are potential outliers, sometimes plotted as a convex hull surrounding *all* of the observations, but better rendered simply as a scatterplot of just those points.

In this way, the bagplot visualizes the bivariate location (median), spread, correlation, skewness, and tails of the data. Compared with a standard data ellipse, it serves as a visual test of normality and provides a simple way to identify outliers.

Bagplots are implemented in `gggda` (Brunson & Gracey, 2025) in the `ggplot2` framework as `geom_bagplot()` with computations done via `stat_bagplot()`. For the Penguin data, the bagplot version of Figure 3.20 is shown in Figure 3.21.

```
ggplot(peng,
       aes(x = bill_length, y = bill_depth,
           color = species, shape = species, fill=species)) +
  geom_smooth(method = "lm", formula = y ~ x,
              se=FALSE, linewidth=2) +
  geom_bagplot(bag.alpha = 0.5,
               outlier.size = 5,
               fraction = 0.5,      # bag fraction
               coef = 2.5,          # fence factor
               show.legend = FALSE) +
  theme_penguins("dark") +
  theme(legend.position = "inside",
        legend.position.inside = c(0.87, 0.15))
```

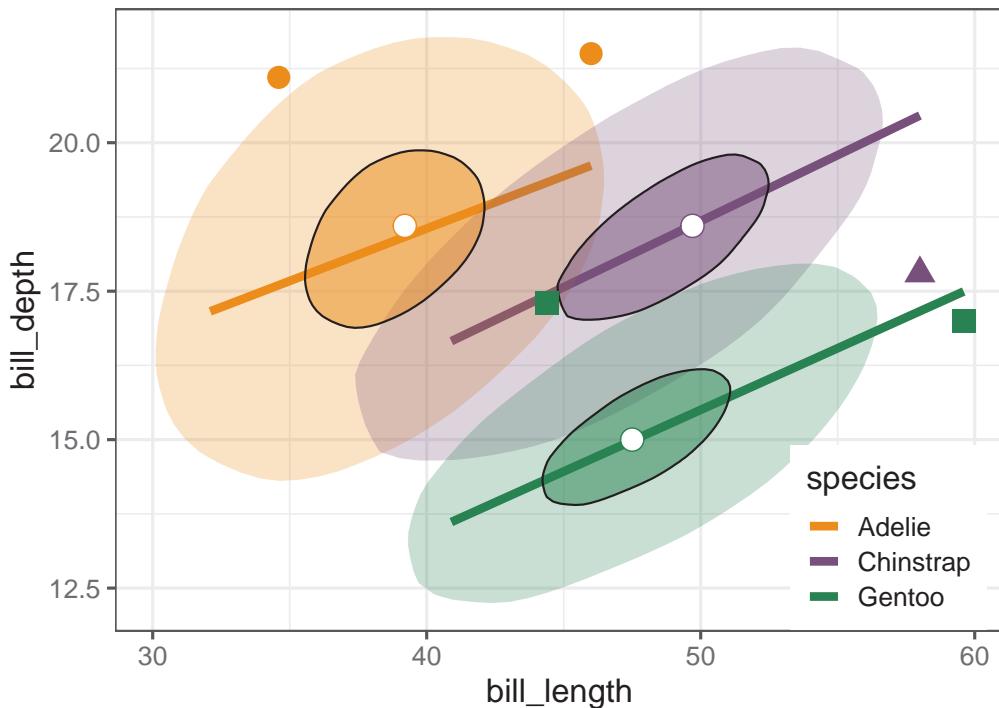


Figure 3.21: Bagplot: For each Penguin species the darker inner (bag) polygon reflects the innermost 50% of the data points. The outer (fence) polygon corresponds to points enclosed by a multiple of the bag. Points outside the fence for each species are plotted individually.

Compared with the data ellipses shown in Figure 3.20, the bag polygons have shapes very close to ellipses, giving credence to the assumption of (approximate) normality. Using the factor `coef = 2.5` causes five points to be flagged as potential outliers. Bivariate skewness is indicated when the bag and fence is not symmetric around the central median in some direction, as is slightly true for all three species.

I discuss other visual tests of multivariate normality in Section 3.9 and consider outlier identification for the Penguin data in Section 3.9.2 below.

3.4 Non-parametric bivariate density plots

While I emphasize data ellipses (because I like their beautiful geometry), other visual summaries of the bivariate density are possible and often useful. To see more detail about the “shape” of bivariate data in a non-parametric way you can use one of the methods described below to see a model-free representation of your data.

For a single variable, `stats::density()` and `ggplot2::geom_density()` calculate a smoothed estimate of the density using nonparametric kernel methods (Silverman, 1986) whose smoothness is controlled by a bandwidth parameter, analogous to the span in a loess smoother. This idea extends to two (and more) variables (Scott, 1992). For bivariate data, `MASS::kde2d()` estimates the density on a square $n \times n$ grid over the ranges of the variables.

`ggplot2` provides `geom_density_2d()` which uses `MASS::kde2d()` and displays these as contours—horizontal slices of the 3D surface at equally-spaced heights and projects these onto the 2D plane. The `ggdensity` package (Otto & Kahle, 2023) extends this with `geom_hdr()`, computing the high density regions that bound given levels of probability and maps these to the `alpha` transparency aesthetic. A `method` argument allows you to specify various nonparametric (`method = "kde"` is the default) and parametric (`method = "mvnorm"` gives normal data ellipses) ways to estimate the underlying bivariate distribution.

Figure 3.22 shows these side-by-side for comparison. With `geom_density_2d()` you can specify either the number of contour `bins` or the width of these bins (`binwidth`). For `geom_hdr()`, the `probs` argument gives a result that is easier to understand.

```
library(ggdensity)
library(patchwork)
p1 <- ggplot(peng,
  aes(x = bill_length, y = bill_depth,
      color = species)) +
  geom_smooth(method = "lm", se=FALSE, linewidth=2) +
  geom_density_2d(binwidth = 1.1, bins = 8) +
  ggtitle("geom_density_2d") +
  theme_bw(base_size = 14) +
  theme_penguins() +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.15))

p2 <- ggplot(peng,
  aes(x = bill_length, y = bill_depth,
      color = species, fill = species)) +
  geom_smooth(method = "lm", se=FALSE, linewidth=2) +
  geom_hdr(probs = c(0.95, 0.68, 0.4), show.legend = FALSE) +
  ggtitle("ggdensity::geom_hdr") +
  theme_bw(base_size = 14) +
  theme_penguins() +
  theme(legend.position = "none")

p1 + p2
```

Compared with the data ellipse, which is highly smoothed by the Gaussian assumption and the bagplot, which is smoothed by the concept of contours of data depth, the plots in Figure 3.22 show considerably more detail, and an intriguing suggestion of two peaks for our Chinstrap penguins.

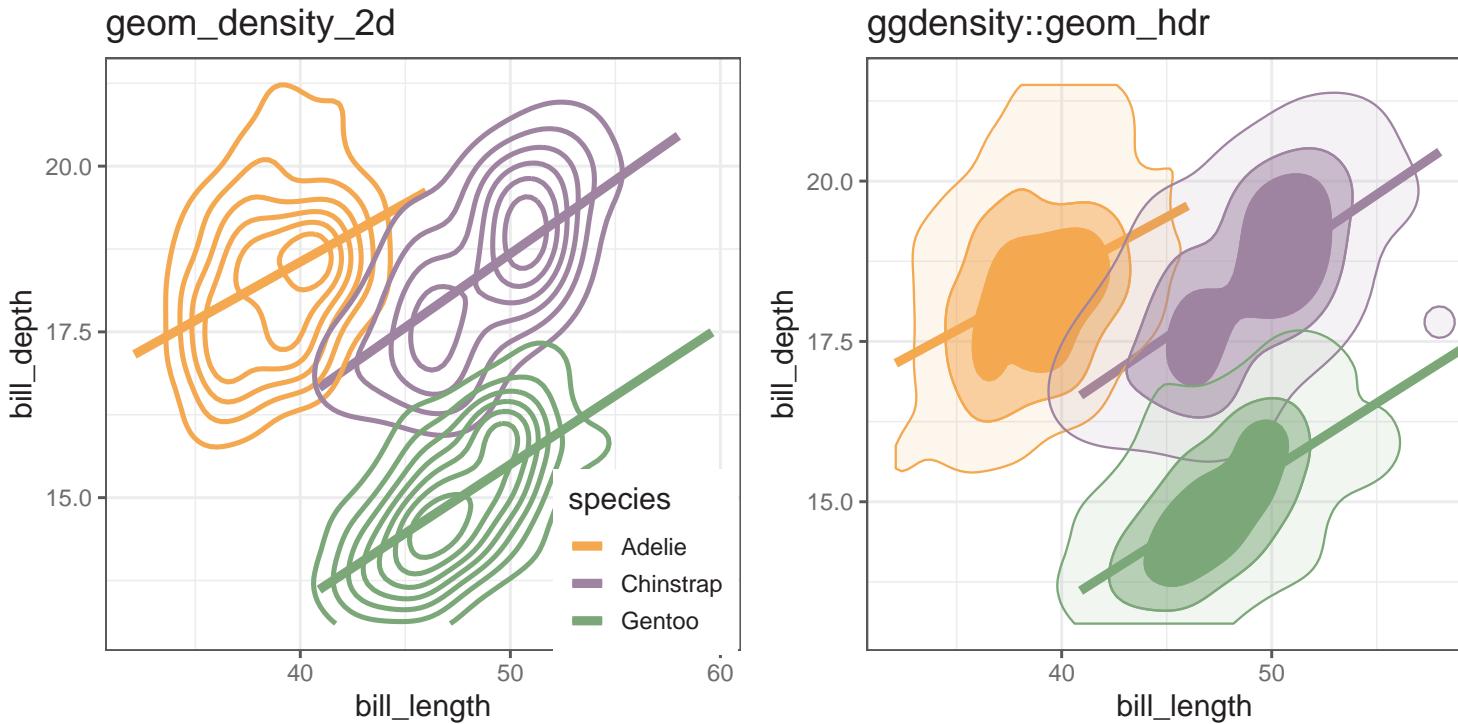


Figure 3.22: Bivariate densities show the contours of the 3D surface representing the frequency in the joint distribution of bill length and bill depth.

3.5 Simpson's paradox: marginal and conditional relationships

Because it provides a visual representation of means, variances, and correlations, the data ellipse is ideally suited as a tool for illustrating and explicating various phenomena that occur in the analysis of linear models. One class of simple, but important, examples concerns the difference between the *marginal relationship* between variables, ignoring some important factor or covariate, and the *conditional* relationship, adjusting (controlling) for that variable.

An important example is **Simpson's paradox** (Simpson, 1951) which occurs when the marginal and conditional relationships differ in direction. That is, the overall correlation in a model $y \sim x$ might be negative, while the within-group correlations in separate models for each group $y[g] \sim x[g]$ might be positive, or vice versa. For Flatlanders in 2D space, this is a puzzlement.

We can see this paradox in the plots of bill length against bill depth for the penguin data shown in Figure 3.23. Ignoring penguin species, the marginal, total-sample correlation is slightly negative as seen in panel (a). The individual-sample ellipses in panel (b) show that the conditional, within-species correlations are all positive, with approximately equal regression slopes. However the group means have a negative relationship, accounting for the negative marginal correlation when species is ignored.

The regression line in panel (a) is that for the linear model `lm(bill_depth ~ bill_length)`, while the separate lines in panel (b) are those for the model `lm(bill_depth ~ bill_length * species)` which allows a different slope and intercept for each species.

A correct analysis of the (conditional) relationship between these variables, controlling or adjusting for mean differences among species, is based on the pooled within-sample covariance matrix, a weighted average of the individual within-group \mathbf{S}_i ,

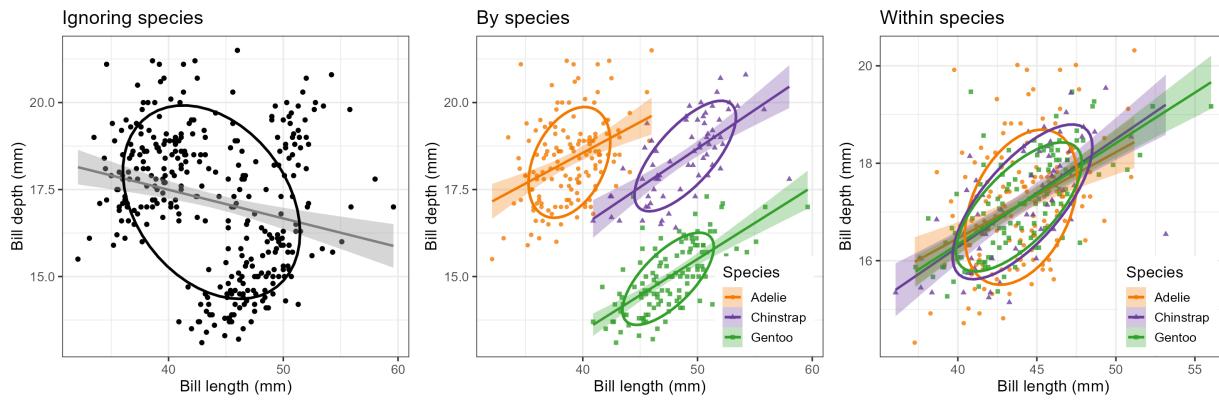


Figure 3.23: Marginal (a), conditional (b), and pooled within-sample (c) relationships of bill length and depth in the Penguins data. Each plot shows the 68% data ellipse and regression line(s) with 95% confidence bands.

$$\mathbf{S}_{\text{within}} = \sum_{i=1}^g (n_i - 1) \mathbf{S}_i / (N - g) ,$$

where $N = \sum n_i$. The result is shown in panel (c) of Figure 3.23.

In this graph, the data for each species were first transformed to deviations from the species means on both variables and then translated back to the grand means. You can also see here that the shapes and sizes of the individual data ellipses are roughly comparable, but perhaps not identical. This visual idea of centering groups to a common mean will become important in Chapter 12 when we want to test the assumption of equality of error covariances in multivariate models.

The `ggplot2` code for the panels in this figure are shown below. Note that for components that will be the same across panels, you can define elements (e.g., `labels`, `theme_penguins()`, `legend_position`) once, and then re-use these across several graphs.

TODO This panel tabset looks fine in HTML but is awkward in PDF

3.6 (a) Ignoring species

```
labels <- labs(
  x = "Bill length (mm)",
  y = "Bill depth (mm)",
  color = "Species",
  shape = "Species",
  fill = "Species")

plt1 <- ggplot(data = peng,
                 aes(x = bill_length,
                     y = bill_depth)) +
  geom_point(size = 1.5) +
  geom_smooth(method = "lm", formula = y ~ x,
```

```

            se = TRUE, color = "gray50") +
stat_ellipse(level = 0.68, linewidth = 1.1) +
ggttitle("Ignoring species") +
labels

plt1

```

3.7 (b) By species

```

legend_position <-
theme(legend.position = "inside",
      legend.position.inside = c(0.83, 0.16))

plt2 <- ggplot(data = peng,
                aes(x = bill_length,
                    y = bill_depth,
                    color = species,
                    shape = species,
                    fill = species)) +
geom_point(size = 1.5,
           alpha = 0.8) +
geom_smooth(method = "lm", formula = y ~ x,
            se = TRUE, alpha = 0.3) +
stat_ellipse(level = 0.68, linewidth = 1.1) +
ggttitle("By species") +
labels +
theme_penguins("dark") +
legend_position

plt2

```

3.8 (c) Within species

```

# center within groups, translate to grand means
means <- colMeans(peng[, 3:4])
peng.centered <- peng |>
  group_by(species) |>
  mutate(bill_length = means[1] + scale(bill_length, scale = FALSE),
         bill_depth = means[2] + scale(bill_depth, scale = FALSE))

plt3 <- ggplot(data = peng.centered,
                aes(x = bill_length,
                    y = bill_depth,
                    color = species,

```

```

      shape = species,
      fill = species)) +
geom_point(size = 1.5,
           alpha = 0.8) +
geom_smooth(method = "lm", formula = y ~ x,
            se = TRUE, alpha = 0.3) +
stat_ellipse(level = 0.68, linewidth = 1.1) +
labels +
ggtitle("Within species") +
theme_penguins("dark") +
legend_position

plt3

```

3.9 Multivariate normality and outliers

The relation of the data ellipsoid for p variables to the χ_p^2 distribution with p degrees of freedom described in Section 3.2 is based on the assumption that the data in \mathbf{y} is a sample from a multivariate normal distribution (MVN), with a mean vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$, so each one implies the other:

$$\mathbf{y}_{p \times 1} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \iff D_M^2(\mathbf{y}) \sim \chi_p^2.$$

This fact can be used to assess whether sample data in \mathbf{y} does indeed follow a MVN distribution by plotting the *quantiles*⁵ of the *sorted* sample D^2 values (denoted $D_{(i)}^2$) calculated from Equation 3.1 against the corresponding χ_p^2 quantiles found from `qchisq(df = p)`. This is called a χ^2 QQ plot.

The essential idea is that the plotted points should then approximately fall along a 45° line of slope = 1 when the data are MVN (and axes are scaled equally). This also provides a simple method to identify potential outliers as those points which are furthest from the centroid; that is those for which $D_{(i)}^2$ is greater than the $1 - \alpha$ quantile of χ_p^2 .

The topics of assessing multivariate normality and detecting multivariate outliers are larger than I consider here. These topics are also intertwined, because outliers inflate the variance (\mathbf{S}) of the data, making extreme observations appear less extreme. A variety of *robust* methods described later (sec ??) work by down-weighting outliers, which is particularly important in assessing whether the residuals from a multivariate linear model are multivariate normal.

In this section, I focus on the χ^2 QQ plot and graphical methods to relate the the points identified as potential outliers to plots in data space.

3.9.1 Galton data

Mahalanobis D^2 values are calculated by `heplots::Mahalanobis()`. I illustrate finding the largest $D_{(i)}^2$ for the `Galton` data as shown below. I've used $\alpha = 0.01$, giving $\chi_p^2(0.99) = 9.21$ as an outlier cutoff, just for the sake of this example. With 928 cases, we would expect about 1% or 9 to have larger squared distances than this cutoff. (Normally, allowing for the fact that we are looking at the largest values in a sample of size n , we would use a much smaller individual significance level, say $\alpha = 0.001$ or smaller still.) Three cases are identified here.

⁵The p^{th} -quantile of a random variable Y is a value, denoted by $Q_Y(p)$, such that $X \leq Q_Y(p)$ with probability p , i.e., $\Pr(Y \leq Q_Y(p)) = p$. For continuous variables, the quantile is the inverse of the cumulative distribution function.

```

data(Galton, package = "HistData")
DSQ <- Mahalanobis(Galton)
alpha <- 0.01
cutoff <- (qchisq(p = 1 - alpha, df = ncol(Galton))) |>
  print()
#> [1] 9.21
outliers <- which(DSQ > cutoff) |>
  print()
#> [1] 1 13 897
GaltonD <- cbind(Galton, DSQ = DSQ)
GaltonD[outliers,]
#>   parent child   DSQ
#> 1     70.5 61.7 13.67
#> 13    70.5 63.2  9.45
#> 897   65.5 72.2  9.49

```

χ^2 QQ plots are constructed by `heplots::cpplot()`. For the Galton data, the result is shown in Figure 3.24, where I've asked for the `id.n = 3` with the greatest $D_{(i)}^2$ values to be identified with their row numbers in the plot. The function returns (invisibly) the $D_{(i)}^2$ and corresponding χ^2 quantile along with the upper-tail p -value.

```

out <- cpplot(Galton, id.n = 3)
out
#>   DSQ quantile      p
#> 1  13.67    15.1 0.000539
#> 897 9.49    12.9 0.001616
#> 13  9.45    11.8 0.002694

```

In the typical use of QQ plots, it is essential to have something in the nature of a confidence band around the points to be able to appreciate whether, and to what degree the observed data points differ from the reference distribution. For `cpplot()`, this helps to assess whether the data are reasonably distributed as multivariate normal and also to flag potential outliers.

The pointwise 95% confidence envelope is calculated as $D_{(i)}^2 \pm 1.96 \times \text{se}(D_{(i)}^2)$ where the standard error is calculated (Chambers et al., 1983, sec. 8.6) as

$$\text{se}(D_{(i)}^2) = \frac{\hat{b}}{d(q_i)} \times \sqrt{p_i(1-p_i)/n} .$$

Here, \hat{b} is an estimate of the *actual* slope of the reference line obtained from the ratio of the interquartile range of the D^2 values to that of the corresponding χ_p^2 distribution. $d(q_i)$ is the density of the chi square distribution at the quantile q_i and p_i is the corresponding percentile.

So, in Figure 3.24, you can see the same three points, with largest D^2 identified earlier. The confidence band shows that uncertainty increases as points get further from the mean, but by and large, nearly all the points lie within it. The fact that larger values fall *beneath* the reference line indicates that the sample D^2 are somewhat *less concentrated* (has a shorter upper tail) than values in the χ^2 distribution.

To help understand what we are seeing in this example, it is helpful to view the data in a scatterplot equivalent of Figure 3.11, with the same three observations labeled and made distinctive in the plot. For this plot, because the heights are recorded in whole inches, I draw the data ellipses first without plotting the points and then draw the points using `jitter()` on top of the data ellipses.

Chi-Square Q-Q Plot of Galton

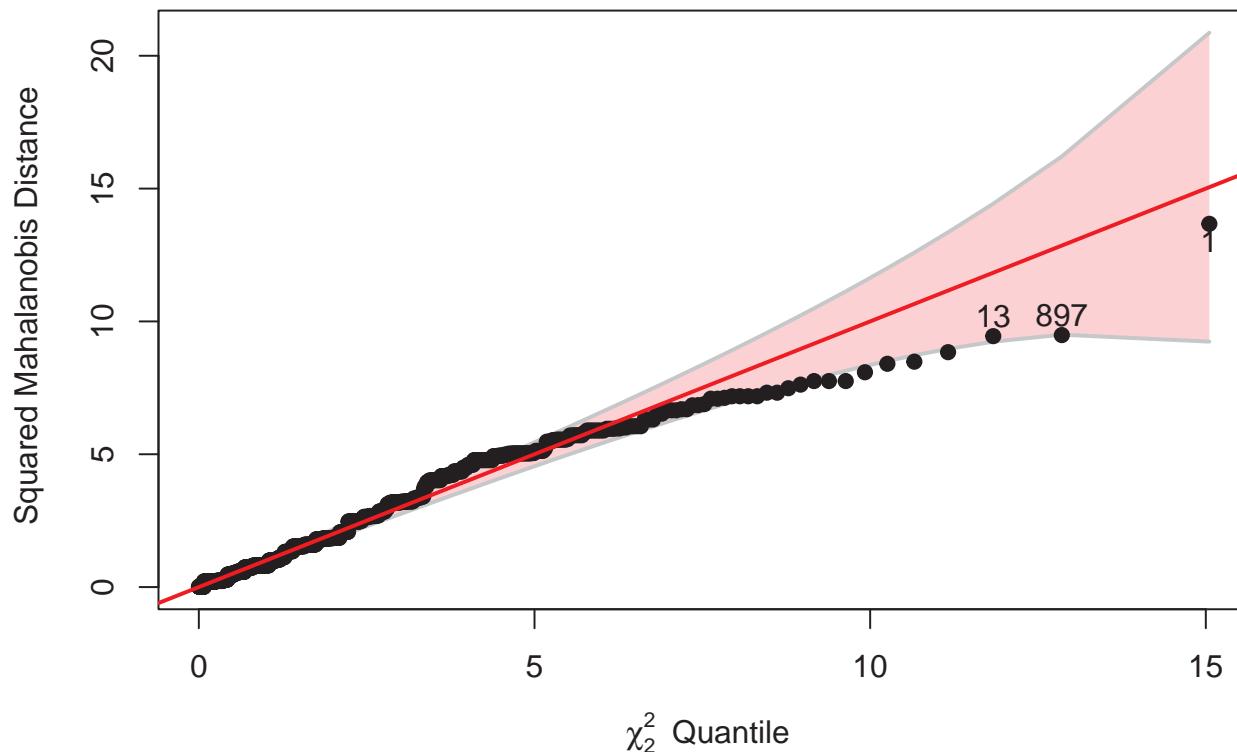


Figure 3.24: Chi-square QQ plot of Galton's data on heights of parents and their offspring, with a 95% pointwise confidence envelope

```

set.seed(47)
dataEllipse(parent ~ child, data = GaltonD,
            levels = c(0.68, 0.95),
            add = FALSE, plot.points = FALSE,
            center.pch = "+", center.cex = 3,
            cex.lab = 1.5)
with(GaltonD,{
  points(jitter(child), jitter(parent),
         col = ifelse(DSQ > cutoff, "red", "black"),
         pch = ifelse(DSQ > cutoff, 16, 1),
         cex = ifelse(DSQ > cutoff, 2, 0.8))
  text(child[outliers], parent[outliers], labels = outliers, pos = 3)
})

```

It is clear from this plot that case 897 is for an exceptionally tall child of quite short parents, while case 1 and 13 are from very short children of tall parents; you could call them poster children for regression toward the mean.

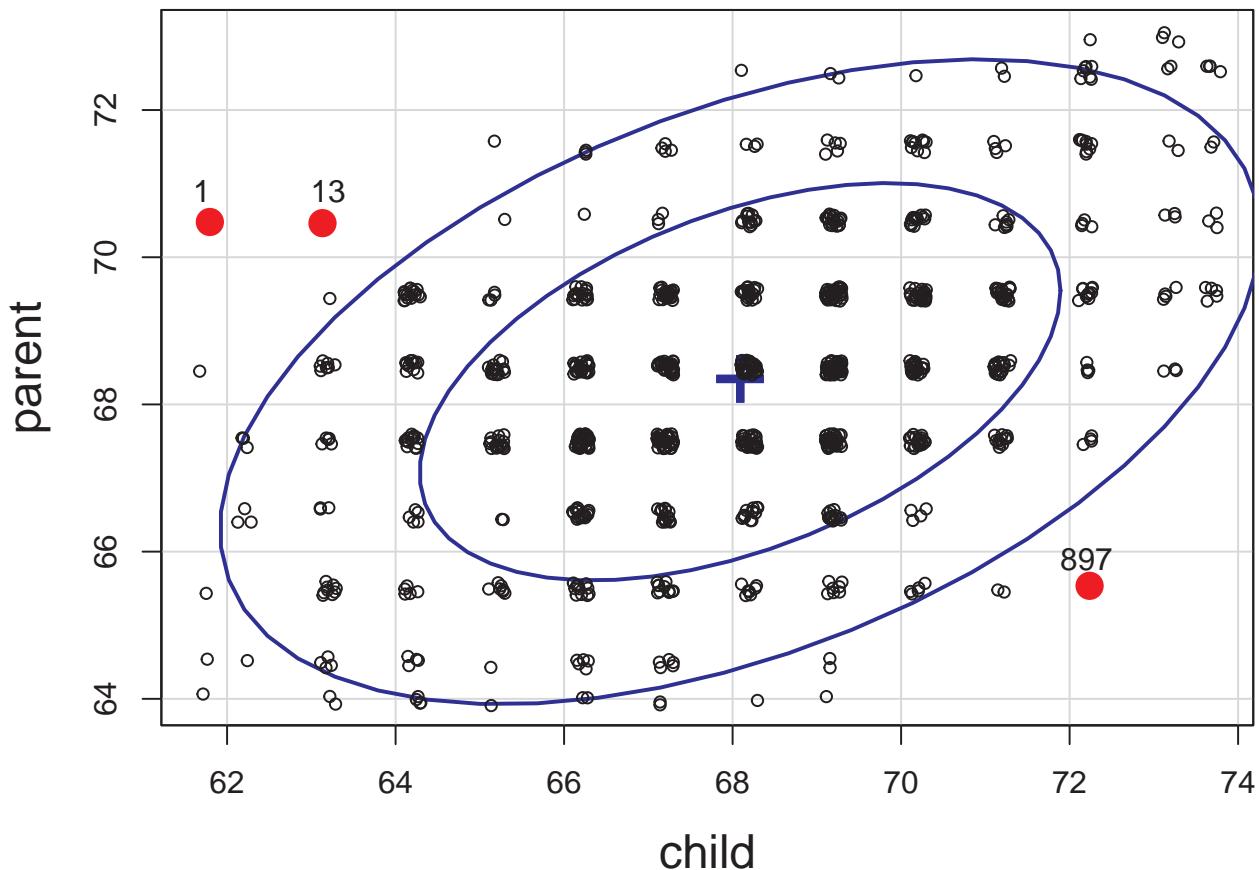


Figure 3.25: Scatterplot of Galton’s data showing 68% and 95% data ellipses for the observations on parent and child height. The discrete points have been jittered to avoid overplotting. The three observations identified as having large D^2 values are labeled with their observation number and given distinctive size, color and shape.

3.9.2 Penguin data

Let’s do a similar analysis to assess multivariate normality and identify possible outliers for the Penguin data. In the `cqplot()` shown in Figure 3.26, I use the same point symbols and colors as in Figure 3.19. For illustration, I again label the three most extreme points.

```

clr <- peng.colors("dark")
pch <- c(19, 17, 15) # ggplot symbol defaults for a factor

out <- cqplot(peng[, 3:6],
  id.n = 3,
  col = clr[peng$species],
  pch = pch[peng$species],
  ref.col = "grey",
  what = "Penguin numeric variables",
  cex.lab = 1.25)
out
#>   DSQ quantile      p
#> 283 27.8      17.6 0.00150

```

```
#> 10 13.3      15.1 0.00450
#> 35 12.4      13.9 0.00751
```

Chi-Square Q-Q Plot of Penguin numeric variables

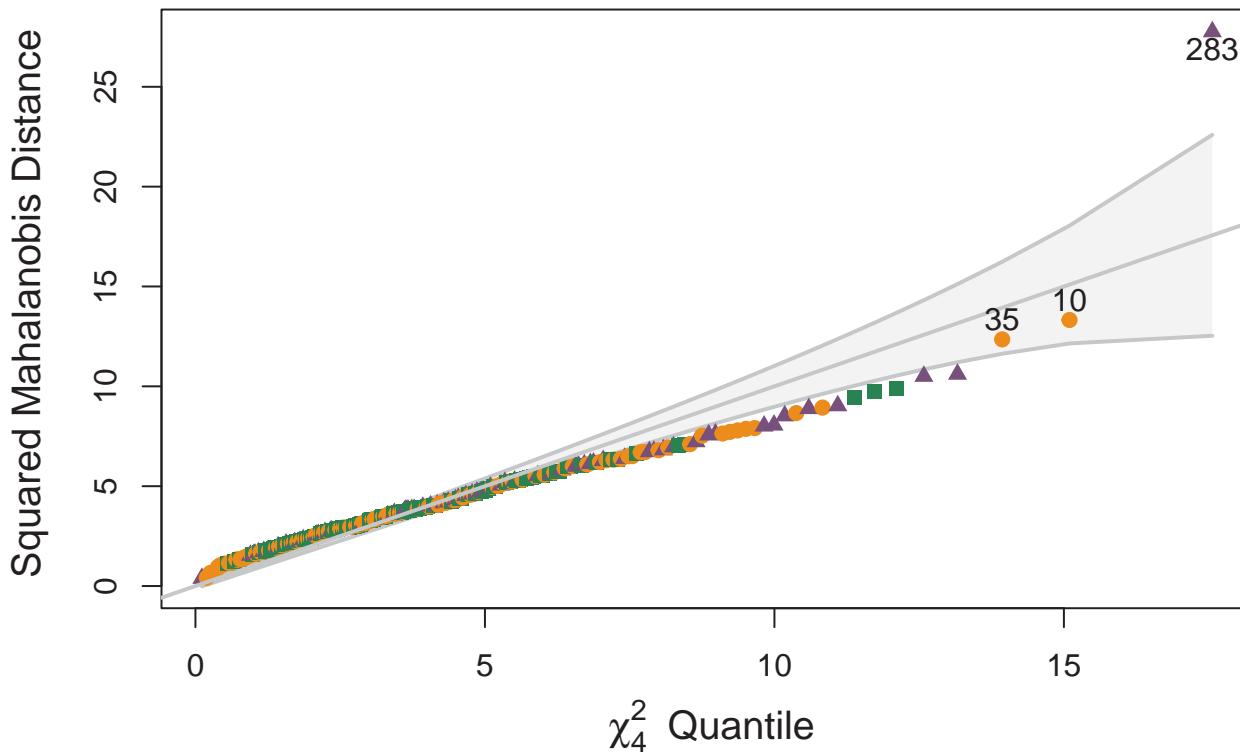


Figure 3.26: Chi-square QQ plot of the Penguin data. The three cases with the largest D^2 values are identified with their case numbers.

One point (case 283) stands out as an extreme multivariate outlier. The other two (10, 35) are well within the confidence envelope.

To relate this to the data, we can plot the data, as was done in Figure 3.19, and label the points identified as noteworthy. It is a bit tricky to label points selectively in `ggplot2` when the criterion for which points to label is complex, or involves variables outside the data frame. Here, I create a logical variable `note`, which is `TRUE` for the noteworthy ones. This is then used to subset the data for `geom_text()` that writes the case id numbers. Figure 3.27 shows the resulting plot.

```
DSQ <- Mahalanobis(peng[, 3:6])
noteworthy <- order(DSQ, decreasing = TRUE)[1:3] |> print()
#> [1] 283 10 35

peng_plot <- peng |>
  tibble::rownames_to_column(var = "id") |>
  mutate(note = id %in% noteworthy)

ggplot(peng_plot,
  aes(x = bill_length, y = bill_depth,
```

```

color = species, shape = species, fill=species)) +
geom_point(aes(size=note), show.legend = FALSE) +
scale_size_manual(values = c(1.5, 4)) +
geom_text(data = subset(peng_plot, note==TRUE),
aes(label = id),
nudge_y = .4, color = "black", size = 5) +
geom_smooth(method = "lm", formula = y ~ x,
se=FALSE, linewidth=2) +
stat_ellipse(geom = "polygon", level = 0.95, alpha = 0.1) +
theme_penguins() +
theme_bw(base_size = 14) +
theme(legend.position = "inside",
legend.position.inside=c(0.85, 0.15))

```

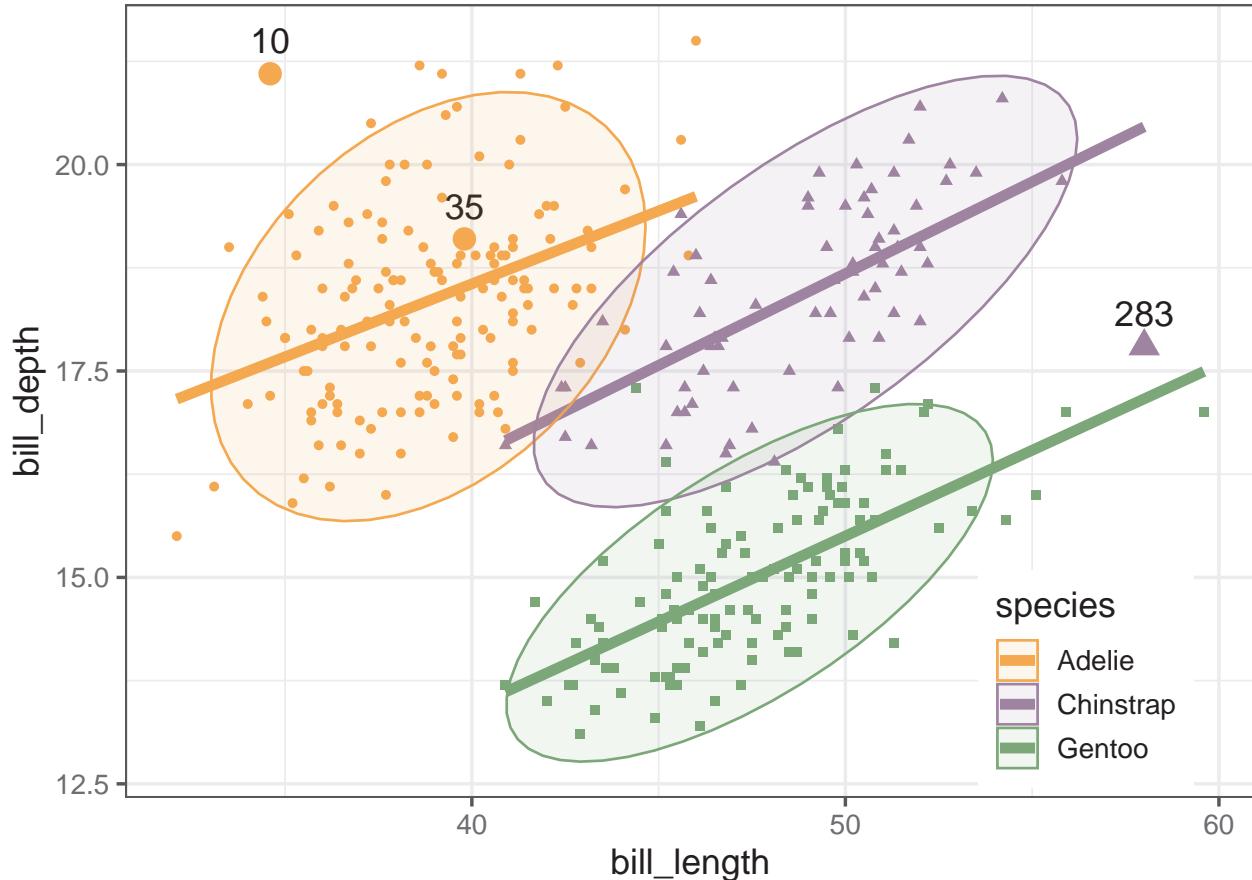


Figure 3.27: Plot of bill length and bill depth, with the noteworthy points labeled. Only one (case 283) is a true multivariate outlier.

Two cases (10, 283) appear to be very unusual in this plot, in relation to other members of their species. But only case 283 is a true multivariate outlier, as shown in the cqplot Figure 3.26. We could call this long-billed penguin “Cyrano”⁶. I’ll call case 10, with a very short and curved bill, “Hook Nose”.

Of course, we are only looking at the data in the 2D space of the bill variables, but possible outliers exist

⁶After the long-nosed Cyrano de Bergerac from Edmund Rostand’s play. Actually, bird 283 is a female. But, it would be a mistake to call her “Rozanne” instead.

in four dimensional Penguin space. Case 35 is well inside the Adelie ellipse, so perhaps it is unusual on one of the other variables. It turns out that multivariate outliers can most often be easily seen as unusual observations in a projection of the data into the space of the *smallest* principal components. I return to this example in Section 4.7.

It bears noting that for linear models, multivariate normality is **not required** for the response variables or predictors, but rather is an assumption for the *residuals* from a model. As well, multivariate outliers in the responses may not turn out to be unusual when the predictor variables are taken into account. In this example, a multivariate model would include the effect of `species`,

```
peng.mod <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~ species,
                 data = peng)
```

and there would be cause for concern if the residuals from this model, `residuals(peng.mod)` were highly non-normal or showed outliers.

3.10 Scatterplot matrices

Going beyond bivariate scatterplots, a *pairs* plot (or *scatterplot matrix*) displays all possible $p \times p$ pairs of p variables in a matrix-like display where variables (x_i, x_j) are shown in a plot for row i , column j . This idea, due to Hartigan (1975b), uses small multiple plots, so that the eye can easily scan across a row or down a column to see how a given variable is related to all the others.

The most basic version is provided by `pairs()` in base R. When one variable is considered as an outcome or response, it is usually helpful to put this in the first row and column. For the `Prestige` data, in addition to income and education, we also have a measure of % women in each occupational category.

Plotting these together gives Figure 3.28. In such plots, the diagonal cells give labels for the variables, but they are also a guide to interpreting what is shown. In each row, say row 2 for `income`, income is the vertical y variable in plots against other variables. In each column, say column 3 for `education`, education is the horizontal x variable.

```
pairs(~ prestige + income + education + women,
      data=Prestige)
```

The plots in the first row show what we have seen before for the relations between prestige and income and education, adding to those the plot of prestige vs. % women. Plots in the first column show the same data, but with x and y interchanged.

But this basic `pairs()` plot is very limited. A more feature-rich version is provided by `car:::scatterplotMatrix()` which can add the regression lines, loess smooths and data ellipses for each pair, as shown in Figure 3.29.

The diagonal panels show density curves for the distribution of each variable; for example, the distribution of `education` appears to be multi-modal and that of `women` shows that most of the occupations have a low percentage of women.

The combination of the regression line with the loess smoothed curve, but without their confidence envelopes, provides about the right amount of detail to take in at a glance where the relations are non-linear. We've already seen (Figure 3.12) the non-linear relation between prestige and income (row 1, column 2) when occupational type is ignored. But all relations with income in column 2 are non-linear, reinforcing our idea (Section 3.2.2.1) that effects of income should be assessed on a log scale.

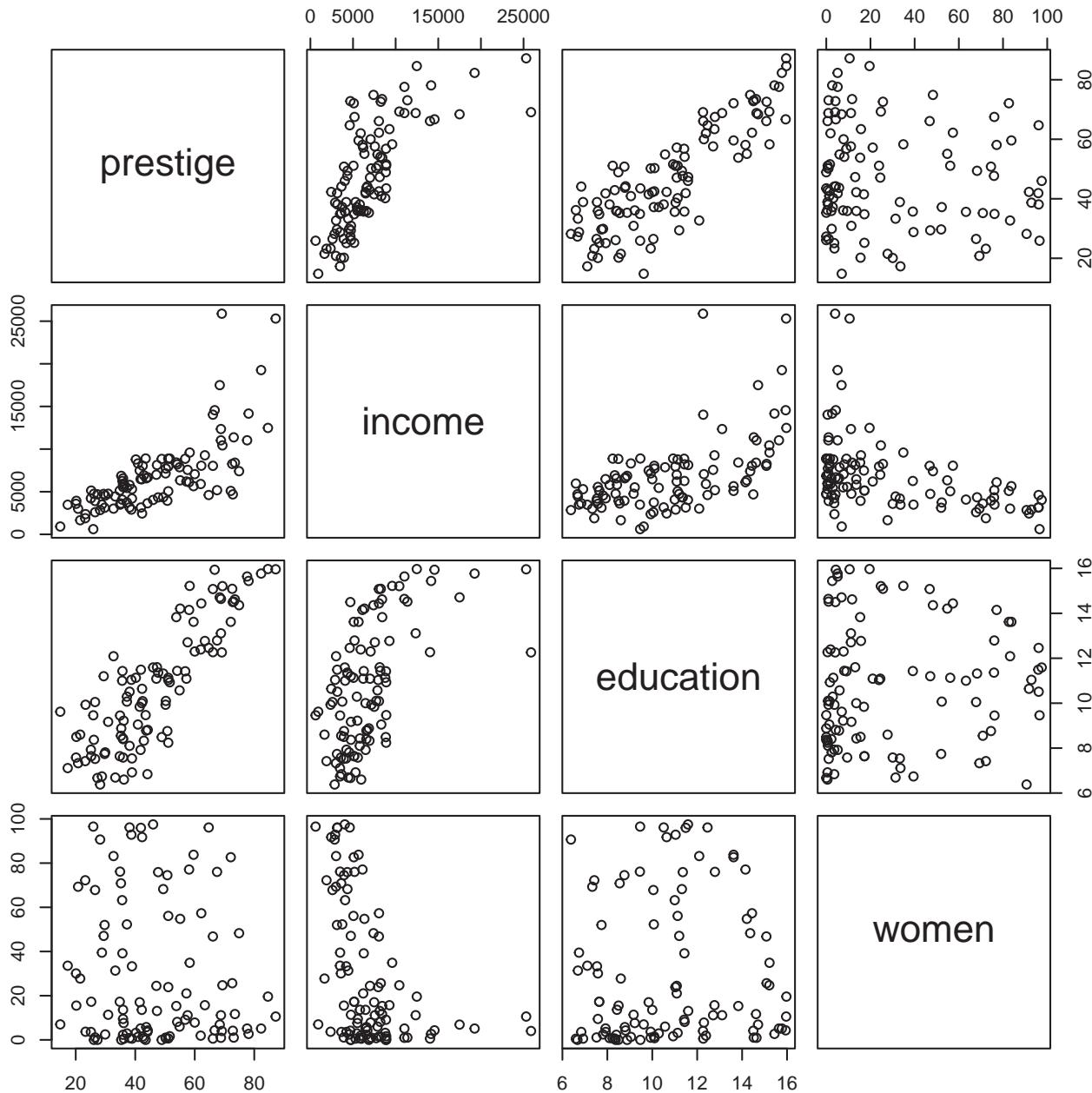


Figure 3.28: Scatterplot matrix of the variables in the Prestige dataset produced by `pairs()`

```
scatterplotMatrix(~ prestige + income + education + women,
  data=Prestige,
  regLine = list(method=lm, lty=1, lwd=2, col="black"),
  smooth=list(smooth=loessLine, spread=FALSE,
    lty.smooth=1, lwd.smooth=3, col.smooth="red"),
  ellipse=list(levels=0.68, fill.alpha=0.1))
```

`scatterplotMatrix()` can also label points using the `id =` argument (though this can get messy) and can stratify the observations by a grouping variable with different symbols and colors. For example, Figure 3.30 uses the syntax `~ prestige + education + income + women | type` to provide separate regression lines,

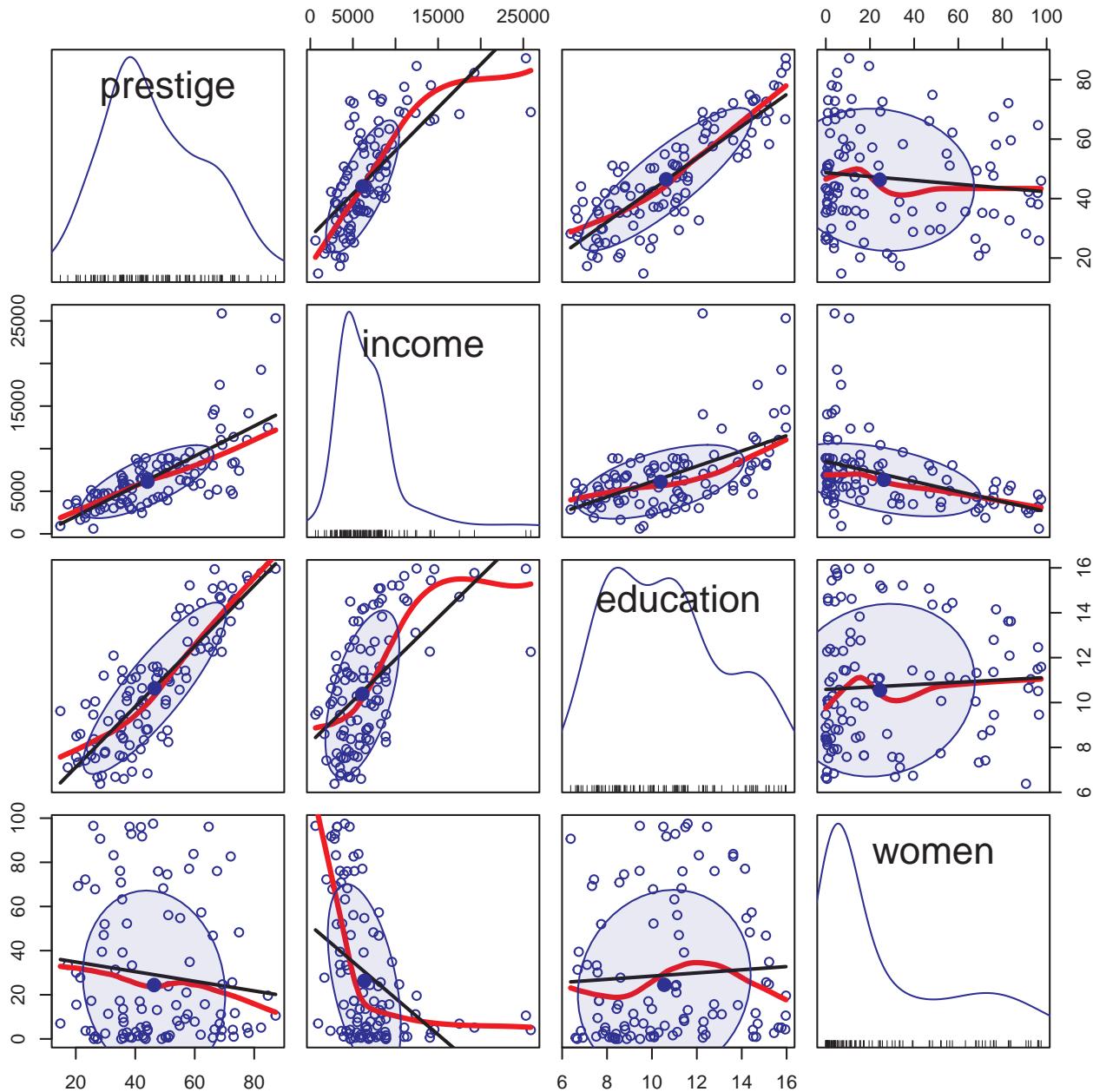


Figure 3.29: Scatterplot matrix of the variables in the Prestige dataset from `car::scatterplotMatrix()`.

smoothed curves and data ellipses for the three types of occupations. (The default colors are somewhat garish, so I use `scales::hue_pal()` to mimic the discrete color scale used in `ggplot2`).

```
scatterplotMatrix(~ prestige + income + education + women | type,
  data = Prestige,
  col = scales::hue_pal()(3),
  pch = 15:17,
  smooth=list(smoker=loessLine, spread=FALSE,
              lty.smooth=1, lwd.smooth=3, col.smooth="black"),
  ellipse=list(levels=0.68, fill.alpha=0.1))
```

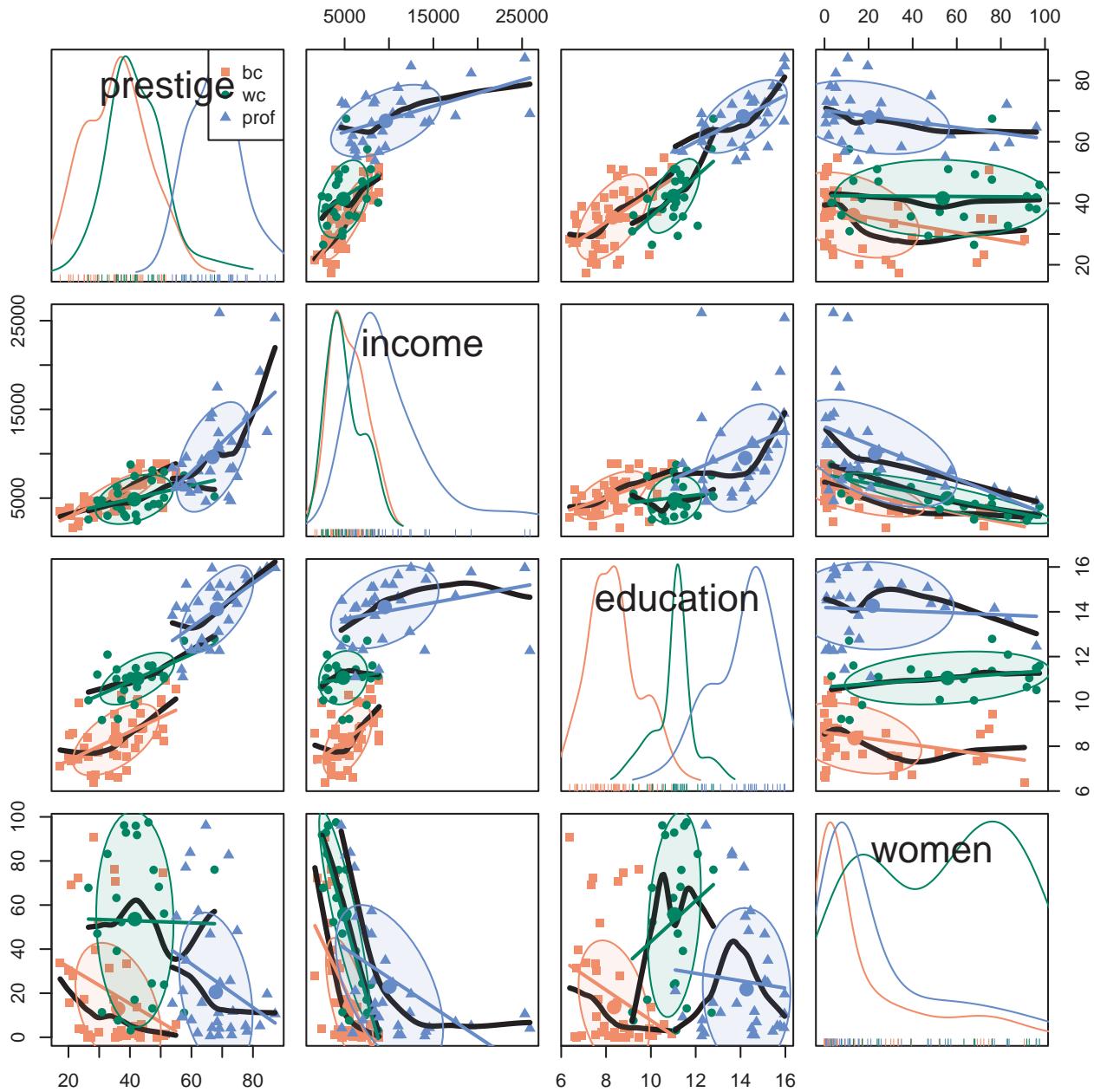


Figure 3.30: Scatterplot matrix of the variables in the Prestige dataset from `car::scatterplotMatrix()`, stratified by type of occupation.

It is now easy to see why education is multi-modal: blue collar, white collar and professional occupations have largely non-overlapping years of education. As well, the distribution of % women is much higher in the white collar category.

For the `penguins` data, given what we've seen before in Figure 3.19 and Figure 3.20, we may wish to suppress details of the points (`plot.points = FALSE`) and loess smooths (`smooth = FALSE`) to focus attention on the similarity of regression lines and data ellipses for the three penguin species. In Figure 3.31, I've chosen to show boxplots rather than density curves in the diagonal panels in order to highlight differences in the means and interquartile ranges of the species, and to show 68% and 95% data ellipses in the off-diagonal panels.

```
scatterplotMatrix(~ bill_length + bill_depth + flipper_length + body_mass | species,
  data = peng,
  col = peng.colors("medium"),
  legend=FALSE,
  ellipse = list(levels = c(0.68, 0.95),
                 fill.alpha = 0.1),
  regLine = list(lwd=3),
  diagonal = list(method = "boxplot"),
  smooth = FALSE,
  plot.points = FALSE,
  cex.labels=1)
```

It can be seen that the species are widely separated in most of the bivariate plots. As well, the regression lines for species have similar slopes and the data ellipses have similar size and shape in most of the plots. From the boxplots, we can also see that **Adelie** penguins have shorter bill lengths than the others, while **Gentoo** penguins have smaller bill depth, but longer flippers and are heavier than **Chinstrap** and **Adelie** penguins.

Looking ahead

Figure 3.31 provides a reasonably complete visual summary of the data in relation to multivariate models that ask “do the species differ in their means on these body size measures?” This corresponds to the MANOVA model,

```
peng.mod <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~ species,
  data=peng)
```

Hypothesis-error (HE) plots, described in Chapter 11 provide a better summary of the evidence for the MANOVA test of differences among means on all variables together. These give an **H** ellipse reflecting the differences among means, to be compared with an **E** ellipse reflecting within-group variation and a visual test of significance.

A related question is “how well are the penguin species distinguished by these body size measures?” Here, the relevant model is linear discriminant analysis (LDA), where **species** plays the role of the response in the model,

```
peng.lda <- MASS:lda( species ~ cbind(bill_length, bill_depth, flipper_length, body_mass),
  data=peng)
```

Both MANOVA and LDA depend on the assumption that the variances and correlations between the variables are the same for all groups. This assumption can be tested and visualized using the methods in Chapter 12.

3.10.1 Visual thinning

What can you do if there are even more variables than in these examples? If what you want is a high-level, zoomed-out display summarizing the pairwise relations more strongly, you can apply the idea of visual thinning to show only the most important features.

This example uses data on the rate of various crimes in the 50 U.S. states from the United States Statistical Abstracts, 1970, used by Hartigan (1975a) and Friendly (1991). These are ordered in the dataset roughly by seriousness of crime or from crimes of violence to property crimes.

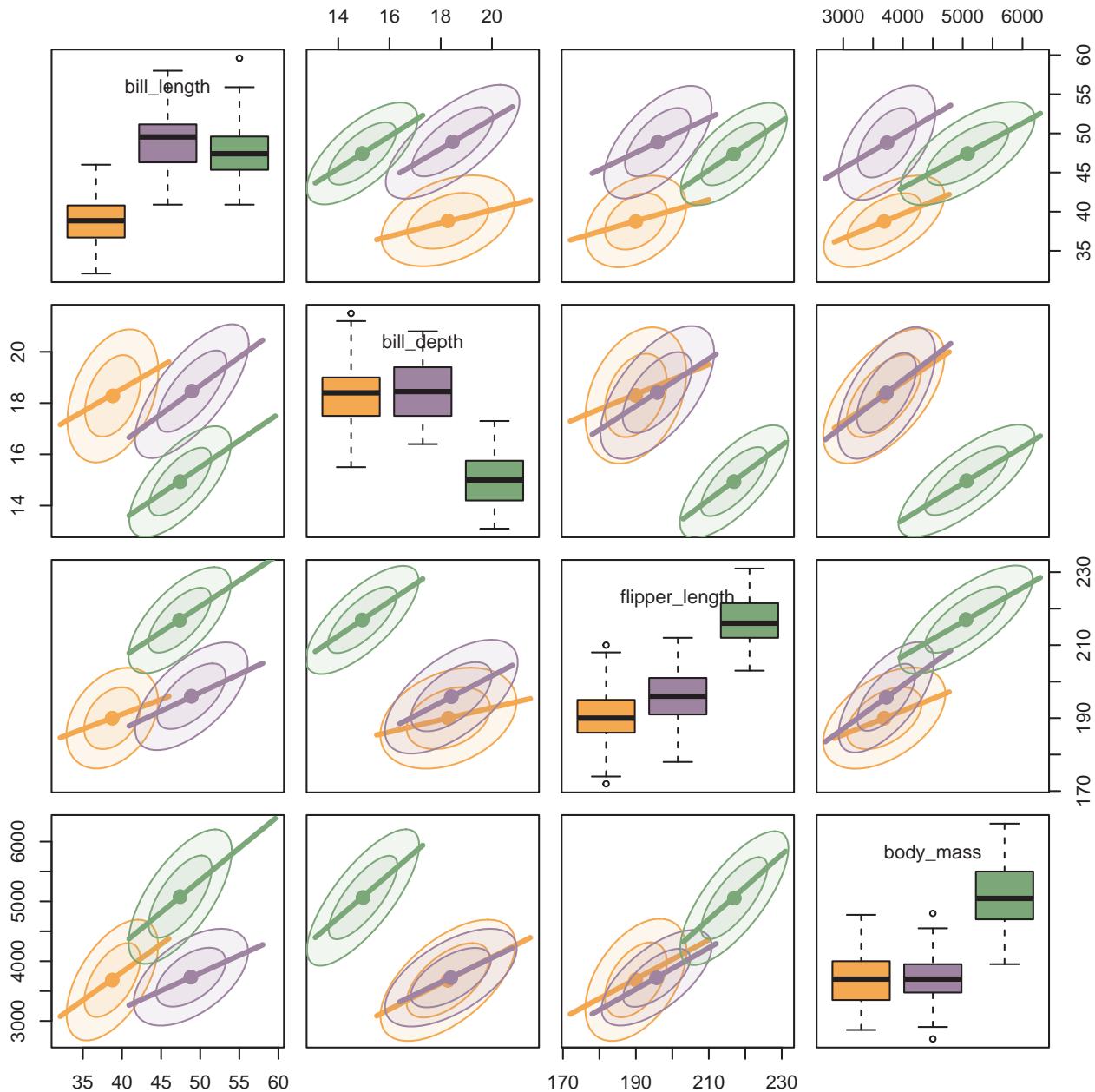


Figure 3.31: Scatterplot matrix of the quantitative variables in the penguins dataset, stratified by species.

```

data(crime, package = "ggbio")
str(crime)
#> 'data.frame': 50 obs. of 10 variables:
#> $ state    : chr "Alabama" "Alaska" "Arizona" "Arkansas" ...
#> $ murder   : num 14.2 10.8 9.5 8.8 11.5 6.3 4.2 6 10.2 11.7 ...
#> $ rape     : num 25.2 51.6 34.2 27.6 49.4 42 16.8 24.9 39.6 31.1 ...
#> $ robbery  : num 96.8 96.8 138.2 83.2 287 ...
#> $ assault  : num 278 284 312 203 358 ...
#> $ burglary: num 1136 1332 2346 973 2139 ...

```

```
#> $ larceny : num 1882 3370 4467 1862 3500 ...
#> $ auto      : num 281 753 440 183 664 ...
#> $ st        : chr "AL" "AK" "AZ" "AR" ...
#> $ region   : Factor w/ 4 levels "Northeast","South",...: 2 4 4 2 4 4 1 2 2 2 ...
```

Figure 3.32 displays the scatterplot matrix for these seven variables, using only the regression line and data ellipse to show the linear relation and the loess smooth to show potential non-linearity. To make this even more schematic, the axis tick marks and labels are also removed using the `par()` settings `xaxt = "n"`, `yaxt = "n"`.

```
crime |>
  select(where(is.numeric)) |>
  scatterplotMatrix(
    plot.points = FALSE,
    ellipse = list(levels = 0.68, fill=FALSE),
    smooth = list(spread = FALSE,
                  lwd.smooth=2, lty.smooth = 1, col.smooth = "red"),
    cex.labels = 2,
    xaxt = "n", yaxt = "n")
```

We can see that all pairwise correlations are positive, pairs closer to the main diagonal tend to be more highly correlated and in most cases the nonparametric smooth doesn't differ much from the linear regression line. Exceptions to this appear mainly in the columns for `robbery` and `auto` (auto theft).

3.11 Corrrgrams

What if you want to summarize the data even further simple visual thinning. For example with many variables you might want to show only the value of the correlation for each pair of variables, but do so in a way to help see patterns in the correlations that would be invisible in just a table.

A **corrgram** (Friendly, 2002) is a visual display of a correlation matrix, where the correlation can be rendered in a variety of ways to show the direction and magnitude: circular “pac-man” (or pie) symbols, ellipses, colored vars or shaded rectangles, as shown in Figure 3.33.

Another aspect is that of **effect ordering** (Friendly & Kwan, 2003), ordering the levels of factors and variables in graphic displays to make important features most apparent. For variables, this means that we can arrange the variables in a matrix-like display in such a way as to make the pattern of relationships easiest to see. Methods to achieve this include using principal components and cluster analysis to put the most related variables together as described in Chapter 4.

In R, these diagrams can be created using the `corrgram` (Wright, 2021) and `corrplot` (Wei & Simko, 2024) packages, with different features. `corrgram::corrgram()` is closest to Friendly (2002), in that it allows different rendering functions for the lower, upper and diagonal panels as illustrated in Figure 3.33. For example, a corrgram similar to Figure 3.32 can be produced as follows (not shown here):

```
crime |>
  select(where(is.numeric)) |>
  corrgram(lower.panel = panel.ellipse,
            upper.panel = panel.ellipse,
            diag.panel = panel.density)
```

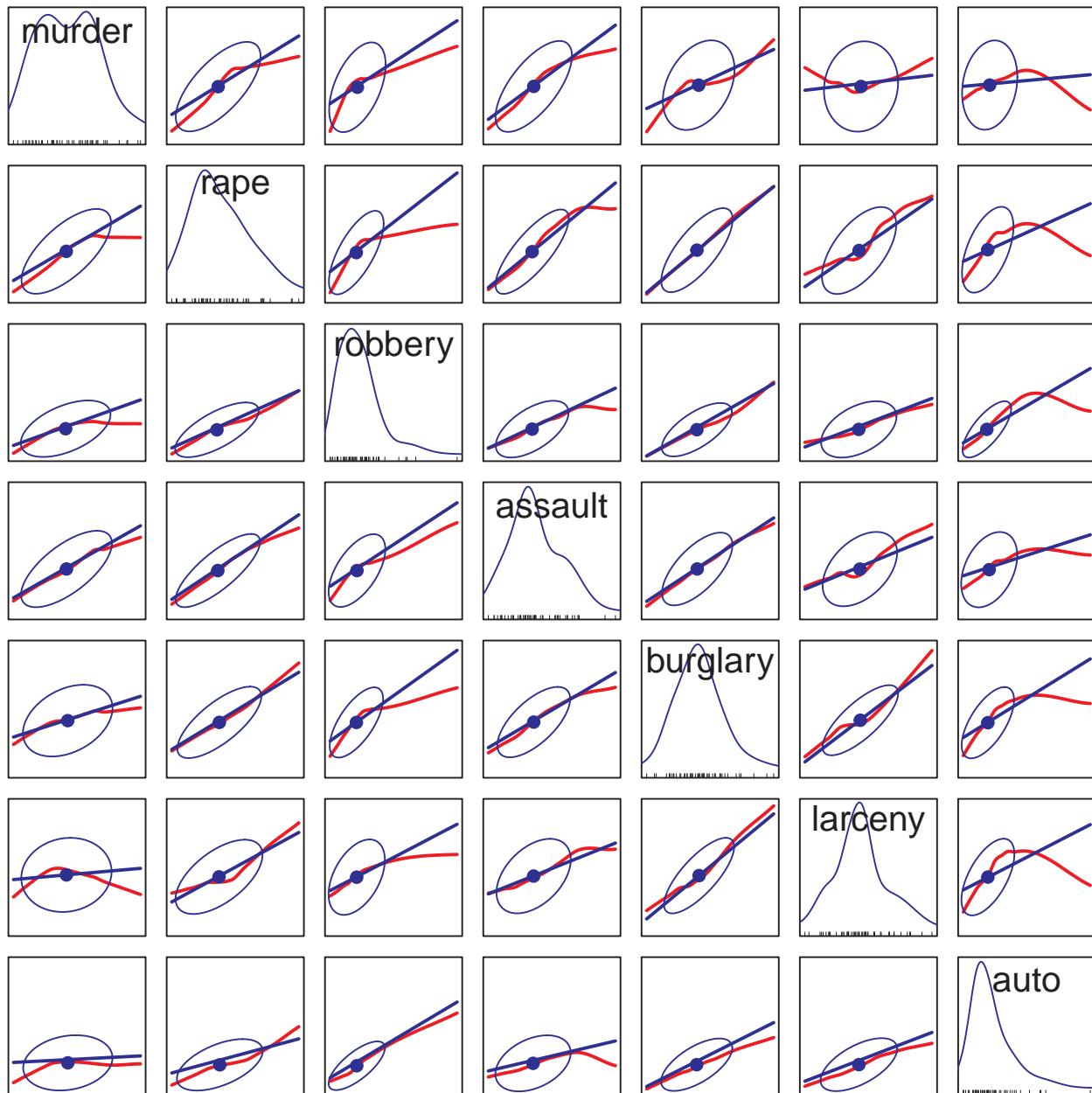


Figure 3.32: Visual thinning: Scatterplot matrix of the crime data, showing only high-level summaries of the linear and nonlinear relations between each pair of variables.

With the `corrplot` package, `corrplot()` provides the rendering methods `c("circle", "square", "ellipse", "number", "shade", "color", "pie")`, but only one can be used at a time. The function `corrplot.mixed()` allows different options to be selected for the lower and upper triangles. The iconic rendering shape is colored with a gradient in relation to the correlation value. For comparison, Figure 3.34 uses ellipses below the diagonal and filled pie charts below the diagonal using a gradient of the fill color in both cases.

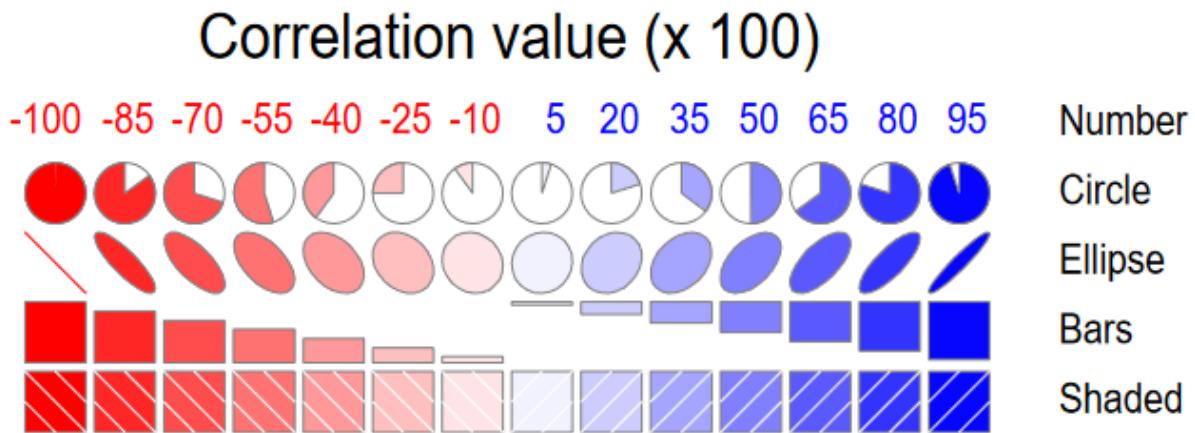


Figure 3.33: Corrgrams: Some renderings for the value of a correlation in a corrgram display, conveying sign and magnitude in different ways.

```
crime.cor <- crime |>
  dplyr::select(where(is.numeric)) |>
  cor()

corrplot.mixed(crime.cor,
  lower = "ellipse",
  upper = "pie",
  tl.col = "black",
  tl.srt = 0,
  tl.cex = 1.25,
  addCoef.col = "black",
  addCoefasPercent = TRUE)
```

The combination of renderings shown in Figure 3.34 is instructive. Small differences among correlation values are easier to see with the pie symbols than with the ellipses; for example, compare the values for murder with larceny and auto theft in row 1, columns 6–7 with those in column 1, rows 6–7, where the former are easier to distinguish. The shading color adds another visual cue.

The variables in Figure 3.32 and Figure 3.34 are arranged by their order in the dataset, which is not often the most useful. A better idea is to arrange the variables so that the most highly correlated variables are adjacent.

A general method described in Section 4.5 orders the variables according to the angles of the first two eigenvectors from a principal components analysis (PCA) around a unit circle. The function `corrMatOrder()` provides several methods (`order = c("AOE", "FPC", "hclust", "alphabet")`) for doing this, and PCA ordering is `order = "AOE"`. Murder and auto theft are still first and last, but some of the intermediate crimes have been rearranged.

```
ord <- corrMatOrder(crime.cor, order = "AOE")
rownames(crime.cor)[ord]
#> [1] "murder"    "assault"   "rape"      "robbery"   "burglary"
#> [6] "larceny"   "auto"
```

Using this ordering in `corrplot()` produces Figure 3.35.

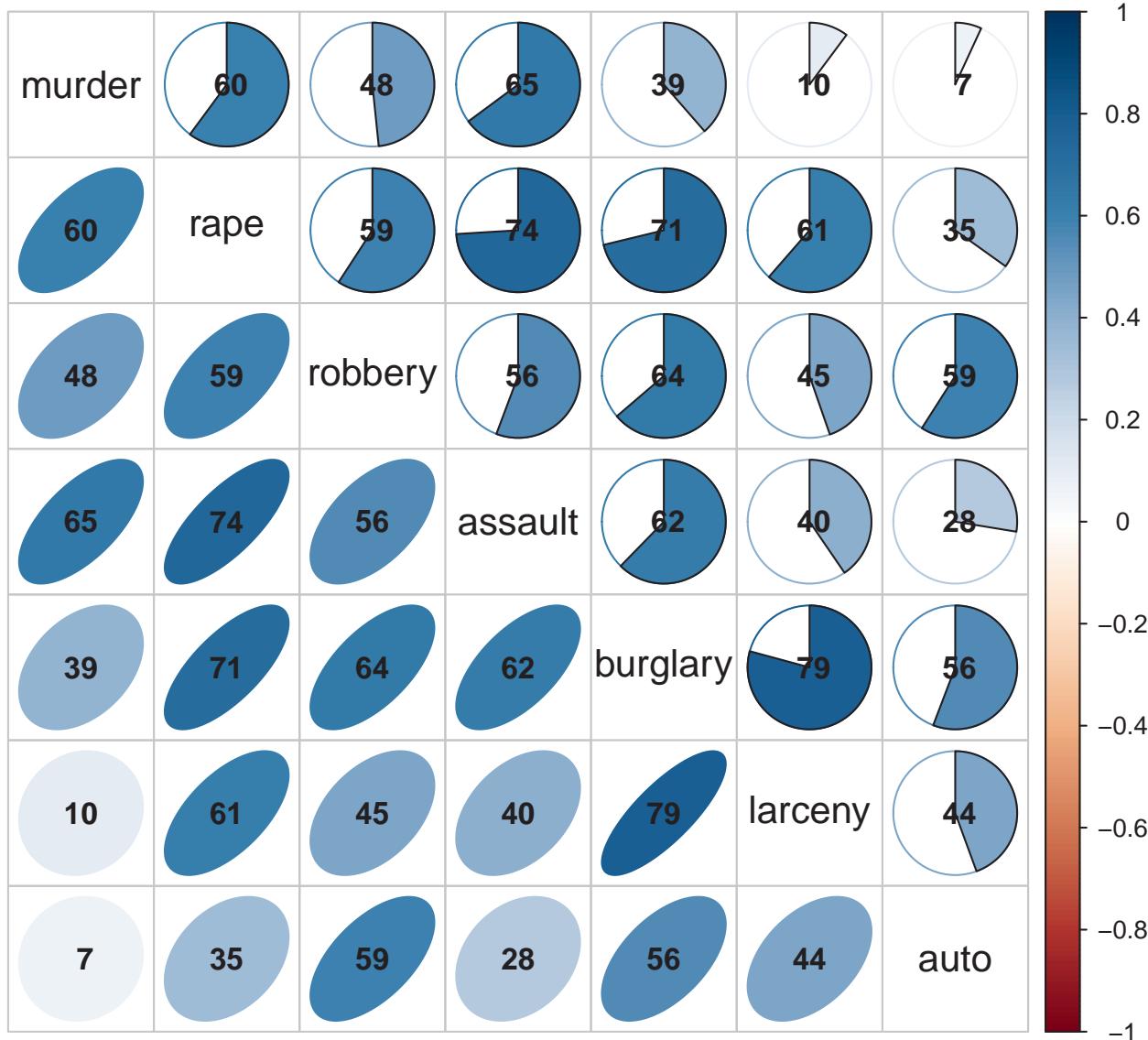


Figure 3.34: Mixed corrplot of the `crime` data, showing the correlation between each pair of variables with an ellipse (lower) and a pie chart symbol (upper), all shaded in proportion to the correlation value, also shown numerically.

```
corrplot.mixed(crime.cor,
  order = "AOE",
  lower = "ellipse",
  upper = "ellipse",
  tl.col = "black",
  tl.srt = 0,
  tl.cex = 1.25,
  addCoef.col = "black",
  addCoefasPercent = TRUE)
```

In this case, where the correlations among the crime variables are all positive, the effect of variable re-ordering is subtle, but note that there is now a slightly pronounced pattern of highest correlations near the diagonal,

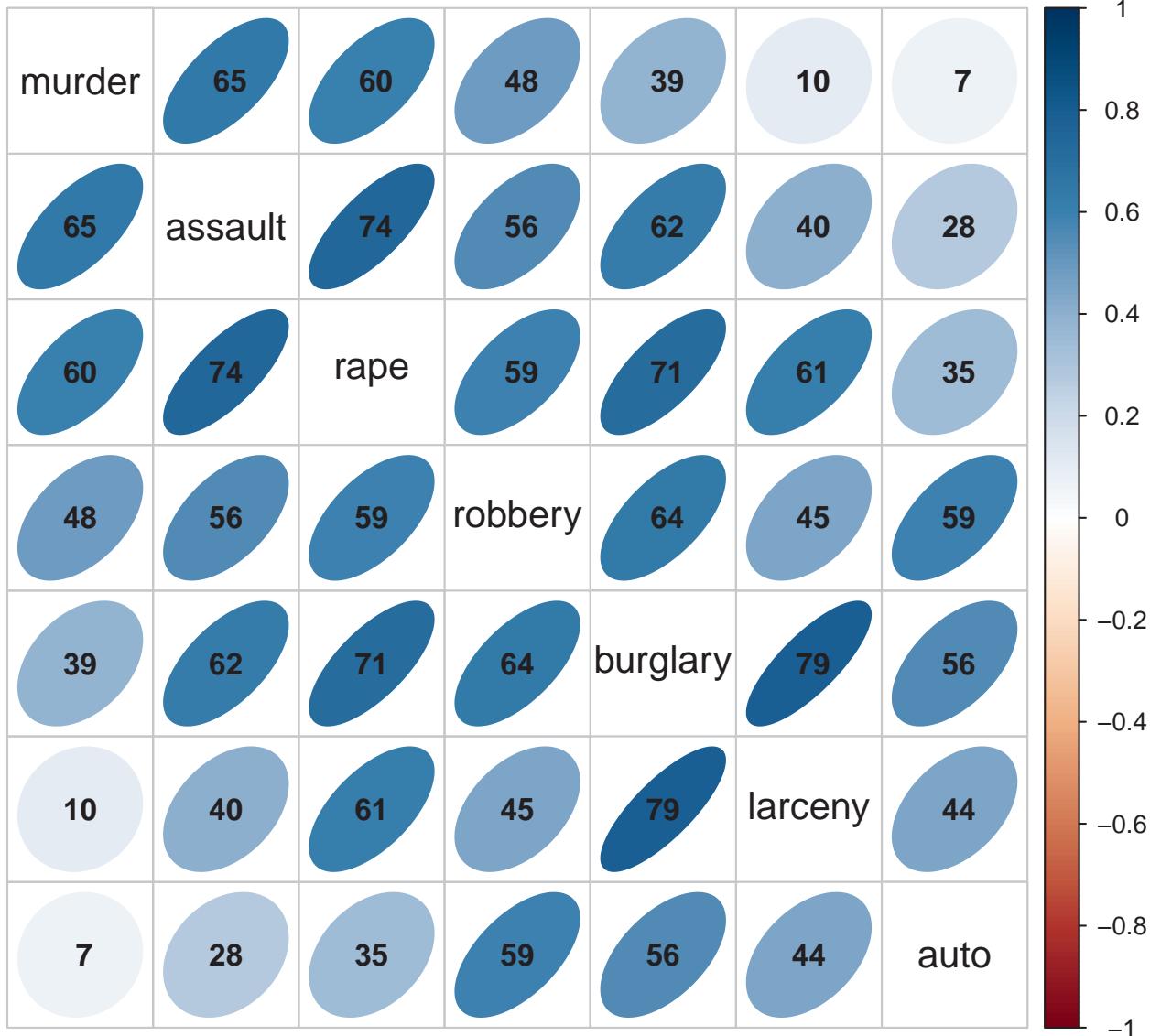


Figure 3.35: Corrplot of the `crime` data with the variables reordered according to the angles of variable eigenvectors. Correlations are rendered with ellipses shaded in proportion to their magnitude.

and decreasing away from the diagonal. Figure 4.25 and Figure 4.27 in Section 4.5 provide a more dramatic example of variable ordering using this method.

Variations of corrgrams are worthy replacements for a numeric table of correlations, which are often presented in publications only for archival value. Including the numeric value (rounded here, for presentation purposes), makes this an attractive alternative to boring tables of correlations.

3.12 Generalized pairs plots

When a dataset contains one or more discrete variables, the traditional pairs plot cannot cope, because the discrete categories would plot as many overlaid points. This cannot be represented using only color and/or point symbols in a meaningful scatterplot.

But the associations between categorical variables in a frequency table *can* be shown in *mosaic displays* (Friendly, 1994), using an array of tiles whose areas are depict the cell frequencies. For an n -way frequency, an analog of the scatterplot matrix uses mosaic plots for each pair of variables. The **vcd** package (Meyer et al., 2024) implements very general `pairs()` methods for "table" objects and **vcdExtra** (Friendly, 2025b) extends this to wide classes of loglinear models (Friendly, 1999). See Friendly (1999) and my book *Discrete Data Analysis with R* (Friendly & Meyer, 2016) for mosaic plots and mosaic matrices.

For example, we can tabulate the distributions of penguin species by sex and the island where they were observed using `xtabs()`. `ftable()` prints this three-way table more compactly. (In this example, and what follows in the chapter, I've changed the labels for sex from ("f", "m") to ("Female", "Male").)

```
# use better labels for sex
peng <- peng |>
  mutate(sex = factor(sex, labels = c("Female", "Male")))
peng.table <- xtabs(~ species + sex + island, data = peng)

ftable(peng.table)
#>           island Biscoe Dream Torgersen
#> species   sex
#> Adelie     Female      22    27     24
#>          Male       22    28     23
#> Chinstrap  Female      0    34     0
#>          Male       0    34     0
#> Gentoo    Female      58     0     0
#>          Male       61     0     0
```

We can see immediately that the penguin species differ by island: only Adelie were observed on all three islands; Biscoe Island had no Chinstraps and Dream Island had no Gentoos.

`vcd::pairs()` produces all pairwise mosaic plots, as shown in Figure 3.36. The diagonal panels show the one-way frequencies by width of the divided bars. Each off-diagonal panel shows the bivariate counts, breaking down each column variable by splitting the bars in proportion to a second variable. Consequently, the frequency of each cell is represented by its' area. The purpose is to show the **pattern of association** between each pair, and so, the tiles in the mosaic are shaded according to the signed standardized residual, $d_{ij} = (n_{ij} - \hat{n}_{ij})/\sqrt{\hat{n}_{ij}}$ in a simple $\chi^2 = \sum_{ij} d_{ij}^2$ test for association— blue where the observed frequency n_{ij} is significantly greater than expected \hat{n}_{ij} under independence, and red where it is less than expected. The tiles are unshaded when $|d_{ij}| < 2$.

```
library(vcd)
pairs(peng.table, shade = TRUE,
      lower_panel_args = list(labeling = labeling_values()),
      upper_panel_args = list(labeling = labeling_values()))
```

The shading patterns in cells (1,3) and (3,1) of Figure 3.36 show what we've seen before in the table of frequencies: The distribution of species varies across island because on each island one or more species did not occur. Row 2 and column 2 show that sex is nearly exactly proportional among species and islands,

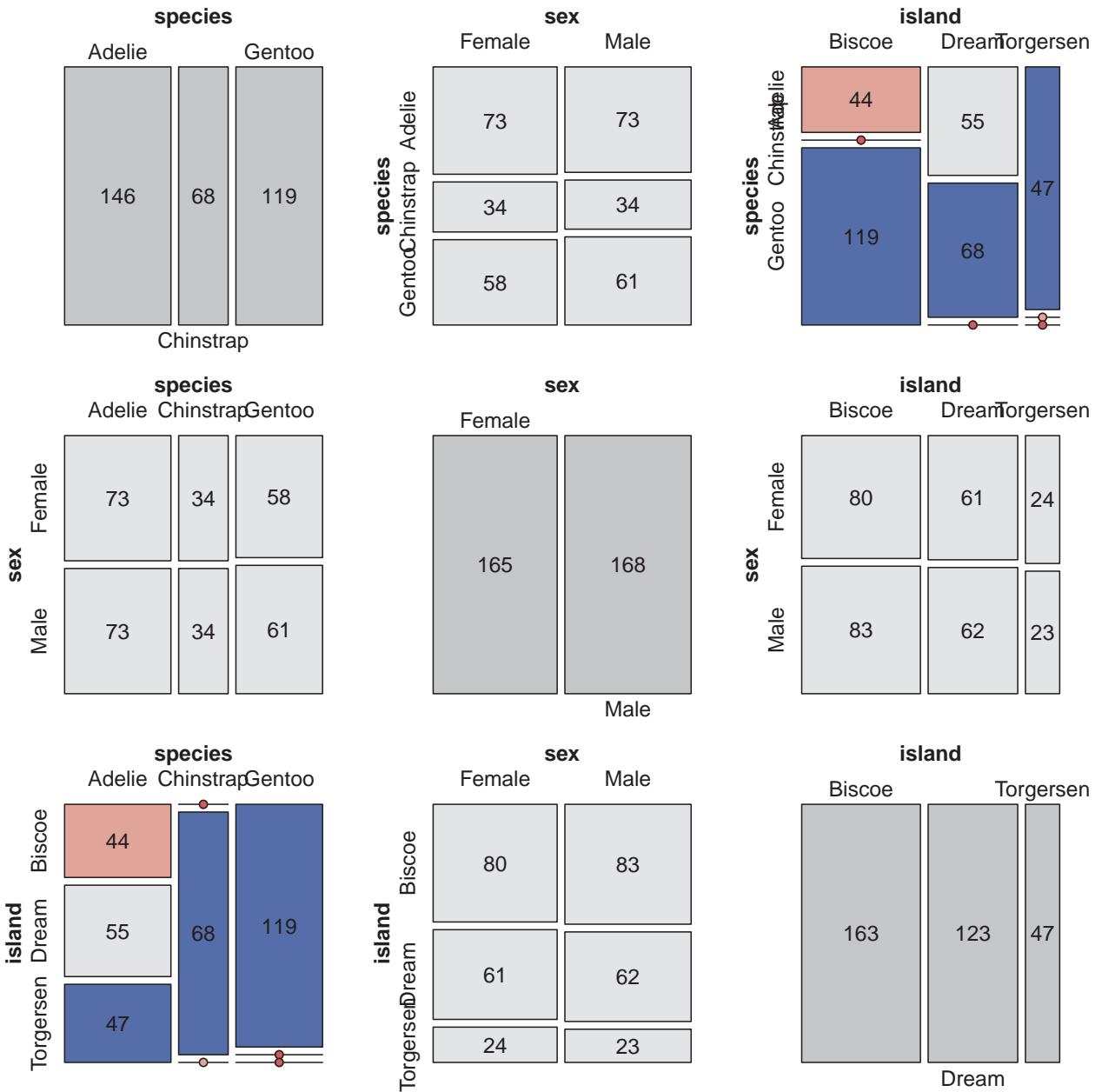


Figure 3.36: Mosaic pairs plot for the combinations of species, sex and island. Diagonal plots show the marginal frequency of each variable by the width of each rectangle. Off-diagonal mosaic plots subdivide by the conditional frequency of the second variable, shown numerically in the tiles.

indicating independence, $\text{sex} \perp \{\text{species}, \text{island}\}$. More importantly, mosaic pairs plots can show, at a glance, all (bivariate) associations among multivariate categorical variables.

The next step, by John Emerson and others (Emerson et al., 2013) was to recognize that combinations of continuous and discrete, categorical variables could be plotted in different ways.

- Two continuous variables can be shown as a standard scatterplot of points and/or bivariate density contours, or simply by numeric summaries such as a correlation value;
- A pair of one continuous and one categorical variable can be shown as side-by-side boxplots or violin plots, histograms or density plots;

- Two categorical variables could be shown in a mosaic plot or by grouped bar plots.

In the `ggplot2` framework, these displays are implemented using the `ggpairs()` function from the `GGally` package (Schloerke et al., 2025) . This allows different plot types to be shown in the lower and upper triangles and in the diagonal cells of the plot matrix. As well, aesthetics such as color and shape can be used within the plots to distinguish groups directly. As illustrated below, you can define custom functions to control exactly what is plotted in any panel.

The basic, default plot shows scatterplots for pairs of continuous variables in the lower triangle and the values of correlations in the upper triangle. A combination of a discrete and continuous variables is plotted as histograms in the lower triangle and boxplots in the upper triangle. Figure 3.37 includes `sex` to illustrate the combinations.

```
ggpairs(peng, columns=c(3:6, 7),
        aes(color=species, alpha=0.5),
        progress = FALSE) +
  theme_penguins() +
  theme(axis.text.x = element_text(angle = -45))
```

To my eye, printing the values of correlations in the upper triangle is often a waste of graphic space. But in this example the correlations show something peculiar and interesting if you look closely: In all pairs among the penguin size measurements, there are positive correlations within each species, as we can see in Figure 3.31. Yet, in three of these panels, the overall correlation ignoring species is negative. For example, the overall correlation between bill depth and flipper length is $r = -0.579$ in row 2, column 3; the scatterplot in the diagonally opposite cell, row 3, column 2 shows the data. These cases, of differing signs for an overall correlation, ignoring a group variable and the within group correlations are examples of **Simpson's Paradox**, explored later in Section 6.4.2.

The last row and column, for `sex` in Figure 3.37, provides an initial glance at the issue of sex differences among penguin species that motivated the collection of these data. We can go further by also examining differences among species and island, but first we need to understand how to display exactly what we want for each pairwise plot.

`ggpairs()` is extremely general in that for each of the `lower`, `upper` and `diag` sections you can assign any of a large number of built-in functions (of the form `ggally_NAME`), or your own custom function for what is plotted, depending on the types of variables in each plot.

- `continuous`: both X and Y are continuous variables, supply this as the NAME part of a `ggally_NAME()` function or the name of a custom function;
- `combo`: one X or Y variable is discrete while the other is continuous, using the same convention;
- `discrete`: both X and Y are discrete variables.

The defaults, which were used in Figure 3.37, are:

```
upper = list(continuous = "cor",           # correlation values
            combo = "box_no_facet",      # boxplots
            discrete = "count")        # rectangles ~ count
lower = list(continuous = "points",         # just data points
            combo = "facethist",       # faceted histograms
            discrete = "facetbar")     # faceted bar plots
diag  = list(continuous = "densityDiag",    # density plots
            discrete = "barDiag")      # bar plots
```

Thus, `ggpairs()` uses `ggally_cor()` to print the correlation values for pairs of continuous variables in the upper triangle, and uses `ggally_points()` to plot scatterplots of points in the lower portion. The

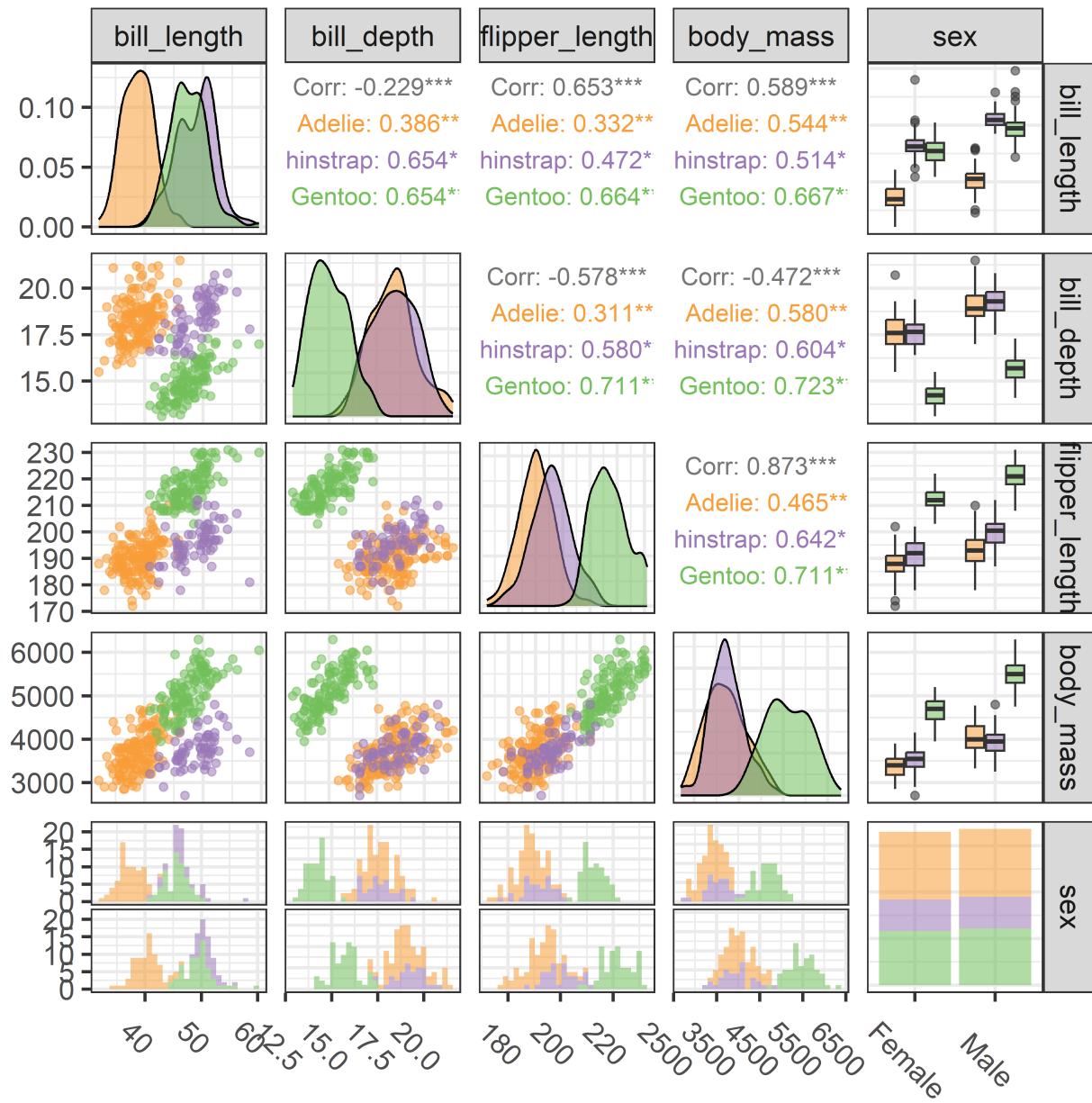


Figure 3.37: Basic `ggpairs()` plot of penguin size variables and sex, stratified by species.

diagonal panels as shown as density plots (`ggally_densityDiag()`) for continuous variables but as bar plots (`ggally_barDiag()`) for discrete factors.

See the vignette, [ggally_plots](#) for an illustrated list of available high-level plots. For our purpose here, which is to illustrate enhanced displays, note that for scatterplots of continuous variables, there are two functions which plot the points and also add a smoother, `_lm` or `_loess`.

```
ls(getNamespace("GGally")) |>
  stringr::str_subset("^ggally_smooth_")
#> [1] "ggally_smooth_lm"    "ggally_smooth_loess"
```

A customized display for scatterplots of continuous variables can be any function that takes `data` and `mapping` arguments and returns a "ggplot" object. The `mapping` argument supplies the aesthetics, e.g., `aes(color=species, alpha=0.5)`, but only if you wish to override what is supplied in the `ggpairs()` call.

Here is a function, `my_panel()` that plots the data points, regression line and loess smooth:

```
my_panel <- function(data, mapping, ...){
  p <- ggplot(data = data, mapping = mapping) +
    geom_point() +
    geom_smooth(method=lm, formula = y ~ x, se = FALSE, ...) +
    geom_smooth(method=loess, formula = y ~ x, se = FALSE, ...)
  p
}
```

For this example, I want only simple summaries of for the scatterplots, so I don't want to plot the data points, but do want to add the regression line and a data ellipse.

```
my_panel1 <- function(data, mapping, ...){
  p <- ggplot(data = data, mapping = mapping) +
    geom_smooth(method=lm, formula = y ~ x, se = FALSE, ...) +
    stat_ellipse(geom = "polygon", level = 0.68, ...)
  p
}
```

Then, to show what can be done, Figure 3.38 uses `my_panel1()` for the scatterplots in the 4 x 4 block of plots in the upper left. The combination of the continuous body size measures and the discrete factors `species`, `island` and `sex` are shown in upper triangle by boxplots but by faceted histograms in the lower portion. The factors are shown as rectangles with area proportional to count (poor-man's mosaic plots) above the diagonal and as faceted bar plots below.

```
ggpairs(peng, columns=c(3:6, 1, 2, 7),
        mapping = aes(color=species, fill = species, alpha=0.2),
        lower = list(continuous = my_panel1),
        upper = list(continuous = my_panel1),
        progress = FALSE) +
  theme_penguins() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  theme(axis.text.x = element_text(angle = -45))
```

There is certainly a lot going on in Figure 3.38, but it does show a high-level overview of all the variables (except `year`) in the penguins dataset. It is probably easiest to "read" this figure by focusing on the four blocks for the combinations of 4 continuous and 3 categorical measures. In the upper left block, visual thinning of the scatterplots, showing only the data ellipses and regression lines gives a simple view as it did in Figure 3.31.

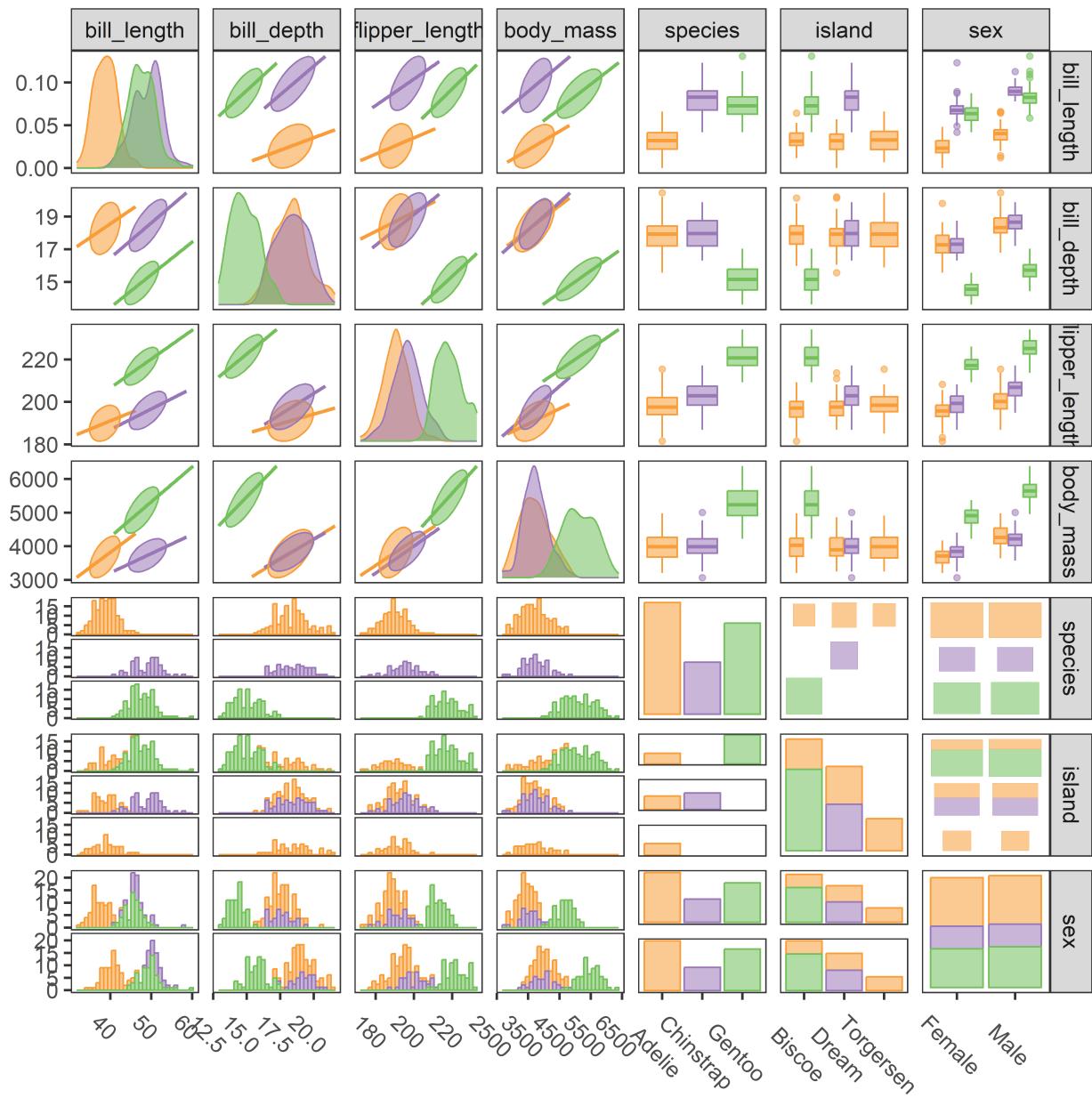


Figure 3.38: Customized `ggpairs()` plot of penguin size variables, together with species, island and sex.

3.13 Parallel coordinate plots

As we have seen above, scatterplot matrices and generalized pairs plots extend data visualization to multivariate data, but these variables share one 2D space, so resolution decreases as the number of variable increase. You need a very large screen or sheet of paper to see more than, say 5-6 variables with any clarity.

Parallel coordinate plots are an attractive alternative, with which we can visualize an arbitrary number of variables to get a visual summary of a potentially high-dimensional dataset, and perhaps recognize outliers and clusters in the data in a different way. In these plots, each variable is shown on a separate, parallel axis.

A multivariate observation is then plotted by connecting their respective values on each axis with lines across all the axes.

The geometry of parallel coordinates is interesting, because what is a point in n -dimensional (Euclidean) *data* space becomes a line in the *projective* parallel coordinate space with n axes, and vice-versa: lines in parallel coordinate space correspond to points in data space. Thus, a collection of points in data space map to lines that intersect in a point in projective space. What this does is to map n -dimensional relations into 2D patterns we can see in a parallel coordinates plot.

History Corner

Those who don't know history are doomed to plagiarize it —The author

The theory of projective geometry originated with the French mathematician Maurice d'Ocagne (1885) who sought a way to provide graphic calculation of mathematical functions with alignment diagrams or *nomograms* using parallel axes with different scales. A three-variable equation, for example, could be solved using three parallel axes, where known values could be marked on their scales, a line drawn between them, and an unknown read on its scale at the point where the line intersects that scale.

Henry Gannet (1880), in work preceding the *Statistical Atlas of the United States* for the 1890 Census (Gannett, 1898), is widely credited with being the first to use parallel coordinates plots to show data, in his case, to show the [rank ordering of US states](#) by 10 measures including population, occupations, wealth, manufacturing, agriculture and so on.

However, both d'Ocagne and Gannet were far preceded in this by Andre-Michel Guerry (1833) who used this method to show how the rank order of various crimes changed with age of the accused. See Friendly (2022), Figure 7 for his version and for an appreciation of the remarkable contributions of this amateur statistician to the history of data visualization.

The use of parallel coordinates for display of multidimensional data was rediscovered by Alfred Inselberg (1985) and extended by Edward Wegman (1990), neither of whom recognized the earlier history. Somewhat earlier, David Andrews (1972) proposed mapping multivariate observations to smooth Fourier functions composed of alternating $\sin()$ and $\cos()$ terms. And in my book, *SAS System for Statistical Graphics* (Friendly, 1991), I implemented what I called *profile plots* without knowing their earlier history as parallel coordinate plots.

Parallel coordinate plots present a challenge for graphic developers, in that they require a different way to think about plot construction for multiple variables, which can be quantitative, as in the original idea, or categorical factors, all to be shown along parallel axes.

Here, I use the `ggpcp` package (Hofmann et al., 2022), best described in VanderPlas et al. (2023), who also review the modern history.⁷ This takes some getting used to, because they develop `pcp_*` extensions of the `ggplot2` grammar of graphics framework to allow:

- `pcp_select()`: selections of the variables to be plotted and their horizontal order on parallel axes,
- `pcp_scale()`: methods for scaling of the variables to each axis,
- `pcp_arrange()`: methods for breaking ties in factor variables to space them out.

Then, it provides `geom_pcp_*` functions to control the display of axes with appropriate aesthetics, labels for categorical factors and so forth. Figure 3.39 illustrates this type of display, using sex and species in addition to the quantitative variables for the penguin data.

```
peng |>
  pcp_select(bill_length:body_mass, sex, species) |>
  pcp_scale(method = "uniminmax") |>
  pcp_arrange() |>
```

⁷Other implementations of parallel coordinate plots in R include: `MASS:::parcoord()`, `GGally:::ggparcoord()` and `PairViz:::pcp()`. The `ggpcp` version used here is the most general.

```
ggplot(aes_pcp()) +
  geom_pcp_axes() +
  geom_pcp(aes(colour = species), alpha = 0.8, overplot = "none") +
  geom_pcp_labels() +
  scale_colour_manual(values = peng.colors()) +
  labs(x = "", y = "") +
  theme(axis.title.y = element_blank(), axis.text.y = element_blank(),
        axis.ticks.y = element_blank(), legend.position = "none")
```

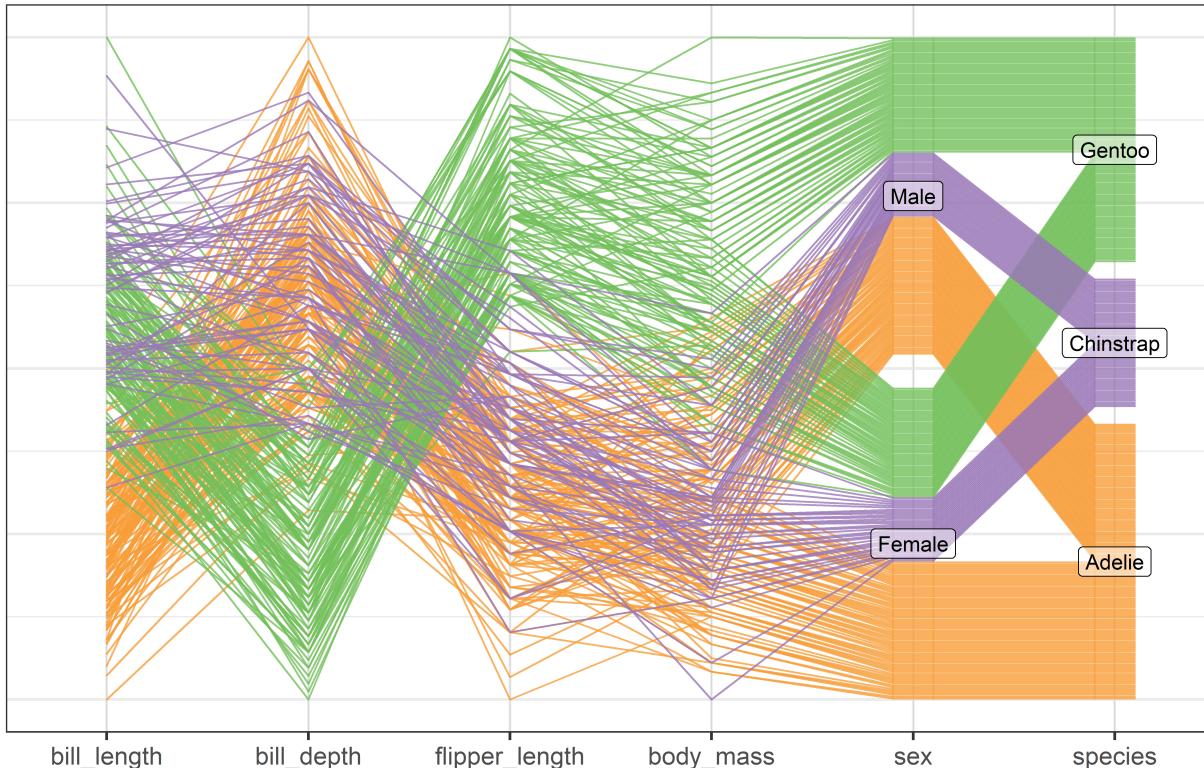


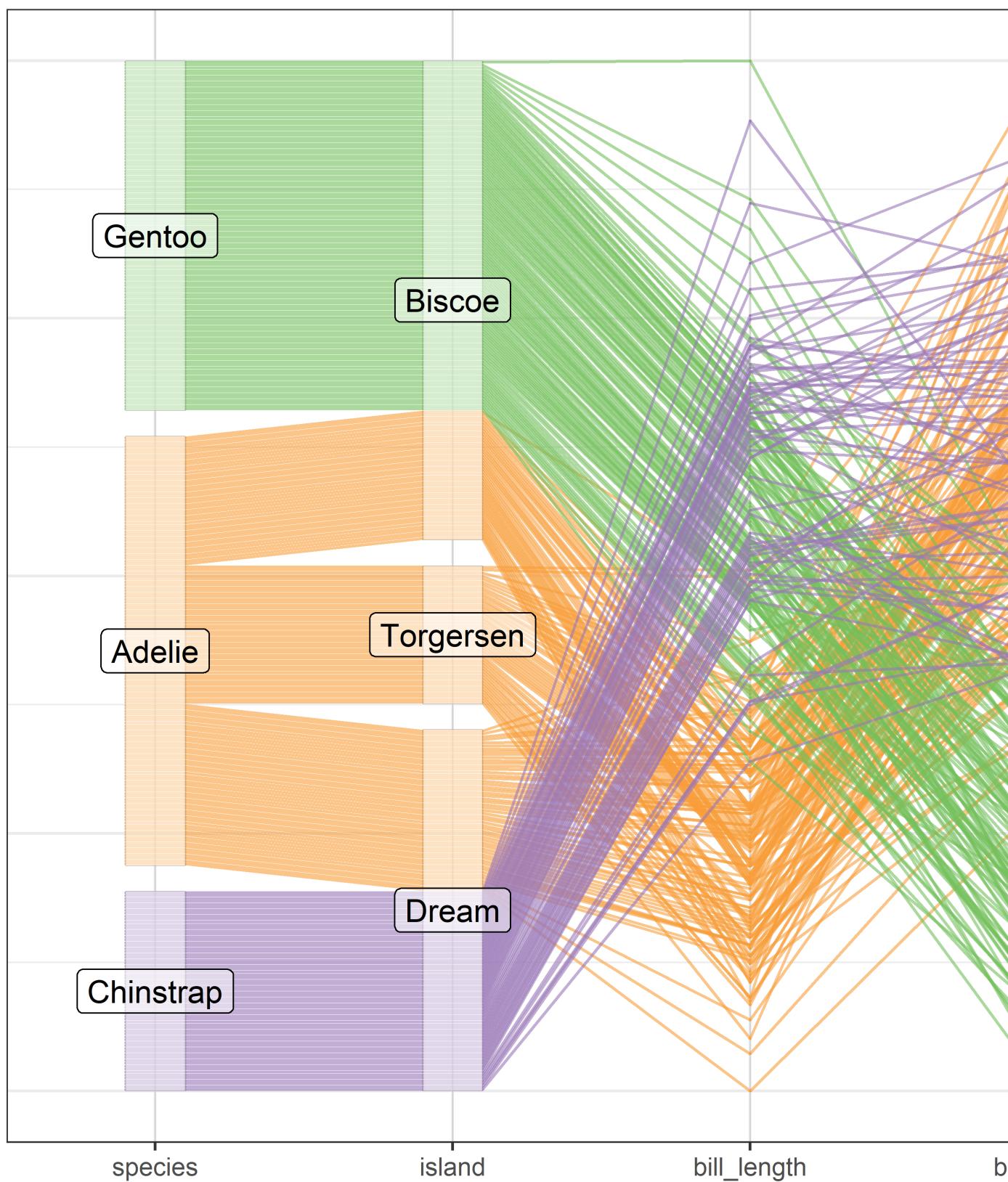
Figure 3.39: Parallel coordinates plot of penguin size variables, together with sex and species.

Rearranging the order of variables and the ordering of factor levels can make a difference in what we can see in such plots. For a simple example (following VanderPlas et al. (2023)), we reorder the levels of species and islands to make it clearer which species occur on each island.

```
peng1 <- peng |>
  mutate(species = factor(species, levels = c("Chinstrap", "Adelie", "Gentoo"))) |>
  mutate(island = factor(island, levels = c("Dream", "Torgersen", "Biscoe")))

peng1 |>
  pcp_select(species, island, bill_length:body_mass) |>
  pcp_scale() |>
  pcp_arrange(method = "from-left") |>
  ggplot(aes_pcp()) +
  geom_pcp_axes()
```

```
geom_pcp(aes(colour = species), alpha = 0.6, overplot = "none") +  
  geom_pcp_boxes(fill = "white", alpha = 0.5) +  
  geom_pcp_labels() +  
  scale_colour_manual(values = peng.colors()[c(2,1,3)]) +  
  theme_bw() +  
  labs(x = "", y = "") +  
  theme(axis.text.y = element_blank(),  
        axis.ticks.y = element_blank(),  
        legend.position = "none")
```



The order of variables in this plot emphasizes the relation between penguin species and the island where they were observed and then shows the values of the quantitative body size measurements. More generally, quantitative variables can, and probably should, be ordered to place the most highly correlated variables adjacently to minimize the degree of crossing lines from one variable to the next (Martí & Laguna, 2003). When variables are highly *negatively* correlated (such as `bill_depth` and `flipper_length` here), crossings can be reduced simply by reversing the scale of one of the variables, e.g., by plotting `-bill_depth`.

3.14 Animated tours

In the mid 17th to early 19th-century the **Grand Tour** became a coming-of-age custom for young Europeans (mainly British nobility and landed gentry) of sufficient rank and means to undertake a journey to the principal sites of Europe (Paris, Geneva, Rome, Athens, ...) to complete their education by learning something of the cultural legacies in history, art, and music from antiquity to the Renaissance. Thereby, they could gain a wider appreciation of history and be prepared to play a role in polite society or in their chosen endeavors.

Travels in high-dimensional data space might be less thrilling than a journey from London through Paris and Milan to Rome. Yet, in both cases it is useful to think of the path taken, and what might be seen along the way. But there are different kinds of tours. We might simply take a meandering tour, exploring all the way, or want to plan a tour to see the most interesting sites in travel or have a tour guided by an expert. Similarly in data space, we might travel randomly to see what we can find or be guided to find interesting features such as clusters, outliers or non-linear relations in data.

Following the demonstration in PRIM-9 (?@sec-discoveries) of exploring multidimensional data space by rotation Asimov (1985) developed the idea of the *grand tour*, a computer method for viewing multivariate statistical data via orthogonal projections onto an animated sequence of low-dimensional subspaces, like a movie. In contrast to a scatterplot matrix which shows a static view of a data cloud projected onto all pairwise variable axes, a statistical tour is like the view of an eye moving smoothly in high-dimensional space, capturing what it sees from a given location onto the 2-d plane of the computer screen.

More generally, statistical tours are a type of dynamic projections onto orthogonal axes (called a *basis*) that embed data in a p -dimensional space into a d -dimensional viewing subspace. Typically, $d = 2$, and the result is displayed as scatterplots, together with vectors representing the projections of the data variables in this space. But the projected data can be rendered in 1-d as densities or histograms, or in other number of dimensions as glyphs, or even as parallel coordinate plots. The essential idea is that we can define, and animate, a *tour path* as a smooth sequence of such projections over small changes to the projection basis, which gives the orientation of the data in the viewing space.

3.14.1 Projections

The idea of a projection is fundamental to touring methods and other visualizations of high-D data, so it is useful to understand what a projection is. Quite simply, you can think of a projection as the shadow of an object or cloud of points. This is nicely illustrated by the cover image (Figure 3.40) used for Douglas Hofstadter's (1979) *Gödel, Bach and Escher* which shows 3D solid shapes illuminated by light sources so their shadows form the letters G, B and E projected onto the planes formed by pairs of the three coordinate axes. The set of three 2D views is essentially the same that we see in a scatterplot matrix, where a 3D dataset is portrayed by the set of shadows of the points on planes formed by pairs of coordinate axes.

In the simplest case, a data point $\mathbf{x} = (x_1, x_2)$ in two dimensions can be represented geometrically as a vector from the origin as shown in Figure 3.41. This point can be projected on any one-dimensional axis \mathbf{p} by dropping a line perpendicular to \mathbf{p} , which is the idea of a shadow. Mathematically, this is calculated as the product $\mathbf{x}^T \mathbf{p} = x_1 p_1 + x_2 p_2$ and suitably normalized to give the correct length. ...

More generally, a projection of an $(n \times p)$ data matrix \mathbf{X} representing n observations in p dimensions onto a

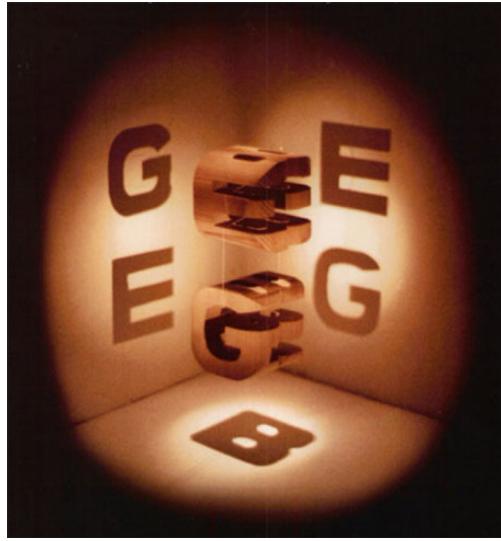


Figure 3.40: The cover image from Hofstadter (1979) illustrates how projections are shadows of an object cast by a light from a given direction.

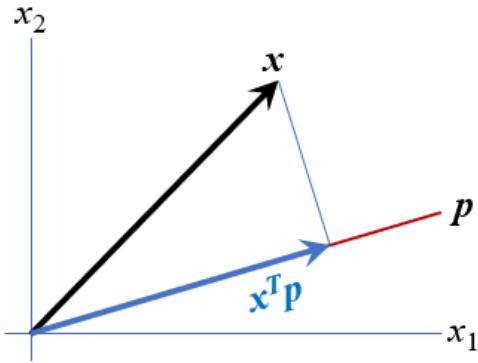


Figure 3.41: Projection of a point \mathbf{x} onto a direction or axis \mathbf{p} .

d -dimensional viewing space $\mathbf{Y}_{n \times d}$ is represented by a $p \times d$ projection matrix \mathbf{P} as $\mathbf{Y} = \mathbf{X}\mathbf{P}$, where the columns of \mathbf{P} are orthogonal and of unit length, i.e., $\mathbf{P}^T\mathbf{P} = \mathbf{I}_{(d \times d)}$.

For example, to project a data matrix \mathbf{X} in three dimensions onto a 2D plane, we would multiply it by a (3×2) orthonormal matrix \mathbf{P} . The matrix \mathbf{P}_1 below simply selects the first two columns of \mathbf{X} .⁸

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 10 \\ 0 & 10 & 0 \\ 0 & 10 & 10 \\ 10 & 0 & 0 \\ 10 & 0 & 10 \\ 10 & 10 & 0 \\ 10 & 10 & 10 \end{bmatrix}_{8 \times 3} ; \mathbf{P}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}_{3 \times 2} \Rightarrow \mathbf{Y} = \mathbf{X} \mathbf{P}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 10 \\ 0 & 10 \\ 10 & 0 \\ 10 & 0 \\ 10 & 10 \\ 10 & 10 \end{bmatrix}_{8 \times 2}$$

An oblique projection using all three dimensions is given by \mathbf{P}_2 below. This produces a new 2D view in \mathbf{Y} :

⁸This example was modified from one used by D. Cook et al. (2008).

$$\mathbf{P}_2 = \begin{bmatrix} 0.71 & -0.42 \\ 0.71 & 0.42 \\ 0 & 0.84 \end{bmatrix}_{3 \times 2} \Rightarrow \mathbf{Y} = \mathbf{X} \mathbf{P}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 8.4 \\ 7.1 & 4.2 \\ 7.1 & 12.6 \\ 7.1 & -4.2 \\ 7.1 & 4.2 \\ 14.2 & 0 \\ 14.2 & 8.4 \end{bmatrix}$$

The columns in \mathbf{Y} are simply the linear combinations of those of \mathbf{X} using the weights in each column of \mathbf{P}_2

$$\begin{aligned}\mathbf{y}_1 &= 0.71\mathbf{x}_1 + 0.71\mathbf{x}_2 + 0\mathbf{x}_3 \\ \mathbf{y}_2 &= -0.42\mathbf{x}_1 + 0.42\mathbf{x}_2 + 0.84\mathbf{x}_3\end{aligned}$$

```
vals <- c(0, 10)
X <- expand.grid(x1 = vals, x2=vals, x3=vals) |> as.matrix()

# project on just x1, x2 plane
P1 <- rbind(diag(2), c(0,0))
Y1 <- X %*% P1

# oblique projection
P2 <- matrix(c(0.71, 0.71, 0, -0.42, .42, 0.84), ncol=2)
Y2 <- X %*% P2
```

In this example, the matrix \mathbf{X} consists of 8 points at the vertices of a cube of size 10, as shown in Figure 3.42 (a). The projections $\mathbf{Y}_1 = \mathbf{P}_1\mathbf{X}$ and $\mathbf{Y}_2 = \mathbf{P}_2\mathbf{X}$ are shown in panels (b) and (c). To make it easier to relate the points in different views, shapes and colors are assigned so that each point has a unique combination of these attributes.⁹

```
pch <- rep(15:18, times = 2)
colors <- c("red", "blue", "darkgreen", "brown")
col <- rep(colors, each = 2)
data.frame(X, pch, col)
#>   x1 x2 x3 pch      col
#> 1  0  0  0  15     red
#> 2 10  0  0  16     red
#> 3  0 10  0  17    blue
#> 4 10 10  0  18    blue
#> 5  0  0 10  15 darkgreen
#> 6 10  0 10  16 darkgreen
#> 7  0 10 10  17   brown
#> 8 10 10 10  18   brown
```

But, if we are traveling in the projection space of \mathbf{Y} , we need some signposts to tell us how the new dimensions relate to those of \mathbf{X} . The answer is provided simply by plotting the rows of \mathbf{P} as vectors, as shown in Figure 3.43. In these plots, each row of \mathbf{P}_1 and \mathbf{P}_2 appears as a vector from the origin. Its direction shows the contribution each of $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ make to the new coordinates \mathbf{y}_1 and \mathbf{y}_2 .

⁹Plot shapes given by `pch = 15:18` correspond to: filled square (15), filled circle (16), filled triangle point-up (17), filled diamond (18).

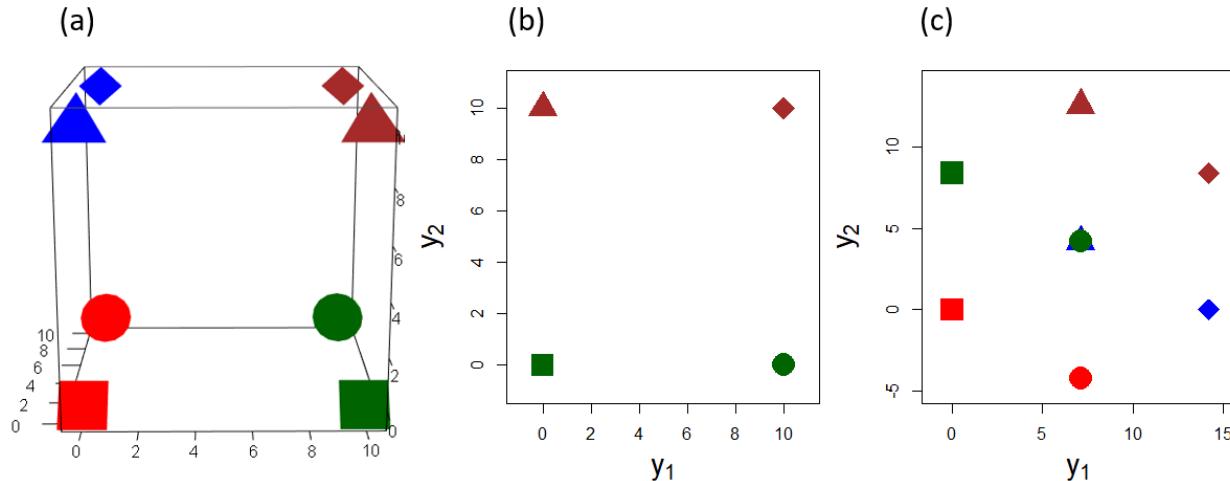


Figure 3.42: Projection example: (a) The 8 points in \mathbf{X} form a cube of size 10; (b) the projection by \mathbf{P}_1 is the view ignoring \mathbf{x}_3 (two points coincide at each vertex); (c) the projection by \mathbf{P}_2 is an oblique view.

In \mathbf{P}_1 , the projected variable y_1 is related only to \mathbf{x}_1 , while y_2 is related only to \mathbf{x}_2 ; \mathbf{x}_3 makes no contribution, and appears at the origin. However in the projection given by \mathbf{P}_2 , \mathbf{x}_1 and \mathbf{x}_2 make the same contribution to y_1 , while \mathbf{x}_3 has no contribution to that horizontal axis. The vertical axis, y_2 here is completely aligned with \mathbf{x}_3 ; \mathbf{x}_1 and \mathbf{x}_2 have vertical components that are half of that for \mathbf{x}_3 in absolute value.

```
library(matlib)
op <- par(mar=c(4, 5, 1, 1)+.1)
xlim <- ylim <- c(-1.1, 1.1)
axes.x <- c(-1, 1, NA, 0, 0)
axes.y <- c(0, 0, NA, -1, 1)
labs <- c(expression(x[1]), expression(x[2]), expression(x[3]))
plot(xlim, ylim, type = "n", asp=1,
      xlab = expression(y[1]), ylab = expression(y[2]),
      cex.lab = 1.8)
circle(0, 0, 1, col = adjustcolor("skyblue", alpha = 0.2))
lines(axes.x, axes.y, col = "grey")
vectors(P1, labels = labs, cex.lab = 1.8, lwd = 3, pos.lab = c(4, 2, 1))

plot(xlim, ylim, type = "n", asp=1,
      xlab = expression(y[1]), ylab = expression(y[2]),
      cex.lab = 1.8)
circle(0, 0, 1, col = adjustcolor("skyblue", alpha = 0.2))
lines(axes.x, axes.y, col = "grey")
vectors(P2, labels = labs, cex.lab = 1.8, lwd = 3)
par(op)
```

3.14.1.1 Vector lengths

In Figure 3.43, the **lengths** of the \mathbf{x} vectors reflect the relative degree to which each variable is represented in the space of the projection, and this is important for interpretation. For the \mathbf{P}_1 projection, \mathbf{x}_3 is of length 0, while \mathbf{x}_1 and \mathbf{x}_2 fill the unit circle. In the projection given by \mathbf{P}_2 , all three \mathbf{x} are approximately the same length.

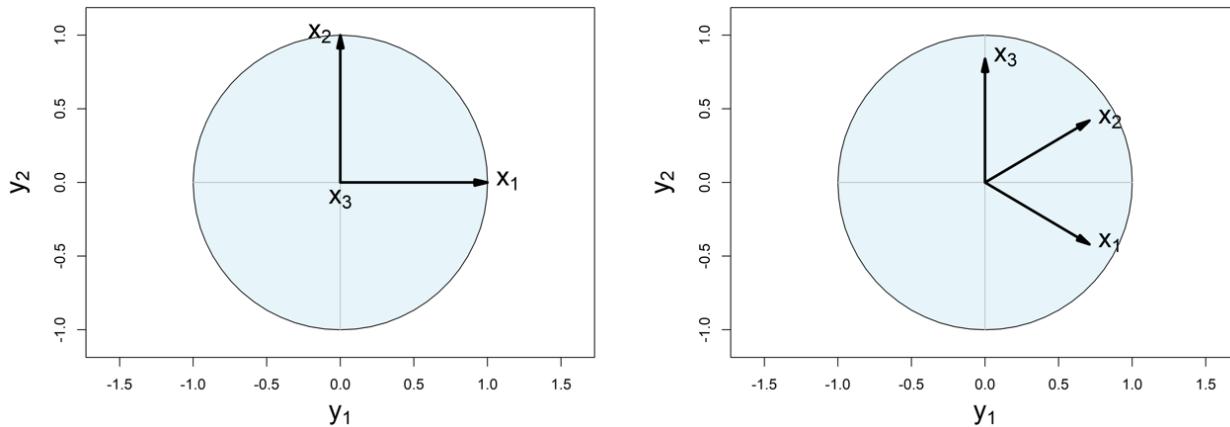


Figure 3.43: Variable vectors: Data variables viewed as vectors in the space of their projections. The angles of the \mathbf{x} vectors with respect to the \mathbf{y} coordinate axes show their relative contributions to each. The lengths of the \mathbf{x} vectors show the relative degree to which they are represented in the space of \mathbf{y} s. Left: the $\mathbf{P1}$ projection; right: the $\mathbf{P2}$ projection.

In algebra, the length of a vector \mathbf{x} is $\|\mathbf{x}\| = (\mathbf{x}^T \mathbf{x})^{1/2} = \sqrt{\sum x_i^2}$, the Euclidean distance of the tip of the vector from the origin. In R, we calculate the lengths of row vectors in a projection matrix by transposing and using `matlib::len()`.

```
P1 |> t() |> matlib::len()
#> [1] 1 1 0
P2 |> t() |> matlib::len()
#> [1] 0.825 0.825 0.840
```

3.14.1.2 Joint-views

To interpret such projections, we want to see **both** the projected data and the signposts that tell us where we are in relation to the original variables. To do this, we can overlay the variable vectors represented by the rows of the projection matrix \mathbf{P} onto plots like Figure 3.42 (b) and Figure 3.42 (c) to see how the axes in a projection relate to those in the data. To place these together on the same plot, we can either center the columns of \mathbf{Y} at their means or shift the the columns of \mathbf{P} to `colMeans(Y)`. It is only the directions of the vectors that matters, so we are free to scale their lengths by any convenient factor.

```
Y2s <- scale(Y2, scale=FALSE)           # center Y2
plot(Y2s, cex = 3,
      asp = 1,
      pch = pch, col = col,
      xlab = expression(y[1]), ylab = expression(y[2]),
      xlim = c(-10, 10), ylim = c(-10, 10), cex.lab = 1.8)
r <- 7
vecs <- (r*diag(3) %*% P2)
vectors(vecs, labels = labs, cex.lab = 1.8, lwd = 2)
vectors(-vecs, labels = NULL, lty = 1, angle = 1, col = "gray")
```

The plot in Figure 3.44 illustrates this, centering \mathbf{Y} , and multiplying the vectors in \mathbf{P} by 7. To check your understanding, try to see if you can relate what is shown in this plot to the 3D plot in Figure 3.42 (a).

The idea of viewing low-dimensional projections of data together with vectors representing the contributions of

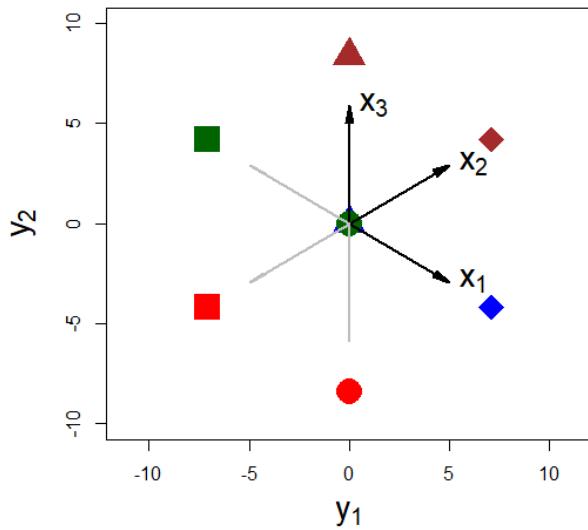


Figure 3.44: The \mathbf{P}_2 projection of the data showing vectors for the original variables in the space of \mathbf{Y} .

the original variables to the dimensions shown in a display is also the basis of **biplot** techniques (Section 4.3) we will use in relation to principal components analysis.

3.14.2 Touring methods

The trick of statistical touring methods is to generate a smooth sequence of interpolated projections $\mathbf{P}_{(t)}$ indexed by time t , $\mathbf{P}_{(1)}, \mathbf{P}_{(2)}, \mathbf{P}_{(3)}, \dots, \mathbf{P}_{(T)}$. This gives a path of views $\mathbf{Y}_{(t)} = \mathbf{XP}_{(t)}$, that can be animated in successive frames, as shown schematically in Figure 3.45.

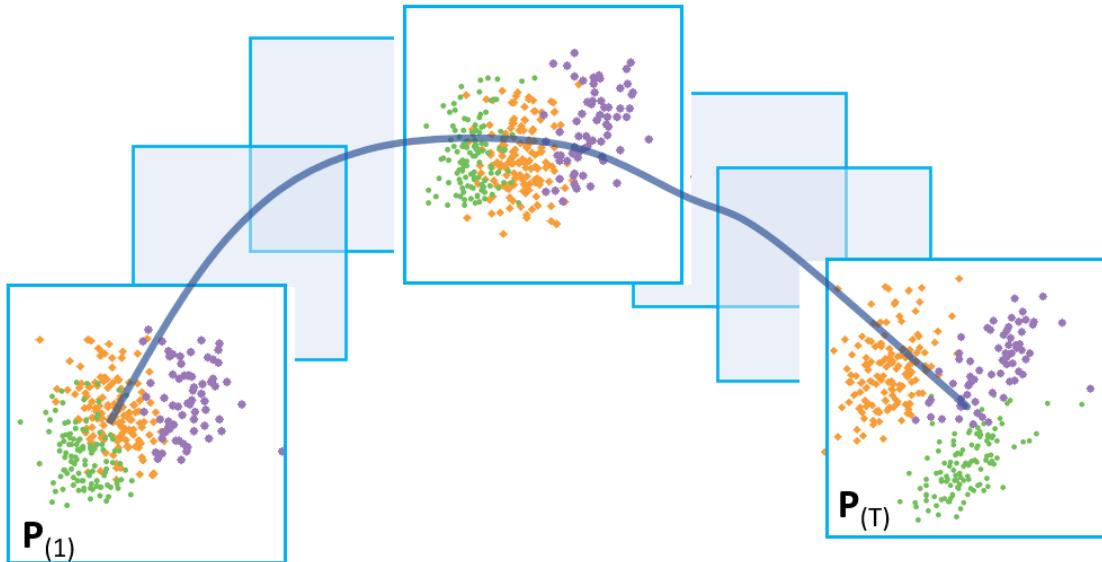


Figure 3.45: Interpolations: Illustration of a grand tour of interpolations of projection planes showing 2D scatterplots of the Penguin dataset. The sequence of views moves smoothly from an initial frame $\mathbf{P}(1)$ to a final frame $\mathbf{P}(T)$ where the penguin species are widely separated.

Asimov's (1985) original idea of the grand tour was that of a random path, picking orthogonal projections

$\mathbf{P}_{(i)}$ at random. Given enough time, the grand tour gives a space-filling path and would eventually show every possible projection of the data. But it does so smoothly, by interpolating from one projection to the next. In the travel analogy, the path by road from London to Paris might go smoothly through Kent to Dover, thence via Amiens and Beauvais before reaching Paris. By air, the tour would follow a smoother *geodesic* path, and this is what the grand tour does. The sense in watching an animation of a statistical grand tour is that of continuous motion. The grand tour algorithm is described in detail by Buja et al. (2005) and D. Cook et al. (2008).

3.14.2.1 Guided tours

The next big idea was that rather than traveling randomly in projection space one could take a *guided tour*, following a path that leads to “interesting projections”, such as those that reveal clusters, gaps in data space or outliers. This idea, called *projection pursuit* (D. Cook et al., 1995), works by defining a measure of interestingness of a data projection. In a guided tour, the next projection is chosen to increase that index, so over time the projection moves toward one that is maximizes that index.

In the time since Asimov (1985), there have been many implementations of touring visualization methods. XGobi (Swayne et al., 1998) for X-Windows displays on Linux systems provided a test-bed for dynamic, interactive graphic methods; its successor, GGobi (D. Cook & Swayne, 2007; Swayne et al., 2003) extended the range of touring methods to include a wider variety of projection pursuit indices.

3.14.2.2 tourr package

The current state of art is best captured in the `tourr` package for R (Wickham et al., 2011; Wickham & Cook, 2025). It defines a tour to consist of three components:

- **data:** An $(n \times p)$ numerical data matrix to be viewed.
- **path:** A tour path function that produces a smoothed sequence of projection matrices $\mathbf{P}_{(p \times d)}$ in d . dimensions, for example `grand_tour(d = 2)` or `guided_tour(index = holes)`.
- **display:** A function that renders the projected data, for example `display_xy()` for a scatterplot, `display_depth()` for a 3D plot with simulated depth, or `display_pcp()` for a parallel coordinates plots

This very nicely separates the aspects of a tour, and allows one to think of and define new tour path methods and display methods. The package defines two general tour functions: `animate()` produces a real-time animation on a display device and `render()` saves image frames to disk, such as a .gif file.

```
animate(data, tour_path, display_method)
render(data, tour_path, display_method)
```

The `tourr` package provides a wide range of tour path methods and display methods:

```
# tour path methods
grep("_tour$", ls.str("package:tourr"), value = TRUE)
#> [1] "dependence_tour"      "frozen_guided_tour"
#> [3] "frozen_tour"          "grand_tour"
#> [5] "guided_anomaly_tour" "guided_section_tour"
#> [7] "guided_tour"          "little_tour"
#> [9] "local_tour"           "new_tour"
#> [11] "planned_tour"        "planned2_tour"
#> [13] "radial_tour"

# display methods
grep("display_", ls.str("package:tourr"), value = TRUE)
#> [1] "display_andrews"    "display_density2d" "display_depth"
#> [4] "display_dist"       "display_faces"     "display_groupxy"
```

```
#> [7] "display_idx"      "display_image"    "display_pca"
#> [10] "display_pcp"     "display_sage"     "display_scatmat"
#> [13] "display_slice"   "display_stars"   "display_stereo"
#> [16] "display_trails"  "display_xy"
```

Tour path methods take a variety of optional arguments to specify the detailed behavior of the method. For example, most allow you to specify the number of dimension (`d =`) of the projections. The `guided_tour()` is of particular interest here.

```
args(guided_tour)
#> function (index_f, d = 2, cooling = 0.99, max.tries = 25, max.i = Inf,
#>   search_f = search_geodesic, n_jellies = 30, n_sample = 100,
#>   alpha = 0.5, ...)
#> NULL
```

In this, `index_f` specifies a function that the method tries to optimize on its path and package defines four indices:

- Holes (`holes()`): This is sensitive to projections with separated clusters of points, with few points near the origin
- Central mass (`cmass()`): Sensitive to projections with lots of points in the center, but perhaps with some outliers
- Linear discriminant analysis (`lda_pp()`): For data with a grouping factor, optimizes a measure of separation of the group means as in MANOVA or linear discriminant analysis.
- PDA analysis (`pda_pp()`): A penalized version of `lda_pp()` for cases of large p relative to sample size n (E.-K. Lee & Cook, 2009).

In addition, there is now a `guided_anomaly_tour()` that looks for the best projection of observations that are outside the data ellipsoid, finding a view showing observations with large Mahalanobis distances from the centroid.

3.14.2.3 Penguin tours

Penguins are a traveling species. They make yearly travels inland to breeding sites in early spring, repeating the patterns of their ancestors. Near the beginning of summer, adult penguins and their chicks return to the sea and spend the rest of the summer feeding there (Black et al., 2018). If they were also data scientists, they might wonder about the relations among among their cousins of different species and take a tour of their measurements...

For example, using the Penguins dataset, the following calls produce grand tours in 2, 3, and 4 dimensions. The 2D tour is displayed as a scatterplot, the 3D tour using simulated depth as shown by variation in point size and transparency, and the 4D tour is shown using a parallel coordinate plot.

```
data(peng, package = "heplots")
peng_scaled <- scale(peng[,3:6])
colnames(peng_scaled) <- c("BL", "BD", "FL", "BM")

animate(peng_scaled, grand_tour(d = 2), display_xy())
animate(peng_scaled, grand_tour(d = 3), display_depth())
animate(peng_scaled, grand_tour(d = 4), display_pcp())
```

To illustrate, I'll start with a grand tour designed to explore this 4D space of penguins. I'll abbreviate the variables to two characters, “BL” = `bill_length`, “BD” = `bill_depth`, “FL” = `flipper_length`, and “BM” = `body_mass` and identify the penguin species using point shape (`pch`) and color (`col`).

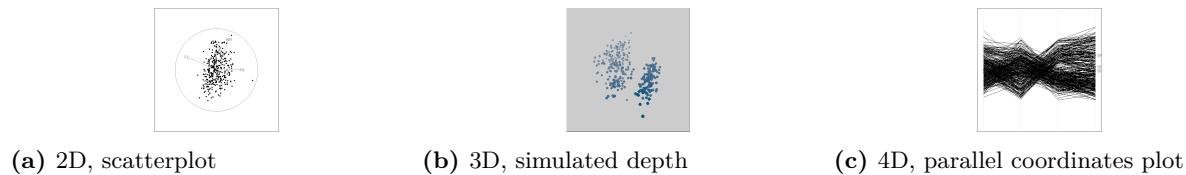


Figure 3.46: Grand tours of the penguin dataset in 2, 3, and 4 dimensions using different `display_*`() methods.

As you watch this pay attention to the separation of the species and any other interesting features. What do you see?

```
data(peng, package = "heplots")
peng_scaled <- scale(peng[,3:6])
colnames(peng_scaled) <- c("BL", "BD", "FL", "BM")

pch <- c(15, 16, 17)[peng$species]
cex = 1.2

set.seed(1234)
animate(peng_scaled,
        tour_path = grand_tour(d=2),
        display_xy(col = peng$species,
                   palette = peng.colors("dark"),
                   pch = pch, cex = cex,
                   axis.col = "black",
                   axis.text.col = "black",
                   axis.lwd = 1.5))
```

Figure 3.47 shows three frames from this movie. The first (a) is the initial frame that shows the projection in the plane of bill depth and bill length. The variable vectors indicate that bill length differentiates Adelie penguins from the others. In frame (b), the three species are widely separated, with bill depth distinguishing Gentoo from the others. In frame (c) the three species are largely mixed, but two points stand out as outliers, with exceptionally long bills compared to the rest.

Let's take the penguins on a guided tour, trying to find views that show the greatest separations among the penguin species; that is, a guided tour, optimizing the `lda_pp()` index.

```
set.seed(1234)
animate(peng_scaled,
        guided_tour(lda_pp(peng$species)),
        display_xy(col = peng$species,
                   palette = peng.colors("dark"),
                   pch = pch,
                   cex = cex)
)
```

TODO: I'm trying to balance what will/can be shown in the HTML version vs. the printed PDF. Needs text here specifically for the PDF version.

These examples are intended to highlight what is possible with dynamic graphics for exploring high-dimensional data visually. D. Cook & Laa (2024) extend the discussion of these methods from D. Cook & Swayne (2007)

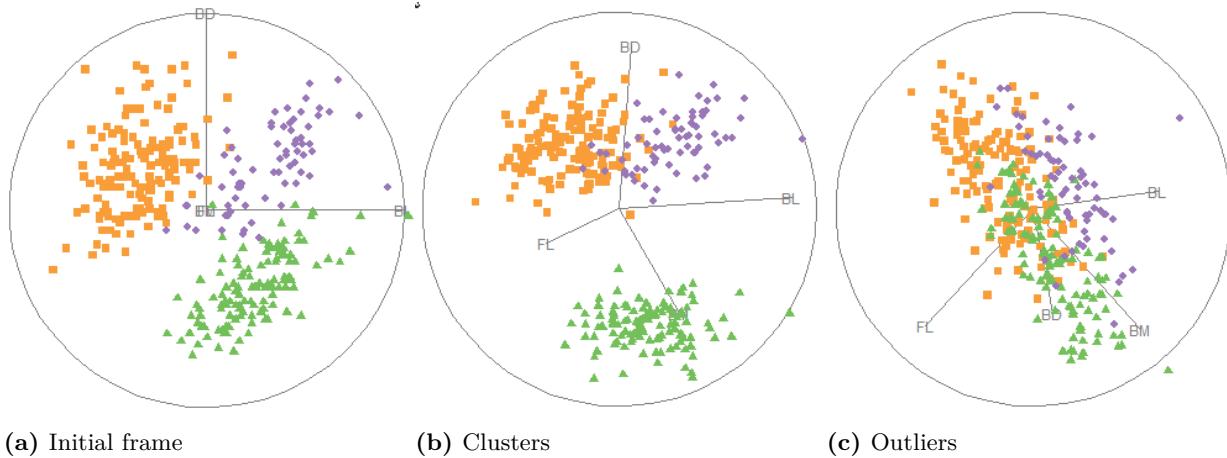


Figure 3.47: Three frames from the grand tour of the Penguin data. (a) The initial frame is the projection showing only BD and BL, where bill length conveniently separates Adelie from the other two species. (b) A frame that shows the three species more widely separated. (c) A frame that shows two outliers with very large bills.

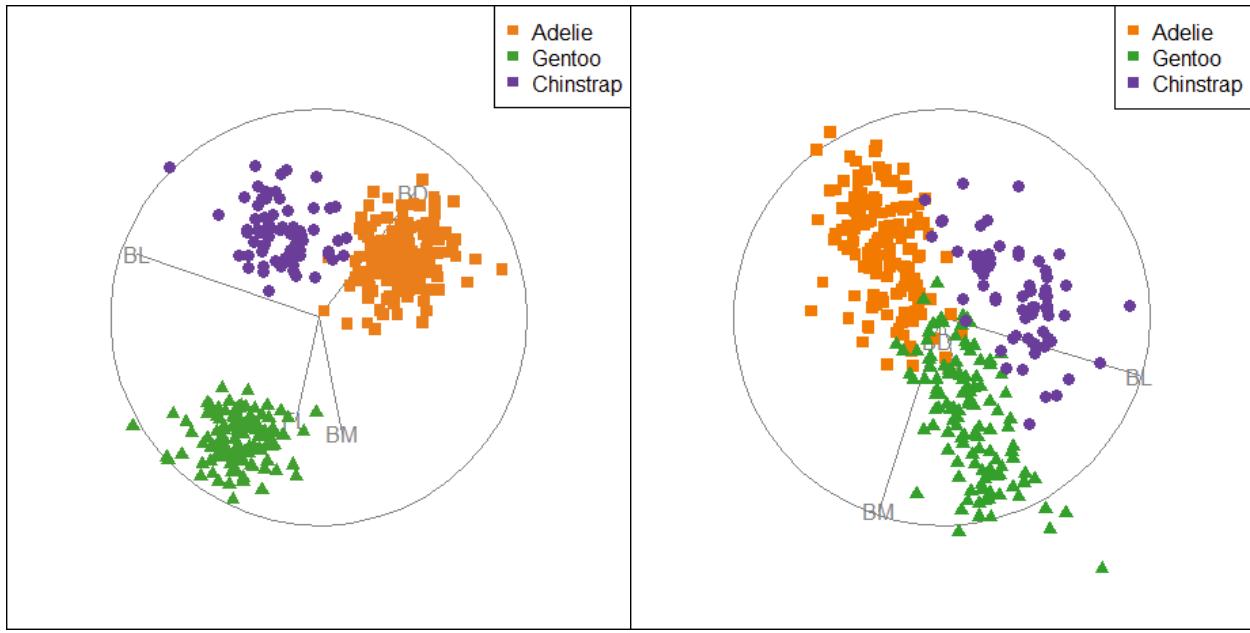


Figure 3.48: Guided tours: These figures show the final frame in the animations of guided tours designed to find the projection that optimize an index. (a) The `lda_pp()` criterion optimizes the separation of the means relative to within-group variation. (b) The `anomalies_index()` optimizes the average Mahalanobis distance of points from the centroid

(which used Ggobi) to the `tourrr` package. They illustrate dimension reduction, various cluster analysis methods, trees and random forests and some machine-learning techniques.

Ideally, we should be able interact with a tour,

- pausing when we see something interesting and saving the view for later analysis;
- selecting or highlighting unusual points,

- changing tour methods or variables displayed on the fly, and so forth.

Some packages that provide these capabilities are: **detourr** (Hart & Wang, 2022) **liminal** (S. Lee, 2021) and **langevitour** (Harrison, 2023, 2025) The **loon** package (Waddell & Oldford, 2023) is a general toolkit that enables highly interactive data visualization. It provides a **loon.tour** package (Xu & Oldford, 2021) for using touring methods within the **loon** environment.

3.15 Network diagrams

A major theme throughout this chapter has been to understand how to extend data visualization from simple bivariate scatterplots to increasingly more complex situations with larger datasets. With a moderate number of variables, techniques such as smoothing, summarizing with data ellipses and fitted curves, and visual thinning can be used to tame “big N ” datasets with thousands of cases.

However “big p ” datasets, with more than a moderate number (p) of variables still remain a challenge. It is hard to see how the more advanced methods (corrgrams, parallel coordinate) described earlier could cope with $p = 20, 50, 100, 500, \dots$ variables. At some point, each of these begins to break down for the purpose of visualizing associations among many variables. We are forced to thin the information presented in graphs more and more as the number of variables increases.

It turns out that there is a way to increase the number of variables displayed dramatically, if we are mainly interested in the pairwise correlations for reasonably normally distributed data. A graphical **network diagram** portrays variables by *nodes* (vertices), connected by (weighted) *edges* whose properties reflect the strength of connections between pairs, such as a correlation. Such diagrams can reveal properties not readily seen by other means.

As an example consider Figure 3.49, which portrays the correlations among 25 self-report items reflecting 5 factors (the “Big Five”) considered in personality psychology to represent the dominant aspects of all of personality. These factors are easily remembered by the acronym **OCEAN**: Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism. The dataset, **bfi**, contains data from an online sample of $n = 2800$ with 5 items for each scale.

In this figure (taken from Rodrigues (2021)), the item nodes are labeled according to the OCEAN factor they are assumed to measure. For 25 items, there are $25 \times 24/2 = 300$ correlations, way too much to see. A clearer picture arises when we reduce the number of edges shown according to some criterion. Here, edges are drawn *only* between nodes where the correlation is considered important by a method (“glasso” = graphical LASSO) designed to make the graph optimally sparse.

The edges shown in Figure 3.49 reflect the Pearson correlation between a given pair of items by the visual attributes of color and line style: magnitude is shown by both the thickness and transparency of the edge; the sign of the correlation is shown by color and line type: solid blue for positive correlations and dashed red for negative ones.

According to some theories, the five personality factors should be largely non-overlapping, so there should not be many edges connecting items of one factor with those of another. Yet, there are quite a few cross-factor connections in Figure 3.49, so perhaps the theory is wrong, or, more likely, the 25 items are not good representatives of these underlying dimensions. The network diagram shown here is a visual tool for thought and refinement. See Costantini et al. (2015) for a tutorial on network analysis of personality data in R.

Network diagrams stem from mathematical graph theory (Bondy & Murty, 2008; West, 2001) of the abstract properties of nodes and edges used to represent pairwise relationships. These can be used to model many types of relations and processes in physical, biological, social and other sciences, where such properties as connectedness, centrality, cliques of connected nodes and so forth provide a vocabulary used to understand and explain complex systems.

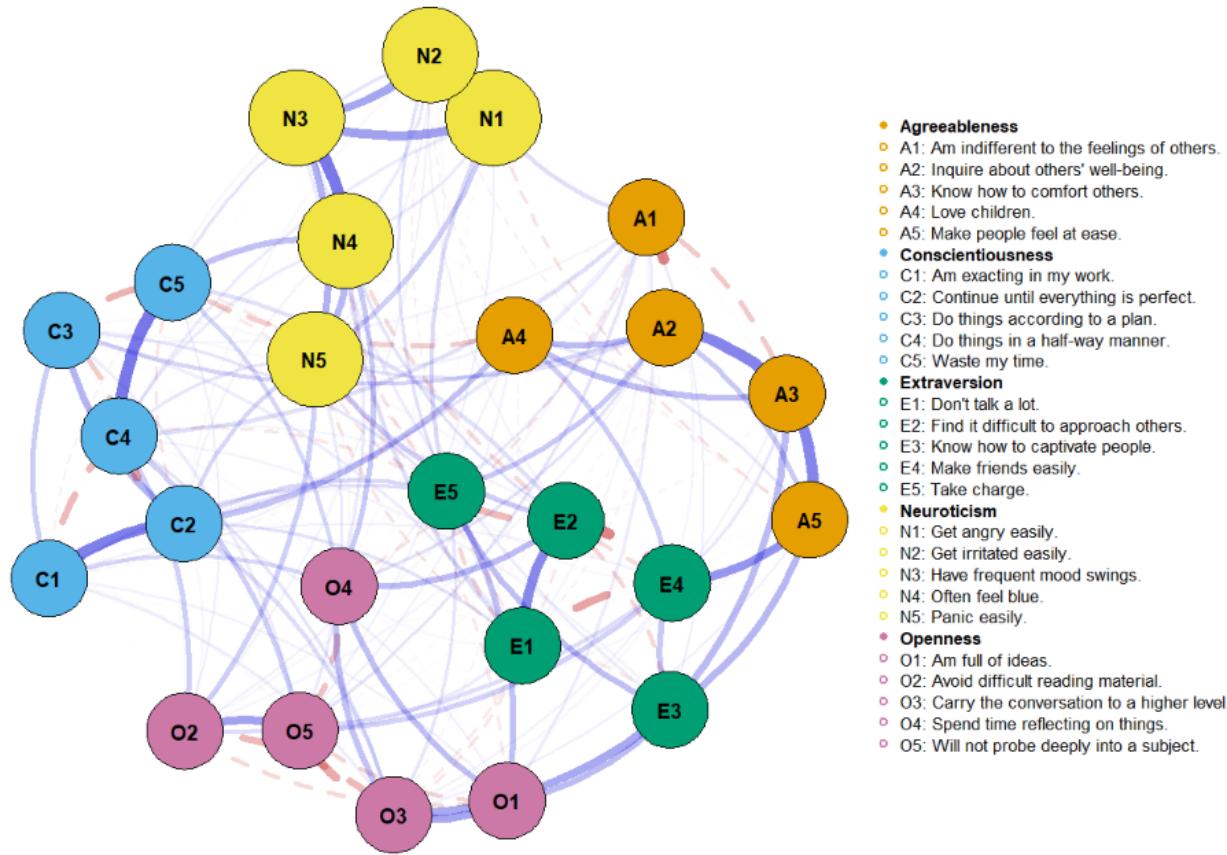


Figure 3.49: Network diagram of the correlations among 25 items from a Big-Five personality scale, 5 items for each scale. The magnitude of a correlation is shown by the thickness and transparency of the edge between two item nodes. The sign of a correlation is shown by edge color and style: solid blue for positive and dashed red for negative. *Source: Rodrigues (2021)*

For one example, Grandjean (2016) used network analysis to study the connections among 2500 Twitter users (nodes) who identified as belonging to a “digital humanities” community from the relations (edges) of who follows whom. Grandjean also used these methods to study the relationships among [characters in Shakespeare’s tragedies](#) in terms of the characters (nodes) and edges representing how often they appeared in the same scene.

The wide applicability of these ideas has led to what is now called *network science* (Barab’asi, 2016) encompassing computer networks, biological networks, cognitive and semantic networks, and social networks. Recent developments in psychology led to a framework of *network psychometrics* (Ivoranu et al., 2022), where, for example, symptoms of psychopathology (phobias, anxiety, substance abuse) can be conceptualized as an interconnected network of clusters and studied for possible causal relations (Robinaugh et al., 2019).

Because a network diagram can potentially reflect hundreds of variables, various [graph layout algorithms](#) have been developed to automatically position the nodes so as to generate aesthetically pleasing network visualizations that emphasize important structural properties, like clusters and central nodes, while minimizing visual clutter (many crossing lines) to promote understandability and usability.

There are quite a few R packages for constructing network diagrams, both static and dynamic / interactive, and these differ considerably in how the information required for a graph is structured as R objects, and the flexibility to produce attractive graphs. Among these, [igraph](#) (Csárdi et al., 2024) structures the data as a dataset of vertices and edges with properties

-> packages: qgraph, ...

3.15.1 Crime data

For the present purposes, let's see what network diagrams can tell us about the crime data analyzed earlier. Here, I first reorder the variables as in Figure 3.35. In the call to `qgraph()`, the argument `minimum = "sig"` says to show only the edges for significant correlations (at $\alpha = 0.01$ here). In Figure 3.50, the variable nodes are positioned around a circle (`layout = "circle"`), which is the default.

```
library(qgraph)
ord <- corrMatOrder(crime.cor, order = "AOE")
rownames(crime.cor)[ord]
#> [1] "murder"    "assault"   "rape"      "robbery"   "burglary"
#> [6] "larceny"   "auto"
crime.cor <- crime.cor[ord, ord]

# "association graph": network of correlations
qgraph(crime.cor,
       title = "Crime data:\ncorrelations", title.cex = 1.5,
       graph = "cor",
       layout = "circle",
       minimum = "sig", sampleSize = nrow(crime), alpha = 0.01,
       color = grey(.9), vsize = 12,
       labels = rownames(crime.cor),
       posCol = "blue")
```

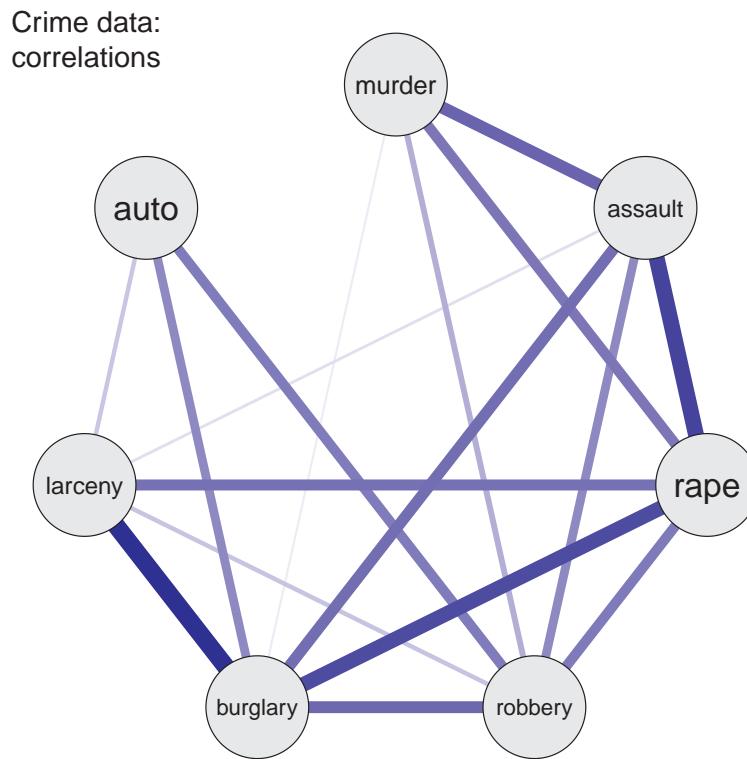


Figure 3.50: Network diagram depicting the correlations among the crime variables. Only edges for correlations that are significant at the $\alpha = 0.01$ level are displayed.

In this figure, you can see the group of property crimes (auto theft, larceny, burglary) at the left separated from the violent crimes against persons at the right.

3.15.2 Partial correlations

Among the more important statistical applications of network graph theory is the idea that you can also use them to study the the *partial* (conditional) associations among variables with the contributions of all other variables removed in what are called Graphical Gaussian Models (GGMs) (Højsgaard et al., 2012; Lauritzen, 1996). In a network diagram of these partial associations,

- The edges between nodes represent the *partial correlations* between those variables.
- The absence of an edge between two nodes indicates their variables are *conditionally independent*, given the other variables.

So, whereas a network diagram of correlations shows *marginal associations* ignoring other variables, one of partial correlations allows you to visualize the *direct* relationship between each pair of variables, removing the indirect effects that might be mediated through all other variables.

For a set of variables $X = \{x_1, x_2, \dots, x_p\}$, the partial correlation between x_i and x_j , controlling for all other variables $Z = X \setminus \{x_i, x_j\} = x_{\text{others}}$ is equivalent to the correlation between the *residuals* of the linear regressions of x_i on all other \mathbf{Z} and x_j on \mathbf{Z} . (The notation $X \setminus \{x_i, x_j\}$ is read as “ X without the set $\{x_i, x_j\}$ ”).

Mathematically, let \hat{x}_i and \hat{x}_j be the predicted values from the linear regressions of x_i on \mathbf{Z} and of x_j on \mathbf{Z} , respectively. The partial correlation p_{ij} between x_i and x_j controlling for \mathbf{Z} is given by:

$$p_{x_i, x_j | \mathbf{Z}} = r(x_i, x_j | \text{others}) = \text{cor}[(x_i - \hat{x}_i), (x_j - \hat{x}_j)] \quad (3.3)$$

But, rather than running all these linear regressions, they can all be computed from the inverse of the correlation matrix (Whittaker, 1990, Ch. 5), a relation first noted by Dempster (1972). Let \mathbf{R} be the correlation matrix of the variables. Then, the matrix \mathbf{P} of partial correlations can be obtained from the negative inverse, $-\mathbf{R}^{-1}$, standardized to a correlation matrix by dividing by the square root of product of its diagonal elements,

$$P_{ij} = -\frac{R_{ij}^{-1}}{\sqrt{(R_{ii}^{-1} \cdot R_{jj}^{-1})}} .$$

The practical implications of this are:

- If a partial correlation is close to zero, it suggests the relationship between two variables is primarily mediated through other variables.
- Non-zero partial correlations indicate a direct relationship that persists after controlling for other variables.

Figure 3.51 shows the partial correlation network for the crime data, using the `qgraph()` argument `graph = "pcor"`. To provide a more interpretable result, the argument `layout = "spring"` positions the nodes using a force-embedded algorithm where edges act like springs, pulling connected nodes together and unconnected nodes repel each other, pushing them apart.

```
qgraph(crime.cor,
       title = "Crime data:\npartial correlations", title.cex = 1.5,
       graph = "pcor",
       layout = "spring", repulsion = 1.2,
       minimum = "sig", sampleSize = nrow(crime), alpha = 0.05,
       color = grey(.9), vsize = 14,
```

```
labels = rownames(crime.cor),
edge.labels = TRUE, edge.label.cex = 1.7,
posCol = "blue")
```

Crime data:
partial correlations

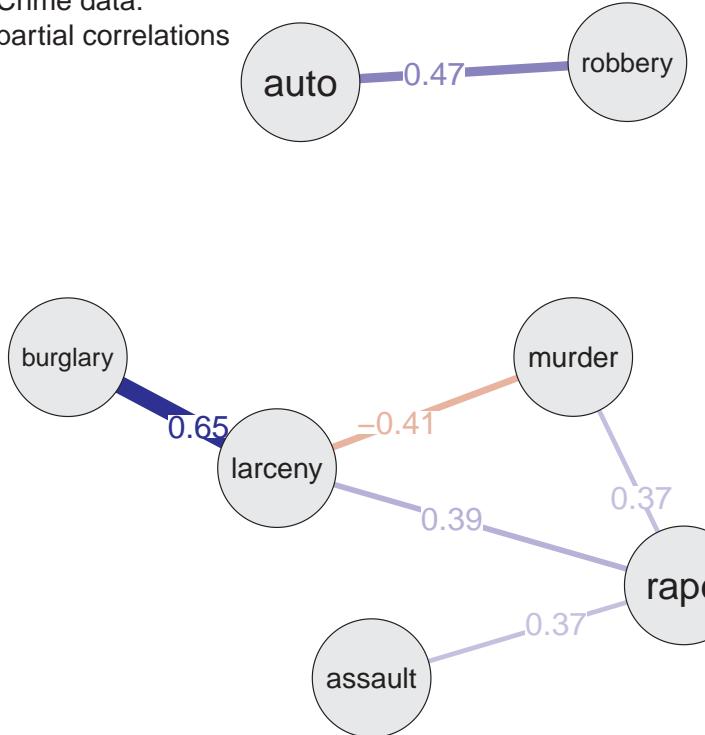


Figure 3.51: Network diagram of partial correlations among the crime variables, controlling for all others. Variable nodes have been positioned by a “spring” layout method ...

Figure 3.51 shows that, once all other crime variables are controlled for each pair, there remain only a few partial correlations at the $\alpha = 0.05$ level. Of these, only the largest three in absolute value are significant at $\alpha = 0.01$.

Thus, once all other variables are taken into account, what remains is mainly a strong positive association between burglary and larceny and a moderate one between auto theft and robbery. There also remains a moderate negative correlation between murder and larceny. The spring layout makes it clear that, with suppression of weak edges, auto theft and robbery form a cluster separated from the other variables.

3.15.3 Visualizing partial correlations

Just as you can visualize *marginal* association between variables in a scatterplot, you can also visualize *conditional* association. A **partial variables plot** is simply a scatterplot of the partial residuals $e_i = (x_i - \hat{x}_i)$ from a regression of x_i on the other variables Z against those $e_j = (x_j - \hat{x}_j)$ for another variable x_j .

In this, you can use all the bells and whistles of standard scatterplots (regression lines, smooths, data ellipses, ...) to listen more attentively to the story partial association has to tell. The function `pvPlot()` calculates the partial residuals and then calls `car::dataEllipse()` for display. The five most “unusual” observations by Mahalanobis D^2 are identified with their abbreviated state labels. Figure 3.52 shows these plots for the variable pairs with the two largest partial correlations.

```

source("R/pvPlot.R")
# select numeric, make `st` into rownames
crime.num <- crime |>
  tibble::column_to_rownames("st") |>
  dplyr::select(where(is.numeric))

pvPlot(crime.num, vars = c("burglary", "larceny"),
       id = list(n=5),
       cex.lab = 1.5)
pvPlot(crime.num, vars = c("robbery", "auto"),
       id = list(n=5),
       cex.lab = 1.5)

```

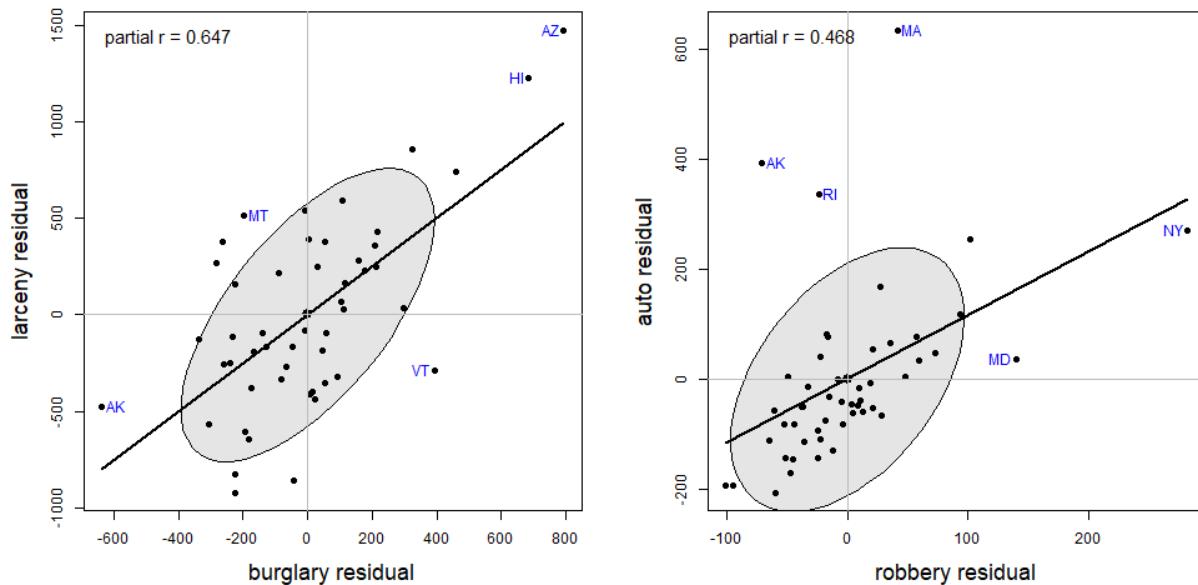


Figure 3.52: Partial variables plots for burglary and larceny (left) and for robbery and auto theft (right) in the network diagram for partial correlations of the crime variables.

In the pvPlot for burglary and larceny, you can see that the high partial correlation is largely driven by the extreme points at the left and right sides. Once all other variables are taken into account, Arizona (AZ) and Hawaii (HI) have larger incidence of both crimes, while Arkansas (AK) are smaller on both.

In the pvPlot for robbery and auto theft, New York stands out as an influential, high-leverage point (see Section 6.6); Massachusetts (MA) is noteworthy because auto theft in that state is considerably higher than what would be predicted from all other variables.

3.16 What have we learned?

This chapter provides a comprehensive toolkit for visualizing multivariate relationships, transforming the humble scatterplot into a powerful engine for data exploration and discovery.

- **Enhance your plots:** The basic scatterplot becomes far more informative when we add **smoothers**

(regression lines, loess curves), **stratifiers** (grouping by color, shape, or panels), and **data ellipses** that capture correlation, variance, and regression relationships in a single elegant geometric form. These annotations help turn static dots into informative stories about relationships in your data.

- **Data ellipses are a visualization Multi-Tool:** These simple geometric summaries encode a remarkable amount of information—means, standard deviations, correlation, regression slopes, and confidence regions—all in one visual package. When your data are roughly bivariate normal, the ellipse becomes a sufficient summary, telling you everything you need to know about the relationship between two variables. Relaxing the normality assumption, robust methods like the bagplot can serve nearly as well.
- **Simpson’s Paradox lurks everywhere:** The grouping variables you ignore can completely reverse the relationships you see, once you properly account for the grouping variables. Marginal correlations can be negative while all within-group correlations—conditional on the grouping variable—are positive (or vice versa). This fundamental lesson reminds us that **context matters**—always consider what variables you might be overlooking.
- **Scatterplot matrices scale gracefully:** When you have multiple variables, pairs plots let you see all pairwise relationships simultaneously. Add **visual thinning** (removing points, keeping only smoothers and ellipses) and **effect ordering** (arranging variables by similarity) to handle even larger numbers of variables while maintaining interpretability.
- **Generalized pairs plots bridge data types:** Modern extensions handle mixtures of continuous and categorical variables elegantly, using appropriate plot types for each combination—scatterplots for continuous pairs, box plots for continuous-categorical combinations, and mosaic plots for categorical pairs. This unified framework means no variable gets left behind.
- **Parallel coordinates reveal high-dimensional patterns:** When scatterplot matrices reach their limits, parallel coordinate plots let you visualize dozens of variables simultaneously. Each observation becomes a connected line across parallel axes, revealing clusters, outliers, and multivariate patterns that would be invisible in traditional 2D projections.
- **Tours provide dynamic exploration:** Animated statistical tours take you on guided journeys through high-dimensional data space, smoothly rotating through different 2D projections. Whether taking a random grand tour or a guided tour optimized for specific features (clusters, outliers, group separation), these methods reveal structures hidden in static displays.
- **Network diagrams conquer “big p” data:** When correlations become too numerous to display traditionally, network graphs show variables as nodes connected by edges representing associations. Partial correlation networks reveal **direct** relationships between variables with all other influences removed—the difference between marginal and conditional independence becomes visually apparent.
- **Visual thinning is a superpower:** As datasets grow in size and complexity, strategic removal of visual elements (points, axes, labels) while retaining essential summaries (trends, ellipses, connections) lets you maintain clarity and insight. Less can indeed be more when every remaining element carries a greater share of graphic information.

Having understood these lessons of multivariate thinking and visualization in *data space*, you are now prepared to take the next step in Chapter 4 to consider how to enhance your understanding of multivariate data using **multivariate juicers** which project high-dimensional data into a lower-dimensional space in which important effects can be more easily seen.

3.17 Exercises

Exercise 3.1. Using the `Salaries` dataset, create one or more plots to compare different smoothing methods for the relationship between `yrs.since.phd` and `salary` shown in Figure 3.5. Include linear regression, quadratic polynomial, and loess smoothers,

```
library(ggplot2)
data(Salaries, package = "carData")
# Your code here
```

Exercise 3.2. One alternative to a loess smooth, which allows a `span` argument to control the degree of smoothing is a **natural spline**, that can be used in `geom_smooth()` using the argument `formula = y ~ splines::ns(x, df=)`, where `df` is the equivalent number of degrees of freedom for the spline smoother. Re-do Exercise 3.1, but trying out this smoothing method for several values of `df`.

Package summary

For development, keep track of the packages used in each chapter.

19 packages used here: `broom`, `car`, `carData`, `corrgram`, `corrplot`, `dplyr`, `GGally`, `ggdensity`, `ggda`, `ggpcp`, `ggplot2`, `grid`, `heplots`, `knitr`, `patchwork`, `qgraph`, `tidyverse`, `tourr`, `vcd`

4

Dimension Reduction

4.1 Flatland and Spaceland

It is high time that I should pass from these brief and discursive notes about Flatland to the central event of this book, my initiation into the mysteries of Space. THAT is my subject; all that has gone before is merely preface — Edwin Abbott, *Flatland*, p. 57.

There was a cloud in the sky above *Flatland* one day. But it was a huge, multidimensional cloud of sparkly points that might contain some important message, perhaps like the hidden EUREKA (Figure 3), or perhaps forecasting the upcoming harvest, if only Flatlanders could appreciate it.

A leading citizen, A SQUARE, who had traveled once to Spaceland and therefore had an inkling of its majesty beyond the simple world of his life in the plane looked at that cloud and had a brilliant thought, an OMG moment:

“Oh, can I, in my imagination, rotate that cloud and squeeze its juice so that it rains down on Flatland with greatest joy?”

As it happened, our Square friend, although he could never really *see* in three dimensions, he could now at least *think* of a world described by **height** as well as breadth and width, and think of the **shadow** cast by a cloud as something mutable, changing size and shape depending on its’ orientation over Flatland.

And what a world it was, inhabited by Pyramids, Cubes and wondrous creatures called Polyhedrons with many *Corners*, *Faces* and *Edges*. Not only that, but all those Polyhedra were forced in Spaceland to obey a magic formula: $C + F - E = 2$.¹ How cool was that!

Indeed, there were even exalted Spheres, having so many faces that its surface became as smooth as a baby’s bottom with no need for pointed corners or edges, just as Circles were the smoothest occupants of his world with far too many sides to count. It was his dream of a Sphere passing through Flatland (Figure 1) that first awakened him to a third dimension.

He also marveled at Ellipsoids, as smooth as Spheres, but in Spaceland having three natural axes of different extent and capable of being appearing fatter or slimmer when rotated from different views. An Ellipsoid had magical properties: it could appear as so thin in one or more dimensions that it became a simple 2D ellipse, or a 1D line, or even a 0D point (Friendly et al., 2013).

All of these now arose in Square’s richer 3D imagination. And, all of this came from just one more dimension than his life in Flatland.

4.1.1 Multivariate juicers

Up to now, we have also been living in Flatland. We have been trying to understand data in **data space** of possibly many dimensions, but confined to the 2D plane of a graph window. Scatterplot matrices and

¹This is Euler’s (1758) formula, which states that any convex polyhedron must obey the formula $V + F - E = 2$ where V is the number of vertexes (corners), F is the number of faces and E is the number of edges. For example, a tetrahedron or pyramid has $(V, F, E) = (4, 4, 6)$ and a cube has $(V, F, E) = (8, 6, 12)$. Stated in words, for all solid bodies confined by planes, the sum of the number of vertexes and the number of faces is two less than the number of edges.

parallel coordinate plots provided some relief. The former did so by **projecting** the data into sets of 2D views in the coordinates of data space; the latter did so by providing multiple axes in a 2D space along which we could trace the paths of individual observations.

This chapter is about seeing data in a different projection, a low-dimensional (usually 2D) space that squeezes out the most juice from multidimensional data for a particular purpose (Figure 4.1), where what we want to understand can be more easily seen.

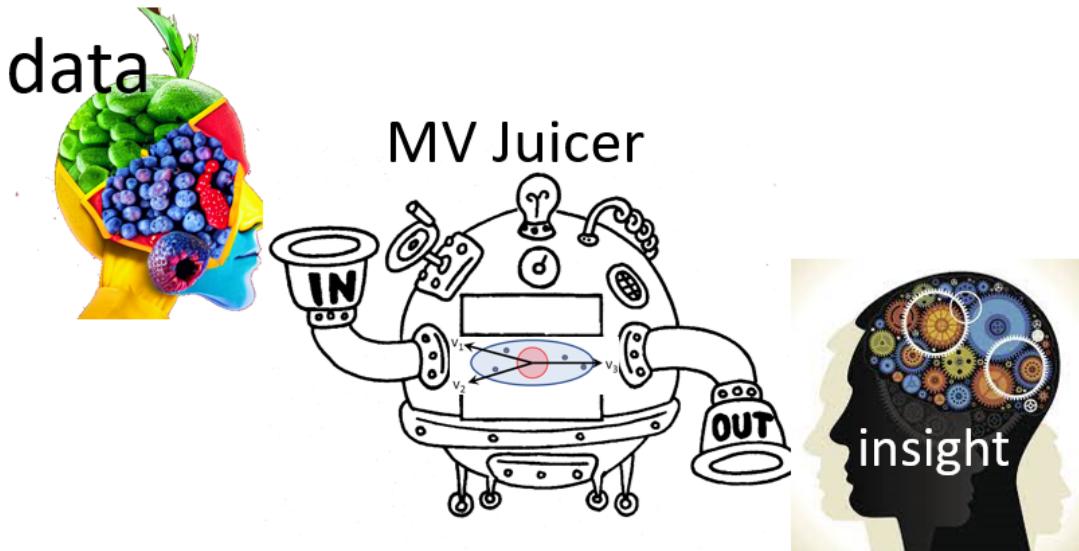


Figure 4.1: A multivariate juicer takes data from possibly high-dimensional data space and transforms it to a lower-dimensional space in which important effects can be more easily seen.

Here, I concentrate on **principal components analysis** (PCA), whose goal reflects A Square's desire to see that sparkly cloud of points in n D space in the plane showing the greatest variation (squeezing the most juice) among all other possible views. This appealed to his sense of geometry, but left him wondering how the variables in that high-D cloud were related to the dimensions he could see in a best-fitting plane.

The idea of a **biplot**, showing the data points in the plane, together with thick pointed arrows—variable vectors—in one view is the other topic explained in this chapter (Section 4.3). The biplot is the simplest example of a multivariate juicer. The essential idea is to project the cloud of data points in n dimensions into the 2D space of principal components and simultaneously show how the original variables relate to this space. For exploratory analysis to get an initial, incisive view of a multivariate dataset, a biplot is often my first choice.

i Looking ahead

I'm using the term *multivariate juicer* here to refer the wider class of **dimension reduction** techniques, used for various purposes in data analysis and visualization. PCA is the simplest example and illustrates the general ideas.

The key point is that these methods are designed to transform the data into a low-dimensional space for a particular goal or purpose. In PCA, the goal is to extract the greatest amount of total variability in the data. In the context of univariate multiple regression, the goal is often to reduce the number of predictors necessary to account for an outcome variable, called *feature extraction* in the machine learning literature.

When the goal is to best distinguish among groups **discriminant analysis** finds uncorrelated weighted

sums of predictors on which the means of groups are most widely separated in a reduced space of hopefully fewer dimensions.

The methods I cover in this book are all linear methods, but there is also a wide variety of non-linear dimension reduction techniques.

Packages

In this chapter I use the following packages. Load them now:

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(patchwork)
library(ggbiplot)
library(FactoMineR)
library(factoextra)
library(car)
library(ggpubr)
library(matlib)
```

4.2 Principal components analysis (PCA)

When Francis Galton (1886) first discovered the idea of regression toward the mean and presented his famous diagram (Figure 3.10), he had little thought that he had provided a window to a higher-dimensional world, beyond what even A Square could imagine. His friend, Karl Pearson (1896) took that idea and developed it into a theory of regression and a measure of correlation that would bear his name, Pearson's r .

But then Pearson (1901) had a further inspiration, akin to that of A Square. If he also had a cloud of sparkly points in $2, 3, 4, \dots, p$ dimensions, could he find a point ($0D$), or line ($1D$), or plane ($2D$), or even a hyperplane (nD) that best summarized — squeezed out the most juice—from multivariate data? This was the first truly multivariate problem in the history of statistics (Friendly & Wainer, 2021, p. 186).

The best $0D$ point was easy—it was simply the centroid, the means of each of the variables in the data, $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_p)$, because that was “closest” to the data in the sense of minimizing the sum of squared differences, $\sum_i \sum_j (x_{ij} - \bar{x}_j)^2$. In higher dimensions, his solution was also an application of the method of least squares, but he argued it geometrically and visually as shown in Figure 4.2.

For a $1D$ summary, the line of best fit to the points P_1, P_2, \dots, P_n is the line that goes through the centroid and made the average squared length of the *perpendicular* segments from those points to a line as small as possible. This was different from the case in linear regression, for fitting y from x , where the average squared length of the *vertical* segments, $\sum_i (y_i - \hat{y}_i)^2$ was minimized by least squares.

He went on to prove the visual insights from simple smoothing of Galton (1886) (shown in Figure 3.10) regarding the regression lines of $y \sim x$ and $x \sim y$. More importantly, he proved that the cloud of points is captured, for the purpose of finding a best line, plane or hyperplane, by the ellipsoid that encloses it, as seen in his diagram, Figure 4.3. The major axis of the 2D ellipse is the line of best fit, along which the data points have the smallest average squared distance from the line. The axis at right angles to that—the minor axis—is labeled “line of worst fit” with the largest average squared distance.

Even more importantly—and this is the basis for PCA—he recognized that the two orthogonal axes of the ellipse gave new coordinates for the data which were uncorrelated, whatever the correlation of x and y .

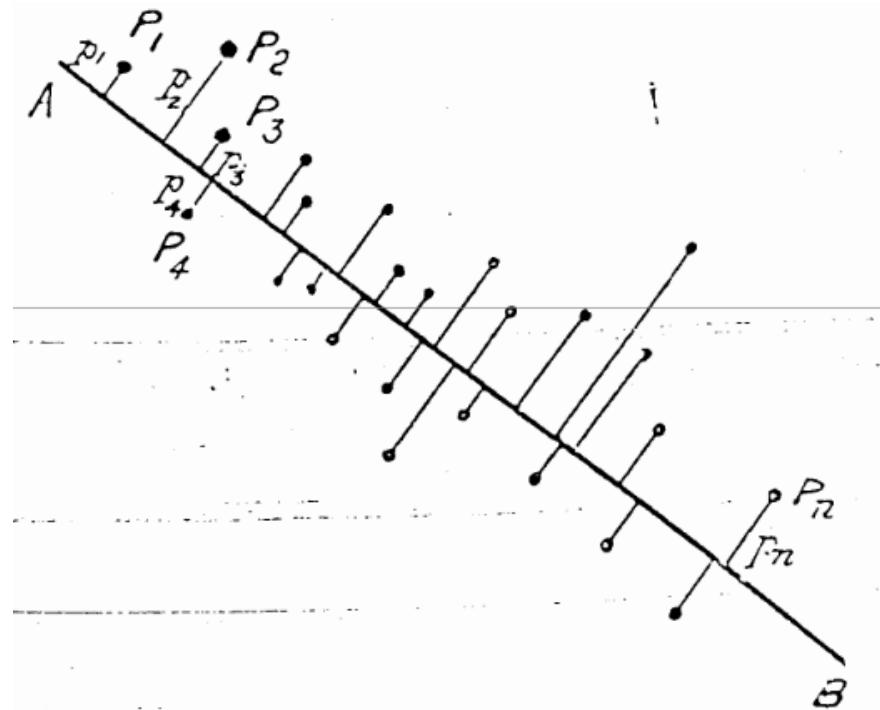


Figure 4.2: Karl Pearson's (1901) geometric, visual argument for finding the line or plane of closest fit to a collection of points, P_1, P_2, P_3, \dots

Physically, the axes of the correlation type-ellipse are the directions of independent and uncorrelated variation. — Pearson (1901), p. 566.

It was but a small step to recognize that for two variables, x and y :

- The line of best fit, the major axis (PC1) had the greatest variance of points projected onto it.
- The line of worst fit, the minor axis (PC2), had the least variance.
- These could be seen as a rotation of the data space of (x, y) to a new space (PC1, PC2) with uncorrelated variables.
- The total variation of the points in data space, $\text{Var}(x) + \text{Var}(y)$, being unchanged by rotation, was equally well expressed as the total variation $\text{Var}(\text{PC1}) + \text{Var}(\text{PC2})$ of the scores on what are now called the principal component axes.

It would have appealed to Pearson (and also to A Square) to see these observations demonstrated in a 3D video. `?@fig-pca-animation` shows a 3D plot of the variables `Sepal.Length`, `Sepal.Width` and `Petal.Length` in Edgar Anderson's `iris` data, with points colored by species and the 95% data ellipsoid. This is rotated smoothly by interpolation until the first two principal axes, PC1 and PC2 are aligned with the horizontal and vertical dimensions. Because this is a rigid rotation of the cloud of points, the total variability is obviously unchanged.

4.2.1 PCA by springs

Before delving into the mathematics of PCA, it is useful to see how Pearson's problem, and fitting by least squares generally, could be solved in a physical realization.

From elementary statistics, you may be familiar with a physical demonstration that the mean, \bar{x} , of a sample is the value for which the sum of deviations, $\sum_i(x_i - \bar{x})$ is zero, so the mean can be visualized as the point of balance on a line where those differences $(x_i - \bar{x})$ are placed. Equally well, there is a physical realization of

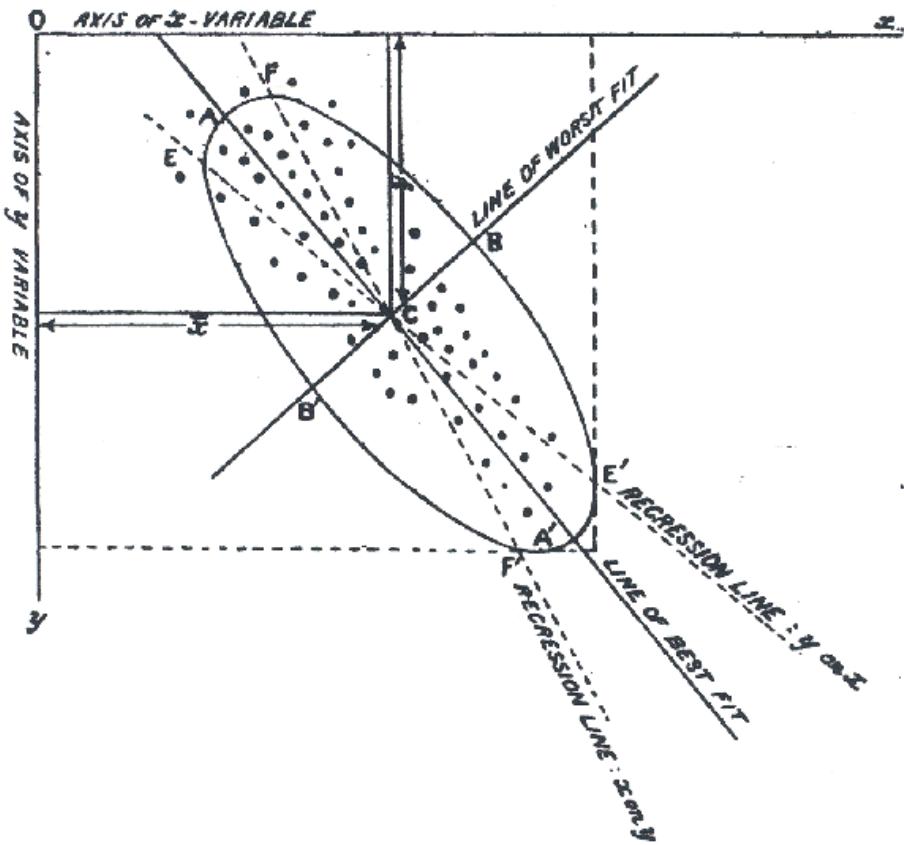


Figure 4.3: Karl Pearson's diagram showing the elliptical geometry of regression and principal components analysis . . . *Source:* Pearson (1901), p. 566.

the mean as the point along an axis where weights connected by springs will minimize the sum of squared differences, because springs with a constant stiffness, k , exert forces proportional to $k(x_i - \bar{x})^2$. That's the reason it is useful as a measure of central tendency: it minimizes the average squared error.

In two dimensions, imagine that we have points, (x_i, y_i) and these are attached by springs of equal stiffness k , to a line anchored at the centroid, (\bar{x}, \bar{y}) as shown in Figure 4.4. If we rotate the line to some initial position and release it, the springs will pull the line clockwise or counterclockwise and the line will bounce around until the forces, proportional to the squares of the lengths of the springs, will eventually balance out at the position (shown by the red fixed line segments at the ends). This is the position that minimizes the the sum of squared lengths of the connecting springs, and also minimizes the kinetic energy in the system.

If you look closely at Figure 4.4 you will see something else: When the line is at its final position of minimum squared length and energy, the positions of the red points on this line are spread out furthest, i.e., have **maximum** variance. Conversely, when the line is at right angles to its final position (shown by the black line at 90°) the projected points have the smallest possible variance.

Figure 4.4: Animation of PCA fitted by springs. The blue data points are connected to their projections on the red line by springs perpendicular to that line. From an initial position, the springs pull that line in proportion to their squared distances, until the line finally settles down to the position where the forces are balanced and the minimum is achieved. *Source:* Amoeba, <https://bit.ly/46tAicu>.

4.2.2 Mathematics and geometry of PCA

As the ideas of principal components developed, there was a happy marriage of Galton's geometrical intuition and Pearson's mathematical analysis. The best men at the wedding were ellipses and higher-dimensional ellipsoids. The bridesmaids were eigenvectors, pointing in as many different directions as space would allow, each sized according to their associated eigenvalues. Attending the wedding were the ghosts of uncles, Leonhard Euler, Jean-Louis Lagrange, Augustin-Louis Cauchy and others who had earlier discovered the mathematical properties of ellipses and quadratic forms in relation to problems in physics.

The key idea in the statistical application was that, for a set of variables $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$, the $p \times p$ covariance matrix \mathbf{S} could be expressed **exactly** as a matrix product involving a matrix \mathbf{V} , whose columns are *eigenvectors* (\mathbf{v}_i) and a diagonal matrix $\mathbf{\Lambda}$, whose diagonal elements (λ_i) are the corresponding *eigenvalues*.

To explain this, it is helpful to use a bit of matrix math:

$$\mathbf{S}_{p \times p} = \mathbf{V}_{p \times p} \quad \mathbf{\Lambda}_{p \times p} \quad \mathbf{V}_{p \times p}^T \quad (4.1)$$

$$= (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_p^T \end{pmatrix} \quad (4.2)$$

$$= \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{v}_2 \mathbf{v}_2^T + \cdots + \lambda_p \mathbf{v}_p \mathbf{v}_p^T \quad (4.3)$$

{#eq-S-eigen}

In this equation,

1. The last line follows because $\mathbf{\Lambda}$ is a diagonal matrix, so \mathbf{S} is expressed as a sum of outer products of each \mathbf{v}_i with itself, times the eigenvalue λ_i .
2. The columns of \mathbf{V} are the eigenvectors of \mathbf{S} . They are orthogonal and of unit length, so $\mathbf{V}^T \mathbf{V} = \mathbf{I}$ and thus they represent orthogonal (uncorrelated) directions in data space.
3. The columns \mathbf{v}_i are the weights applied to the variables to produce the scores on the principal components. For example, the first principal component is the weighted sum:

$$\text{PC}_1 = v_{11} \mathbf{x}_1 + v_{12} \mathbf{x}_2 + \cdots + v_{1p} \mathbf{x}_p .$$

4. The matrix of *all* scores on the principal components can be calculated by multiplying the data matrix \mathbf{X} by the eigenvectors, $\mathbf{PC} = \mathbf{X}\mathbf{V}$.
5. The eigenvalues, $\lambda_1, \lambda_2, \dots, \lambda_p$ are the variances of the components, because $\mathbf{v}_i^T \mathbf{S} \mathbf{v}_i = \lambda_i$.
6. It is usually the case that the variables $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$ are linearly independent, which means that none of these is an exact linear combination of the others. In this case, all eigenvalues λ_i are positive and the covariance matrix \mathbf{S} is said to have **rank** p . (Rank is the number of non-zero eigenvalues.)
7. Here is a key fact: If, as usual, the eigenvalues are arranged in order, so that $\lambda_1 > \lambda_2 > \cdots > \lambda_p$, then the first d components give a d -dimensional approximation to \mathbf{S} , which accounts for $\sum_i^d \lambda_i$ of the $\sum_i^p \lambda_i$ total variance, usually interpreted as the proportion, $(\sum_i^d \lambda_i) / (\sum_i^p \lambda_i)$.

For the case of two variables, \mathbf{x}_1 and \mathbf{x}_2 Figure 4.5 shows the transformation from data space to component space. The eigenvectors, $\mathbf{v}_1, \mathbf{v}_2$ are the major and minor axes of the data ellipse, whose lengths are the square roots $\sqrt{\lambda_1}, \sqrt{\lambda_2}$ of the eigenvalues.

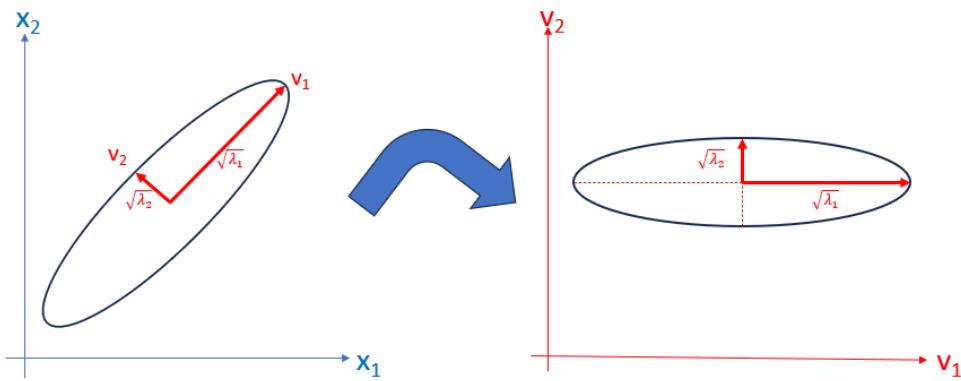


Figure 4.5: Geometry of PCA as a rotation from data space to principal component space, defined by the eigenvectors v_1 and v_2 of a covariance matrix

4.2.2.1 Example: Workers' experience and income

For a small example, consider the relation between years of experience and income in a small (contrived) sample ($n = 10$) of workers in a factory. The dataset `workers` contains these and other variables. In a wider context, we might want to fit a regression model to predict `Income`, but here we focus on a PCA of just these two variables.

```
data(workers, package = "matlib")
head(workers)
#>           Income Experience Skill Gender
#> Abby        20          0    2 Female
#> Betty       35          5    5 Female
#> Charles     40          5    8  Male
#> Doreen      30         10    6 Female
#> Ethan        50         10   10  Male
#> Francie     50         15    7 Female
```

Let's start with a simple scatterplot of `Income` vs. `Experience`, with points labeled by `Name` (and colored by `Gender`). There's a fairly strong correlation ($r = 0.853$). How does a PCA capture this?

```
vars <- c("Experience", "Income")
plot(workers[, vars],
      pch = 16, cex = 1.5,
      cex.lab = 1.5)
text(workers[, vars],
      labels = rownames(workers),
      col = ifelse(workers$Gender == "Female", "red", "blue"),
      pos = 3, xpd = TRUE)
```

To carry out a PCA of these variables, first calculate the vector of means (\bar{x}) and covariance matrix S .

```
mu <- colMeans(workers[, vars]) |> print()
#> Experience      Income
#>      15.5        46.5
S <- cov(workers[, vars]) |> print()
```

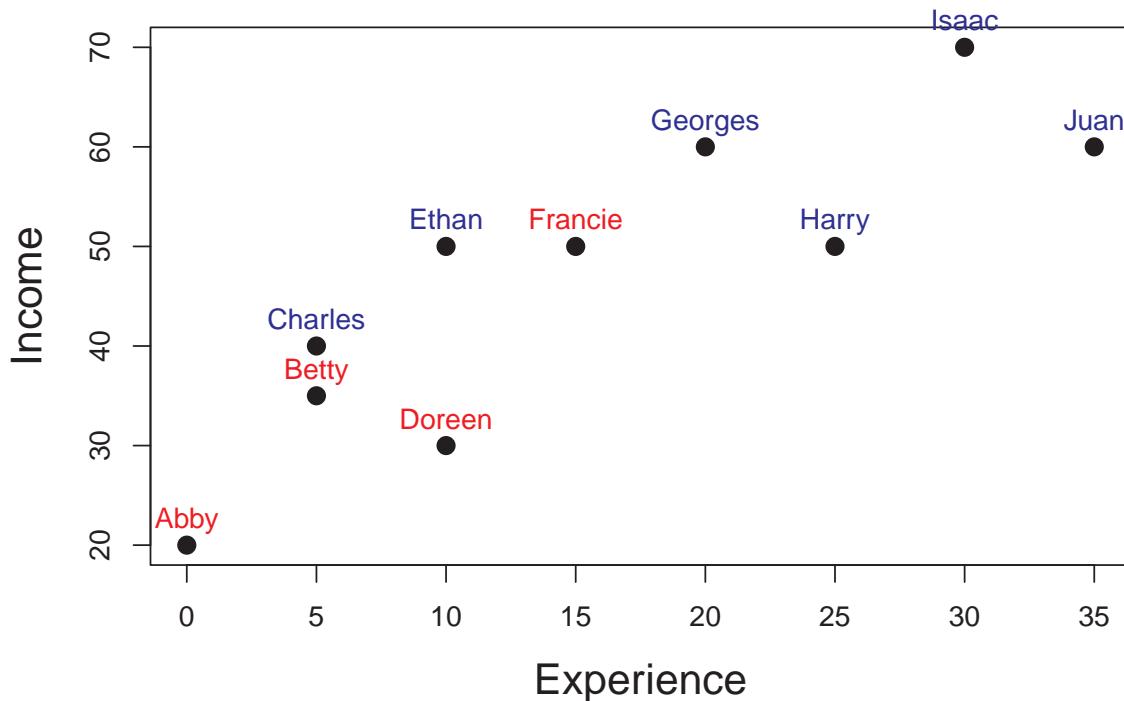


Figure 4.6: Scatterplot of Income vs. Experience for the `workers` data.

```
#>           Experience Income
#> Experience      136    152
#> Income          152    234
```

The eigenvalues and eigenvectors of \mathbf{S} are calculated by `eigen()`. This returns a list with components `values` for the λ_i and `vectors` for \mathbf{V} .

```
S.eig <- eigen(S)
Lambda <- S.eig$values |> print()
#> [1] 344.3 25.1
V <- S.eig$vectors |> print()
#> [,1]  [,2]
#> [1,] 0.589 -0.808
#> [2,] 0.808  0.589
```

From this, you can verify the points above regarding the relations between variances of the variables and the eigenvalues:

```
#total variances of the variables = sum of eigenvalues
sum(diag(S))
#> [1] 369
sum(Lambda)
#> [1] 369

# percent of variance of each PC
100 * Lambda / sum(Lambda)
#> [1] 93.2 6.8
```

Using these, you can express the eigenvalue decomposition of \mathbf{S} in ?@eq-S-eigen with `latexMatrix()` and `Eqn` from the `matlib` package (Friendly et al., 2024) as:

```
options(digits = 3)
rownames(S) <- colnames(S) <- c("\\small \text{Exp}",
                                     "\\small \text{Inc}")
spacer <- "\\phantom{0000000000000000}"
Eqn("\\mathbf{S} &= \\mathbf{V}", spacer,
     "\\mathbf{\\Lambda}", spacer,
     "\\mathbf{V}^\\top", Eqn_newline(),
     latexMatrix(S), "&=",
     latexMatrix(V), " ", diag(Lambda), " ", latexMatrix(V, transpose=TRUE),
     align = TRUE)
```

$$\mathbf{S} = \mathbf{V} \quad \mathbf{\Lambda} \quad \mathbf{V}^\top$$

$$\begin{matrix} Exp & Inc \\ Exp & \begin{pmatrix} 136 & 152 \\ 152 & 234 \end{pmatrix} \\ Inc & \end{matrix} = \begin{pmatrix} 0.589 & -0.808 \\ 0.808 & 0.589 \end{pmatrix} \begin{pmatrix} 344.3 & 0.0 \\ 0.0 & 25.1 \end{pmatrix} \begin{pmatrix} 0.589 & -0.808 \\ 0.808 & 0.589 \end{pmatrix}^\top$$

The “scores” on the principal components can be calculated (point (5) above) as $\mathbf{PC} = \mathbf{X}\mathbf{V}$:

```
PC <- as.matrix(workers[, vars]) %*% V
colnames(PC) <- paste0("PC", 1:2)
head(PC)
#>      PC1   PC2
#> Abby  16.2 11.78
#> Betty 31.2 16.57
#> Charles 35.3 19.52
#> Doreen 30.1  9.59
#> Ethan  46.3 21.37
#> Francie 49.2 17.32
```

Then, you can visualize the geometry of PCA as in Figure 4.5 (left) by plotting the data ellipse for the points, along with the PCA axes (`heplots::ellipse.axes()`). Figure 4.7 also shows the bounding box of the data ellipse, which are parallel to the PC axes and scaled to have the same “radius” as the data ellipse.

```
# calculate conjugate axes for PCA factorization
pca.fac <- function(x) {
  xx <- svd(x)
  ret <- t(xx$v) * sqrt(pmax( xx$d, 0))
  ret
}

dataEllipse(Income ~ Experience, data=workers,
            pch = 16, cex = 1.5,
            center.pch = "+", center.cex = 2,
            cex.lab = 1.5,
            levels = 0.68,
            grid = FALSE,
            xlim = c(-10, 40),
            ylim = c(10, 80),
```

```

asp = 1)
abline(h = mu[2], v = mu[1],
lty = 2, col = "grey")

# axes of the ellipse = PC1, PC2
radius <- sqrt(2 * qf(0.68, 2, nrow(workers)-1 ))
heplots::ellipse.axes(S, mu,
  radius = radius,
  labels = TRUE,
  col = "red", lwd = 2,
  cex = 1.8)

# bounding box of the ellipse
lines(spida2::ellplus(mu, S, radius = radius,
  box = TRUE, fac = pca.fac),
  col = "darkgreen",
  lwd = 2, lty="longdash")

```

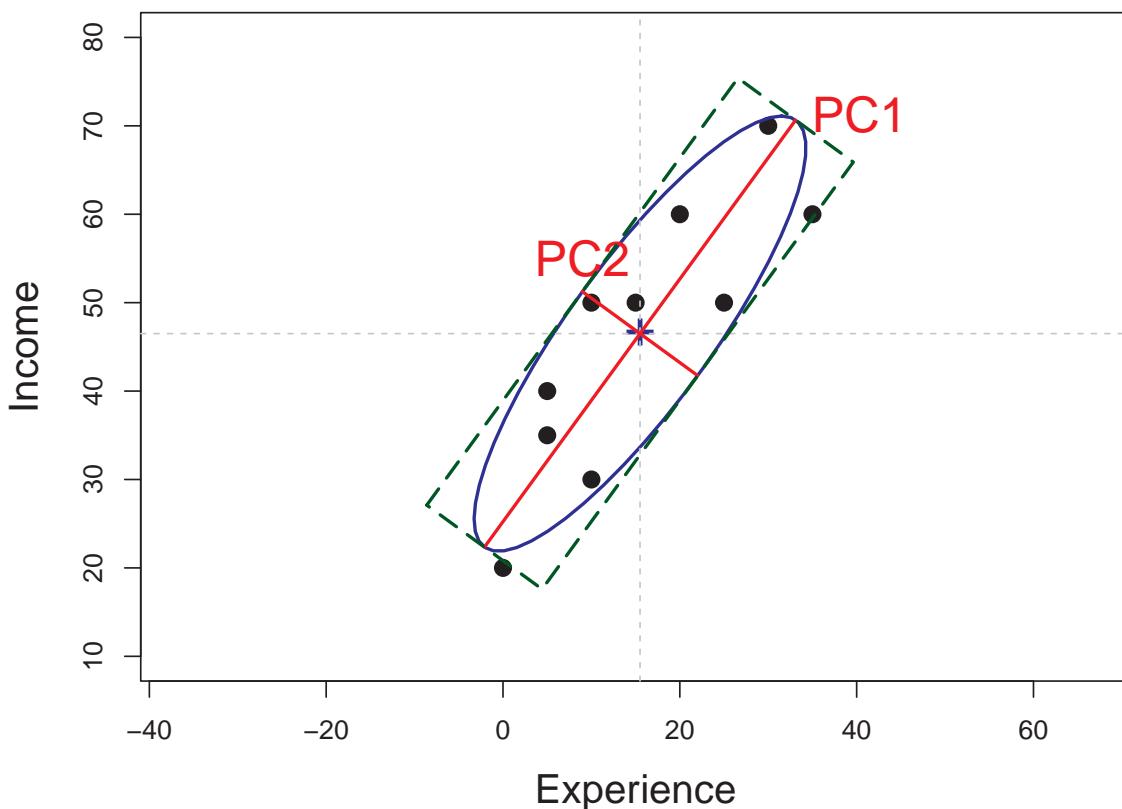


Figure 4.7: Geometry of the PCA for the `workers` data, showing the data ellipse, the eigenvectors of \mathbf{S} , whose half-lengths are the square roots $\sqrt{\lambda_i}$ of the eigenvalues, and the bounding box of the ellipse.

Finally, to preview the methods of the next section, the results calculated “by hand” above can be obtained using `prcomp()`. The values labeled “Standard deviations” are the square roots $\sqrt{\lambda_i}$ of the two eigenvalues. The eigenvectors are labeled “Rotation” because \mathbf{V} is the matrix that rotates the data matrix to produce the component scores.

```
workers.pca <- prcomp(workers[, vars]) |> print()
#> Standard deviations (1, ..., p=2):
#> [1] 18.56  5.01
#>
#> Rotation (n x k) = (2 x 2):
#>          PC1     PC2
#> Experience 0.589  0.808
#> Income      0.808 -0.589
```

4.2.3 Finding principal components

In R, PCA is most easily carried out using `stats:::prcomp()` or `stats:::princomp()` or similar functions in other packages such as `FactoMineR::PCA()`. The **FactoMineR** package (Husson et al., 2017, 2025) has extensive capabilities for exploratory analysis of multivariate data (PCA, correspondence analysis, cluster analysis).

A particular strength of FactoMineR for PCA is that it allows the inclusion of *supplementary variables* (which can be categorical or quantitative) and *supplementary points* for individuals. These are not used in the analysis, but are projected into the plots to facilitate interpretation. For example, in the analysis of the `crime` data described below, it would be useful to have measures of other characteristics of the U.S. states, such as poverty and average level of education (Section 4.3.5).

Unfortunately, although all of these functions perform similar calculations, the options for analysis and the details of the result they return differ.

The important options for analysis include:

- whether or not the data variables are **centered**, to a mean of $\bar{x}_j = 0$
- whether or not the data variables are **scaled**, to a variance of $\text{Var}(x_j) = 1$.

It nearly always makes sense to center the variables. The choice of scaling determines whether the correlation matrix is analyzed, so that each variable contributes *equally* to the total variance that is to be accounted for versus analysis of the covariance matrix, where each variable contributes its *own variance* to the total. Analysis of the covariance matrix makes little sense when the variables are measured on different scales², unless you want to interpret total variance on the scales of the different variables.

You don't need to scale your data in advance, but be aware of the options: `prcomp()` has default options `center = TRUE`, `scale. = FALSE`³ so in most cases you should specify `scale. = TRUE`. I mostly use this. The older `princomp()` has only the option `cor = FALSE` which centers the data and uses the covariance matrix, so in most cases the default is OK.

To illustrate, the analysis of the `workers` data presented above used `scale. = FALSE` by default, so the eigenvalues reflected the variances of Experience and Income. The analogous result, using standardized variables (*z-scores*) can be computed in any of the forms shown below, using either `scale. = FALSE` or standardizing first using `scale()`:

```
prcomp(workers[, vars], scale. = TRUE)
#> Standard deviations (1, ..., p=2):
#> [1] 1.361 0.383
#>
#> Rotation (n x k) = (2 x 2):
```

²For example, if two variables in the analysis are height and weight, changing the unit of height from inches to centimeters would multiply its variance by 2.54^2 ; changing weight from pounds to kilograms would divide its variance by 2.2^2 .

³The unfortunate default `scale. = FALSE` was for consistency with S, the precursor to R but in general scaling is usually advisable.

```
#>          PC1    PC2
#> Experience 0.707  0.707
#> Income      0.707 -0.707

# same as (output suppressed):
workers[, vars] |> prcomp(scale. = TRUE) |> invisible()
workers[, vars] |> scale() |> prcomp() |> invisible()
```

In this form, each of Experience and Income have variance = 1, and the "Standard deviations" reported are the square roots ($\sqrt{\lambda_i}$) of the eigenvalues λ_i of the correlation matrix \mathbf{R} . The eigenvalues of a correlation matrix always sum to p , the number of variables. This fact prompted the rough *rule of thumb* to extract principal components whose eigenvalues exceed 1.0, which is their average value, $\bar{\lambda} = (\sum^p \lambda_i)/p = p/p$.

```
prcomp(workers[, vars], scale. = TRUE)$sdev
#> [1] 1.361 0.383

# eigen values of correlation matrix
R <- cor(workers[, vars])
R.eig <- eigen(R)
Lambda <- R.eig$values |> print()
#> [1] 1.853 0.147
sum(Lambda)
#> [1] 2
```

Example: Crime data

The dataset `crime`, analysed in Section 3.11, showed all positive correlations among the rates of various crimes in the corrgram, Figure 3.34. What can we see from a PCA? Is it possible that a few dimensions can account for most of the juice in this data?

In this example, you can easily find the PCA solution using `prcomp()` in a single line in base-R. You need to specify the numeric variables to analyze by their columns in the data frame. The most important option here is `scale. = TRUE`.

```
data(crime, package = "ggbioplot")
crime.pca <- prcomp(crime[, 2:8], scale. = TRUE)
```

The tidy equivalent is more verbose, but also more expressive about what is being done. It selects the variables to analyze by a function, `is.numeric()` applied to each of the columns and feeds the result to `prcomp()`.

```
crime.pca <-
  crime |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)
```

As is typical with models in R, the result, `crime.pca` of `prcomp()` is an object of class "prcomp", a list of components, and there are a variety of methods for "prcomp" objects. Among the simplest is `summary()`, which gives the contributions of each component to the total variance in the dataset.

```
summary(crime.pca) |> print(digits=2)
#> Importance of components:
```

```
#> PC1   PC2   PC3   PC4   PC5   PC6   PC7
#> Standard deviation 2.03 1.11 0.85 0.563 0.508 0.471 0.352
#> Proportion of Variance 0.59 0.18 0.10 0.045 0.037 0.032 0.018
#> Cumulative Proportion 0.59 0.76 0.87 0.914 0.951 0.982 1.000
```

The object, `crime.pca` returned by `prcomp()` is a list of the following the following elements:

```
names(crime.pca)
#> [1] "sdev"      "rotation"   "center"    "scale"     "x"
```

Of these, for n observations and p variables,

- `sdev` is the length p vector of the standard deviations of the principal components (i.e., the square roots $\sqrt{\lambda_i}$ of the eigenvalues of the covariance/correlation matrix). When the variables are standardized, the sum of squares of the eigenvalues is equal to p .
- `rotation` is the $p \times p$ matrix of weights or **loadings** of the variables on the components; the columns are the eigenvectors of the covariance or correlation matrix of the data;
- `x` is the $n \times p$ matrix of **scores** for the observations on the components, the result of multiplying (rotating) the data matrix by the loadings. These are uncorrelated, so `cov(x)` is a $p \times p$ diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i = sdev^2$.
- `center` gives the means of the variables when the option `center.` = `TRUE` (the default)

4.2.4 Visualizing variance proportions: screeplots

For a high-D dataset, such as the crime data in seven dimensions, a natural question is how much of the variation in the data can be captured in 1D, 2D, 3D, ... summaries and views. This is answered by considering the proportions of variance accounted by each of the dimensions, or their cumulative values. The components returned by various PCA methods have (confusingly) different names, so `broom::tidy()` provides methods to unify extraction of these values.

```
(crime.eig <- crime.pca |>
  broom::tidy(matrix = "eigenvalues"))
#> # A tibble: 7 x 4
#>   PC std.dev percent cumulative
#>   <dbl>    <dbl>   <dbl>      <dbl>
#> 1     1     2.03   0.588     0.588
#> 2     2     1.11   0.177     0.765
#> 3     3     0.852  0.104     0.868
#> 4     4     0.563  0.0452    0.914
#> 5     5     0.508  0.0368    0.951
#> 6     6     0.471  0.0317    0.982
#> 7     7     0.352  0.0177    1
```

Then, a simple visualization is a plot of the proportion of variance for each component (or cumulative proportion) against the component number, usually called a **screeplot**. The idea, introduced by Cattell (1966), is that after the largest, dominant components, the remainder should resemble the rubble, or scree formed by rocks falling from a cliff.

From this plot, imagine drawing a straight line through the plotted eigenvalues, starting with the largest one. The typical rough guidance is that the last point to fall on this line represents the last component to extract, the idea being that beyond this, the amount of additional variance explained is non-meaningful. Another rule of thumb is to choose the number of components to extract a desired proportion of total variance, usually in the range of 80 - 90%.

`stats::plot(crime.pca)` would give a bar plot of the variances of the components, however `ggbiplots::ggscreeplot()` gives nicer and more flexible displays as shown in Figure 4.8.

```
p1 <- ggscreeplot(crime.pca) +
  stat_smooth(data = crime.eig |> filter(PC>=4),
              aes(x=PC, y=percent), method = "lm",
              se = FALSE,
              fullrange = TRUE) +
  theme_bw(base_size = 14)

p2 <- ggscreeplot(crime.pca, type = "cev") +
  geom_hline(yintercept = c(0.8, 0.9), color = "blue") +
  theme_bw(base_size = 14)

p1 + p2
```

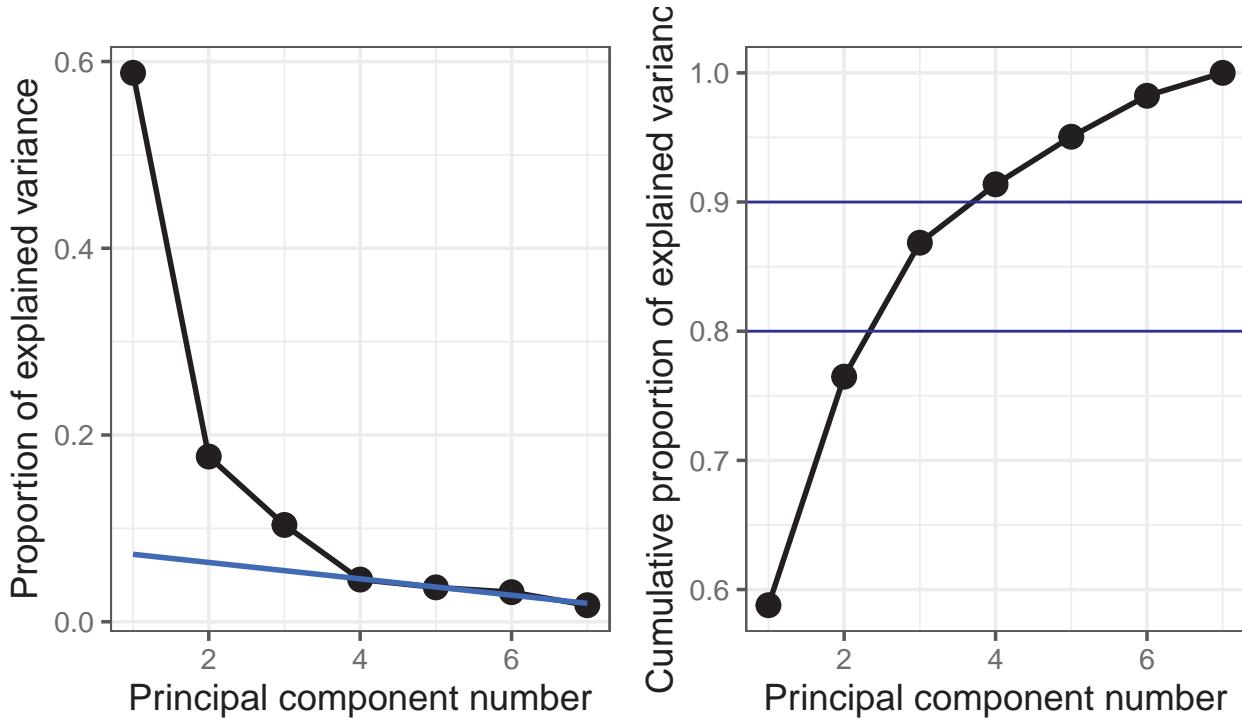


Figure 4.8: Screeplots for the PCA of the crime data. The left panel shows the traditional version, plotting variance proportions against component number, with linear guideline for the scree rule of thumb. The right panel plots cumulative proportions, showing cutoffs of 80%, 90%.

From this we might conclude that four components are necessary to satisfy the scree criterion or to account for 90% of the total variation in these crime statistics. However two components, giving 76.5%, might be enough juice to tell a reasonable story.

4.2.5 Visualizing PCA scores and variable vectors

To see and attempt to understand PCA results, it is useful to plot both the scores for the observations on a few of the largest components and also the loadings or variable vectors that give the weights for the variables in determining the principal components.

In Section 4.3 I discuss the biplot technique that plots both in a single display. However, I do this directly here, using tidy processing to explain what is going on in PCA and in these graphical displays.

Scores

The (uncorrelated) principal component scores can be extracted as `crime.pca$x` or using `purrr::pluck("x")`. As noted above, these are uncorrelated and have variances equal to the eigenvalues of the correlation matrix.

```
scores <- crime.pca |> purrr::pluck("x")
cov(scores) |> zapsmall()
#>      PC1   PC2   PC3   PC4   PC5   PC6   PC7
#> PC1  4.11  0.00  0.00  0.00  0.00  0.00  0.00
#> PC2  0.00  1.24  0.00  0.00  0.00  0.00  0.00
#> PC3  0.00  0.00  0.73  0.00  0.00  0.00  0.00
#> PC4  0.00  0.00  0.00  0.32  0.00  0.00  0.00
#> PC5  0.00  0.00  0.00  0.00  0.26  0.00  0.00
#> PC6  0.00  0.00  0.00  0.00  0.00  0.22  0.00
#> PC7  0.00  0.00  0.00  0.00  0.00  0.00  0.12
```

For plotting, it is more convenient to use `broom::augment()` which extracts the scores (named `.fittedPC*`) and appends these to the variables in the dataset.

```
crime.pca |>
  broom::augment(crime) |> head()
#> # A tibble: 6 x 18
#>   .rownames state    murder  rape robbery assault burglary larceny
#>   <chr>     <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
#> 1 1         Alabama  14.2    25.2   96.8   278.   1136.   1882.
#> 2 2         Alaska   10.8    51.6   96.8   284    1332.   3370.
#> 3 3         Arizona  9.5     34.2   138.   312.   2346.   4467.
#> 4 4         Arkansas 8.8     27.6   83.2   203.   973.    1862.
#> 5 5         California 11.5    49.4   287.   358    2139.   3500.
#> 6 6         Colorado  6.3     42     171.   293.   1935.   3903.
#> # i 10 more variables: auto <dbl>, st <chr>, region <fct>,
#> #   .fittedPC1 <dbl>, .fittedPC2 <dbl>, .fittedPC3 <dbl>,
#> #   .fittedPC4 <dbl>, .fittedPC5 <dbl>, .fittedPC6 <dbl>,
#> #   .fittedPC7 <dbl>
```

Then, we can use `ggplot()` to plot any pair of components. To aid interpretation, I label the points by their state abbreviation and color them by `region` of the U.S.. A geometric interpretation of the plot requires an aspect ratio of 1.0 (via `coord_fixed()`) so that a unit distance on the horizontal axis is the same length as a unit distance on the vertical. To demonstrate that the components are uncorrelated, I also added their data ellipse.

```
crime.pca |>
  broom::augment(crime) |> # add original dataset back in
  ggplot(aes(.fittedPC1, .fittedPC2, color = region)) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_point(size = 1.5) +
  geom_text(aes(label = st), nudge_x = 0.2) +
  stat_ellipse(color = "grey") +
  coord_fixed() +
```

```
labs(x = "PC Dimension 1", y = "PC Dimension 2") +
  theme_minimal(base_size = 14) +
  theme(legend.position = "top")
```

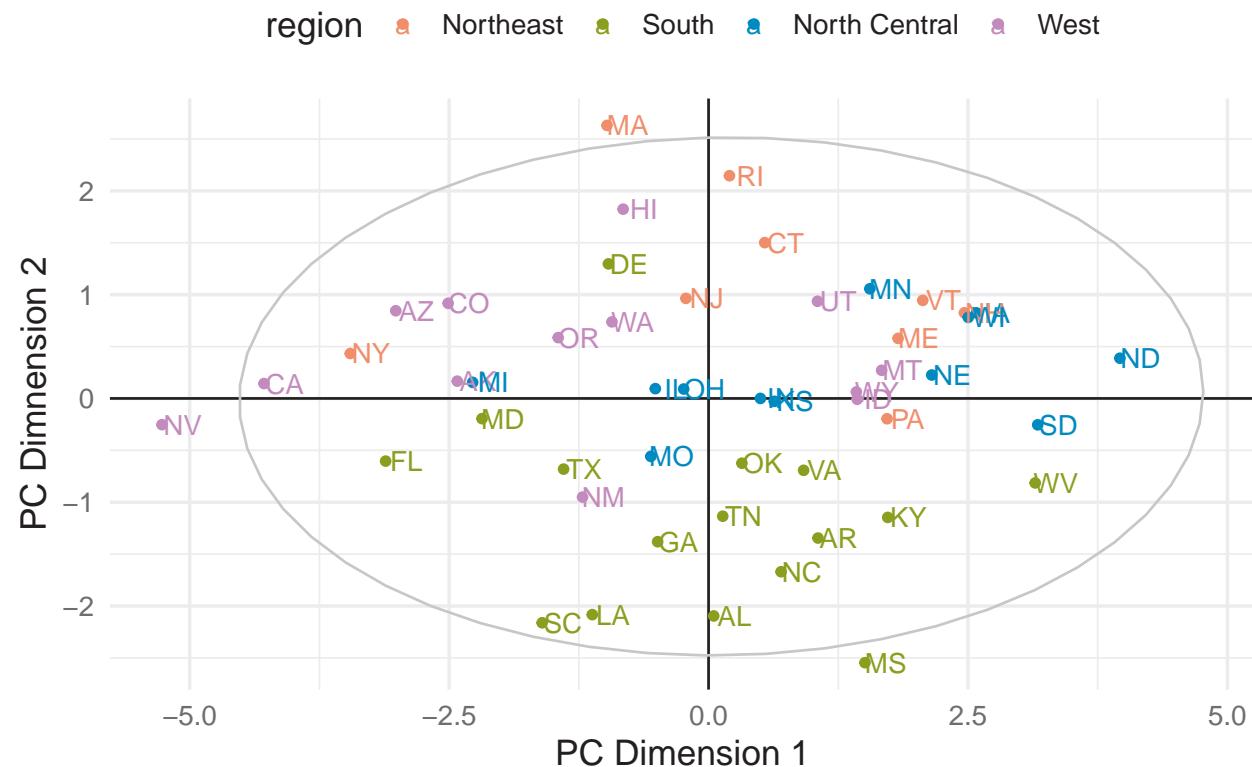


Figure 4.9: Plot of component scores on the first two principal components for the `crime` data. States are colored by `region`.

To interpret such plots, it is useful consider the observations that are a high and low on each of the axes as well as other information, such as region here, and ask how these differ on the crime statistics. The first component, PC1, contrasts Nevada and California with North Dakota, South Dakota and West Virginia. The second component has most of the southern states on the low end and Massachusetts, Rhode Island and Hawaii on the high end. However, interpretation is easier when we also consider how the various crimes contribute to these dimensions.

When, as here, there are more than two components that seem important in the scree plot, we could obviously go further and plot other pairs.

Variable vectors

You can extract the variable loadings using either `crime.pca$rotation` or `purrr::pluck("rotation")`, similar to what I did with the scores.

```
crime.pca |> purrr::pluck("rotation")
#>          PC1      PC2      PC3      PC4      PC5      PC6      PC7
#> murder   -0.300  -0.6292  -0.1782  0.2321  0.5381  0.2591  0.2676
#> rape     -0.432  -0.1694  0.2442 -0.0622  0.1885 -0.7733 -0.2965
#> robbery  -0.397  0.0422 -0.4959  0.5580 -0.5200 -0.1144 -0.0039
#> assault  -0.397 -0.3435  0.0695 -0.6298 -0.5067  0.1724  0.1917
```

```
#> burglary -0.440  0.2033  0.2099  0.0576  0.1010  0.5360 -0.6481
#> larceny   -0.357  0.4023  0.5392  0.2349  0.0301  0.0394  0.6017
#> auto       -0.295  0.5024 -0.5684 -0.4192  0.3698 -0.0573  0.1470
```

But note something important in this output: All of the weights for the first component are negative. In PCA, the directions of the eigenvectors are completely arbitrary, in the sense that the vector $-\mathbf{v}_i$ gives the same linear combination as \mathbf{v}_i , but with its' sign reversed. For interpretation, it is useful (and usually recommended) to reflect the loadings to a positive orientation by multiplying them by -1. In general, you are free to reflect any of the components for ease of interpretation, and not necessarily if all the signs are negative.

To reflect the PCA loadings (multiplying PC1 and PC2 by -1) and get them into a convenient format for plotting with `ggplot()`, it is necessary to do a bit of processing, including making the `row.names()` into an explicit variable for the purpose of labeling.

rownames in R

R software evolved over many years, particularly in conventions for labeling cases in printed output and graphics. In base-R, the convention was that the `row.names()` of a matrix or `data.frame` served as observation labels in all printed output and plots, with a default to use numbers `1:n` if there were no rownames. In `ggplot2` and the `tidyverse` framework, the decision was made that observation labels had to be an **explicit** variable in a “tidy” dataset, so it could be used as a variable in constructs like `geom_text(aes(label = label))` as in this example. This change often requires extra steps in software that uses the rownames convention.

```
vectors <- crime.pca |>
  purrr::pluck("rotation") |>
  as.data.frame() |>
  mutate(PC1 = -1 * PC1, PC2 = -1 * PC2) |>      # reflect axes
  tibble::rownames_to_column(var = "label")

vectors[, 1:3]
#>      label    PC1     PC2
#> 1    murder  0.300  0.6292
#> 2    rape    0.432  0.1694
#> 3    robbery  0.397 -0.0422
#> 4    assault  0.397  0.3435
#> 5    burglary 0.440 -0.2033
#> 6    larceny  0.357 -0.4023
#> 7    auto     0.295 -0.5024
```

Then, I plot these using `geom_segment()`, taking some care to use arrows from the origin with a nice shape and add `geom_text()` labels for the variables positioned slightly to the right. Again, `coord_fixed()` ensures equal scales for the axes, which is important because we want to interpret the angles between the variable vectors and the PCA coordinate axes.

```
arrow_style <- arrow(
  angle = 20, ends = "first", type = "closed",
  length = grid::unit(8, "pt")
)

vectors |>
```

```

ggplot(aes(PC1, PC2)) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_segment(xend = 0, yend = 0,
               linewidth = 1,
               arrow = arrow_style,
               color = "brown") +
  geom_text(aes(label = label),
            size = 5,
            hjust = "outward",
            nudge_x = 0.05,
            color = "brown") +
  ggforce::geom_circle(aes(x0 = 0, y0 = 0, r = 0.5), color = gray(.50)) +
  xlim(-0.5, 0.9) +
  ylim(-0.8, 0.8) +
  coord_fixed() +           # fix aspect ratio to 1:1
  theme_minimal(base_size = 14)

```

The variable vectors (arrows) shown in Figure 4.10 have the following interpretations:

- (1) The lengths of the variable vectors, $\|\mathbf{v}_i\| = \sqrt{\sum_j v_{ij}^2}$ give the relative proportion of variance of each variable accounted for in a two-dimensional display.
- (2) Each vector points in the direction in component space with which that variable is most highly correlated: the value, v_{ij} , of the vector for variable \mathbf{x}_i on component j reflects the correlation of that variable with the j th principal component. Thus,
 - A variable that is perfectly correlated with a component is parallel to it.
 - A variable this is uncorrelated with an component is perpendicular to it.
- (3) The angle between vectors shows the strength and direction of the correlation between those variables: the cosine of the angle θ between two variable vectors, \mathbf{v}_i and \mathbf{v}_j , which is $\cos(\theta) = \mathbf{v}_i' \mathbf{v}_j / \|\mathbf{v}_i\| \cdot \|\mathbf{v}_j\|$ gives the approximation of the correlation r_{ij} between \mathbf{x}_i and \mathbf{x}_j that is shown in this space. This means that:
 - two variable vectors that point in the same direction are highly correlated; $r = 1$ if they are completely aligned.
 - Variable vectors at right angles are approximately uncorrelated, while those pointing in opposite directions are negatively correlated; $r = -1$ if they are at 180° .

To illustrate point (1), the following indicates that almost 70% of the variance of `murder` is represented in the the 2D plot shown in Figure 4.9, but only 40% of the variance of `robbery` is captured. For point (2), the correlation of `murder` with the dimensions is 0.3 for PC1 and 0.63 for PC2. For point (3), the angle between `murder` and `burglary` looks to be about 90° , but the actual correlation is 0.39.

```

vectors |> select(label, PC1, PC2) |>
  mutate(length = sqrt(PC1^2 + PC2^2))
#>   label    PC1    PC2 length
#> 1  murder  0.300  0.6292  0.697
#> 2    rape  0.432  0.1694  0.464
#> 3  robbery  0.397 -0.0422  0.399
#> 4 assault  0.397  0.3435  0.525
#> 5 burglary 0.440 -0.2033  0.485

```

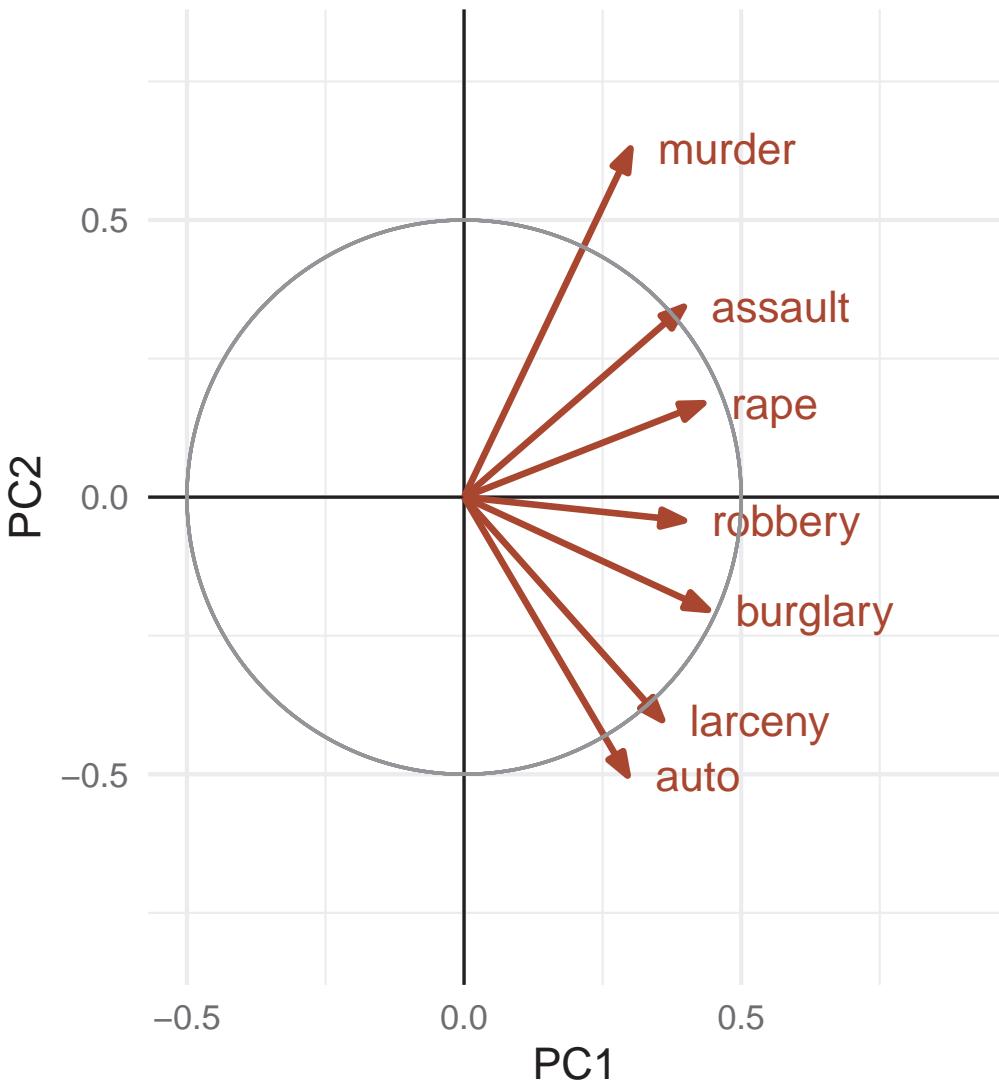


Figure 4.10: Plot of component loadings the first two principal components for the `crime` data. These are interpreted as the contributions of the variables to the components.

```
#> 6  larceny  0.357 -0.4023  0.538
#> 7      auto  0.295 -0.5024  0.583
```

4.3 Biplots

The biplot is a visual multivariate juicer. It is the simple and powerful idea that came from the recognition that you can overlay a plot of observation scores in a principal components analysis with the information of the variable loadings (weights) to give a simultaneous display that is easy to interpret. In this sense, a biplot is generalization of a scatterplot, projecting from data space to PCA space, where the observations are shown by points, as in the plots of component scores in Figure 4.9, but with the variables also shown by vectors (or scaled linear axes aligned with those vectors).

The idea of the biplot was introduced by Ruben Gabriel (1971, 1981) and later expanded in scope by Gower & Hand (1996). The book by Greenacre (2010) gives a practical overview of the many variety of biplots. Gower et al. (2011) *Understanding biplots* provides a full treatment of many topics, including how to calibrate biplot axes, 3D plots, and so forth.

Biplot methodology is far more general than I cover here. Categorical variables can be incorporated in PCA using points that represent the levels of discrete categories. Two-way frequency tables of categorical variables can be analysed using *correspondence analysis*, which is similar to PCA, but designed to account for the maximum amount of the χ^2 statistic for association; *multiple correspondence analysis* extends this to method to multi-way tables (Friendly & Meyer, 2016; Greenacre, 1984).

4.3.1 Constructing a biplot

The biplot is constructed by using the singular value decomposition (SVD) to obtain a low-rank approximation to the data matrix $\mathbf{X}_{n \times p}$ (centered, and optionally scaled to unit variances) whose n rows are the observations and whose p columns are the variables.

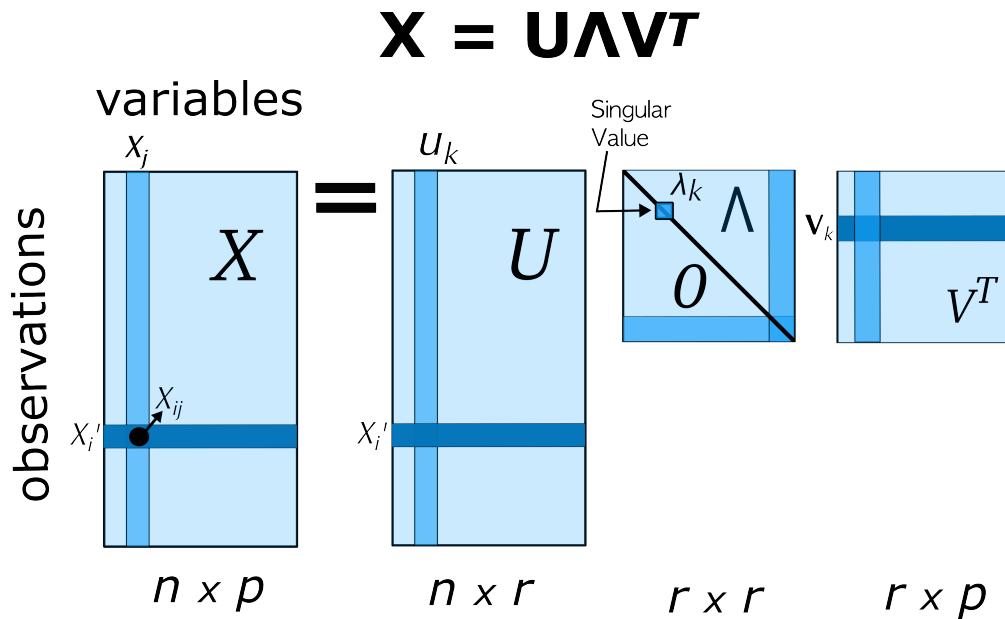


Figure 4.11: The singular value decomposition expresses a data matrix \mathbf{X} as the product of a matrix \mathbf{U} of observation scores, a diagonal matrix Λ of singular values and a matrix \mathbf{V}^T of variable weights.

Using the SVD, the matrix \mathbf{X} , of rank $r \leq p$ can be expressed *exactly* as:

$$\mathbf{X} = \mathbf{U} \Lambda \mathbf{V}^T = \sum_i^r \lambda_i \mathbf{u}_i \mathbf{v}_i^T, \quad (4.4)$$

where

- \mathbf{U} is an $n \times r$ orthonormal matrix of uncorrelated observation scores; these are also the eigenvectors of $\mathbf{X}\mathbf{X}'$.
- Λ is an $r \times r$ diagonal matrix of singular values, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r$, which are also the square roots of the eigenvalues of $\mathbf{X}\mathbf{X}'$.
- \mathbf{V} is an $r \times p$ orthonormal matrix of variable weights and also the eigenvectors of $\mathbf{X}'\mathbf{X}$.

Then, a rank 2 (or 3) PCA approximation $\hat{\mathbf{X}}$ to the data matrix used in the biplot can be obtained from the first 2 (or 3) singular values λ_i and the corresponding $\mathbf{u}_i, \mathbf{v}_i$ as:

$$\mathbf{X} \approx \widehat{\mathbf{X}} = \lambda_1 \mathbf{u}_1 \mathbf{v}'_1 + \lambda_2 \mathbf{u}_2 \mathbf{v}'_2 .$$

The variance of \mathbf{X} accounted for by each term is λ_i^2 .

A biplot is then obtained by overlaying two scatterplots that share a common set of axes and have a between-set scalar product interpretation. Typically, the observations (rows of \mathbf{X}) are represented as points and the variables (columns of \mathbf{X}) are represented as vectors from the origin.

The **scale** factor, α allows the variances of the components to be apportioned between the row points and column vectors, with different interpretations, by representing the approximation $\widehat{\mathbf{X}}$ as the product of two matrices,

$$\widehat{\mathbf{X}} = (\mathbf{U} \boldsymbol{\Lambda}^\alpha) (\boldsymbol{\Lambda}^{1-\alpha} \mathbf{V}') = \mathbf{AB}'$$

This notation uses a little math trick involving a power, $0 \leq \alpha \leq 1$: When $\alpha = 1$, $\boldsymbol{\Lambda}^\alpha = \boldsymbol{\Lambda}^1 = \boldsymbol{\Lambda}$, and $\boldsymbol{\Lambda}^{1-\alpha} = \boldsymbol{\Lambda}^0 = \mathbf{I}$. $\alpha = 1/2$ gives the diagonal matrix $\boldsymbol{\Lambda}^{1/2}$ whose elements are the square roots of the singular values.

The choice $\alpha = 1$ assigns the singular values totally to the left factor; then, the angle between two variable vectors, reflecting the inner product $\mathbf{x}_j^\top, \mathbf{x}_{j'}$ approximates their correlation or covariance, and the distance between the points approximates their Mahalanobis distances. $\alpha = 0$ gives a distance interpretation to the column display. $\alpha = 1/2$ gives a symmetrically scaled biplot. *TODO**: Explain this better.

When the singular values are assigned totally to the left or to the right factor, the resultant coordinates are called *principal coordinates* and the sum of squared coordinates on each dimension equal the corresponding singular value. The other matrix, to which no part of the singular values is assigned, contains the so-called *standard coordinates* and have sum of squared values equal to 1.0.

4.3.2 Biplots in R

There are a large number of R packages providing biplots. The most basic, `stats::biplot()`, provides methods for "prcomp" and "princomp" objects. Among other packages, `factoextra` (Kassambara & Mundt, 2020), an extension of `FactoMineR` (Husson et al., 2025), is perhaps the most comprehensive and provides `ggplot2` graphics. In addition to biplot methods for quantitative data using PCA (`fviz_pca()`), it offers biplots for categorical data using correspondence analysis (`fviz_ca()`) and multiple correspondence analysis (`fviz_mca()`); factor analysis with mixed quantitative and categorical variables (`fviz_fAMD()`) and cluster analysis (`fviz_cluster()`). The `adegraphics` package (Dray & Siberchicot, 2025) produces lovely biplots using `lattice` graphics, but with its own analytic framework.

Here, I use the `ggbio` package (Vu & Friendly, 2024), which aims to provide a simple interface to biplots within the `ggplot2` framework. I also use some convenient utility functions from `factoextra`.

4.3.3 Example: Crime data

A basic biplot of the `crime` data, using standardized principal components and labeling the observation by their state abbreviation is shown in Figure 4.12. The correlation circle reflects the data ellipse of the standardized components. This reminds us that these components are uncorrelated and have equal variance in the display.

```
crime.pca <- reflect(crime.pca) # reflect the axes

ggbio(crime.pca,
      obs.scale = 1, var.scale = 1,
      labels = crime$st ,
      circle = TRUE,
```

```
varname.size = 4,
varname.color = "brown") +
theme_minimal(base_size = 14)
```

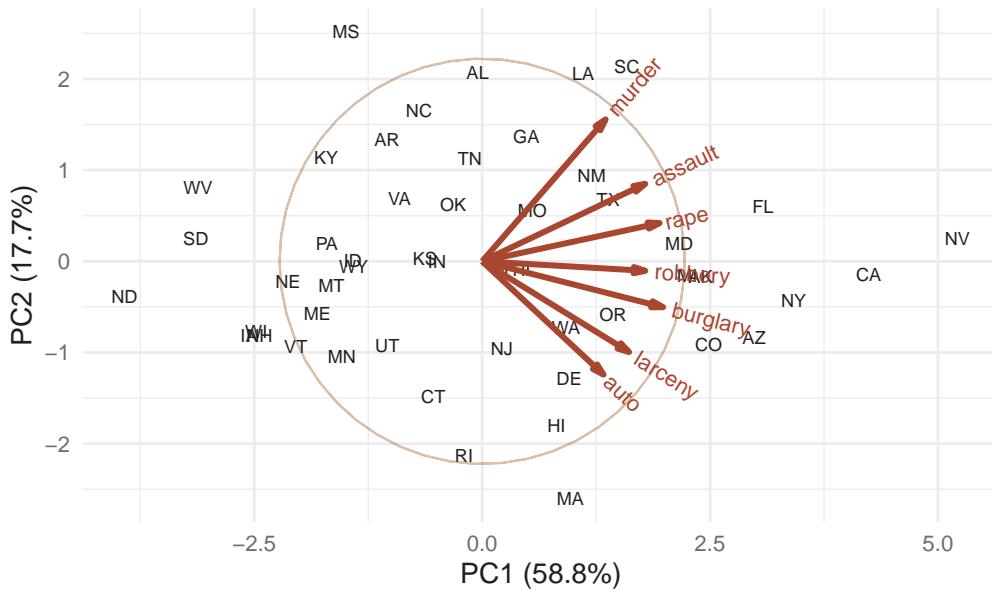


Figure 4.12: Basic biplot of the crime data. State abbreviations are shown at their standardized scores on the first two dimensions. The variable vectors reflect the correlations of the variables with the biplot dimensions.

In this dataset the states are grouped by region and we saw some differences among regions in the plot (Figure 4.9) of component scores. `ggbiplots()` provides options to include a `groups = variable`, used to color the observation points and also to draw their data ellipses, facilitating interpretation.

```
ggbiplots(crime.pca,
obs.scale = 1, var.scale = 1,
groups = crime$region,
labels = crime$st,
labels.size = 4,
var.factor = 1.4,
ellipse = TRUE,
ellipse.prob = 0.5, ellipse.alpha = 0.1,
circle = TRUE,
varname.size = 4,
varname.color = "black",
clip = "off") +
labs(fill = "Region", color = "Region") +
theme_minimal(base_size = 14) +
theme(legend.direction = 'horizontal', legend.position = 'top')
```

This plot provides what is necessary to interpret the nature of the components and also the variation of the states in relation to these. In this, the data ellipses for the regions provide a visual summary that aids interpretation.

- From the variable vectors, it seems that PC1, having all positive and nearly equal loadings, reflects a total

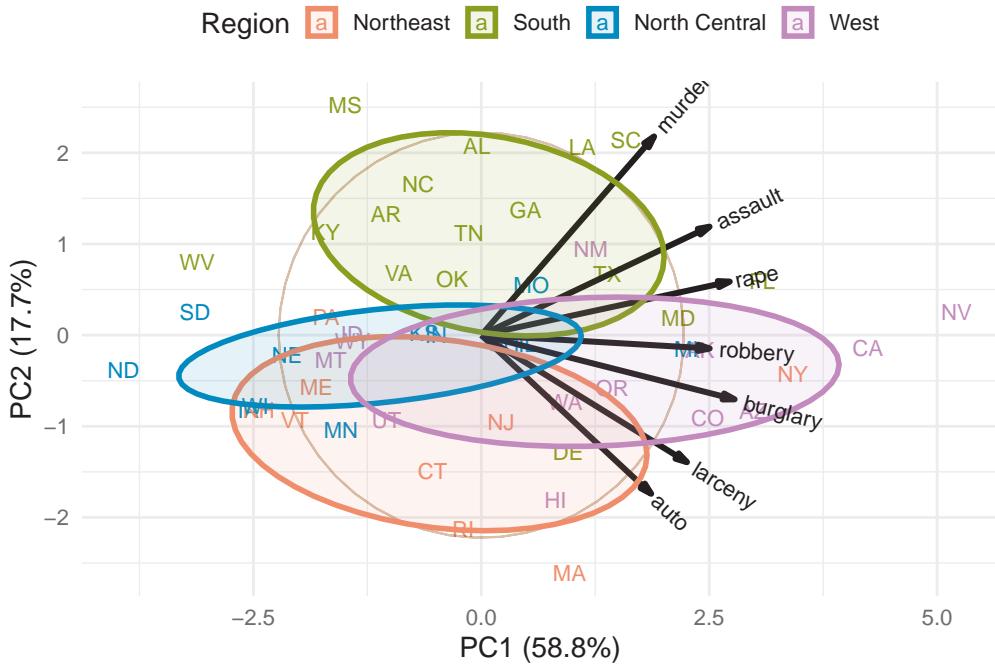


Figure 4.13: Enhanced biplot of the crime data, grouping the states by region and adding data ellipses.

or overall index of crimes. Nevada, California, New York and Florida are highest on this, while North Dakota, South Dakota and West Virginia are lowest.

- The second component, PC2, shows a contrast between crimes against persons (murder, assault, rape) at the top and property crimes (auto theft, larceny) at the bottom. Nearly all the Southern states are high on personal crimes; states in the North East are generally higher on property crimes.
- Western states tend to be somewhat higher on overall crime rate, while North Central are lower on average. In these states there is not much variation in the relative proportions of personal vs. property crimes.

Moreover, in this biplot you can interpret the value for a particular state on a given crime by considering its projection on the variable vector, where the origin corresponds to the mean, positions along the vector have greater than average values on that crime, and the opposite direction have lower than average values. For example, Massachusetts has the highest value on auto theft, but a value less than the mean. Louisiana and South Carolina on the other hand are highest in the rate of murder and slightly less than average on auto theft.

These 2D plots account for only 76.5% of the total variance of crimes, so it is useful to also examine the third principal component, which accounts for an additional 10.4%. The `choices =` option controls which dimensions are plotted.

```
ggbiplots(crime.pca,
  choices = c(1,3),
  obs.scale = 1, var.scale = 1,
  groups = crime$region,
  labels = crime$st,
  labels.size = 4,
  var.factor = 2,
  ellipse = TRUE,
  ellipse.prob = 0.5, ellipse.alpha = 0.1,
```

```

circle = TRUE,
varname.size = 4,
varname.color = "black",
clip = "off") +
labs(fill = "Region", color = "Region") +
theme_minimal(base_size = 14) +
theme(legend.direction = 'horizontal', legend.position = 'top')

```

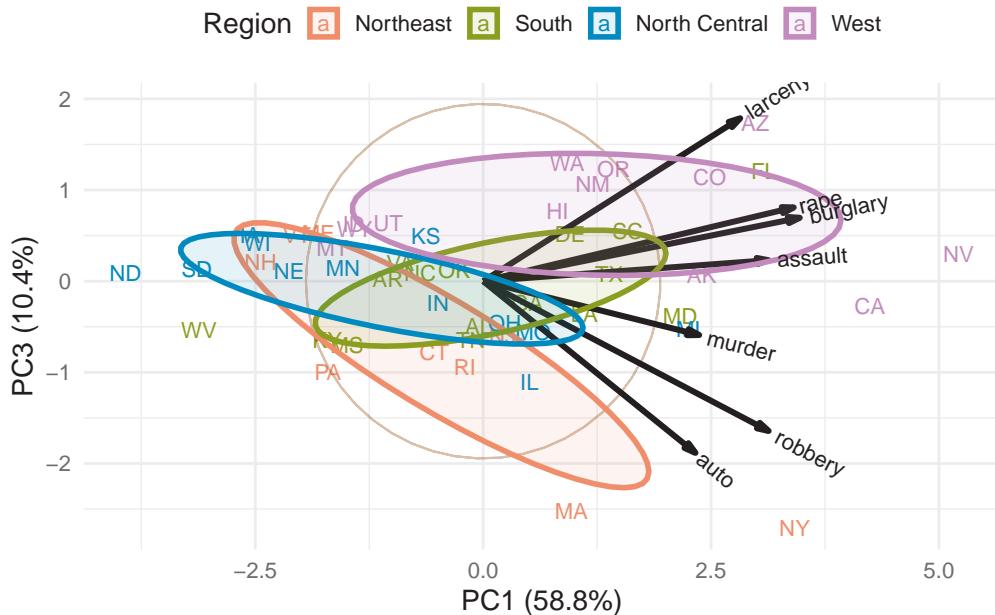


Figure 4.14: Biplot of dimensions 1 & 3 of the crime data, with data ellipses for the regions.

Dimension 3 in Figure 4.14 is more subtle. One interpretation is a contrast between larceny, which is a larceny (simple theft) and robbery, which involves stealing something from a person and is considered a more serious crime with an element of possible violence. In this plot, murder has a relatively short variable vector, so does not contribute very much to differences among the states.

4.3.4 Biplot contributions and quality

To better understand how much each variable contributes to the biplot dimensions, it is helpful to see information about the variance of variables along each dimension. Graphically, this is nothing more than a measure of the lengths of projections of the variables on each of the dimensions. `factoextra::get_pca_var()` calculates a number of tables from a "prcomp" or similar object.

```

var_info <- factoextra::get_pca_var(crime.pca)
names(var_info)
#> [1] "coord"    "cor"       "cos2"      "contrib"

```

The component `cor` gives correlations of the variables with the dimensions and `contrib` gives their variance contributions as percents, where each row and column sums to 100.

```

contrib <- var_info$contrib
cbind(contrib, Total = rowSums(contrib)) |>

```

```

rbind(Total = c(colSums(contrib), NA)) |>
  round(digits=2)
#>           Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7 Total
#> murder      9.02  39.59   3.18   5.39  28.96   6.71   7.16  100
#> rape        18.64   2.87   5.96   0.39   3.55  59.79   8.79  100
#> robbery     15.75   0.18  24.59  31.14  27.04   1.31   0.00  100
#> assault     15.73  11.80   0.48  39.67  25.67   2.97   3.68  100
#> burglary    19.37   4.13   4.41   0.33   1.02  28.73  42.01  100
#> larceny     12.77  16.19  29.08   5.52   0.09   0.16  36.20  100
#> auto         8.71  25.24  32.31  17.58  13.67   0.33   2.16  100
#> Total       100.00 100.00 100.00 100.00 100.00 100.00 100.00   NA

```

These contributions can be visualized as sorted barcharts for a given axis using `factoextra::fviz_contrib()`. The dashed horizontal lines are at the average value for each dimension.

```

p1 <- fviz_contrib(crime.pca, choice = "var", axes = 1,
                    fill = "lightgreen", color = "black")
p2 <- fviz_contrib(crime.pca, choice = "var", axes = 2,
                    fill = "lightgreen", color = "black")
p1 + p2

```

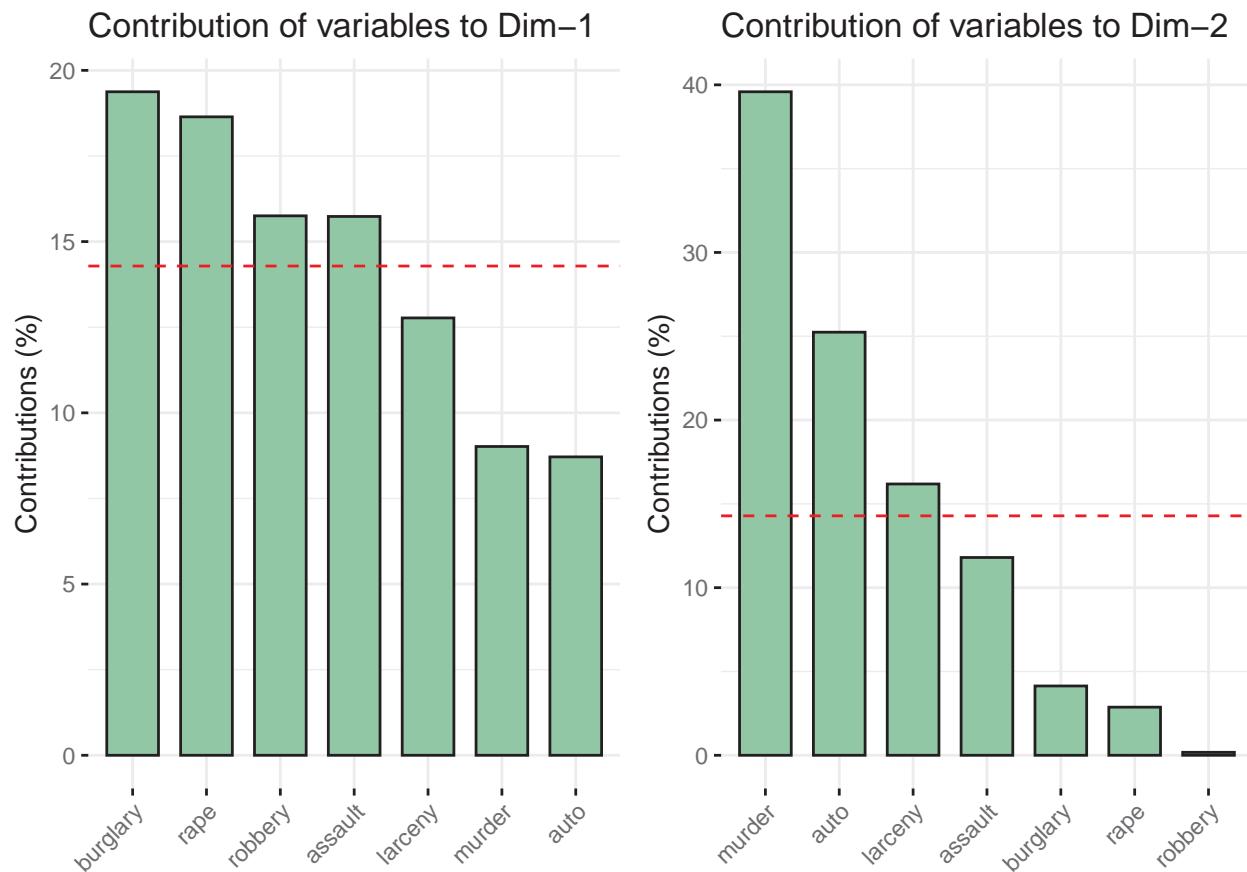


Figure 4.15: Contributions of the crime variables to dimensions 1 (left) & 2 (right) of the PCA solution

A simple rubric for interpreting the dimensions in terms of the variable contributions is to mention those that

are largest or above average on each dimension. So, burglary and rape contribute most to the first dimension, while murder and auto theft contribute most to the second.

Another useful measure is called `cos2`, the *quality* of representation, meaning how much of a variable is represented in a given component. The columns sum to the eigenvalue for each dimension. The rows each sum to 1.0, meaning each variable is completely represented on all components, but we can find the quality of a k -D solution by summing the values in the first k columns. These can be plotted in a style similar to Figure 4.15 using `factoextra::fviz_cos2()`.

```
quality <- var_info$cos2
rowSums(quality)
#>   murder      rape    robbery assault burglary larceny      auto
#>       1         1        1        1        1        1        1

colSums(quality)
#> Dim.1 Dim.2 Dim.3 Dim.4 Dim.5 Dim.6 Dim.7
#> 4.115 1.239 0.726 0.316 0.258 0.222 0.124

cbind(quality[, 1:2],
      Total = rowSums(quality[, 1:2])) |>
  round(digits = 2)
#>           Dim.1 Dim.2 Total
#> murder      0.37  0.49  0.86
#> rape        0.77  0.04  0.80
#> robbery     0.65  0.00  0.65
#> assault     0.65  0.15  0.79
#> burglary    0.80  0.05  0.85
#> larceny     0.53  0.20  0.73
#> auto        0.36  0.31  0.67
```

In two dimensions, murder and burglary are best represented; robbery and larceny are the worst, but as we saw above (Figure 4.14), these crimes are implicated in the third dimension.

4.3.5 Supplementary variables

An important feature of biplot methodology is that once you have a reduced-rank display of the relations among a set of variables, you can use other available data to help interpret what is shown in the biplot. In a sense, this is what I did above in Figure 4.13 and Figure 4.14 using `region` as a grouping variable and summarizing the variability in the scores for states with their data ellipses by region.

When we have other quantitative variables on the same observations, these can be represented as supplementary variables in the same space. Geometrically, this amounts to projecting the new variables on the space of the principal components. It is carried out by regressions of these supplementary variables on the scores for the principal component dimensions.

For example, the left panel of Figure 4.16 depicts the vector geometry of a regression of a variable \mathbf{y} on two predictors, \mathbf{x}_1 and \mathbf{x}_2 . The fitted vector, $\hat{\mathbf{y}}$, is the perpendicular projection of \mathbf{y} onto the plane of \mathbf{x}_1 and \mathbf{x}_2 . In the same way, in the right panel, a **supplementary variable** is projected into the plane of two principal component axes shown as an ellipse. The black fitted vector shows how that additional variable relates to the biplot dimensions.

For this example, it happens that some suitable supplementary variables to aid interpretation of crime rates are available in the dataset `datasets::state.x77`, which was obtained from the U.S. Bureau of the Census *Statistical Abstract of the United States* for 1977. I select a few of these below and make the state name a column variable so it can be merged with the `crime` data.

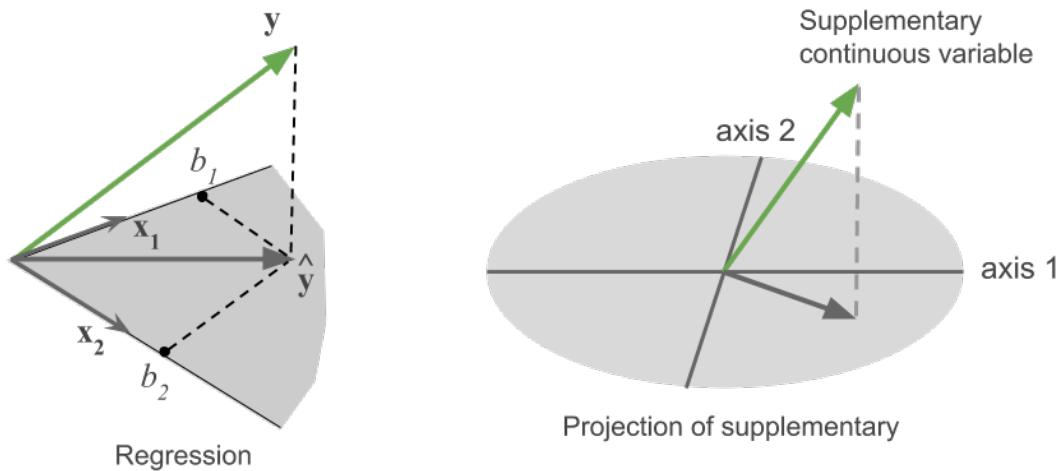


Figure 4.16: Fitting supplementary variables in a biplot is analogous (right) to regression on the principal component dimensions (left). *Source:* Aluja et al. (2018), Figure 2.11

```

supp_data <- state.x77 |>
  as.data.frame() |>
  tibble::rownames_to_column(var = "state") |>
  rename(Life_Exp = `Life_Exp`,
         HS_Grad = `HS Grad`) |>
  select(state, Income:Life_Exp, HS_Grad)

head(supp_data)
#>      state Income Illiteracy Life_Exp HS_Grad
#> 1    Alabama   3624      2.1    69.0   41.3
#> 2     Alaska   6315      1.5    69.3   66.7
#> 3    Arizona   4530      1.8    70.5   58.1
#> 4   Arkansas   3378      1.9    70.7   39.9
#> 5 California   5114      1.1    71.7   62.6
#> 6 Colorado    4884      0.7    72.1   63.9

```

Then, we can merge the `crime` data with the `supp_data` dataset to produce something suitable for analysis using `factoMineR::PCA()`.

```

crime_joined <-
  dplyr::left_join(crime[, 1:8], supp_data, by = "state")
names(crime_joined)
#> [1] "state"        "murder"       "rape"        "robbery"
#> [5] "assault"      "burglary"     "larceny"     "auto"
#> [9] "Income"       "Illiteracy"   "Life_Exp"    "HS_Grad"

```

`PCA()` can only get the labels for the observations from the `row.names()` of the dataset, so I assign them explicitly. The supplementary variables are specified by the argument `quanti.sup` as the indices of the columns in what is passed as the data argument.

```
row.names(crime_joined) <- crime$st
crime.PCA_sup <- PCA(crime_joined[,c(2:8, 9:12)],
                      quanti.sup = 8:11,
                      scale.unit=TRUE,
                      ncp=3,
                      graph = FALSE)
```

The essential difference between the result of `prcomp()` used earlier to get the `crime.pca` object and the result of `PCA()` with supplementary variables is that the `crime.PCA_sup` object now contains a `quanti.sup` component containing the coordinates for the supplementary variables in PCA space.

These can be calculated directly as the coefficients of a multivariate regression of the *standardized* supplementary variables on the PCA scores for the dimensions, with no intercept—which forces the fitted vectors to go through the origin. For example, in the plot below (Figure 4.17), the vector for Income has coordinates (0.192, -0.530) on the first two PCA dimensions.

```
reg.data <- cbind(scale(supp_data[, -1]),
                    crime.PCA_sup$ind$coord) |>
  as.data.frame()

sup.mod <- lm(cbind(Income, Illiteracy, Life_Exp, HS_Grad) ~
               0 + Dim.1 + Dim.2 + Dim.3,
               data = reg.data )

(coefs <- t(coef(sup.mod)))
#>           Dim.1   Dim.2   Dim.3
#> Income      0.192   0.530   0.0482
#> Illiteracy   0.112  -0.536   0.1689
#> Life_Exp    -0.131   0.649  -0.2158
#> HS_Grad     0.103   0.610  -0.4095
```

Note that, because the supplementary variables are standardized, these coefficients are the same as the correlations between the supplementary variables and the scores on the principal components, up to a scaling factor for each dimension. This provides a general way to relate dimensions found in other methods to the original data variables using vectors as in biplot techniques.

```
cor(reg.data[, 1:4], reg.data[, 5:7]) |>
  print() -> R
#>           Dim.1   Dim.2   Dim.3
#> Income      0.393   0.596   0.0415
#> Illiteracy   0.230  -0.602   0.1453
#> Life_Exp    -0.268   0.730  -0.1857
#> HS_Grad     0.211   0.686  -0.3524

R / coefs
#>           Dim.1   Dim.2   Dim.3
#> Income      2.05   1.12  0.861
#> Illiteracy   2.05   1.12  0.861
#> Life_Exp    2.05   1.12  0.861
#> HS_Grad     2.05   1.12  0.861
```

The `PCA()` result can then be plotted using `FactoMiner::plot()` or various `factoextra` functions like

`fviz_pca_var()` for a plot of the variable vectors or `fviz_pca_biplot()` for a biplot. When a `quanti.sup` component is present, supplementary variables are also shown in the displays.

For simplicity I use `FactoMiner::plot()` here and only show the variable vectors. For consistency with earlier plots, I first reflect the orientation of the 2nd PCA dimension so that crimes of personal violence are at the top, as in Figure 4.10.

```
# reverse coordinates of Dim 2
crime.PCA_sup <- ggbioplot::reflect(crime.PCA_sup, columns = 2)
# also reverse the orientation of coordinates for supplementary vars on Dim 2
# crime.PCA_sup$quanti.sup$coord[, 2] <- -crime.PCA_sup$quanti.sup$coord[, 2]
plot(crime.PCA_sup, choix = "var")
```

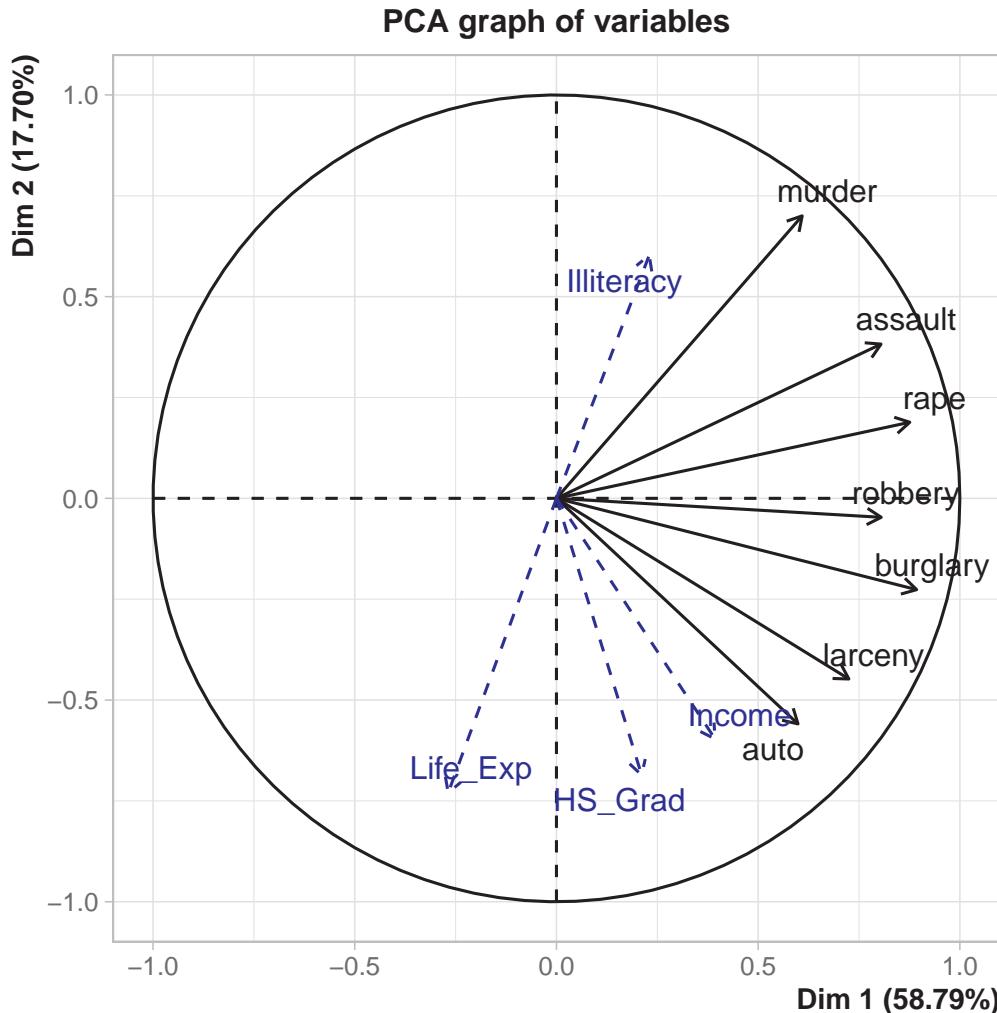


Figure 4.17: PCA plot of variables for the crime data, with vectors for the supplementary variables showing their association with the principal component dimensions.

Recall that from earlier analyses, I interpreted the dominant PC1 dimension as reflecting overall rate of crime. The contributions to this dimension, which are the projections of the variable vectors on the horizontal axis in Figure 4.10 and Figure 4.13 were shown graphically by barcharts in the left panel of Figure 4.15.

But now in Figure 4.17, with the addition of variable vectors for the supplementary variables, you can see

how income, rate of illiteracy, life expectancy and proportion of high school graduates are related to the variation in rates of crimes for the U.S. states.

On dimension 1, what stands out is that life expectancy is associated with lower overall crime, while other supplementary variable have positive associations. On dimension 2, crimes against persons (murder, assault, rape) are associated with greater rates of illiteracy among the states, which as we earlier saw (Figure 4.13) were more often Southern states. Crimes against property (auto theft, larceny) at the bottom of this dimension are associated with higher levels of income and high school graduates.

4.3.6 Example: Diabetes data

As another example, consider the data from Reaven & Miller (1979) on measures of insulin and glucose shown in Figure 4 and that led to the discovery of two distinct types of development of Type 2 diabetes ([?@sec-discoveries](#)). This dataset is available as `Diabetes`. The three groups are `Normal`, `Chemical_Diabetic` and `Overt_Diabetic`, and the (numerical) diagnostic variables are:

- `relwt`: relative weight, the ratio of actual to expected weight, given the person's height,
- `glufast`: fasting blood plasma glucose level
- `glutest`: test blood plasma glucose level, a measure of glucose intolerance
- `instest`: plasma insulin during test, a measure of insulin response to oral glucose
- `sspg`: steady state plasma glucose, a measure of insulin resistance

TODO: Should introduce 3D plots earlier, in Ch3 before Section 3.10.

First, let's try to create a 3D plot, analogous to the artist's drawing from PRIM-9 shown in Figure 5. For this, I use `car::scatter3d()` which can show data ellipsoids summarizing each group. The formula notation, `z ~ x + y` assigns the `z` variable to the vertical direction in the plot, and the `x` and `y` variable form a base plane.

```
cols <- c("darkgreen", "blue", "red")
scatter3d(sspg ~ instest + glutest, data=Diabetes,
          groups = Diabetes$group,
          ellipsoid = TRUE,
          surface = FALSE,
          col = cols,
          surface.col = cols)
```

`car::scatter3d()` uses the `rgl` package (Adler & Murdoch, 2023) to render 3D graphics on a display device, which means that it has facilities for perspective, lighting and other visual properties. You can interactively zoom in or out or rotate the display in any of the three dimensions and use `rgl::spin3d()` to animate rotations around any axes and record this a a `movie3d()`. Figure 4.18 shows two views of this plot, one from the front and one from the back. The data ellipsoids are not as evocative as the artist's rendering, but they give a sense of the relative sizes and shapes of the clouds of points for the three diagnostic groups.

The normal group is concentrated near the origin, with relatively low values on all three diagnostic measures. The chemical diabetic group forms a wing with higher values on insulin response to oral glucose (`instest`), while the overt diabetics form the other wing, with higher values on glucose intolerance (`glutest`). The relative sizes and orientations of the data ellipsoids are also informative.

Given this, what can we see in a biplot view based on PCA? The PCA of these data shows that 83% of the variance is captured in two dimensions and 96% in three. The result for 3D is interesting, in that the view from PRIM-9 shown in Figure 5 and Figure 4.18 nearly captured all available information.

```
data(Diabetes, package="heplots")

diab.pca <-
```

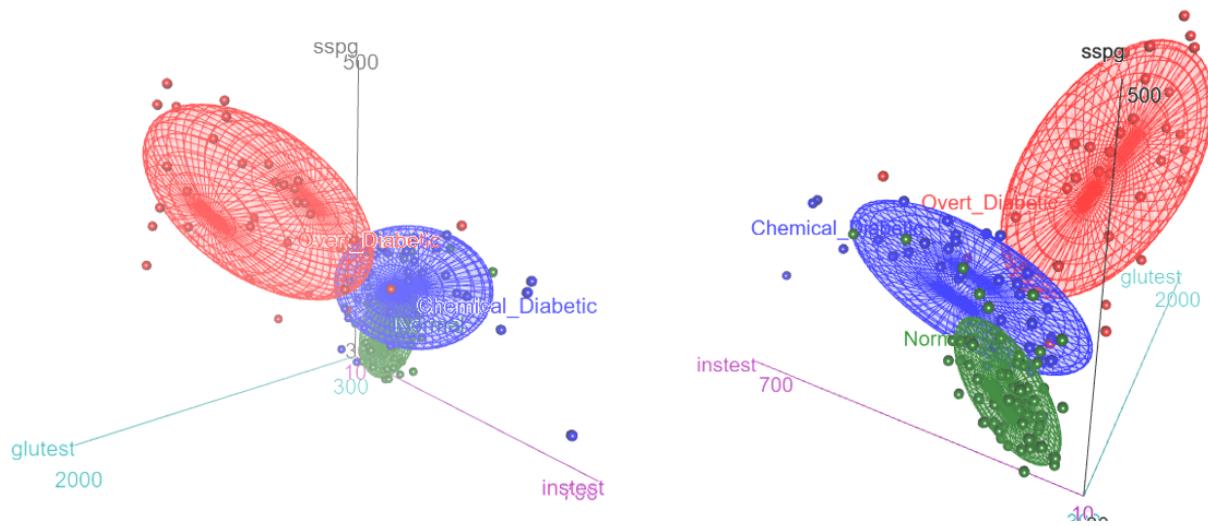


Figure 4.18: Two views of a 3D scatterplot of three main diagnostic variables in the `Diabetes` dataset. The left panel shows an orientation similar to that of Figure 5; the right panel shows a view from the back.

```
Diabetes |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)
summary(diab.pca)

## Importance of components:
##                               PC1    PC2    PC3    PC4    PC5
## Standard deviation     1.662  1.177  0.818  0.3934 0.17589
## Proportion of Variance 0.552  0.277  0.134  0.0309 0.00619
## Cumulative Proportion   0.552  0.829  0.963  0.9938 1.00000
```

A 2D biplot, with data ellipses for the groups, can be produced as before, but I also want to illustrate labeling the groups directly, rather than in a legend.

```
plt <- ggbiplot(diab.pca,
  obs.scale = 1, var.scale = 1,
  groups = Diabetes$group,
  var.factor = 1.4,
  ellipse = TRUE,
  ellipse.prob = 0.5, ellipse.alpha = 0.1,
  circle = TRUE,
  point.size = 2,
  varname.size = 4) +
  labs(fill = "Group", color = "Group") +
  theme_minimal(base_size = 14) +
  theme(legend.position = "none")
```

Then, find the centroids of the component scores and use `geom_label()` to plot the group labels.

```
scores <- data.frame(diab.pca$x[, 1:2], group = Diabetes$group)
centroids <- scores |>
```

```
group_by(group) |>
  summarize(PC1 = mean(PC1),
            PC2 = mean(PC2))

plt + geom_label(data = centroids,
                  aes(x = PC1, y = PC2,
                      label=group, color = group),
                  nudge_y = 0.2)
```

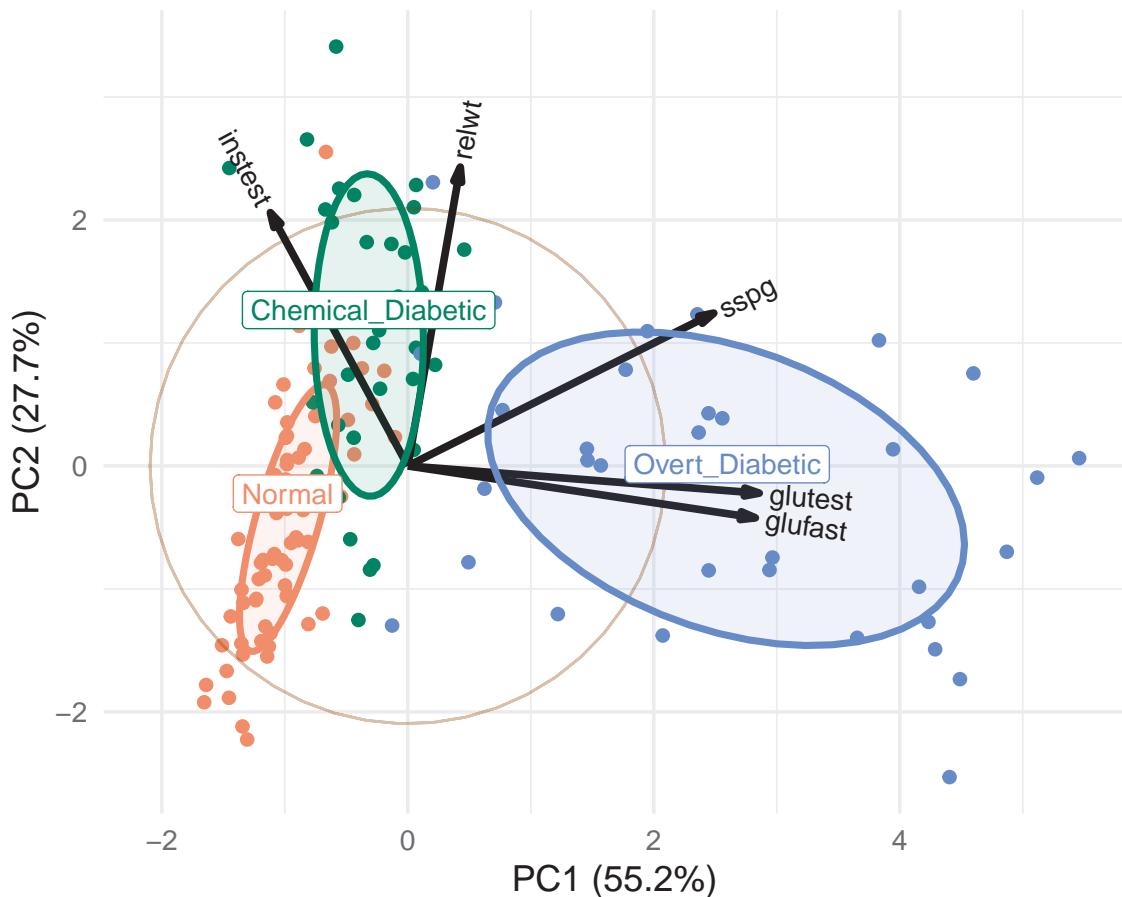


Figure 4.19: 2D biplot of the Diabetes data

What can we see here, and how does it relate to the artist's depiction in Figure 5? The variables `institest`, `sspg` and `glutest` correspond approximately to the coordinate axes in the artist's drawing. `glutest` and `glufast` primarily separate the overt diabetics from the others. The chemical diabetics are distinguished by having larger values of insulin response (`institest`) and are also higher in relative weight (`relwt`).

4.4 Nonlinear dimension reduction

The world of dimension reduction methods reflected by PCA is a simple and attractive one in which relationships among variable are at least approximately linear, and can be made visible in a lower-dimensional view by linear transformations and projections. PCA does an optimal job of capturing *global* linear relationships

in the data. But many phenomena defy linear description or involve *local* nonlinear relationships and clusters within the data. Our understanding of high-D data can sometimes be improved by nonlinear dimension reduction techniques.

To see why, consider the data shown in the left panel of Figure 4.20 and suppose we want to be able to separate the two classes by a line. The groups are readily seen in this simple 2D example, but there is no linear combination or projection that shows them as distinct categories. The right panel shows the same data after a nonlinear transformation to polar coordinates, where the two groups are readily distinguished by radius. Such problems arise in higher dimensions where direct visualization is far more difficult and nonlinear methods become attractive.

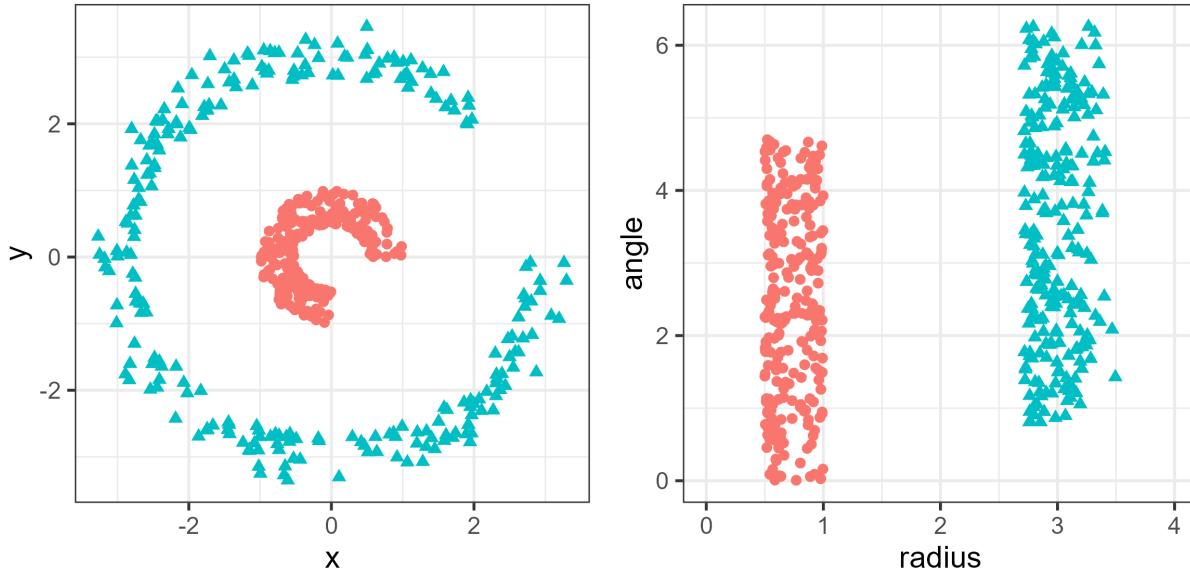


Figure 4.20: *Nonlinear patterns*: Two representations of the same data are shown. In the plot at the left, the clusters are clear to the eye, but there is no linear relation that separates them. Transforming the data nonlinearly, to polar coordinates in the plot at the right, makes the two groups distinct.

4.4.1 Multidimensional scaling

One way to break out of the “linear-combination, maximize-variance PCA” mold is to consider a more intrinsic property of points in *Spaceland*: similarity or distance. The earliest expression of this idea was in **multidimensional scaling** (MDS) by Torgerson (1952), which involved trying to determine a metric low-D representation of objects from their interpoint distances via an application of the SVD.

The break-through for nonlinear methods came from Roger Shepard and William Kruskal (Kruskal, 1964; Shepard, 1962a, 1962b) who recognized that a more general, *nonmetric* version (nMDS) could be achieved using only the *rank order* of input distances d_{ij} among objects. nMDS maps these into a low-D spatial representation of points, $\mathbf{x}_i, \mathbf{x}_j$ whose fitted distances, $\hat{d}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ matches the order of the d_{ij} as closely as possible. That is, rather than assume that the observed distances are linearly related to the fitted \hat{d}_{ij} , nMDS assumes only that their order is the same. Borg & Groenen (2005) and Borg et al. (2018) give a comprehensive overview of modern developments in MDS.

The impetus for MDS stemmed largely from psychology and the behavioral sciences, where simple experimental measures of similarity or dissimilarity of psychological objects (color names, facial expressions, words, Morse code symbols) could be obtained by direct ratings, confusions, or other tasks (Shepard et al., 1972b, 1972a). MDS was revolutionary in that it provided a coherent method to study the dimensions of perceptual and

cognitive space in applications where the explanation of a cognitive process was derived directly from an MDS solution ([Shoben, 1983](#)).

To perform nMDS, you need to calculate the matrix of distances between all pairs of observations (`dist()`). The basic function is `MASS::isoMDS()`.⁴ In the call, you can specify the number of dimensions (`k`) desired, with `k=2` as default. It returns the coordinates in a dataset called `points`.

```
diab.dist <- dist(Diabetes[, 1:5])
mds <- diab.dist |>
  MASS::isoMDS(k = 2, trace = FALSE) |>
  purrr::pluck("points")

colnames(mds) <- c("Dim1", "Dim2")
mds <- bind_cols(mds, group = Diabetes$group)
mds |> sample_n(6)

#> # A tibble: 6 x 3
#>   Dim1    Dim2 group
#>   <dbl>   <dbl> <fct>
#> 1 -213.   -42.1 Normal
#> 2  191.    47.3 Overt_Diabetic
#> 3  12.0   -63.2 Overt_Diabetic
#> 4  25.0   -38.1 Chemical_Diabetic
#> 5 774.     9.44 Overt_Diabetic
#> 6  79.0   136.  Overt_Diabetic
```

The method works by trying to minimize a measure, “Stress”, of the average difference between the fitted distances \hat{d}_{ij} and an optimal monotonic (order-preserving) transformation, $f_{\text{mon}}(d_{ij})$, of the distances in the data. Values of Stress around 5-8% and smaller are generally considered adequate.

Unlike PCA, where you can fit all possible dimensions once and choose the number of components to retain by examining the eigenvalues or variance proportions, in MDS it is necessary to fit the data for several values of `k` and consider the trade-off between goodness of fit and complexity.

```
stress <- vector(length = 5)
for(k in 1:5){
  res <- MASS::isoMDS(diab.dist, k=k, trace = FALSE)
  stress[k] <- res$stress
}
round(stress, 3)
#> [1] 17.755  3.525  0.256  0.000  0.000
```

Plotting these shows that a 3D solution is nearly perfect, while a 2D solution is certainly adequate. This plot is the MDS analog of a screeplot for PCA.

```
plot(stress, type = "b", pch = 16, cex = 2,
      xlab = "Number of dimensions",
      ylab = "Stress (%)")
```

To plot the 2D solution, I'll use `ggpubr::ggscatter()` here because it handles grouping, provides concentration ellipses and other graphical features.

⁴The `vegan` package ([Oksanen et al., 2025](#)) provides `vegan::metaMDS()` which allows a wide range of distance measures ...

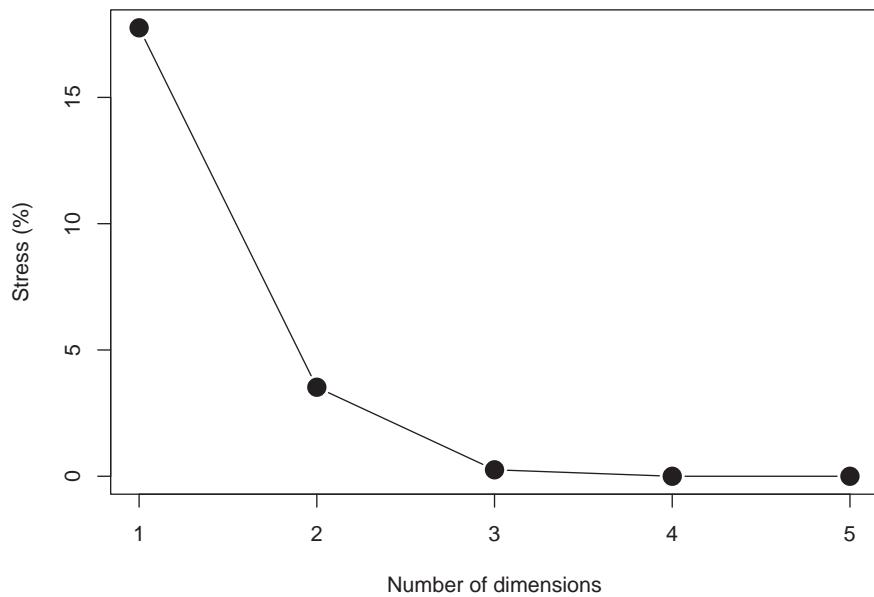


Figure 4.21: Badness of fit (Stress) of the MDS solution in relation to number of dimensions.

```
library(ggpubr)
cols <- scales::hue_pal()(3) |> rev()
mplot <-
ggscatter(mds, x = "Dim1", y = "Dim2",
          color = "group",
          shape = "group",
          palette = cols,
          size = 2,
          ellipse = TRUE,
          ellipse.level = 0.5,
          ellipse.type = "t") +
geom_hline(yintercept = 0, color = "gray") +
geom_vline(xintercept = 0, color = "gray")
```

For this and other examples using MDS, it would be nice to also show how the dimensions of this space relate to the original variables, as in a biplot. Using the idea of correlations between variables and dimensions from Section 4.3.5, I do this as shown below. Only the relative directions and lengths of the variable vectors matter, so you can choose any convenient scale factor to make the vectors fill the plot region.

```
vectors <- cor(Diabetes[, 1:5], mds[, 1:2])
scale_fac <- 500
mplot +
coord_fixed() +
geom_segment(data=vectors,
             aes(x=0, xend=scale_fac*Dim1, y=0, yend=scale_fac*Dim2),
             arrow = arrow(length = unit(0.2, "cm"), type = "closed"),
             linewidth = 1.1) +
geom_text(data = vectors,
          aes(x = 1.15*scale_fac*Dim1, y = 1.07*scale_fac*Dim2,
              label=row.names(vectors)),
```

```

nudge_x = 4,
size = 4) +
theme(legend.position = "inside",
legend.position.inside = c(.8, .8))

```

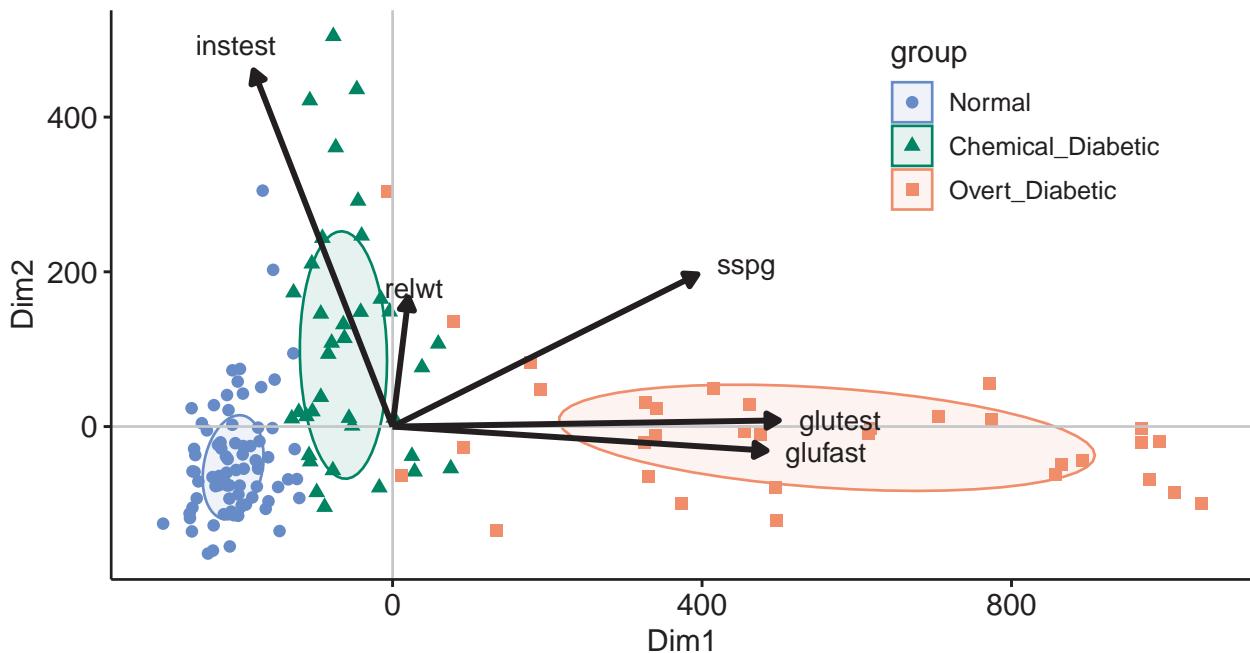


Figure 4.22: Nonmetric MDS representation of the Diabetes data. The vectors reflect the correlations of the variables with the MDS dimensions.

The configuration of the groups in Figure 4.22 is similar to that of the biplot in Figure 4.19, but the groups are more widely separated along the first MDS dimension. The variable vectors are also similar, except that *relwt* is not well-represented in the MDS solution.

4.4.2 t-SNE

With the rise of “machine learning” methods for “feature extraction” in “supervised” vs. “unsupervised” settings, a variety of new algorithms have been proposed for the task of finding low-D representations of high-D data. Among these, t-distributed Stochastic Neighbor Embedding (t-SNE) developed by Maaten & Hinton (2008) is touted as method for revealing *local structure* and clustering better in possibly complex high-D data and at different scales.

t-SNE differs from MDS in what it tries to preserve in the mapping to low-D space: Multidimensional scaling aims to preserve the distances between pairs of data points, focusing on pairs of *distant points* in the original space. t-SNE, on the other hand focuses on preserving *neighboring* data points. Data points that are close in the original data space will be tight in the t-SNE embeddings.

“The t-SNE algorithm models the probability distribution of neighbors around each point. Here, the term *neighbors* refers to the set of points which are closest to each point. In the original, high-dimensional space, this is modeled as a Gaussian distribution. In the 2-dimensional output space this is modeled as a *t*-distribution. The goal of the procedure is to find a mapping onto the 2-dimensional space that minimizes the differences between these two distributions over all points. The fatter tails of a *t*-distribution compared to a Gaussian help to spread the points more evenly in the 2-dimensional space.” (Jake Hoare, [How t-SNE works and Dimensionality Reduction](#)).

t-SNE also uses the idea of mapping distance measures into a low-D space, but converts Euclidean distances into conditional probabilities. Stochastic neighbor embedding means that t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects are assigned a higher probability while dissimilar points are assigned a lower probability.

As van der Maaten and Hinton explained: “The similarity of datapoint \mathbf{x}_j to datapoint \mathbf{x}_i is the conditional probability, $p_{j|i}$, that \mathbf{x}_i would pick \mathbf{x}_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian distribution centered at \mathbf{x}_i .” For $i \neq j$, they define:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)} .$$

and set $p_{i|i} = 0$. σ_i^2 is the variance of the normal distribution that centered on datapoint \mathbf{x}_i and serves as a tuning bandwidth so smaller values of σ_i are used in denser parts of the data space. These conditional probabilities are made symmetric via averaging, giving $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$.

t-SNE defines a similar probability distribution q_{ij} over the points \mathbf{y}_i in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map,

$$D_{\text{KL}}(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} ,$$

a measure of how different the distribution of P in the data is from that of Q in the low-D representation. The t in t-SNE comes from the fact that the probability distribution of the points \mathbf{y}_i in the embedding space is taken to be a heavy-tailed $t_{(1)}$ distribution with one degree of freedom to spread the points more evenly in the 2-dimensional space, rather than the Gaussian distribution for the points in the high-D data space.

t-SNE is implemented in the `Rtsne` package (Krijthe, 2023) which is capable of handling thousands of points in very high dimensions. It uses a tuning parameter, “perplexity” to choose the bandwidth σ_i^2 for each point. This value effectively controls how many nearest neighbors are taken into account when constructing the embedding in the low-dimensional space. It can be thought of as the means to balance between preserving the global and the local structure of the data.⁵

`Rtsne::Rtsne()` finds the locations of the points in the low-D space, of dimension `k=2` by default. It returns the coordinates in a component named `Y`. The package has no `print()`, `summary()` or plot methods, so you’re on your own.

```
library(Rtsne)
set.seed(123)
diab.tsne <- Rtsne(Diabetes[, 1:5], scale = TRUE)
df2 <- data.frame(diab.tsne$Y, group = Diabetes$group)
colnames(df2) <- c("Dim1", "Dim2", "group")
```

You can plot this as shown below:

```
p2 <- ggplot(df2, aes(x=Dim1, y=Dim2, color = group, shape=group)) +
  geom_point(size = 3) +
  stat_ellipse(level = 0.68, linewidth=1.1) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
```

⁵The usual default, `perplexity = 30` focuses on preserving the distances to the 30 nearest neighbors and puts virtually no weight on preserving distances to the remaining points. For data sets with a small number of points e.g. $n = 100$, this will uncover the global structure quite well since each point will preserve distances to a third of the data set. For larger problems, e.g., $n = 10,000$ points, using a higher perplexity value e.g. 500, will do a much better job for of uncovering the global structure. (This description comes from <https://opentsne.readthedocs.io/en/latest/parameters.html>)

```

scale_color_manual(values = cols) +
  labs(x = "Dimension 1",
       y = "Dimension 2") +
  ggtitle("tSNE") +
  theme_bw(base_size = 16) +
  theme(legend.position = "bottom")
p2

```

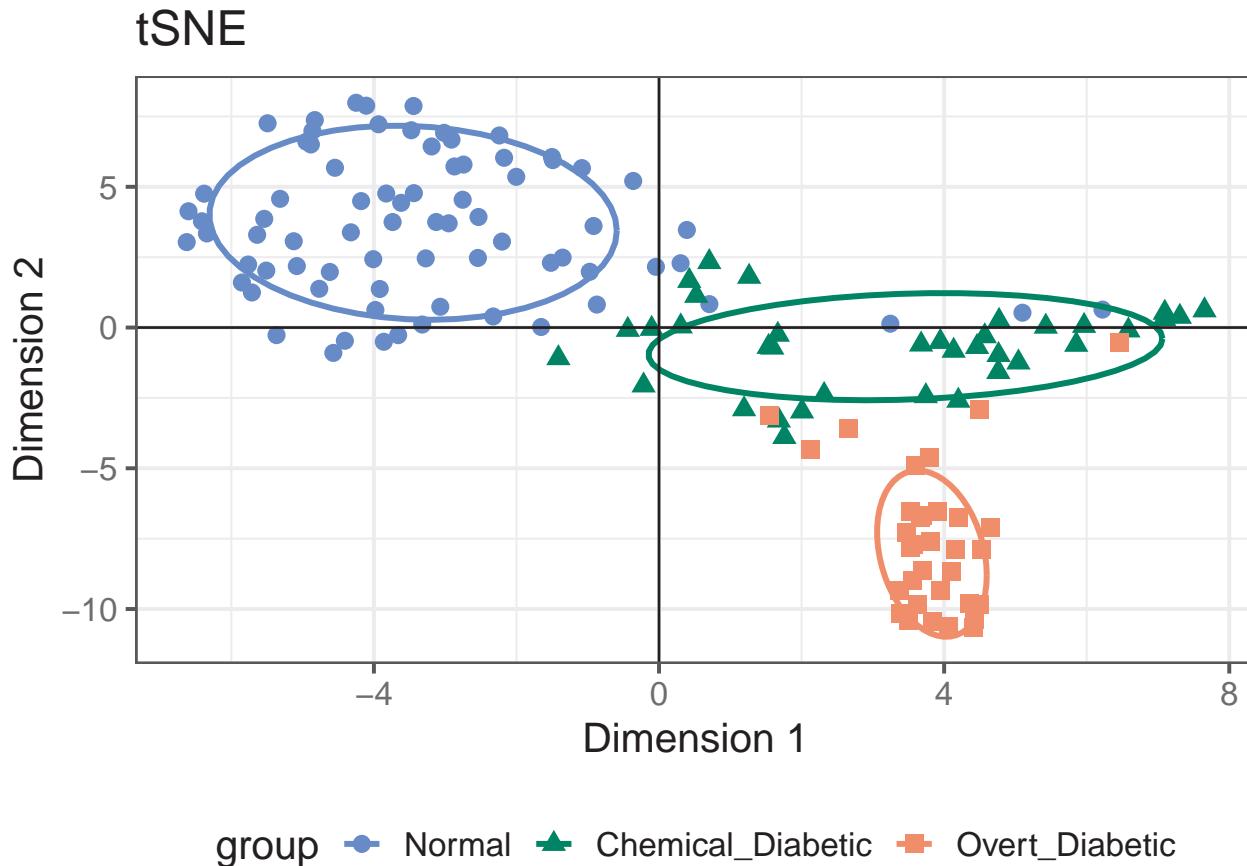


Figure 4.23: t-SNE representation of the Diabetes data.

4.4.2.1 Comparing solutions

For the Diabetes data, I've shown the results of three different dimension reduction techniques, PCA (Figure 4.19), MDS (Figure 4.22), and t-SNE (Figure 4.23). How are these similar, and how do they differ?

One way is to view them side by side as shown in Figure 4.24. To an initial glance, the t-SNE solution looks like a rotated version of the PCA solution, but there are differences in the shapes of the clusters as well.

Another way to compare these two views is to animate the transition from the PCA to the t-SNE representation by a series of smooth interpolated views. This is a more generally useful visualization technique, so it is useful to spell out the details.

The essential idea is calculate interpolated views as a weighted average of the two endpoints using a weight γ that is varied from 0 to 1.

$$\mathbf{X}_{\text{View}} = \gamma \mathbf{X}_{\text{PCA}} + (1 - \gamma) \mathbf{X}_{\text{t-SNE}}$$

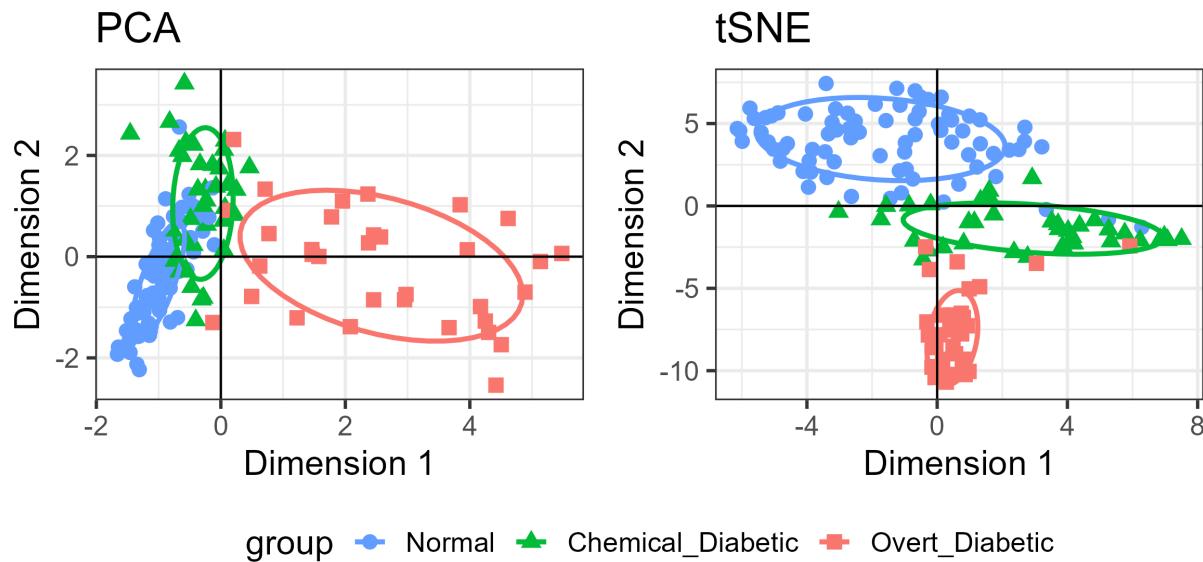


Figure 4.24: Comparison of the PCA and t-SNE 2D representations of the Diabetes data.

The same idea can be applied to other graphical features: lines, paths (ellipses), and so forth. These methods are implemented in the [gganimate](#) package ([Pedersen & Robinson, 2025](#)) .

In this case, to create an animation you can extract the coordinates for the PCA, \mathbf{X}_{PCA} , as a data.frame `df1`, and those for the t-SNE, $\mathbf{X}_{\text{t-SNE}}$ as `df2`, each with a constant `method` variable. These two are then stacked (using `rbind()`) to give a combined `df3`. The animation can then interpolate over `method` going from pure PCA to pure t-SNE.

```
diab.pca <- prcomp(Diabetes[, 1:5], scale = TRUE, rank.=2)
df1 <- data.frame(diab.pca$x, group = Diabetes$group)
colnames(df1) <- c("Dim1", "Dim2", "group")
df1 <- cbind(df1, method="PCA")

set.seed(123)
diab.tsne <- Rtsne(Diabetes[, 1:5], scale = TRUE)
df2 <- data.frame(diab.tsne$Y, group = Diabetes$group)
colnames(df2) <- c("Dim1", "Dim2", "group")
df2 <- cbind(df2, method="tSNE")

# stack the PCA and t-SNE solutions
df3 <- rbind(df1, df2)
```

Then, plot the configuration of the points and add data ellipses as before. The key thing for animating the difference between the solutions is to add `transition_states(method, ...)`, tweening from PCA to t-SNE. The `state_length` argument `transition_states()` controls the relative length of the pause between states.

This animated graphic is shown only in the online version of the book.

```
library(gganimate)
animated_plot <-
```

```

ggplot(df3, aes(x=Dim1, y=Dim2, color=group, shape=group)) +
  geom_point(size = 3) +
  stat_ellipse(level = 0.68, linewidth=1.1) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  scale_color_manual(values = cols) +
  labs(title = "PCA vs. tSNE Dimension Reduction: {closest_state}",
       subtitle = "Frame {frame} of {nframes}",
       x = "Dimension 1",
       y = "Dimension 2") +
  transition_states( method, transition_length = 3, state_length = 2 ) +
  view_follow() +
  theme_bw(base_size = 16) +
  theme(legend.position = "bottom")

animated_plot

```

4.5 Application: Variable ordering for data displays

In many multivariate data displays, such as scatterplot matrices, parallel coordinate plots and others reviewed in Chapter 3, the order of different variables might seem arbitrary. They might appear in alphabetic order, or more often in the order they appear in your dataset, for example when you use `pairs(mydata)`. Yet, the principle of *effect ordering* (Friendly & Kwan (2003)) for variables says you should try to arrange the variables so that adjacent ones are as similar as possible.⁶

For example, the `mtcars` dataset contains data on 32 automobiles from the 1974 U.S. magazine *Motor Trend* and consists of fuel consumption (`mpg`) and 10 aspects of automobile design (`cyl`: number of cylinders; `hp`: horsepower, `wt`: weight) and performance (`qsec`: time to drive a quarter-mile). What can we see from a simple `corrplot()` of their correlations? No coherent pattern stands out in Figure 4.25.

```

data(mtcars)
library(corrplot)
R <- cor(mtcars)
corrplot(R,
         method = 'ellipse',
         title = "Dataset variable order",
         tl.srt = 0, tl.col = "black", tl.pos = 'd',
         mar = c(0,0,1,0))

```

In this display you can scan the rows and columns to “look up” the sign and approximate magnitude of a given correlation; for example, the correlation between `mpg` and `cyl` appears to be about -0.9, while that between `mpg` and `gear` is about 0.5. Of course, you could print the correlation matrix to find the actual values (-0.86 and 0.48 respectively):

⁶The general topic of arranging items (variables, factor values) in an orderly sequence is called *seriation*, and stems from methods of dating in archaeology, used to arrange stone tools, pottery fragments, and other artifacts in time order. In R, the `seriation` package (Hahsler et al., 2024) provides a wide range of techniques. . . .

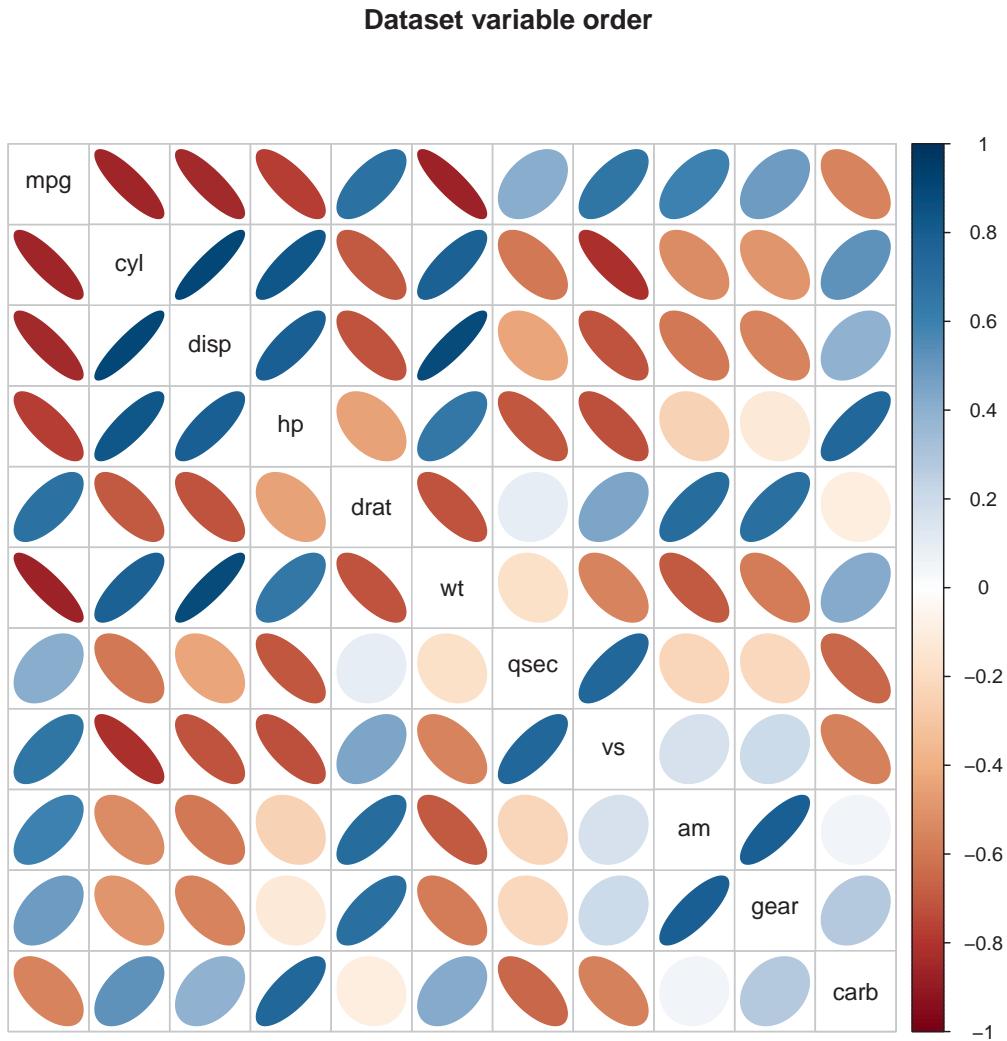


Figure 4.25: Corrplot of `mtcars` data, with the variables arranged in the order they appear in the dataset.

```
print(floor(100*R))
#>      mpg cyl disp  hp drat   wt qsec   vs   am gear carb
#> mpg  100 -86 -85 -78   68 -87   41   66   59   48 -56
#> cyl   -86 100  90  83  -70   78  -60  -82  -53  -50   52
#> disp   90 100  79  79  -72   88  -44  -72  -60  -56   39
#> hp    -78  83  79 100  -45   65  -71  -73  -25  -13   74
#> drat   68 -70  -72 -45  100 -72    9   44   71   69 -10
#> wt    -87  78   88  65  -72  100  -18  -56  -70  -59   42
#> qsec   41 -60  -44 -71    9  -18  100   74  -23  -22  -66
#> vs     66 -82  -72 -73   44  -56   74  100   16   20  -57
#> am     59 -53  -60 -25   71  -70  -23   16  100   79    5
#> gear   48 -50  -56 -13   69  -59  -22   20   79  100   27
#> carb  -56  52   39  74  -10   42  -66  -57    5   27  100
```

Because the angles between variable vectors in the biplot reflect their correlations, Friendly & Kwan (2003) defined **principal component variable ordering** as the order of angles, a_i of the first two eigenvectors, $\mathbf{v}_1, \mathbf{v}_2$ around the unit circle. These values are calculated going counter-clockwise from the 12:00 position as:

$$a_i = \begin{cases} \tan^{-1}(v_{i2}/v_{i1}), & \text{if } v_{i1} > 0; \\ \tan^{-1}(v_{i2}/v_{i1}) + \pi, & \text{otherwise.} \end{cases} \quad (4.5)$$

In Equation 4.5 $\tan^{-1}(x)$ is read as “the angle whose tangent is x ”, and so the angles are determined by the tangent ratios “opposite” / “adjacent” = v_{i2}/v_{i1} in the right triangle defined by the vector and the horizontal axis.

For the `mtcars` data the biplot in Figure 4.26 accounts for 84% of the total variance so a 2D representation is fairly good. The plot shows the variables as widely dispersed. There is a collection at the left of positively correlated variables and another positively correlated set at the right.

```
mtcars.pca <- prcomp(mtcars, scale. = TRUE)
ggbiplot(mtcars.pca,
         circle = TRUE,
         point.size = 2.5,
         varname.size = 6,
         varname.color = "brown") +
  theme_minimal(base_size = 14)
```

In `corrplot()` principal component variable ordering is implemented using the `order = "AOE"` option. There are a variety of other methods based on hierarchical cluster analysis described in the [package vignette](#).

Figure 4.27 shows the result of ordering the variables by this method. A nice feature of `corrplot()` is the ability to manually highlight blocks of variables that have a similar pattern of signs by outlining them with rectangles. From the biplot, the two main clusters of positively correlated variables seemed clear, and are outlined in the plot using `corrplot::corrRect()`. What became clear in the corrplot is that `qsec`, the time to drive a quarter-mile from a dead start didn't quite fit this pattern, so I highlighted it separately.

```
corrplot(R,
         method = 'ellipse',
         order = "AOE",
         title = "PCA variable order",
         tl.srt = 0, tl.col = "black", tl.pos = 'd',
         mar = c(0,0,1,0)) |>
  corrRect(c(1, 6, 7, 11))
```

But wait, there is something else to be seen in Figure 4.27. Can you see one cell that doesn't fit with the rest?

Yes, the correlation of number of forward gears (`gear`) and number of carburetors (`carb`) in the upper left and lower right corners stands out as moderately positive (0.27) while all the others in their off-diagonal blocks are negative. This is another benefit of effect ordering: when you arrange the variables so that the most highly related variable are together, features that deviate from dominant pattern become visible.

4.6 Application: Eigenfaces

There are many applications of principal components analysis beyond the use for visualization for multivariate data covered here, that rely on its' ability as a **dimension reduction** technique, that is, to find a low-dimensional approximation to a high-dimensional dataset.

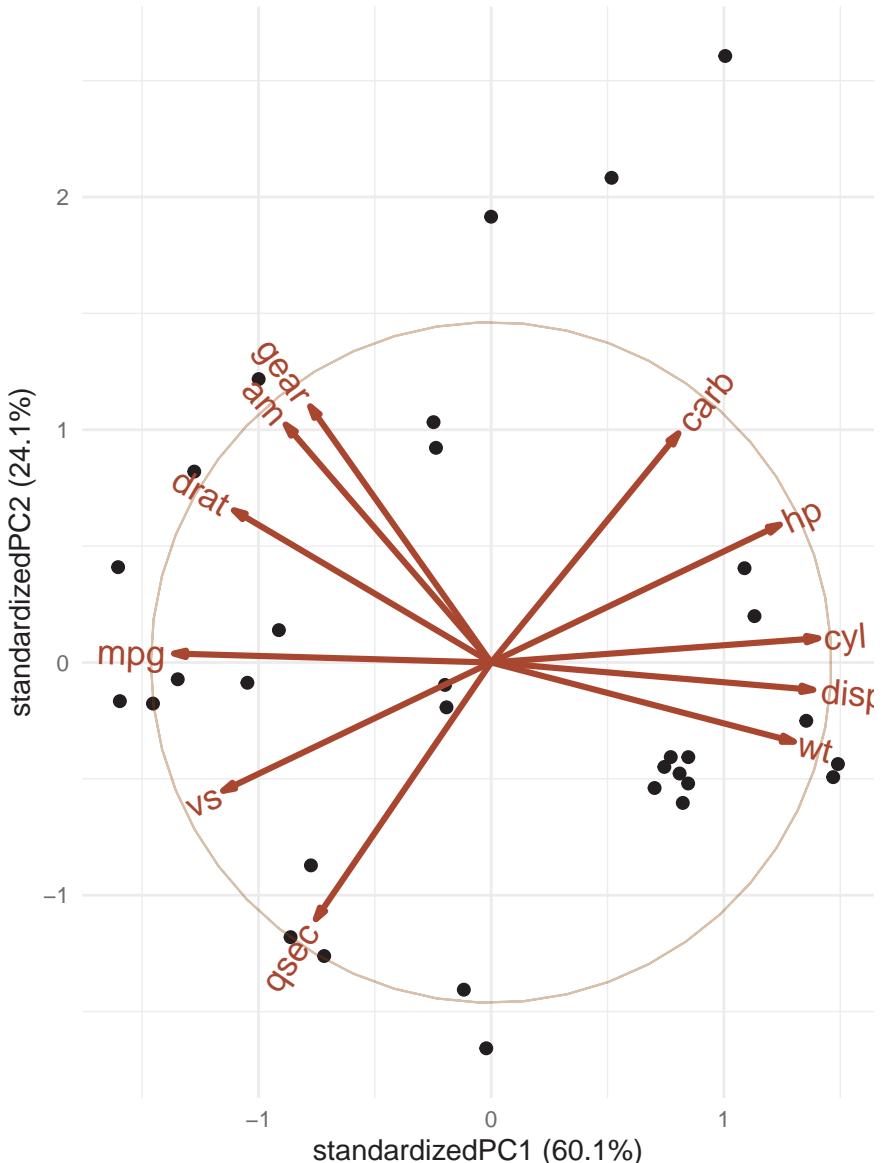


Figure 4.26: Biplot of the `mtcars` data. The order of the variables around the circle, starting from “gear” (say) arranges them so that the most similar variables are adjacent in graphical displays.

i Machine learning uses

In machine learning, for example, PCA is a method used to reduce model complexity and avoid overfitting by *feature extraction*, which amounts to fitting a response variable in a low-D space of the predictors. This is just another name for *principal components regression*, where, instead of regressing the dependent variable on all the explanatory variables directly, a smaller number principal components of the explanatory variables is used as predictors. This has the added benefit that it avoids problems of collinearity (section-ref) due to high correlations of the predictors, because the principal component scores are necessarily uncorrelated. When the goal is model explanation rather than pure prediction, it has the disadvantage that the components may be hard to interpret.

An interesting class of problems have to do with image processing, where an image of size width \times height in

PCA variable order

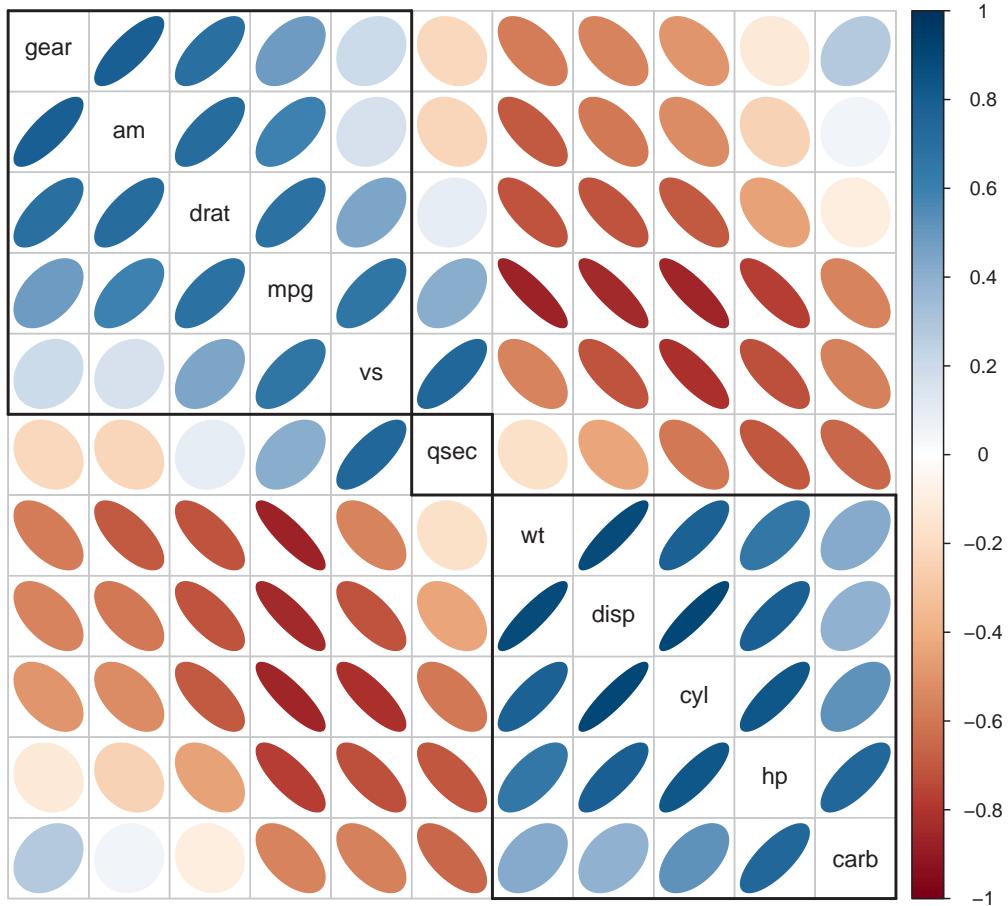


Figure 4.27: Corrplot of `mtcars` data, with the variables ordered according to the variable vectors in the biplot.

pixels can be represented by a $w \times h$ array of greyscale values x_{ij} in the range of $[0, 1]$ or $h \times w \times 3$ array x_{ijk} of (red, green, blue) color values. For example a single 640×640 photo is comprised of about 400K pixels in B/W and 1200K pixels in color.

The uses here include

- **Image compression:** a process applied to a graphics file to minimize its size in bytes for storage or transmission, without degrading image quality below an acceptable threshold
- **image enhancement:** improving the quality of an image, with applications in Computer Vision tasks, remote sensing, and satellite imagery.
- **facial recognition:** classifying or matching a facial image against a large corpus of stored images.

When PCA is used on facial images, you can think of the process as generating **eigenfaces** (Turk & Pentland (1991)) a representation of the pixels in the image in terms of an eigenvalue decomposition. Dimension reduction means that a facial image can be considerably compressed by removing the components associated with small dimensions.

As an example, consider the black and white version of the Mona Lisa shown in Figure 4.28. The idea and code for this example is adapted from this [blog post](#) by Kieran Healy.⁷



Figure 4.28: 640 x 954 black and white image of the *Mona Lisa*. Source: [Wikipedia](#)

It would take too long to explain the entire method, so I'll just sketch the essential parts here. The complete script for this example is contained in [PCA-MonaLisa.R](#). . . .

TODO: Show the necessary parts, including the screeplot.

An image can be imported using `imager::load.image()` which creates a "cimg" object, a 4-dimensional array with dimensions named `x`, `y`, `z`, `c`. `x` and `y` are the usual spatial dimensions, `z` is a depth dimension (which would correspond to time in a movie), and `c` is a color dimension containing R, G, B values.

```
library(imager)
img <- imager::load.image(here::here("images", "MonaLisa-BW.jpg"))
dim(img)
#> [1] 640 954    1    1
```

An `as.data.frame()` method converts this to a data frame with `x` and `y` coordinates. Each `x-y` pair is a location in the 640 by 954 pixel grid, and the `value` is a grayscale value ranging from zero to one.

```
img_df_long <- as.data.frame(img)
head(img_df_long)
#>   x y value
#> 1 1 1 0.431
#> 2 2 1 0.337
#> 3 3 1 0.467
```

⁷<https://kieranhealy.org/blog/archives/2019/10/27/reconstructing-images-using-pca/>

```
#> 4 4 1 0.337
#> 5 5 1 0.376
#> 6 6 1 0.361
```

However, to do a PCA we will need a matrix of data in wide format containing the grayscale pixel values. We can do this using `tidy::pivot_wider()`, giving a result with 640 rows and 954 columns.

```
img_df <- pivot_wider(img_df_long,
                       names_from = y,
                       values_from = value) |>
  select(-x)
dim(img_df)
#> [1] 640 954
```

Mona's PCA is produced from this `img_df` with `prcomp()`:

```
img_pca <- img_df |>
  prcomp(scale = TRUE, center = TRUE)
```

With 955 columns, the PCA comprises 955 eigenvalue/eigenvector pairs. However, the rank of a matrix is the smaller of the number of rows and columns, so only 640 eigenvalues can be non-zero. Printing the first 10 shows that the first three dimensions account for 46% of the variance and we only get to 63% with 10 components.

```
img_pca |>
  broom::tidy(matrix = "eigenvalues") |> head(10)
#> # A tibble: 10 x 4
#>   PC std.dev percent cumulative
#>   <dbl>    <dbl>    <dbl>      <dbl>
#> 1 1     14.1  0.209     0.209
#> 2 2     11.6  0.141     0.350
#> 3 3     10.1  0.107     0.457
#> 4 4     7.83  0.0643    0.522
#> 5 5     6.11  0.0392    0.561
#> 6 6     4.75  0.0237    0.585
#> 7 7     3.70  0.0143    0.599
#> 8 8     3.52  0.0130    0.612
#> 9 9     3.12  0.0102    0.622
#> 10 10   2.86  0.00855   0.631
```

Figure 4.29 shows a screeplot of proportions of variance. Because there are so many components and most of the information is concentrated in the largest dimensions, I've used a `log10()` scale on the horizontal axis. Beyond 10 or so dimensions, the variance of additional components looks quite tiny.

```
ggscreeplot(img_pca) +
  scale_x_log10()
```

Then, if \mathbf{M} is the 640×955 matrix of pixel values, a best approximation $\widehat{\mathbf{M}}_k$ using k dimensions can be obtained as $\widehat{\mathbf{M}}_k = \mathbf{X}_k \mathbf{V}_k^T$ where \mathbf{X}_k are the principal component scores and \mathbf{V}_k are the eigenvectors corresponding to the k largest eigenvalues. The function `approx_pca()` does this, and also undoes the scaling and centering carried out in PCA.

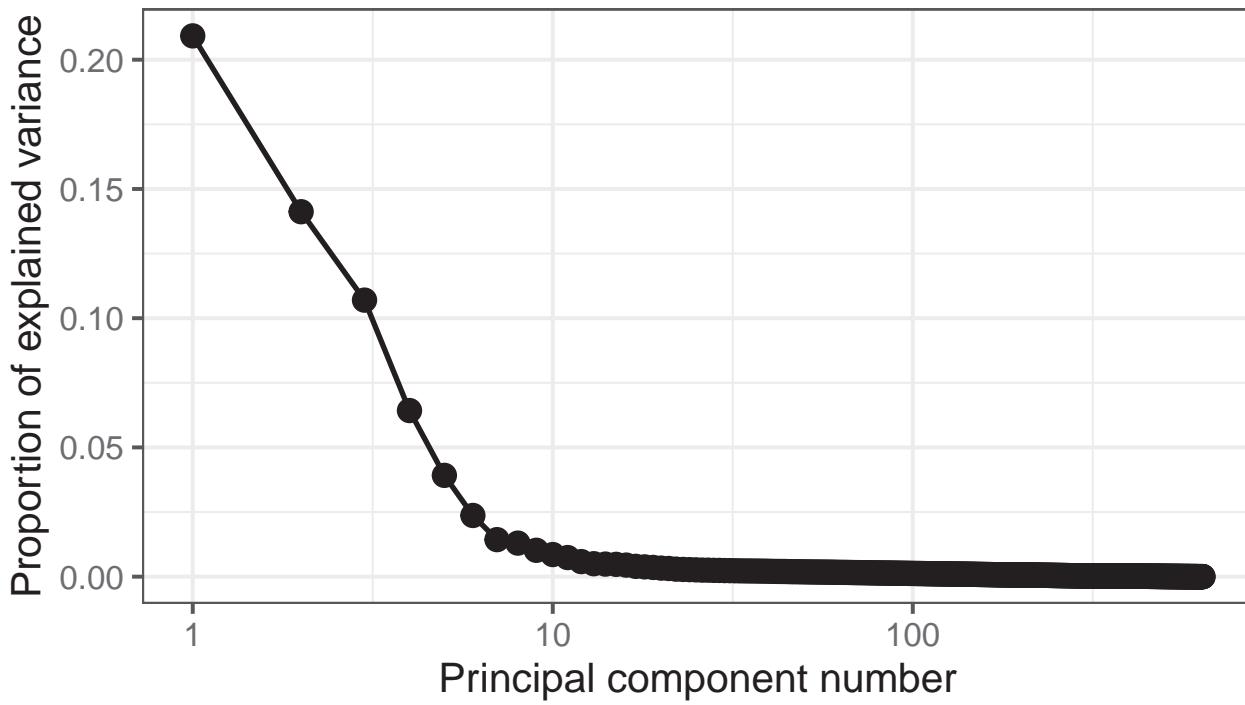


Figure 4.29: Screeplot of the variance proportions in the Mona Lisa PCA.

```

approx_pca <- function(n_comp = 20, pca_object = img_pca){
  ## Multiply the matrix of rotated data (component scores) by the transpose of
  ## the matrix of eigenvectors (i.e. the component loadings) to get back to a
  ## matrix of original data values

  recon <- pca_object$x[, 1:n_comp] %*% t(pca_object$rotation[, 1:n_comp])

  ## Reverse any scaling and centering that was done by prcomp()
  if(all(pca_object$scale != FALSE)){
    ## Rescale by the reciprocal of the scaling factor, i.e. back to
    ## original range.
    recon <- scale(recon, center = FALSE, scale = 1/pca_object$scale)
  }
  if(all(pca_object$center != FALSE)){
    ## Remove any mean centering by adding the subtracted mean back in
    recon <- scale(recon, scale = FALSE, center = -1 * pca_object$center)
  }

  ## Make it a data frame that we can easily pivot to long format
  ## for drawing with ggplot
  recon_df <- data.frame(cbind(1:nrow(recon), recon))
  colnames(recon_df) <- c("x", 1:(ncol(recon_df)-1))

  ## Return the data to long form
  recon_df_long <- recon_df |>
    tidyverse::pivot_longer(cols = -x,
                           names_to = "y",

```

```

            values_to = "value") |>
mutate(y = as.numeric(y)) |>
arrange(y) |>
as.data.frame()

recon_df_long
}

```

Finally, the recovered images, using 2, 3 , 4, 5, 10, 15, 20, 50, and 100 principal components can be plotted using ggplot. In the code below, the `approx_pca()` function is run for each of the 9 values specified by `n_pcs` giving a data frame `recovered_imgs` containing all reconstructed images, with variables `x`, `y` and `value` (the greyscale pixel value).

```

n_pcs <- c(2:5, 10, 15, 20, 50, 100)
names(n_pcs) <- paste("First", n_pcs, "Components", sep = "_")

recovered_imgs <- map_dfr(n_pcs,
                           approx_pca,
                           .id = "pcs") |>
mutate(pcs = stringr::str_replace_all(pcs, "_", " "),
       pcs = factor(pcs, levels = unique(pcs), ordered = TRUE))

```

In `ggplot()`, each is plotted using `geom_raster()`, using `value` to as the fill color. A quirk of images imported to R is that origin is taken as the upper left corner, so the Y axis scale needs to be reversed. The 9 images are then plotted together using `facet_wrap()`.

```

p <- ggplot(data = recovered_imgs,
             mapping = aes(x = x, y = y, fill = value))
p_out <- p + geom_raster() +
scale_y_reverse() +
scale_fill_gradient(low = "black", high = "white") +
facet_wrap(~ pcs, ncol = 3) +
guides(fill = "none") +
labs(title = "Recovering Mona Lisa from PCA of her pixels") +
theme(strip.text = element_text(face = "bold", size = rel(1.2)),
      plot.title = element_text(size = rel(1.5)))

p_out

```

The result, in Figure 4.30 is instructive about how much visual information is contained in lower-dimensional reconstructions, or conversely, how much the image can be compressed by omitting the many small dimensions.

In this figure, with 4-5 components most people would recognize this as a blurry image of the world's most famous portrait. It is certainly clear that this is the Mona Lisa with 10–15 components. Details of the portrait and background features become recognizable with 20–50 components, and with 100 components it compares favorably with the original in Figure 4.28. In numbers, the original 640×955 image is of size 600 Kb. The 100 component version is only 93 Kb, 15.6% of this.

4.7 Elliptical insights: Outlier detection

The data ellipse (Section 3.2), or ellipsoid in more than 2D is fundamental in regression. But, as Pearson showed, it is also key to understanding principal components analysis, where the principal component directions are simply the axes of the ellipsoid of the data. As such, observations that are unusual in data space may not stand out in univariate views of the variables, but will stand out in principal component space, usually on the *smallest* dimension.

As an illustration, I created a dataset of $n = 100$ observations with a linear relation, $y = x + \mathcal{N}(0, 1)$ and then added two discrepant points at $(1.5, -1.5)$, $(-1.5, 1.5)$.

```
set.seed(123345)
x <- c(rnorm(100), 1.5, -1.5)
y <- c(x[1:100] + rnorm(100), -1.5, 1.5)
```

When these are plotted with a data ellipse in Figure 4.31 (left), you can see the discrepant points labeled 101 and 102, but they do not stand out as unusual on either x or y . The transformation to from data space to principal components space, shown in Figure 4.31 (right), is simply a rotation of (x, y) to a space whose coordinate axes are the major and minor axes of the data ellipse, (PC_1, PC_2) . In this view, the additional points appear a univariate outliers on the smallest dimension, PC_2 .

To see this more clearly, `?@fig-outlier-animation` shows an animation of the rotation from data space to PCA space. This uses `heplots::interpPlot()` to interpolate linearly from the positions of the points in data space to their locations in PCA space.

4.7.1 Example: Penguin data

In Section 3.9.2 we examined the questions of multivariate normality and outliers for the penguin data. From a χ^2 QQ plot (Figure 3.26) of the Mahalanobis D^2 values, three Penguins (283, 10, 35) were identified as noteworthy, deserving a closer look to see why they are unusual. It was pointed out (Figure 3.27) that 2D plots of the data variables were only partially revealing. Let's see where they appear in biplots.

First, I find the noteworthy points with the three the largest D^2 values as before:

```
data(peng, package="heplots")

# find potential multivariate outliers
DSQ <- heplots::Mahalanobis(peng[, 3:6])
noteworthy <- order(DSQ, decreasing = TRUE)[1:3] |> print()
#> [1] 283 10 35
```

The PCA shows that the first two components account for 88% of variance, so this is probably an adequate representation of the overall structure of our penguins:

```
peng.pca <- prcomp(
  ~ bill_length + bill_depth + flipper_length + body_mass,
  data=peng, scale. = TRUE
)
summary(peng.pca)
#> Importance of components:
#>                               PC1    PC2    PC3    PC4
#> Standard deviation     1.657 0.882 0.6072 0.328
```

```
#> Proportion of Variance 0.686 0.195 0.0922 0.027
#> Cumulative Proportion 0.686 0.881 0.9730 1.000
```

Figure 4.32 gives the biplot for the first two dimensions. It can be seen that:

- PC1 is largely determined by flipper length and body mass. We can interpret this as an overall measure of **penguin size**. On this dimension, Gentoos are the largest, by quite a lot, compared with Adelie and Chinstrap.
- PC2 is mainly determined by variation in the two beak variables: bill length and depth. Chinstrap are lower than the other two species on bill length and depth, but bill length further distinguishes the Gentoos from the Adelies. A penguin biologist could almost certainly provide an explanation, but I'll call this **beak shape**.
- But, our three suspected outliers are well-within the bulk of their species.

That's the point of this exercise. The projection of the data into the space that accounts for the greatest **total** variance usually does not reveal a few unusual points.

```
source("R/penguin/penguin-colors.R")
# create vector of labels, blank except for the noteworthy
lab <- 1:nrow(peng)
lab <- ifelse(lab %in% noteworthy, lab, "")

ggbiplot(peng.pca,
  choices = 1:2,
  groups = peng$species,
  ellipse = TRUE, ellipse.alpha = 0.1,
  circle = TRUE,
  var.factor = 1,
  geom.ind = c("point", "text"),
  point.size = 1,
  labels = lab, labels.size = 6,
  varname.size = 5,
  clip = "off") +
  theme_minimal(base_size = 14) +
  theme_penguins("dark") +
  scale_shape_discrete() +
  theme(legend.direction = 'horizontal', legend.position = 'top')
```

Now, plotting dimensions 3 and 4 gives Figure 4.33. Dimension 3, accounting for 9%, is largely determined by a contrast of bill length with bill depth and body mass, while dimension 4 involves a contrast between body mass and flipper length. The Chinstraps here have the longest, straightest beaks.

Recall that the perpendicular projection of observation i on the vector for variable j gives an approximation of \hat{x}_{ij} shown in that space. Our friend Cyrano (case 283), the only true multivariate outlier, lies at the extreme ends of both dimensions, with his exceptionally long, straight bill. Case 10 (Hook Nose) stands out at the high end of dimension 3 with a highly curved beak. case 35 is at the high end of dimension 4, so probably is much heavier than most and has short flippers.

4.8 What have we learned?

Welcome to the world of **multivariate juicers**—those magical tools that squeeze the most meaningful information from high-dimensional data clouds! This chapter has taken us on a journey from Flatland to Spaceland, revealing how dimension reduction methods can transform overwhelming complexity into interpretable insights.

- **PCA is your geometric friend, helping you compress N-dimensional data:** Principal Components Analysis finds the directions of maximum variance in your data, creating uncorrelated orthogonal dimensions that capture the most “juice” from your multivariate cloud. Think of it as finding the best viewpoint to see a 3D sculpture when you can only look at a 2D photograph—PCA rotates and compresses your data to show you the best 2D viewing angle.
- **Biplots are visualization gold, helping you view compressed N-dimensional data:** These elegant displays build off of PCA by simultaneously showing both your observations (as points) and your variables (as vectors) in the same reduced space. The magic lies in the interpretation: variable vectors pointing in similar directions are correlated, and you can read approximate values by projecting points onto variable vectors. It’s like having X-ray vision for multivariate relationships!
- **Eigenvalues tell the variance story:** The screeplot becomes your guide for deciding how many dimensions to keep. Look for the “elbow” where the eigenvalues start to resemble scree (rubble) rather than meaningful signal. Generally, 80-90% cumulative variance gives you a solid foundation for interpretation.
- **Supplementary variables enhance interpretation:** Once you’ve found your reduced-dimension view, you can project additional variables into the space to aid interpretation—like adding helpful annotations to a map. This technique bridges the gap between statistical discovery and domain knowledge.
- **Nonlinear methods reveal hidden structures:** When relationships aren’t linear, techniques like multidimensional scaling (MDS) and t-SNE can uncover patterns that PCA might miss. These methods focus on preserving local neighborhoods and distances rather than global variance, often revealing clusters and nonlinear manifolds lurking in your data.
- **Variable ordering creates visual clarity:** When similar variables are placed adjacent to each other, patterns emerge and anomalies become visible—it’s like organizing a messy bookshelf by subject. In biplots, the angles of variable vectors provide a natural ordering that are used to make correlation matrices and other displays much more interpretable.
- **Outlier detection gets multidimensional power:** Points that seem normal in individual variables can reveal themselves as true multivariate outliers when viewed in principal component space, especially along the smallest dimensions. The data ellipse becomes your guide to understanding what’s typical versus what deserves a closer look.
- **Real applications abound:** From compressing the Mona Lisa using eigenfaces to understanding crime patterns across U.S. states, dimension reduction methods bridge the gap between statistical technique and practical insight. These aren’t just mathematical curiosities—they’re essential tools for making sense of our increasingly high-dimensional world.

Recovering Mona Lisa from PCA of her pixels

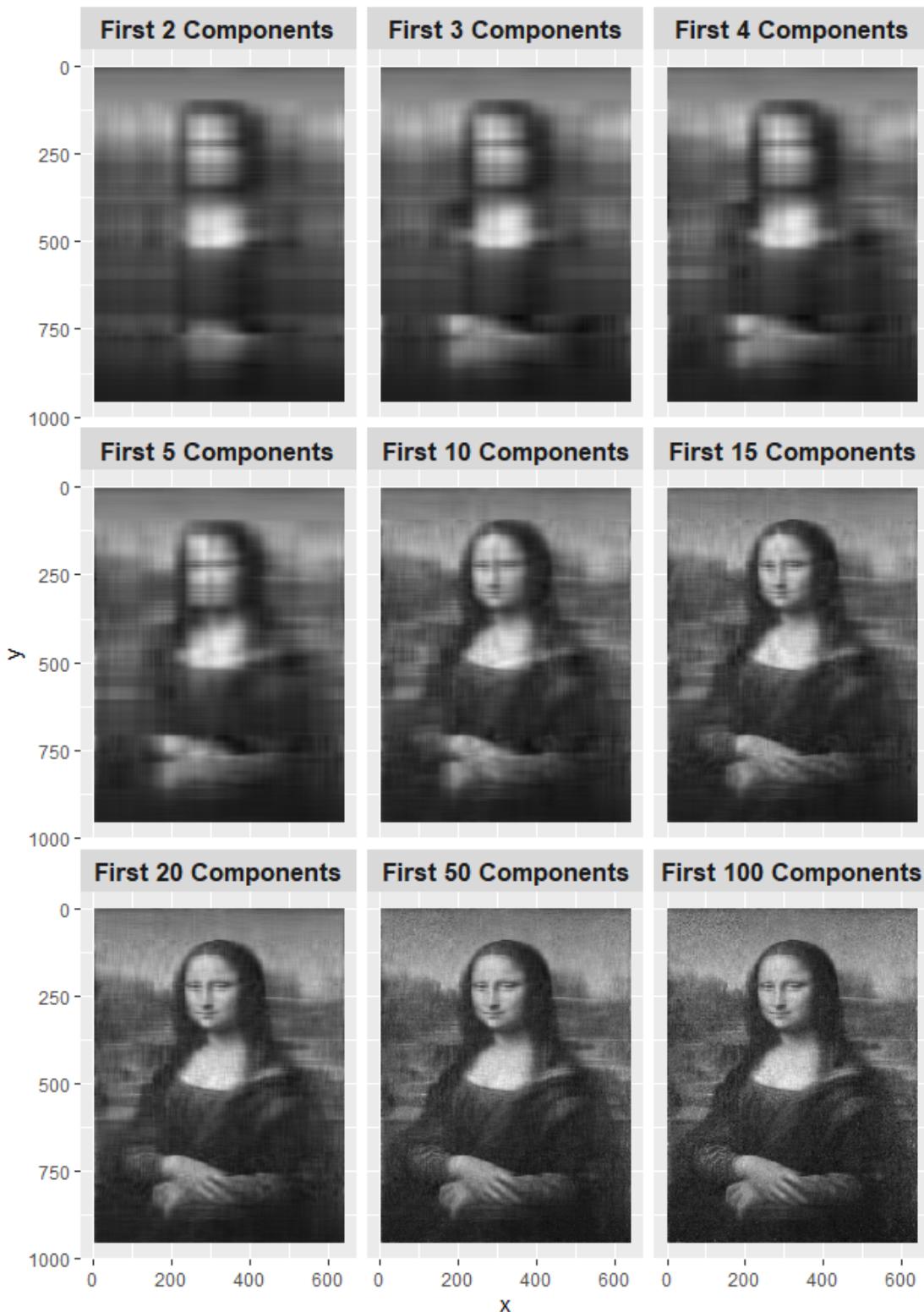


Figure 4.30: Re-construction of the Mona Lisa using 2, 3 , 4, 5, 10, 15, 20, 50, and 100 principal components.

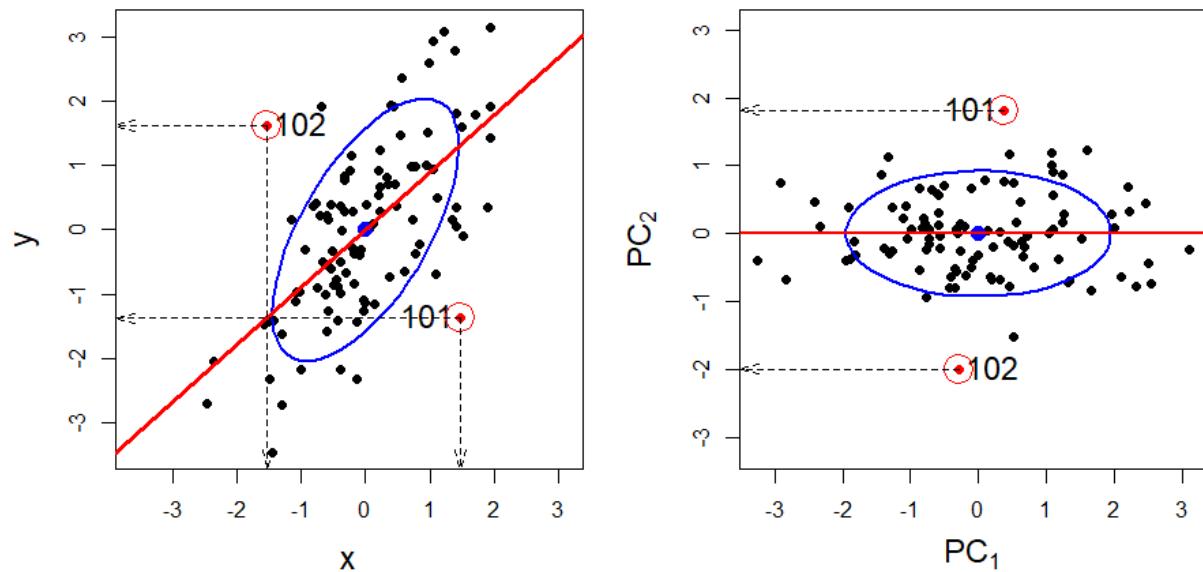


Figure 4.31: Outlier demonstration: The left panel shows the original data and highlights the two discrepant points, which do not appear to be unusual on either x or y . The right panel shows the data rotated to principal components, where the labeled points stand out on the smallest PCA dimension.

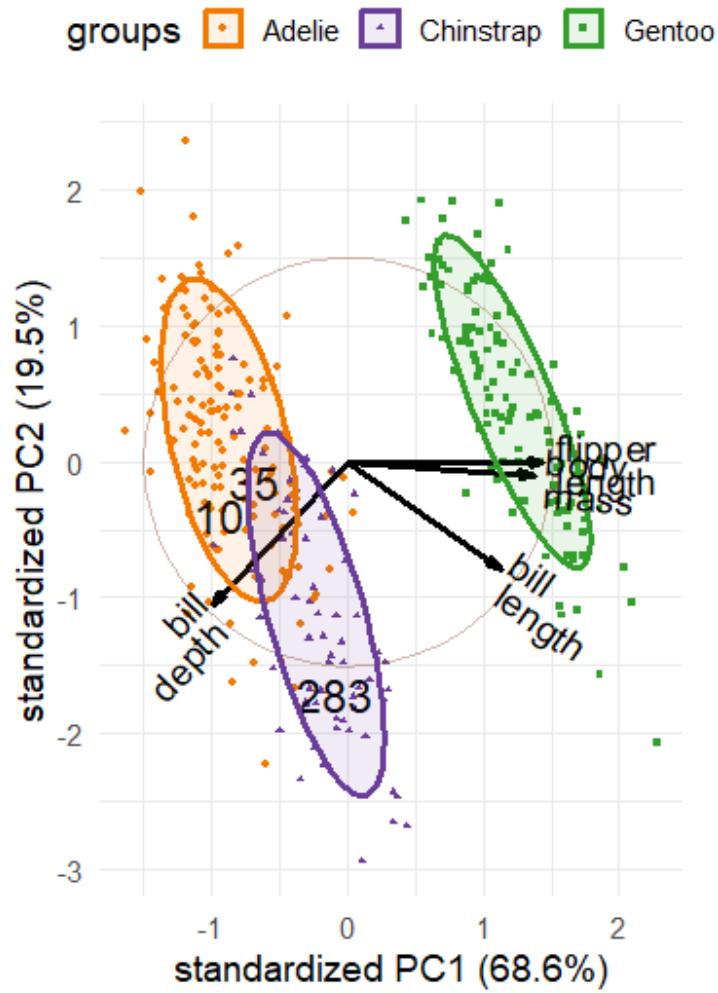


Figure 4.32: Biplot of the first two dimensions of the Penguin data. The points for the three noteworthy cases are labeled, but none of these appear to be unusual in this view.

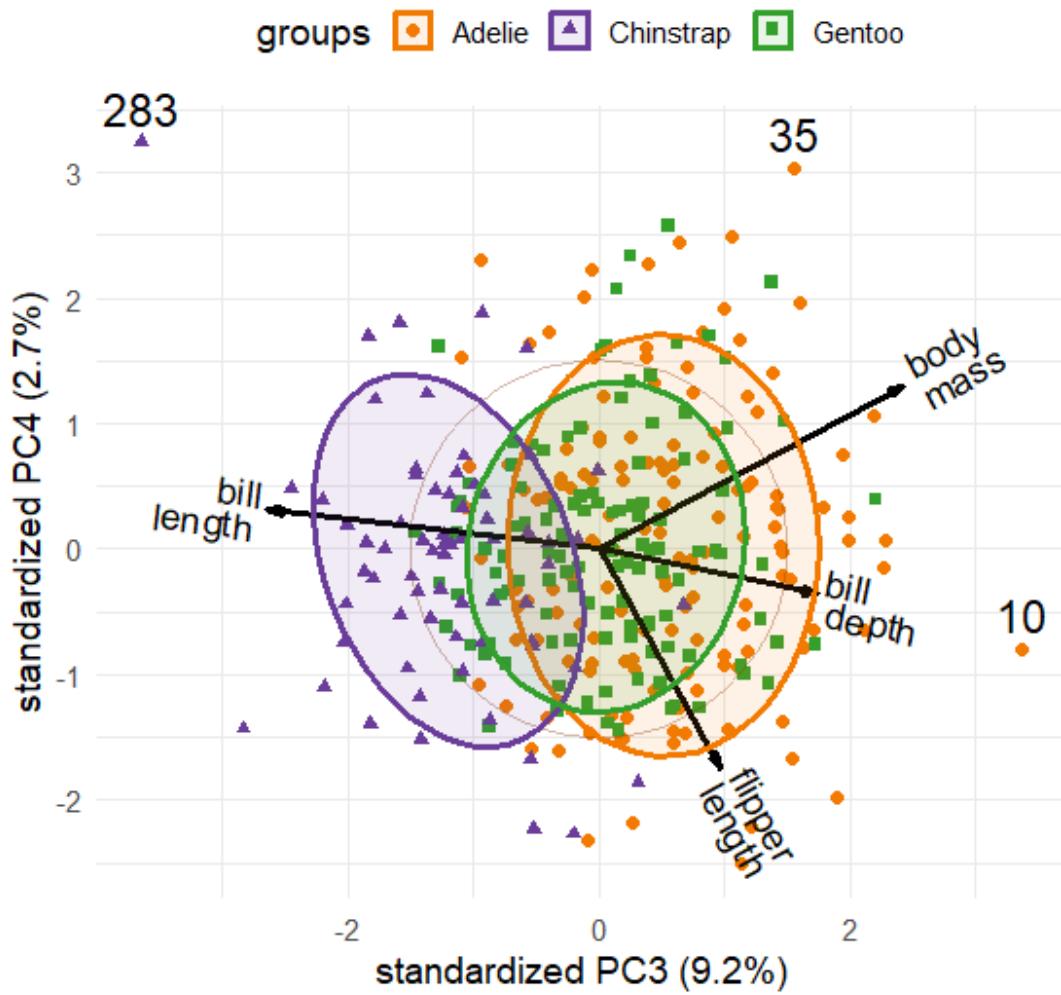


Figure 4.33: Biplot of dimensions 3-4 of the Penguin data. The three noteworthy birds stand out in this view.

Part III

Univariate Linear Models

5

Overview of Linear models

Although this book is primarily about multivariate models, it is useful to have an overview of the range of available techniques for univariate response models to see their uses and to appreciate how easily univariate models generalize to multivariate ones. Hence, this chapter reviews the characteristics of the standard univariate methods for explaining or predicting a single outcome variable from a set of predictors.

The key ideas are:

- For a single quantitative outcome variable \mathbf{y} , methods of linear regression and analysis of variance (ANOVA) are comprised within a single framework of the **general linear model** (GLM). Regression and ANOVA differ only in that the predictors in the former are quantitative, while those in the latter are discrete factors. They can all be fit using `lm()`.
- These models extend directly to the multivariate case of $q > 1$ outcomes, $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q)$, and are also fit using `lm()`.
- A binary outcome $y = (0, 1)$ and categorical outcomes like marital status (“never married”, “married”, “separated”, “divorced”) can be handled within a different extension, the **generalized** linear model as logistic regression or multinomial regression, fit using `glm()`.
- All of these models involve linear combinations of predictors (weighted sums) fit to optimize some criterion, for example minimizing some function of the residuals or maximizing some measure of fit.
- Models and data can be more easily understood with graphics, and many statistical ideas have a visual representation in geometry.

Figure 5.1 summarizes a variety of methods for linear models, classified by number of predictors and number of response variables, and whether these are quantitative vs. discrete. For the present purposes, the key columns are the first two, for the case of one or more quantitative outcome variables.

When the *predictors* are also quantitative, simple regression ($p = 1$) generalizes to multivariate regression with two or more outcomes ($q > 1$). For example we might want to predict weight and body mass index *jointly* from a person’s height.

The situation is more interesting when there are $p > 1$ predictors. The most common multivariate generalization is *multivariate multiple regression* (MMRA), where each outcome is regressed on the predictors, as if done separately for each outcome, but using multivariate tests that take correlations among the predictors into account. Other methods for this case include canonical correlation analysis, which tries to explain all relations between \mathbf{Y} and a set of \mathbf{x} s through maximally correlated linear combinations of each.

When the predictor variables are all discrete or categorical, such as gender or level of education, methods like the simple *t*-test, one-way ANOVA and factorial ANOVA with $q = 1$ outcome measures all have simple extensions to the case of $q > 1$ outcomes.

History Corner

Why are there so many different names for regression vs. ANOVA concepts, statistics and techniques? In regression, we use notation like x_1, x_2, \dots to refer to *predictors* in a model, while in ANOVA, factors A, B, \dots are called *main effects*. In regression applications, we often test *linear hypotheses*, are interested

Response variables: $\mathbf{Y} = (y_1, \dots y_q)$

	Quantitative		Discrete	
	$q=1$	$q>1$	$q=1$	$q>1$
Quantitative	Simple regression	Multivariate regression	Simple logistic regression	
	Multiple regression	Multivariate regression Canonical corr. Partial corr.	Mult. logistic regression Discriminant analysis	Multivariate logistic regression
Discrete	t-test 1-way ANOVA	Hotelling T ² 1-way MANOVA	Simple χ^2	Loglinear models
	Factorial ANOVA	Factorial MANOVA	Logit models Loglinear models	Loglinear models

Figure 5.1: Techniques for linear models classified by number of predictors and number of response variables, and whether these are quantitative vs. discrete

in *coefficients* and evaluate a model with an R^2 statistic, while in ANOVA we may test *contrasts* among factor levels, and use F -tests to evaluate models.

Well, like twins separated at birth, they grew up in homes in different places and with different parents, who were each free to choose their own names, not recognizing their shared DNA.

Methods of regression began in evolutionary biology with Francis Galton's (1886, 1889) studies of heritability of traits, trying to understand how strongly the physical characteristics of one generation of living things resembled those in the next generation. From a study of the diameters of sweet peas in parent plants and their size in the next generation, and another on the relationship between heights of human parents and their offspring, he developed the fundamental ideas of regression. Karl Pearson (1896)

In contrast, analysis of variance methods were raised on farms, notably the Rothamsted Experimental Station, where R. A. Fisher analyzed vast amounts of data on crop experiments designed to determine the conditions (soil condition, fertilizer treatments) that gave the greatest yields, while controlling for extraneous determiners (plots of planting). With multiple factors determining the outcome, Fisher (1923), in an experiment on yields of different varieties of potatoes given various manure treatments, devised the method of breaking down the total variance into portions attributable to each factor and presented the first ANOVA table. The method became well-known after Fisher's (1925b) *Statistical Methods for Research Workers*.

The great synthesis of regression and ANOVA did not take place until the 1960s. At that time, methods for computing were beginning to move from programmable desk calculators to mainframe computers, largely using a collection of separate FORTRAN programs, designed for regression, one-way ANOVA, two-way ANOVA, models with interactions, and so forth. To complete an analysis, as a graduate student I often had to use three or more different programs.

Then, something remarkable happened on two fronts: theory and computation. First, in quick succession

textbooks by Scheffé (1960), Graybill (1961), Winer (1962) ... began to layout a general theory of linear models that encompassed all of these separate models, giving the “General Linear Model” a well-deserved name. Second, two symposiums, one at IBM Yorkdown Heights (IBM (1965)) and the other at the University of Georgia (Bashaw & Findley (1967)) resulted in the first general programs to handle all these cases in an understandable way.

A bit of matrix algebra thrown into the mix showed that most of the ideas for univariate models could be extended to multiple response variables, and so the “Multivariate Linear Model” was born. R. Darrell Bock (Bock, 1963, 1964) sketched a flowchart of the computational steps, which was implemented at the University of Chicago by Jeremy Finn (1967) in the MULTIVARIANCE program. A group at the University of North Carolina headed by Elliot Cramer developed their MANOVA program (Clyde et al., 1966) and Willard Dixon (1965) at UCLA developed the BMD programs incorporating these ideas. The ANOVA and regression twins had finally become part of a larger family.

Packages

In this chapter I use the following packages. Load them now:

```
library(ggplot2)
```

TODO: This stuff on linear combinations seems out of place here. Where to move it?

5.1 The General Linear Model

To establish notation and terminology, it is worthwhile to state the the general linear model formally. For convenience, I use vector and matrix notation. This expresses a response variable, $\mathbf{y} = (y_1, y_2, \dots, y_n)^T$ for n observations, as a sum of terms involving p regressors, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$, each of length n .

$$\begin{aligned}\mathbf{y} &= \beta_0 + \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \cdots + \beta_p \mathbf{x}_p + \epsilon \\ &= [\mathbf{1}, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p] \boldsymbol{\beta} + \epsilon\end{aligned}$$

{#eq-glm}

or, expressed in matrices,

$$\mathbf{y}_{n \times 1} = \mathbf{X}_{n \times (p+1)} \boldsymbol{\beta}_{(p+1) \times 1} + \epsilon$$

The matrix \mathbf{X} is called the *model matrix* and contains the numerical representations of the predictor variables called *regressors*. The essential thing about a linear model is that it is linear in the **parameters** β_i . That is, the predicted value of \mathbf{y} is a linear combination of some \mathbf{x}_i with weights β_i . An example of a nonlinear model is the exponential growth model, $y = \beta_0 + e^{\beta_1 x}$, where the parameter β_1 appears as an exponent.¹

- These can be quantitative variables like `age`, `salary` or `years` of education. But they can also be transformed versions, like `sqrt(age)` or `log(salary)`.
- A quantitative variable can be represented by more than one model regressor, for example if it is expressed as a polynomial like `poly(age, degree=2)` or a natural spline like `ns(salary, df=5)`. The model matrix portion for such terms contains one column for each degree of freedom (`df`) and there are `df` coefficients in the corresponding portion of $\boldsymbol{\beta}$.

¹Taking logarithms of both sides would yield the linear model, $\log(y) = c + \beta_1 x$.

- A categorical or discrete predictor– a factor variable in R– with d levels is expressed as $d - 1$ columns in **X**. Typically these are contrasts or comparisons between a baseline or reference level and each of the remaining ones, but any set of $d - 1$ linearly independent contrasts can be used by assigning to **contrasts(factor)**. For example, **contrasts(factor) <- contr.treatment(4)** for a 4-level factor assigns 3 contrasts representing comparisons with a baseline level, typically the first (in alphabetic order). For an *ordered* factor, such as one for political knowledge with levels “low”, “medium”, “high”, **contrasts.poly()** returns the coefficients of orthogonal polynomial contrasts representing linear and quadratic trends.
- Interactions between predictors are represented as the direct products of the corresponding columns of **X**. This allows the effect of one predictor on the response to depend on values of other predictors. For example, the interaction of two quantitative variables, **x₁, x₂** is represented by the product **x₁ × x₂**. More generally, for variables or factors *A* and *B* with degrees of freedom df_A and df_B the regressors in **X** are the $df_A \times df_B$ products of each column for *A* with each column for *B*.

5.1.1 Model formulas

Statistical models in R, such as those fit by **lm()**, **glm()** and many other modelling function in R are expressed in a simple notation that was developed by Wilkinson & Rogers (1973) for the GENSTAT software system at the Rothamsted Research Station. It solves the problem of having a compact way to specify any model consisting of any combinations of quantitative and discrete factor variables, interactions of these and arbitrary transformations of these.

In this, a **model formula** take the forms

```
response ~ terms
response ~ term1 + term2 + ...
```

where the left-hand side, **response** specifies the response variable in the model and the right-hand side specifies the **terms** in the model specifying the columns in the **X** matrix of **?@eq-glm**; the coefficients β are implied and not represented explicitly in the formula.

The notation **y ~ x** is read as “**y is modeled by x**”. The left-hand side is usually a variable name (such as **height**), but it could be an expression that evaluates to the the response, such as **log(salary)** or **weight/height^2** which represents the body mass index.

On the right-hand side (RHS), the usual arithmetic operator, **+**, **-**, *****, **/**, **^** have special meanings as described below. The most fundamental is that **y ~ a + b** is interpreted as “**y is modeled by a and b**”; that is, the sum of linear terms for **a** and **b**.

Some examples for regression-like models using only quantitative variables, **x**, **x₁**, **x₂**, **x₃**, ... are shown below:

```
y ~ x                      # simple linear regression
y ~ x - 1                   # no intercept: regression through the origin
y ~ x + I(x^2)              # quadratic model
y ~ poly(x, 3)              # cubic model
y ~ x1 * x2                 # crossing: x1 + x2 + x1 : x2
y ~ x1 + x2 + x3            # multiple regression
y ~ (x1 + x2 + x3)^2        # response surface: all quadratics & two-way interactions
log(y) ~ x1 + poly(x, 2)     # arbitrary transformation of response
y1 + y2 ~ x1 + x2 + x3      # response is sum of y1 and y2
```

The intercept β_0 is automatically included in the model without need to specify it explicitly. The minus sign, **-** on the right-hand side removes terms from the model, so a model with no intercept $\beta_0 = 0$ can be specified as **y ~ X -1** (or perhaps more naturally, **y ~ 0 + X**).

Function calls on the RHS, such as `poly(x, 3)` are evaluated directly, but to use a special model operator, like `^` must be “protected” by wrapping the term in `I()`, meaning “identity” or “inhibit”. Thus, the model $y \sim x + I(x^2)$ means the quadratic model $y = \beta_0 + \beta_1 x + \beta_2 x^2$. This differs from the model $y \sim \text{poly}(x, 2)$ in that the former uses the raw x , x^2 values (which are necessarily positively correlated) while `poly()` converts these to orthogonal polynomial scores, which are uncorrelated (and therefore free from problems of collinearity).

Example 5.1. Workers data: Regression models

For the `workers` data (Section 4.2.2.1) you can fit simple regression models predicting income from years of experience using a linear and quadratic model as follows:

```
data(workers, package = "matlib")
workers.mod1 <- lm(Income ~ Experience, data=workers)
coef(workers.mod1) |> t() |> t()
#>           [,1]
#> (Intercept) 29.16
#> Experience   1.12

workers.mod2 <- lm(Income ~ poly(Experience, 2), data=workers)
coef(workers.mod2) |> t() |> t()
#>           [,1]
#> (Intercept)      46.5
#> poly(Experience, 2)1 39.1
#> poly(Experience, 2)2 -11.2
```

It is simplest to understand these models by plotting the data overlaid with the fitted regressions. This uses `geom_smooth()` and specifies the smoothing model as `method = "lm"` with a `formula`, which is $y \sim x$ for the linear model and $y \sim \text{poly}(x, 2)$ for the quadratic.

```
ggplot(data = workers,
       aes(x = Experience, y = Income)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", formula = y ~ x,
              se = FALSE, linewidth = 2,
              color = "blue") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2),
              se = FALSE, linewidth = 2,
              color = "red")
```

The coefficients of the linear model are also easy to interpret:

$$\widehat{\text{Income}} = 29.162 + 1.119(\text{Experience}) \quad (5.1)$$

So a worker with zero years of experience can expect an income of \$29162 and this should increase by \$1119 for each additional year. However, it is not so simple to interpret the coefficients when a `poly()` term is used. Naively plugging in the coefficients for `workers.mod2` gives

$$\widehat{\text{Income}} = 46.5 + 39.111(\text{Experience}) - 11.16(\text{Experience}^2) \quad (5.2)$$

The problem is that $x = \text{Experience}$ in model `workers.mod` is represented not by the raw values, but rather by values of x and x^2 that have been made to be uncorrelated. If you really want to interpret the coefficient values in terms of years of experience, use the option `raw = TRUE` in `poly()`:

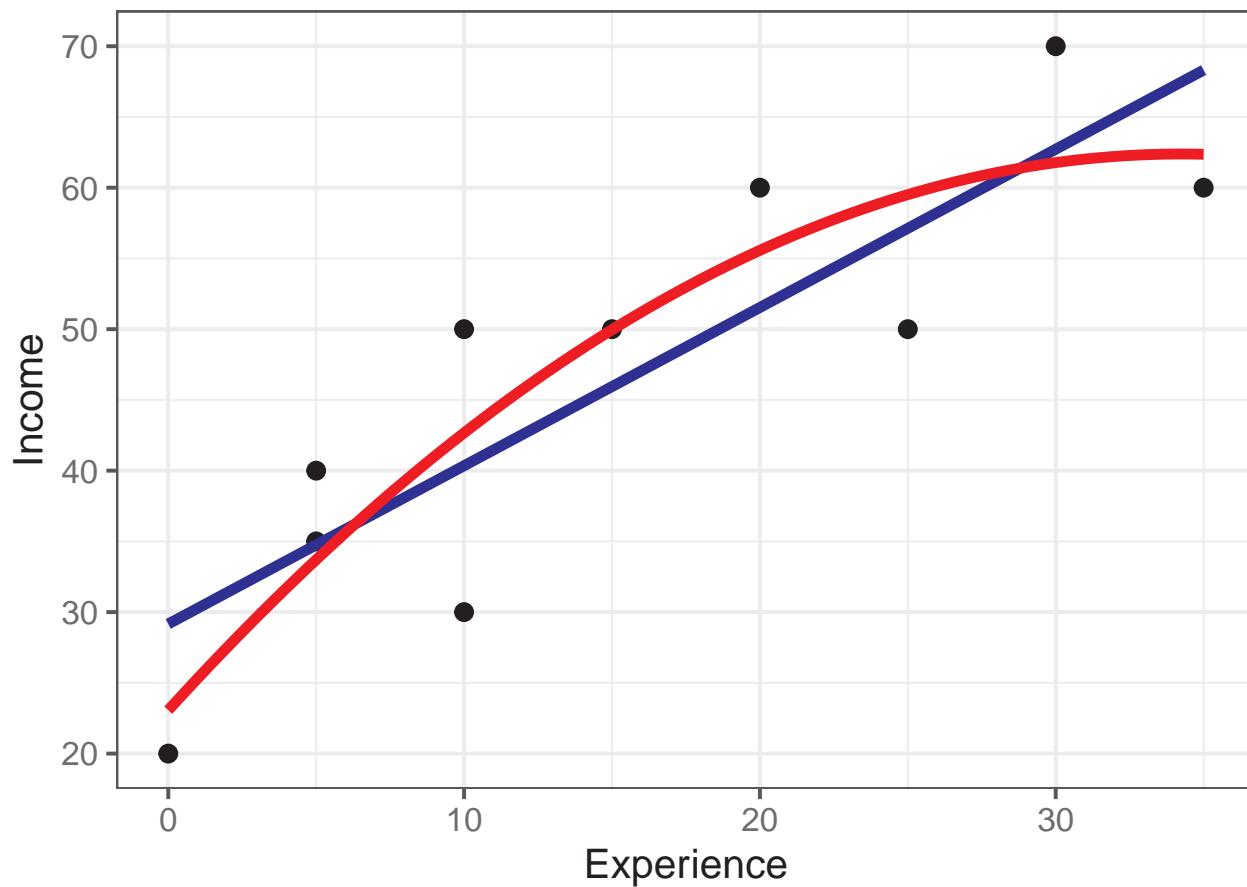


Figure 5.2: Workers data with fitted linear and quadratic models for years of experience.

```
workers.mod2b <- lm(Income ~ poly(Experience, 2, raw = TRUE),
                      data=workers) |>
  print()
#>
#> Call:
#> lm(formula = Income ~ poly(Experience, 2, raw = TRUE), data = workers)
#>
#> Coefficients:
#>              (Intercept)  poly(Experience, 2, raw = TRUE)1
#>                  23.0680                      2.2952
#> poly(Experience, 2, raw = TRUE)2
#>                  -0.0335
```

$$\widehat{\text{Income}} = 23.07 + 2.3(\text{Experience}) - 0.03(\text{Experience}^2)$$

This says that income is predicted to be \$23,068 with no experience, increase initially by \$2295, but that yearly increase decreases by \$330. Some further details of orthogonal polynomials are explained below.

5.1.1.1 Factors

Factor variables are treated specially in linear models, but have simple notations in R formulas. The following examples use A, B, C to represent discrete factors with two or more levels.

```

y ~ A                      # one-way ANOVA
y ~ A + B                  # two-way, main effects only
y ~ A * B                  # full two-way, with interaction
y ~ A + B + A:B            # same, in long-hand
y ~ x + A                  # one-way ANCOVA
y ~ (A + B + C)^2          # three-way ANOVA, incl. all two-way interactions

```

5.1.1.2 Crossing

The `*` operator has special meaning used to specify the crossing of variables and factors and `:` specifies interactions (products of variables). So, the model `y ~ x1 * x2` is expanded to give `y ~ x1 + x2 + x1:x2` and the interaction term `x1:x2` is calculated as $x_1 \times x_2$. In algebraic notation (omitting the error term) this works out to the model,

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_1 x_1 * \beta_2 x_2 \\ &= \beta_0 + (\beta_1 + \beta_2 x_2)x_1 + \beta_2 x_2 , \end{aligned}$$

which means that the coefficient for x_1 in the model is not constant for all values of x_2 , but rather changes with the value of x_2 . If $\beta_2 > 0$, the slope for x_1 increases with x_2 and vice-versa.

`y ~ A * B` for factors is similar, expanding to `y ~ A + B + A:B`, but the columns in the model matrix represent contrasts among the factor levels as described in detail below (Section 5.1.3). The main effects, `A` and `B` come from contrasts among the means of their factor levels and the interaction term `A:B` reflects *differences* among means of `A` across the levels of `B` (and vice-versa).

The model formula `y ~ x + A` specifies an ANCOVA model with different intercepts for the levels of `A`, but with a common slope for `x`. Adding an interaction of `x:A` in the model `y ~ x * A` allow separate slopes and intercepts for the groups.

5.1.1.3 Powers

The `^` exponent operator indicates *powers of a term expression* to a specified degree. Thus the term `(A + B)^2` is identical to `(A + B) * (A + B)` which expands to the main effects of `A`, `B` and their interaction, also identical to `A * B`. In general, the product of parenthesized terms expands as in ordinary algebra,

```
y ~ (A + B) * (C + D) -> A + B + C + D + A:C + A:D + B:C + B:D
```

Powers get more interesting with more terms, so `(A + B + C)^2` is the same as `(A + B + C) * (A + B + C)`, which includes main effects of `A`, `B` and `C` as well as all two-way interactions, `A:B`, `A:C` and `B:C`. The model formula `(A + B + C)^3` expands to include all two-way interactions and the three-way interaction `A:B:C`.

```
(A + B + C)^3 -> A + B + C + A:B + A:C + B:C + A:B:C
```

In this context `-` can be used to remove terms, as shown in the following examples

```

(A + B + C)^2 <-> (A + B + C)^3 - A:B:C
(A + B + C)^3 - B:C - A:B:C <-> A + B + C + A:B + A:C

```

Finally, the symbol `.` on the right-hand side specifies *all terms* in the current dataset other than the response. Thus if you have a data.frame containing `y`, `x1`, `x2`, ..., `x6`, you can specify a model with all variables except `x6` as predictors as

```
y ~ . - x6
```

To test what we've covered above,

- What do you think the model formula $y \sim .^2$ means in a data set containing variables x_1, x_2, x_3 , and x_4 ?
- What about the formula with $y \sim .^2 - A:B:C:D$ with factors A, B, C, D ?

You can work out questions like these or explore model formulae using `terms()` for a "formula" object. The labels of these terms can then be concatenated to a string and turned back into a formula using `as.formula()`:

```
f <- formula(y ~ (x1 + x2 + x3 + x4)^2)
terms = attr(terms(f), "term.labels")

terms |> paste(collapse = " + ")
#> [1] "x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x1:x4 + x2:x3 + x2:x4 + x3:x4"
# convert back to a formula
as.formula(sprintf("y ~ %s", paste(terms, collapse=" + ")))
#> y ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x1:x4 + x2:x3 + x2:x4 +
#>     x3:x4
```

5.1.2 Model matrices

As noted above, a model formula is used to generate the $n \times (p + 1)$ model matrix, \mathbf{X} , typically containing the column of 1s for the intercept β_0 in the model, followed by p columns representing the regressors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$. Internally, `lm()` uses `stats::model.matrix()` and you can use this to explore how factors, interactions and other model terms are represented in a model object.

For a small example, here are a few observations representing income (`inc`) and type of occupation, taking on values `bc` (blue collar), `wc` (white collar) and `prof` (professional). `model.matrix()` takes a *one-sided* formula with the terms on the right-hand side. The main effect model looks like this:

```
set.seed(42)
inc <- round(runif(n=9, 20, 40))
type <- rep(c("bc", "wc", "prof"), each =3)

mm <- model.matrix(~ inc + type)
data.frame(type, mm)
#>   type X.Intercept. inc typeprof typewc
#> 1   bc          1  38      0      0
#> 2   bc          1  39      0      0
#> 3   bc          1  26      0      0
#> 4   wc          1  37      0      1
#> 5   wc          1  33      0      1
#> 6   wc          1  30      0      1
#> 7 prof         1  35      1      0
#> 8 prof         1  23      1      0
#> 9 prof         1  33      1      0
```

As you can see, `type`, with 2 degrees of freedom is represented by two **dummy** (0/1) variables, labeled `typeprof` and `typewc` here. Together, these represent *treatment contrasts* (comparisons) between the *baseline* group `type=="bc"`, which is coded (0, 0) and each of the others: `type=="prof"`, coded (1, 0) and `type=="wc"`, codes (0, 1). Different coding schemes are described in the following section.

In a model with the interaction `inc * type`, additional columns are constructed as the **product** of `inc` with each of the columns for `type`. We will see below how this generalizes to an arbitrary number of predictor terms and their possible interactions.

```
model.matrix(~ inc * type)
#>   (Intercept) inc typeprof typewc inc:typeprof inc:typewc
#> 1          1   38      0     0        0        0
#> 2          1   39      0     0        0        0
#> 3          1   26      0     0        0        0
#> 4          1   37      0     1        0       37
#> 5          1   33      0     1        0       33
#> 6          1   30      0     1        0       30
#> 7          1   35      1     0       35        0
#> 8          1   23      1     0       23        0
#> 9          1   33      1     0       33        0
#> attr("assign")
#> [1] 0 1 2 2 3 3
#> attr("contrasts")
#> attr("contrasts")$type
#> [1] "contr.treatment"
```

5.1.3 Coding factors and contrasts

Discrete explanatory variables, such as race, type of occupation or level of education require special attention in linear models because, unlike continuous variables, they cannot be entered into the model equation just as they are. Instead, we need some way to code those variables numerically.

A key insight is that your choice of a coding scheme changes the meaning of the model parameters, and allows you to perform different comparisons (test different statistical hypotheses) about the means of the category levels according to meaningful questions in your research design. For a more general discussion of coding schemes, see Fox & Weisberg (2018a), sec. 4.7 and the vignette [Coding Matrices, Contrast Matrices and Linear Models](#) for the `codingMatrices` package.

Each coding scheme for a factor represents the **same** model in terms of fitted values and overall significance for that term, but they differ in how the coefficients are parameterized and interpreted. This is crucial to understand, because tests of the coefficients can directly answer different research questions depending on the coding scheme used.

In R, categorical variables are called **factors** usually created by `g <- factor(g)` or `g <- as.factor(g)` for a discrete variable `g`. If levels of the variable `g` are ordered, such as `type` of occupation with levels "`bc`" < "`wc`" < "`prof`" or dose of a drug, "`low`" < "`medium`" < "`high`", you can use `g <- ordered(g)` to reflect this.

In any case, a factor with k levels is reflected in an overall test with $k - 1$ degrees of freedom corresponding to the null hypothesis $\mathcal{H}_0 : \mu_1 = \mu_2 = \dots = \mu_k$. This can be represented as $k - 1$ comparisons among the factor level means, or $k - 1$ separate questions asking how the means differ.

Base R provides several coding schemes via assignment to the `contrasts()` function for a factor, as in `contrasts(df$Drug) <- ...` one of:

- `contr.treatment()`: Compares each level to a *reference* level using $k - 1$ dummy (0, 1) variables. This is the default, and the reference level is taken as the first (in alphabetic or numerical order). You can change the reference level using `relevel()` or `reorder()` for the factor, or simply using `factor(A, levels = ...)`.
- `contr.sum()`: Compares each level to the grand mean.

- `contr.helmert()`: Compares each level to the mean of the *previous* levels, which is useful for ordered categories such as type of occupation with levels "bc" < "wc" < "prof" or dose of a drug, "low" < "medium" < "high".
- `contr.poly()`: For ordered factors with *numeric* levels, this creates orthogonal polynomial contrasts, representing the linear, quadratic, cubic ... trends in the factor means, as if these appeared as x, x^2, x^3, \dots terms in a model with x as a numeric variable.

TODO: Move some stuff from 10.3.1 on contrasts here

I take up some of the details of these coding schemes below. But first, it is useful to define exactly what I mean by a **contrast**. For a factor with k groups, a contrast is simply a comparison of the mean of one subset of groups against the mean of another subset. This is specified as a weighted sum, L of the means μ with weights \mathbf{c} that sum to zero,

$$L = \mathbf{c}^\top \boldsymbol{\mu} = \sum_i^k c_i \mu_i \quad \text{such that} \quad \sum c_i = 0 .$$

Two contrasts, \mathbf{c}_1 and \mathbf{c}_2 are *orthogonal* if the sum of products of their weights is zero, i.e., $\mathbf{c}_1^\top \mathbf{c}_2 = \sum c_{1i} \times c_{2i} = 0$. When contrasts are placed as columns of a matrix \mathbf{C} , they are all *mutually orthogonal* if each pair is orthogonal, which means $\mathbf{C}^\top \mathbf{C}$ is a diagonal matrix. If the columns of \mathbf{C} are normalized to have sums of squares = 1, then $\mathbf{C}^\top \mathbf{C} = \mathbf{I}$.

Orthogonal contrasts correspond to statistically independent tests. This is nice because they reflect separate, non-overlapping research questions. Another consequence is that the sums of squares for the overall hypothesis of differences among the groups is *exactly decomposed* as the sum of the sum of squares accounted for by the $k - 1$ contrasts L_i :

$$\text{SS}_{\text{group}} = \sum_i^{k-1} \text{SS}(L_i) .$$

Treatment coding

Let's examine R's default coding scheme, `contr.treatment` (also called dummy coding), for a factor with 4 levels: 'a', 'b', 'c', and 'd', with a view to understanding the relationship between the true population means, μ_a, μ_b, μ_c , and μ_d and the parameters β estimated in a linear model. We get the following:

```
C <- contr.treatment(letters[1:4]) |> print()
#>   b  c  d
#> a  0  0  0
#> b  1  0  0
#> c  0  1  0
#> d  0  0  1
```

Here, the columns of \mathbf{C} correspond to three dummy variables for the levels b, c, d compared to the reference level a. If we denote these columns as x_b, x_c , and x_d , then:

$$x_b = \begin{cases} 1 & \text{if factor=b} \\ 0 & \text{otherwise} \end{cases} ; \quad X_c = \begin{cases} 1 & \text{if factor=c} \\ 0 & \text{otherwise} \end{cases} ; \quad X_d = \begin{cases} 1 & \text{if factor=d} \\ 0 & \text{otherwise} \end{cases}$$

The design matrix $\mathbf{X}_{(4 \times 4)} = [\mathbf{1}, \mathbf{C}] = [\mathbf{1}, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d]$ includes the constant column $\mathbf{1}$ representing the intercept, which averages over the factor levels when there are other terms in the model.

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

With this coding, the parameters β in the model are related to the means μ as,

$$\begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_d \end{pmatrix} = \mathbf{X}\beta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \beta_0 \\ \beta_b \\ \beta_c \\ \beta_d \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_0 + \beta_b \\ \beta_0 + \beta_c \\ \beta_0 + \beta_d \end{pmatrix}$$

Thus we have,

$$\begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_d \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_0 + \beta_b \\ \beta_0 + \beta_c \\ \beta_0 + \beta_d \end{pmatrix}$$

Note that \mathbf{X} is non-singular as long as the comparisons in \mathbf{C} are linearly independent. Its inverse, \mathbf{X}^{-1} determines how the transformed parameters relate to the original class means, that is, it determines the *interpretation* of the parameters in terms of the means.

```
X <- cbind(1, C)
Xinv <- solve(X) |> print()
#>      a b c d
#>      1 0 0 0
#> b -1 1 0 0
#> c -1 0 1 0
#> d -1 0 0 1
```

Thus, you can solve for the parameters in terms of the means symbolically as follows

$$\begin{pmatrix} \beta_0 \\ \beta_b \\ \beta_c \\ \beta_d \end{pmatrix} = \mathbf{X}^{-1}\mu = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_d \end{pmatrix} = \begin{pmatrix} \mu_a \\ \mu_b - \mu_a \\ \mu_c - \mu_a \\ \mu_d - \mu_a \end{pmatrix}$$

Deviation coding

Another common coding scheme, useful for unordered factors, is *deviation coding* given by `contrast.sum()`. This has the property that the parameters are constrained to sum to zero, $\sum \beta_i = 0$.

```
C <- contr.sum(letters[1:4]) |> print()
#>      [,1] [,2] [,3]
#> a      1     0     0
#> b      0     1     0
#> c      0     0     1
#> d     -1    -1    -1
```

The parameters estimated with this coding are: With this coding, the intercept is $\beta_0 = \bar{\mu}$, the grand mean across all levels and the parameters are the deviations from that,

- $\beta_1 = \mu_a - \bar{\mu}$
- $\beta_2 = \mu_b - \bar{\mu}$

- $\beta_3 = \mu_c - \bar{\mu}$

A (redundant) coefficient for the last group is omitted, because with this coding a coefficient for that group would be equal to the negative of the sum of the others, $\beta_d = \mu_d - \bar{\mu} = -(\beta_1 + \beta_2 + \beta_3)$.

Helmert contrasts

For **ordered factors**, it is sensible to take the ordering into account in interpreting the results. *Helmert contrasts* are designed for this purpose. The intercept is again the grand mean across all levels. Each contrast compares the mean of a given level to the *average of all previous ones* in the order; they contrast the second level with the first, the third with the average of the first two, and so on.

```
C <- contr.helmert(letters[1:4]) |> print()
#>   [,1] [,2] [,3]
#> a    -1    -1    -1
#> b     1    -1    -1
#> c     0     2    -1
#> d     0     0     3
```

It is easier to understand these if the columns are normalized so that the largest value in each column is 1.

```
C %*% diag(1/c(1, 2, 3)) |> MASS::fractions()
#>   [,1] [,2] [,3]
#> a    -1 -1/2 -1/3
#> b     1 -1/2 -1/3
#> c     0     1 -1/3
#> d     0     0     1
```

Then we would have the coefficients as:

- $\beta_0 = \bar{\mu}$
- $\beta_1 = \mu_b - \mu_a$
- $\beta_2 = \mu_c - \frac{\mu_a + \mu_b}{2}$
- $\beta_3 = \mu_d - \frac{\mu_a + \mu_b + \mu_c}{3}$

Note that you can easily reverse the ordering of the comparisons to contrast each of the first $k - 1$ means with the average of all those *higher* in the order.

```
C.rev <- C[4:1, 3:1]
row.names(C.rev) <- letters[1:4]
C.rev
#>   [,1] [,2] [,3]
#> a     3     0     0
#> b    -1     2     0
#> c    -1    -1     1
#> d    -1    -1    -1
```

Polynomial contrasts

For ordered factors that are also numeric, like `Age = c(8, 9, 10, 11)` or those that can be considered equally spaced along some continuum, polynomial contrasts, given by `contr.poly()`, provide *orthogonal* (uncorrelated) contrasts that assess the degree to which the factor means vary linearly, quadratically, and so on with the factor levels.

`contr.poly()` scales each column so that its sum of squares is 1. Each pair of columns is orthogonal, $\mathbf{c}_i^\top \mathbf{c}_j = 0$, so that $\mathbf{C}^\top \mathbf{C} = \mathbf{I}$.

```
C <- contr.poly(4) |> print()
#>      .L     .Q     .C
#> [1,] -0.671  0.5 -0.224
#> [2,] -0.224 -0.5  0.671
#> [3,]  0.224 -0.5 -0.671
#> [4,]  0.671  0.5  0.224

# show they are orthonormal
t(C) %*% C |> zapsmall()
#>      .L     .Q     .C
#> [1,] 1 0 0
#> [2,] 0 1 0
#> [3,] 0 0 1
```

We can get a better sense of orthogonal polynomial contrasts by taking a numeric vector x , and raising it to successive powers, 1, 2, 3. Here $x^0 = 1$ is the constant term or intercept.

```
M <- outer(1:8, 0:3, `^^`)
colnames(M) <- c("int", "lin", "quad", "cubic")
rownames(M) <- paste0("x", 1:8)
M
#>      int lin quad cubic
#> x1    1   1    1    1
#> x2    1   2    4    8
#> x3    1   3    9   27
#> x4    1   4   16   64
#> x5    1   5   25  125
#> x6    1   6   36  216
#> x7    1   7   49  343
#> x8    1   8   64  512
```

Then we can make the columns of M orthogonal using the Gram-Schmidt method, where each successive column after the first is made orthogonal to all previous columns by subtracting their projections on that column. Plotting these, as in Figure 5.3 shows that the coefficients for a linear term plot as a line, those for x^2 follow a quadratic, and so forth.

```
op <- par(mar = c(4, 4, 1, 1)+.1)
M1 <- matlib::GramSchmidt(M, normalize = FALSE)
matplot(M1,
  type = "b",
  pch = as.character(0:3),
  cex = 1.5,
  cex.lab = 1.5,
  lty = 1,
  lwd = 3,
  xlab = "X",
  ylab = "Coefficient")
```

Custom contrasts

You don't have to be constrained by the kinds of comparisons available in `contr.*` functions. For a factor with k levels you are free to make up *any* $k - 1$ contrasts that correspond to $k - 1$ different questions or tests

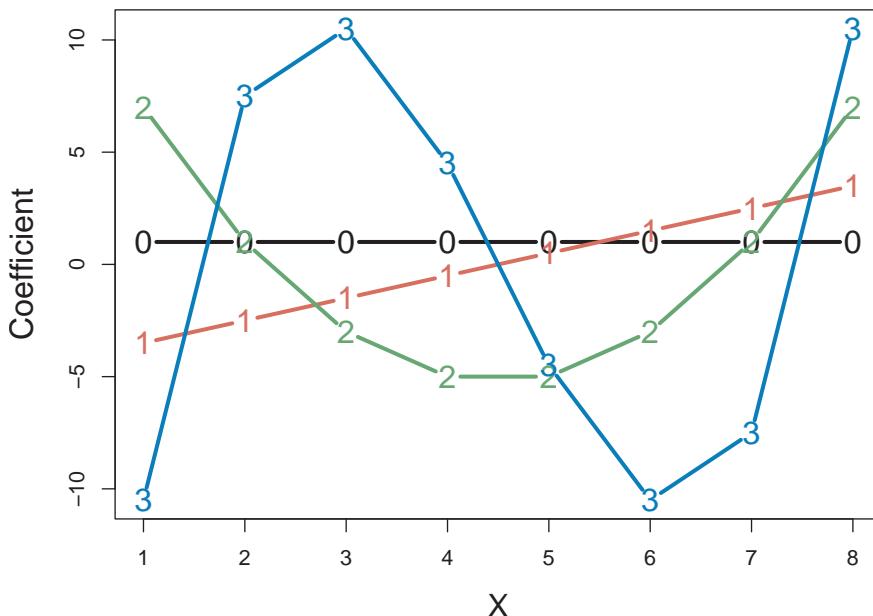


Figure 5.3: The coefficients for orthogonal polynomial contrasts for linear (1), quadratic (2) and cubic (3) terms for a numeric variable X . The intercept or constant term is represented as 0. Orthogonality means that each pair of values is uncorrelated.

of hypotheses about the factor level means. Even better, if your contrasts are orthogonal, their tests are statistically independent.

One useful idea for defining orthogonal comparisons of substantive interest is the idea of **nested dichotomies**. Here you would start with contrasting one meaningful subset of the groups against all the others. Then, successive contrasts are defined by making dichotomies among the groups within each subset.

TODO: Make a figure for the two examples

As one example, suppose we are looking at support on some issue among four political parties: A, B, C and D, where A and B are left-of-center and C and D are to the right of the political spectrum. The following comparisons might of interest:

AB.CD: {A, B} vs. {C, D}
 A.B: {A} vs. {B}
 C.D: {C} vs. {D}

You could set up these comparisons as the following contrasts:

```
# contrasts
AB.CD <- c(1, 1, -1, -1)
A.B   <- c(1, -1, 0, 0)
C.D   <- c(0, 0, 1, -1)

# put them in a matrix
C <- cbind(AB.CD, A.B, C.D)
rownames(C) <- LETTERS[1:4]
C
#>   AB.CD A.B C.D
#> A      1    1    0
```

```
#> B      1   -1    0
#> C     -1    0    1
#> D     -1    0   -1
```

With a data frame like `df` with the factor `party`, you would then use these contrasts in a model by assigning `C` to `contrasts(df$party)`:

```
set.seed(47)
df <- data.frame(
  party = factor(rep(LETTERS[1:4], each = 3)),
  support = c(35, 25, 25, 15) + round(rnorm(12, 0, 1), 1)
)
contrasts(df$party) <- C
```

Then, in a linear model, the coefficients estimate the mean difference between the averages of the subset of parties in each comparison. For example, parties A and B on average are 2.11 higher in support than parties C and D; support for party A is 2.37 greater than party B and so forth.

```
party.mod <- lm(support ~ party, data = df)
coef(party.mod)
#> (Intercept)  partyAB.CD  partyA.B  partyC.D
#>       24.83        2.11        2.37        1.85
```

For another example, say we are examining differences among three psychiatric diagnostic patient groups, “bipolar”, “manic”, “depressed” and also have a matched normal group. One set of meaningful comparisons would be as follows:

```
D1: {Normal} vs. {Bipolar, Depressed, Manic}
D2: {Bipolar} vs. {Depressed, Manic}
D3: {Depressed} vs. {Manic}
```

Weights for these contrasts are assigned by making them positive values for the groups in one subset and negative for the other, and giving numbers that sum to zero for each one:

```
D1 <- c(3, -1, -1, -1)
D2 <- c(0, 2, -1, -1)
D3 <- c(0, 0, 1, -1)

C <- cbind(D1, D2, D3)
rownames(C) <- c("Normal", "Bipolar", "Depressed", "Manic")
C
#>           D1 D2 D3
#> Normal      3  0  0
#> Bipolar     -1  2  0
#> Depressed   -1 -1  1
#> Manic      -1 -1 -1
```

These have the same form as the reversed Helmert contrasts considered earlier.

5.2 What have we learned?

This chapter introduced the fundamental building blocks of linear modeling in R, focusing on the versatile `lm()` function as our primary tool for fitting and understanding linear relationships in data. The goal for the chapter is to help you understand the mechanics of translating between the algebraic formulation of a linear model and your research questions using the tools in the `lm()` family.

Here are the essential takeaways:

- **The `lm()` function is your Swiss Army knife for linear modeling** – Whether you’re fitting simple regression, multiple regression, ANOVA, or ANCOVA models, `lm()` provides a unified interface through R’s elegant formula syntax. The beauty lies in how a model formula like `y ~ x1 + x2 + x1:x2` captures complex relationships with intuitive notation.
- **Model objects contain a wealth of information** – An “`lm`” object isn’t just coefficients; it’s a comprehensive container holding fitted values, residuals, the design matrix, and diagnostic information. Learning to extract and manipulate these components with functions like `coef()`, `fitted()`, `residuals()`, and `model.matrix()` unlocks the full power of linear modeling.
- **Model matrices reveal the algebraic heart of your analysis** – Understanding how R transforms your formula into a design matrix via `model.matrix()` is fundamental to grasping what linear models actually compute. The journey from a formula like `y ~ treatment + block` to a matrix of 0s and 1s illuminates how categorical predictors are used in mathematical operations to calculate fitted values.
- **Contrasts control how factors speak to your research questions** – The choice between treatment, sum, or Helmert contrasts isn’t just technical housekeeping – it determines which comparisons your model coefficients represent. Using `contrasts()` and `C()` strategically means your model output directly answers your scientific questions rather than leaving you to decode cryptic parameter estimates.

However fitting a model is just the first step. While `summary()` and `anova()` provide essential numerical summaries, diagnostic plots created with `plot(lm_object)` expose patterns in residuals. Other plots help to identify influential observations, and validate model assumptions. The interplay between numerical and graphical summaries is where true understanding emerges. This is the topic of Chapter 6.

6

Plots for univariate response models

For a univariate linear model fit using `lm()`, `glm()` and similar functions, the standard `plot()` method gives basic versions of *diagnostic* plots of residuals and other calculated quantities for assessing possible violations of the model assumptions. Some of these can be considerably enhanced using other packages.

Beyond this,

- tables of model coefficients, standard errors and test statistics can often be usefully supplemented or even replaced by suitable *coefficient plots* providing essentially the same information.
- when there are two or more predictors, standard plots of y vs. each x can be misleading because they ignore the other predictors in the model. *Added-variable plots* (also called *partial regression plots*) allow you to visualize *partial* relations between the response and a predictor by adjusting (controlling for) all other predictors. *Component + residual* plots help to diagnose nonlinearity.
- in this same situation, you can more easily understand their separate impact on the response by plotting the marginal *predicted effects* of one or more focal variables, averaging over other variables not shown in a given plot.
- when there are highly correlated predictors, some specialized plots are useful to understand the nature of *multicollinearity*.

The classic reference on regression diagnostics is Belsley et al. (1980). My favorite modern texts are the brief Fox (2020) and the more complete Fox & Weisberg (2018a), both of which are supported by the `car` package (Fox et al., 2023). Some of the examples in this chapter are inspired by Fox & Weisberg (2018a).

Packages

In this chapter I use the following packages. Load them now:

```
library(car)
library(dplyr)
library(easystats)
library(effects)
library(ggeffects)
library(ggplot2)
library(ggstats)
library(marginaleffects)
library(modelsummary)
library(performance)
library(tidyr)
```

6.1 The “regression quartet”

For a fitted model, plotting the model object with `plot(model)` provides for any of six basic plots, of which four are produced by default, giving rise to the term *regression quartet* for this collection. These are:

- **Residuals vs. Fitted:** For well-behaved data, the points should hover around a horizontal line at residual = 0, with no obvious pattern or trend.
- **Normal Q-Q plot:** A plot of sorted standardized residuals e_i (obtained from `rstudent(model)`) against the theoretical values those values would have in a standard normal $\mathcal{N}(0, 1)$ distribution.
- **Scale-Location:** Plots the square-root of the absolute values of the standardized residuals $\sqrt{|e_i|}$ as a measure of “scale” against the fitted values \hat{y}_i as a measure of “location”. This provides an assessment of homogeneity of variance. Violations appear as a tendency for scale (variability) to vary with location.
- **Residuals vs. Leverage:** Plots standardized residuals against leverage to help identify possibly influential observations. Leverage, or “hat” values (given by `hat(model)`) are proportional to the squared Mahalanobis distances of the predictor values \mathbf{x}_i from the means, and measure the potential of an observation to change the fitted coefficients if that observation was deleted. Actual influence is measured by Cooks’s distance (`cooks.distance(model)`) and is proportional to the product of residual times leverage. Contours of constant Cook’s D are added to the plot.

One key feature of these plots is providing **reference** lines or smoothed curves for ease of judging the extent to which a plot conforms to the expected pattern; another is the **labeling** of observations which deviate from an assumption.

The base-R `plot(model)` plots are done much better in a variety of packages. I illustrate some versions from the `car` (Fox et al., 2023) and `performance` (Lüdecke et al., 2021) packages, part of the `easystats` (Lüdecke et al., 2022) suite of packages.

6.1.1 Example: Duncan’s occupational prestige

In a classic study in sociology, Duncan (1961) used data from the U.S. Census in 1950 to study how one could predict the prestige of occupational categories — which is hard to measure — from available information in the census for those occupations. His data is available in `carData:Duncan`, and contains

- `type`: the category of occupation, one of `prof` (professional), `wc` (white collar) or `bc` (blue collar);
- `income`: the percentage of occupational incumbents with a reported income > 3500 (about 40,000 in current dollars);
- `education`: the percentage of occupational incumbents who were high school graduates;
- `prestige`: the percentage of respondents in a social survey who rated the occupation as “good” or better in prestige.

These variables are a bit quirky in they are measured in percents, 0-100, rather dollars for `income` and years for `education`, but this common scale permitted Duncan to ask an interesting sociological question: Assuming that both income and education predict prestige, are they equally important, as might be assessed by testing the hypothesis $H_0 : \beta_{\text{income}} = \beta_{\text{education}}$. If so, this would provide a very simple theory for occupational prestige.

A quick look at the data shows the variables and a selection of the occupational categories, which are the `row.names()` of the dataset.

```
data(Duncan, package = "carData")
set.seed(42)
car::some(Duncan)
```

```
#>           type income education prestige
#> accountant     prof    62      86      82
#> professor      prof    64      93      93
#> engineer       prof    72      86      88
#> factory.owner   prof    60      56      81
#> store.clerk      wc     29      50      16
#> carpenter        bc     21      23      33
#> machine.operator bc     21      20      24
#> barber          bc     16      26      20
#> soda.clerk      bc     12      30       6
#> janitor         bc      7      20       8
```

Let's start by fitting a simple model using just income and education as predictors. The results look very good! Both `income` and `education` are highly significant and the $R^2 = 0.828$ for the model indicates that `prestige` is very well predicted by just these variables.

```
duncan.mod <- lm(prestige ~ income + education, data=Duncan)
summary(duncan.mod)

#>
#> Call:
#> lm(formula = prestige ~ income + education, data = Duncan)
#>
#> Residuals:
#>   Min     1Q Median     3Q    Max
#> -29.54  -6.42   0.65   6.61  34.64
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -6.0647    4.2719  -1.42   0.16
#> income       0.5987    0.1197   5.00  1.1e-05 ***
#> education     0.5458    0.0983   5.56  1.7e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 13.4 on 42 degrees of freedom
#> Multiple R-squared:  0.828, Adjusted R-squared:  0.82
#> F-statistic: 101 on 2 and 42 DF, p-value: <2e-16
```

Beyond this, Duncan was interested in the coefficients and whether income and education could be said to have equal impacts on predicting occupational prestige. A nice display of model coefficients with confidence intervals is provided by `parameters::model_parameters()`.

```
parameters::model_parameters(duncan.mod)
#> Parameter | Coefficient | SE |      95% CI | t(42) |      p
#> -----
#> (Intercept) |      -6.06 | 4.27 | [-14.69, 2.56] | -1.42 | 0.163
#> income      |       0.60 | 0.12 | [ 0.36, 0.84] |  5.00 | < .001
#> education    |       0.55 | 0.10 | [ 0.35, 0.74] |  5.56 | < .001
```

We can also test Duncan's hypothesis that income and education have equal effects on prestige with `car::linearHypothesis()`. This is constructed as a test of a restricted model in which the two coefficients are forced to be equal against the unrestricted model. Duncan was very happy with this result.

```

car::linearHypothesis(duncan.mod, "income = education")
#>
#> Linear hypothesis test:
#> income - education = 0
#>
#> Model 1: restricted model
#> Model 2: prestige ~ income + education
#>
#>   Res.Df  RSS Df Sum of Sq    F Pr(>F)
#> 1     43 7519
#> 2     42 7507  1      12.2 0.07    0.8

```

Equivalently, the linear hypothesis that $\beta_{\text{Inc}} = \beta_{\text{Educ}}$ can be tested with a Wald test for difference between these coefficients, expressed as $\mathcal{H}_0 : \mathbf{C}\beta = 0$, using $\mathbf{C} = (0, -1, 1)$. The estimated value, -0.053 has a confidence interval [-0.462, 0.356], consistent with Duncan's hypothesis.

```

wtest <- spida2::wald(duncan.mod, c(0, -1, 1))[[1]]
wtest$estimate
#>
#>           Estimate Std. Error DF t-value p-value Lower 0.95 Upper 0.95
#> Larg -0.0529     0.203 42 -0.261   0.795     -0.462     0.356

```

We can visualize this test and confidence intervals using a joint confidence ellipse for the coefficients for income and education in the model `duncan.mod`. In Figure 6.1 the size of the ellipse is set to $\sqrt{F_{1,\nu}^{0.95}} = t_\nu^{0.95}$, so that its shadows on the horizontal and vertical axes correspond to 1D 95% confidence intervals. In this plot, the line through the origin with slope = 1 corresponds to equal coefficients for income and education and the line with slope = -1 corresponds to their difference, $\beta_{\text{Educ}} - \beta_{\text{Inc}}$. The orthogonal projection of the coefficient vector $(\hat{\beta}_{\text{Inc}}, \hat{\beta}_{\text{Educ}})$ (the center of the ellipse) is the point estimate of $\hat{\beta}_{\text{Educ}} - \hat{\beta}_{\text{Inc}}$ and the shadow of the ellipse along this axis is the 95% confidence interval for the difference in slopes.

```

confidenceEllipse(duncan.mod, col = "blue",
  levels = 0.95, dfn = 1,
  fill = TRUE, fill.alpha = 0.2,
  xlim = c(-.4, 1),
  ylim = c(-.4, 1), asp = 1,
  cex.lab = 1.3,
  grid = FALSE,
  xlab = expression(paste("Income coefficient, ", beta[Inc])),
  ylab = expression(paste("Education coefficient, ", beta[Educ])))

abline(h=0, v=0, lwd = 2)

# confidence intervals for each coefficient
beta <- coef( duncan.mod )[-1]
CI <- confint(duncan.mod)      # confidence intervals
lines( y = c(0,0), x = CI["income",] , lwd = 5, col = 'blue')
lines( x = c(0,0), y = CI["education",] , lwd = 5, col = 'blue')
points(rbind(beta), col = 'black', pch = 16, cex=1.5)
points(diag(beta) , col = 'black', pch = 16, cex=1.4)
arrows(beta[1], beta[2], beta[1], 0, angle=8, len=0.2)
arrows(beta[1], beta[2], 0, beta[2], angle=8, len=0.2)

```

```

# add line for equal slopes
abline(a=0, b = 1, lwd = 2, col = "darkgreen")
text(0.8, 0.8, expression(beta[Educ] == beta[Inc]),
     srt=45, pos=3, cex = 1.5, col = "darkgreen")

# add line for difference in slopes
col <- "darkred"
x <- c(-1.5, .5)
lines(x=x, y=-x)
text(-.15, -.15, expression(~beta["Educ"] - ~beta["Inc"]),
     col=col, cex=1.5, srt=-45)

# confidence interval for b1 - b2
wtest <- spida2::wald(duncan.mod, c(0, -1, 1))[[1]]
lower <- wtest$estimate$Lower / 2
upper <- wtest$estimate$Upper / 2
lines(-c(lower, upper), c(lower,upper), lwd=6, col=col)

# projection of (b1, b2) on b1-b2 axis
beta <- coef( duncan.mod )[-1]
bdiff <- beta %*% c(1, -1)/2
points(bdiff, -bdiff, pch=16, cex=1.3)
arrows(beta[1], beta[2], bdiff, -bdiff,
       angle=8, len=0.2, col=col, lwd = 2)

# calibrate the diff axis
ticks <- seq(-0.3, 0.3, by=0.2)
ticklen <- 0.02
segments(ticks, -ticks, ticks-sqrt(2)*ticklen, -ticks-sqrt(2)*ticklen)
text(ticks-2.4*ticklen, -ticks-2.4*ticklen, ticks, srt=-45)

```

6.1.2 Diagnostic plots

But, should Duncan be so happy? It is unlikely that he ran any model diagnostics or plotted his model; we do so now. Here is the regression quartet (Figure 6.2) for this model. Each plot shows some trend lines, and importantly, labels some observations that stand out and might deserve attention.

```

op <- par(mfrow = c(2,2),
          mar = c(4,4,3,1)+.1)
plot(duncan.mod, lwd=2, pch=16)
par(op)

```

Some points to note:

- A few observations (minister, reporter, conductor, contractor) are flagged in multiple panels. It turns out (Section 6.6.3) that the observations for minister and reporter noted in the residuals vs. leverage plot are highly influential and largely responsible for Duncan's finding that the slopes for income and education could be considered equal.
- The red trend line in the scale-location plot indicates that residual variance is not constant, but rather increases from both ends. This is a consequence of the fact that `prestige` is measured as a percentage, bounded at [0, 100], and the standard deviation of a percentage p is proportional to $\sqrt{p \times (1-p)}$ which is maximal at $p = 0.5$.

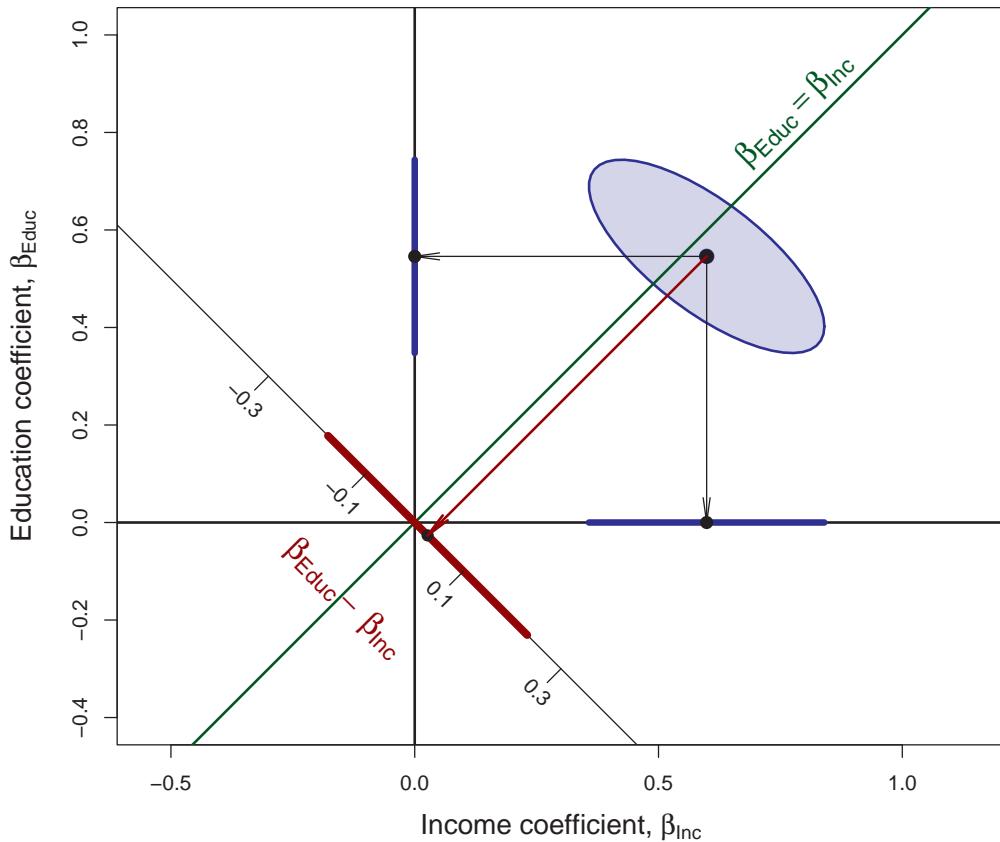


Figure 6.1: Joint 95% confidence ellipse for $(\beta_{\text{Inc}}, \beta_{\text{Educ}})$, together with their 1D shadows, which give 95% confidence intervals for the separate coefficients and the linear hypothesis that the coefficients are equal. Projecting the confidence ellipse along the line with unit slope gives a confidence interval for the difference between coefficients, shown by the dark red line.

Similar, but nicer-looking diagnostic plots are provided by `performance::check_model()` which uses `ggplot2` for graphics. These include helpful captions indicating what should be observed for each for a good-fitting model. However, they don't have as good facilities for labeling unusual observations as the base R `plot.lm()` method or functions in the `car`.

```
check_model(duncan.mod,
            check=c("linearity", "qq",
                   "homogeneity", "outliers"))
```

6.1.3 Example: Canadian occupational prestige

Following Duncan (1961), occupational prestige was studied in a Canadian context by Bernard Blishen and others at York University, giving the dataset `Prestige` which we looked at in Example 3.2. It differs from the `Duncan` dataset primarily in that the main variables—prestige, income and education were revamped to better reflect the underlying constructs in more meaningful units.

- `prestige`: Rather than a simple percentage of “good+” ratings, this uses a wider and more reliable scale from Pineo & Porter (1967) on a scale from 10–90.
- `income` is measured as the average income of incumbents in each occupation, in 1971 dollars, rather than percent exceeding a given threshold (\$3500)

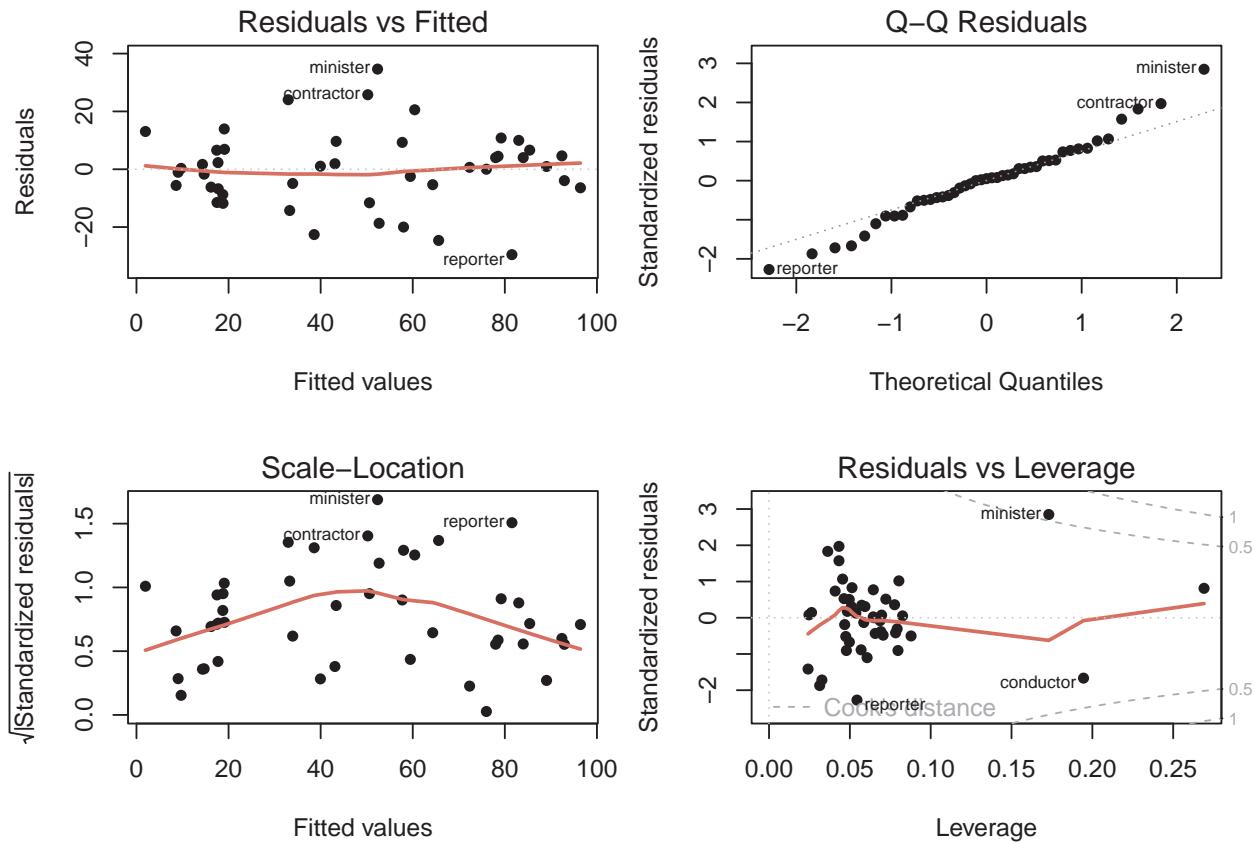


Figure 6.2: Regression quartet of diagnostic plots for the Duncan data. Several possibly unusual observations are labeled.

- education is measured as the average education of occupational incumbents, years.

The dataset again includes type of occupation with the same levels "bc" (blue collar), "wc" (white collar) and "prof" (professional)¹, but in addition includes the percent of women in these occupational categories.

Our interest again is in understanding how prestige is related to census measures of the average education, income, percent women of incumbents in those occupations, but with attention to the scales of measurement and possibly more complex relationships.

```
data(Prestige, package="carData")
# Reorder levels of type
Prestige$type <- factor(Prestige$type,
                         levels=c("bc", "wc", "prof"))
str(Prestige)
#> 'data.frame': 102 obs. of 6 variables:
#> $ education: num 13.1 12.3 12.8 11.4 14.6 ...
#> $ income   : int 12351 25879 9271 8865 8403 11030 8258 14163 11377 11023 ...
#> $ women    : num 11.16 4.02 15.7 9.11 11.68 ...
#> $ prestige  : num 68.8 69.1 63.4 56.8 73.5 77.6 72.6 78.1 73.1 68.8 ...
```

¹Note that the factor type in the dataset has its levels ordered alphabetically. For analysis and graphing it is useful to reorder the levels in the natural increasing order. An alternative is to make type an *ordered* factor, but this would represent it using polynomial contrasts for linear and quadratic trends, which seems useful in this context.

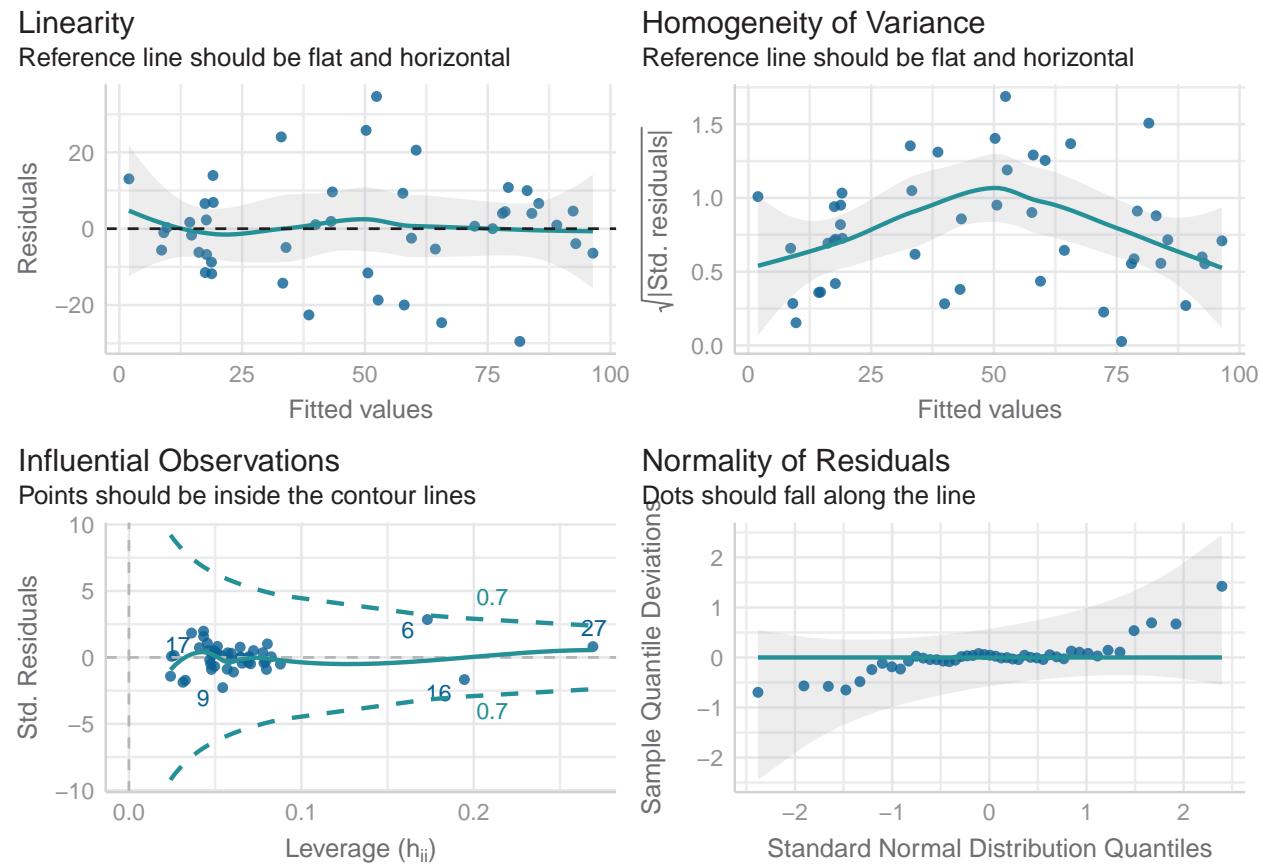


Figure 6.3: Diagnostic plots for the Duncan data, using `performance::check_model()`.

```
#> $ census    : int  1113 1130 1171 1175 2111 2113 2133 2141 2143 2153 ...
#> $ type      : Factor w/ 3 levels "bc","wc","prof": 3 3 3 3 3 3 3 3 3 3 ...
```

We fit a main-effects model using all predictors (ignoring `census`, the Canadian Census occupational code):

```
prestige.mod <- lm(prestige ~ education + income + women + type,
                     data=Prestige)
```

`plot(model)` produces four separate plots. For a quick look, I like to arrange them in a single 2x2 figure.

```
op <- par(mfrow = c(2,2),
           mar=c(4,4,3,1)+.1)
plot(prestige.mod, lwd=2, cex.lab=1.4)
par(op)
```

6.2 Other Model plots

TODO: What goes here?

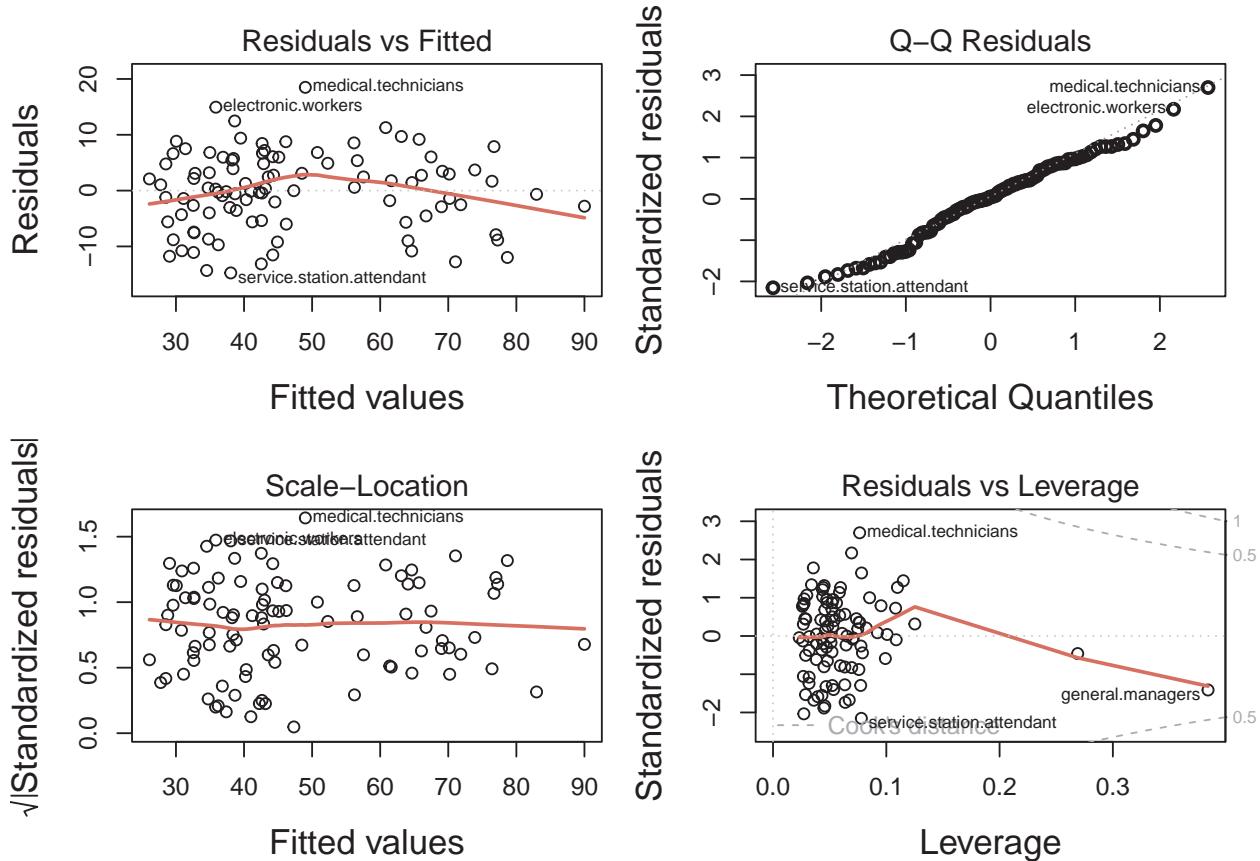


Figure 6.4: Regression quartet of diagnostic plots for the `Prestige` data. Several possibly unusual observations are labeled in each plot.

6.3 Coefficient displays

The results of linear models are most often reported in tables and typically with “significance stars” (*, **, ***) to indicate the outcome of hypothesis tests. These are useful for looking up precise values and you can use this format to compare a small number of competing models side-by-side. However, as illustrated by Kastellec & Leoni (2007), plots of coefficients can increase the clarity of presentation and make it easier to draw correct conclusions. Yet, when you need to present tables, there is a variety of tools in R that can help make them attractive in publications.

For illustration, I'll consider three models for the `Prestige` data of increasing complexity:

- `mod1` fits the main effects of the three quantitative predictors;
- `mod2` adds the categorical variable `type` of occupation;
- `mod3` allows an interaction of `income` with `type`.

```
mod1 <- lm(prestige ~ education + income + women,
            data=Prestige)
mod2 <- lm(prestige ~ education + women + income + type,
            data=Prestige)
```

```
mod3 <- lm(prestige ~ education + women + income * type,
            data=Prestige)
```

From our earlier analyses (`?@sec-prestige`) (Example 3.2) we saw that the marginal relationship between `income` and `prestige` was nonlinear (Figure 3.12), and was better represented in a linear model using `log(income)` (Section 3.2.2.1) shown in Figure 3.15. However, this possibly non-linear relationship could also be explained by stratifying (Section 3.2.2.2) the data by `type` of occupation (Figure 3.16).

6.3.1 Displaying coefficients

`summary()` gives the complete precis of a fitted model, with information about the estimated coefficients, residuals and goodness-of-fit statistics like R^2 . But if you only want to see the coefficients, standard errors, etc. `lmtest::coeftest()` gives these results in the familiar format for console output. `broom::tidy()` places these in a tidy format common to many modeling functions which is useful for further processing (e.g., comparing models).

```
lmtest::coeftest(mod1)
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -6.794334   3.239089  -2.10   0.039 *
#> education    4.186637   0.388701   10.77  < 2e-16 ***
#> income       0.001314   0.000278    4.73  7.6e-06 ***
#> women        -0.008905  0.030407   -0.29   0.770
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

broom::tidy(mod1)
#> # A tibble: 4 x 5
#>   term      estimate std.error statistic p.value
#>   <chr>     <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept) -6.79      3.24     -2.10  3.85e- 2
#> 2 education    4.19      0.389     10.8   2.59e-18
#> 3 income       0.00131   0.000278    4.73  7.58e- 6
#> 4 women        -0.00891  0.0304     -0.293 7.70e- 1
```

The `modelsummary` package (Arel-Bundock, 2025b) is an easy to use, very general package to summarize data and statistical models in R. The main function `modelsummary()` can produce highly customizable tables of coefficients in a wide variety of output formats, including HTML, PDF, LaTeX, Markdown, and MS Word. You can select the statistics displayed for any model term with the `estimate` and `statistic` arguments.

```
modelsummary(list("Model1" = mod1),
            coef.omit = "Intercept",
            shape = term ~ statistic,
            estimate = "{estimate} [{conf.low}, {conf.high}]",
            statistic = c("std.error", "p.value"),
            fmt = fmt_statistic("estimate" = 3, "conf.low" = 3, "conf.high" = 3),
            gof.omit = ".")
```

`gof.omit` allows you to omit or select the goodness-of-fit statistics and other model information available from those listed by `get_gof()`:

Table 6.1: Table of coefficients for the main effects model.

Model1			
	Est.	S.E.	p
education	4.187 [3.415, 4.958]	0.389	0.000
income	0.001 [0.001, 0.002]	0.000	0.000
women	-0.009 [-0.069, 0.051]	0.030	0.770

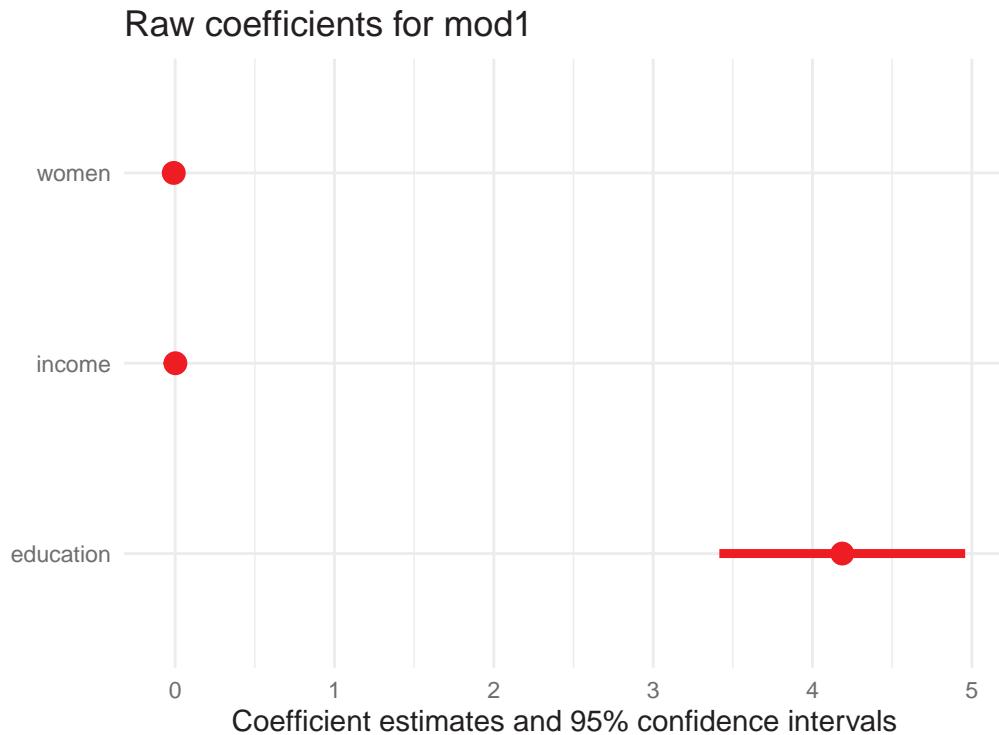
```
get_gof(mod1)
#>   aic bic r.squared adj.r.squared rmse nobs   F logLik
#> 1 716 729      0.798        0.792 7.69 102 129    -353
```

6.3.2 Visualizing coefficients

`modelplot()` is the companion function. It allows you to plot model estimates and confidence intervals. It makes it easy to subset, rename, and customize plots using same mechanics as in `modelsummary()`.

```
theme_set(theme_minimal(base_size = 14))

modelplot(mod1, coef_omit="Intercept",
          color="red", size=1, linewidth=2) +
  labs(title="Raw coefficients for mod1")
```

**Figure 6.5:** Plot of coefficients and their standard error bar for the simple main effects model

But this plot is disappointing and misleading because it shows the **raw** coefficients. From the plot, it looks

Table 6.2: Table of coefficients for three models.

	Model1			Model2			Model3		
	Est.	S.E.	p	Est.	S.E.	p	Est.	S.E.	p
education	4.19***	0.39	<0.01	3.66***	0.65	<0.01	2.80***	0.59	<0.01
income	0.00***	0.00	<0.01	0.00***	0.00	<0.01	0.00***	0.00	<0.01
women	-0.01	0.03	0.77	0.01	0.03	0.83	0.08*	0.03	0.02
typewc				-2.92	2.67	0.28	3.43	5.37	0.52
typeprof				5.91	3.94	0.14	27.55***	5.41	<0.01
income × typewc							-0.00	0.00	0.21
income × typeprof							-0.00***	0.00	<0.01
RMSE	7.69			6.91			6.02		
R2	0.798			0.835			0.875		

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001

like only `education` has a non-zero effect, but the effect of `income` is also highly significant. The problem is that the magnitude of the coefficient $\hat{b}_{\text{education}}$ is more than 40,000 times that of the other coefficients, because education is measured years, while income is measured in dollars. The 95% confidence interval for $\hat{b}_{\text{income}} = [0.0008, 0.0019]$, but this is invisible in the plot.

Before figuring out how to fix this issue, I show the comparable displays from `modelsummary()` and `modelplot()` for all three models. When you give `modelsummary()` a list of models, it displays their coefficients side-by-side as shown in Table 6.2.

```
models <- list("Model1" = mod1, "Model2" = mod2, "Model3" = mod3)
modelsummary(models,
  coef_omit = "Intercept",
  fmt = 2,
  stars = TRUE,
  shape = term ~ statistic,
  statistic = c("std.error", "p.value"),
  gof_map = c("rmse", "r.squared")
)
```

Note that a factor predictor (like `type` here) with d levels is represented by $d - 1$ coefficients in main effects and in interactions with quantitative variables. These levels are coded with *treatment contrasts* by default. Also by default, the first level is set as the reference level in alphabetical order. Here the reference level is blue collar (`bc`), so the coefficient `typeprof = 5.91` indicates that professional occupations on average are rated 5.91 greater on the Prestige scale than blue collar workers.

Note also that unlike the table, the coefficients in Figure 6.5 are ordered from bottom to top, because the Y axis starts at the lower left corner. In Figure 6.6 I use `scale_y_discrete()` to reverse the order. It is also useful to add a vertical reference line at $\beta = 0$.

```
modelplot(models,
  coef_omit="Intercept",
  size=1.3, linewidth=2) +
  ggtitle("Raw coefficients") +
```

```
geom_vline(xintercept = 0, linewidth=1.5) +
scale_y_discrete(limits=rev) +
theme(legend.position = "inside",
legend.position.inside = c(0.85, 0.2))
```

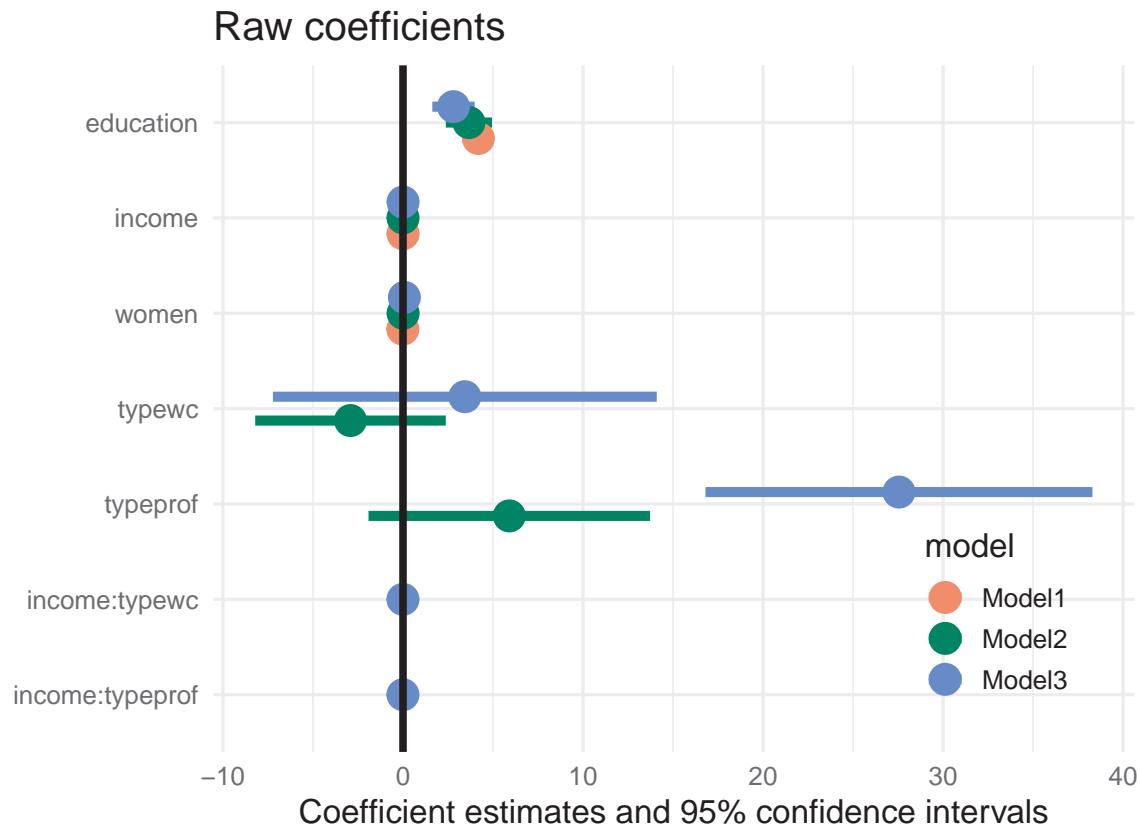


Figure 6.6: Plot of raw coefficients and their confidence intervals for all three models

6.3.3 More useful coefficient plots

The problem with plots of raw coefficients shown in Figure 6.5 and Figure 6.6 is that the coefficients for different predictors are not directly comparable because they are measured in different units.

One alternative is to plot the *standardized coefficients*. Another way is to re-scale the predictors into more comparable and meaningful units. I illustrate these ideas below.

Standardized coefficients

The simplest way to do this is to transform all variables to standardized (z) scores. The coefficients are then interpreted as the standardized change in prestige for a one standard deviation change in the predictors. The syntax below uses `scale` to transform all the numeric variables. Then, we re-fit the models using the standardized data.

```
Prestige_std <- Prestige |>
  as_tibble() |>
  mutate(across(where(is.numeric), scale))
```

```

mod1_std <- lm(prestige ~ education + income + women,
                 data=Prestige_std)
mod2_std <- lm(prestige ~ education + women + income + type,
                 data=Prestige_std)
mod3_std <- lm(prestige ~ education + women + income * type,
                 data=Prestige_std)

```

The plot in Figure 6.7 now shows the significant effect of income in all three models. As well, it offers a more sensitive comparison of the coefficients of other terms across models; for example `women` is not significant in models 1 and 2, but becomes significant in Model 3 when the interaction of `income * type` is included.

```

models <- list("Model1" = mod1_std, "Model2" = mod2_std, "Model3" = mod3_std)
modelplot(models,
          coef_omit="Intercept", size=1.3) +
  ggtitle("Standardized coefficients") +
  geom_vline(xintercept = 0, linewidth = 1.5) +
  scale_y_discrete(limits=rev) +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.2))

```

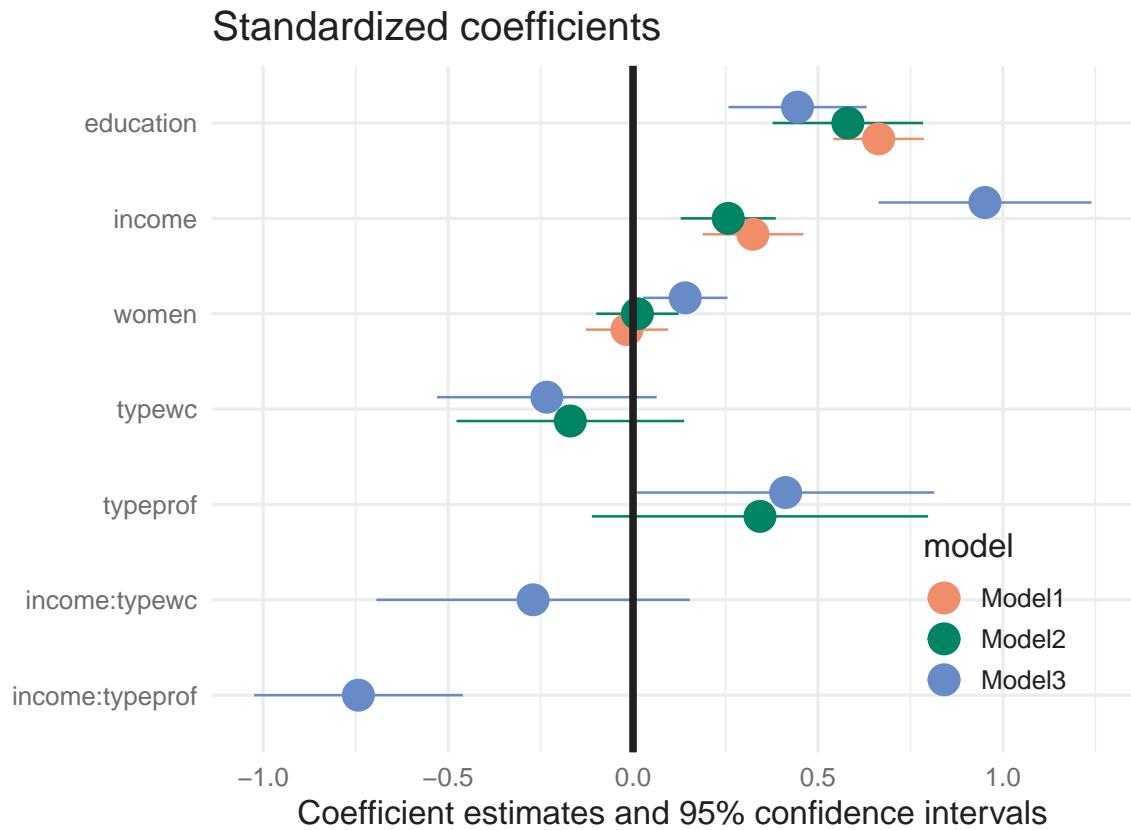


Figure 6.7: Plot of standardized coefficients and their confidence intervals for all three models

It turns out there is an easier way to get plots of standardized coefficients. `modelsummary()` extracts coefficients from model objects using the `parameters` package, and that package offers several options for standardization: See [model parameters documentation](#). We can pass the `standardize="refit"` (or other)

argument directly to `modelsummary()` or `modelplot()`, and that argument will be forwarded to `parameters`. The plot produced by the code below is identical to Figure 6.7 and is not shown.

```
modelplot(list("mod1" = mod1, "mod2" = mod2, "mod3" = mod3),
          standardize = "refit",
          coef OMIT="Intercept", size=1.3) +
  ggtitle("Standardized coefficients") +
  geom_vline(xintercept = 0, linewidth=1.5) +
  scale_y_discrete(limits=rev) +
  theme(legend.position = "inside",
        legend.position.inside = c(0.85, 0.2))
```

The `ggstats` package (Larmarange, 2025) provides even nicer versions of coefficient plots that handle factors in a more reasonable way, as levels within the factor. `ggcoef_model()` plots a single model and `ggcoef_compare()` plots a list of models using sensible defaults. A small but nice feature is that it explicitly shows the 0 value for the reference level of a factor (`type = "bc"` here) and uses better labels for factors and their interactions.

```
models <- list(
  "Base model"      = mod1_std,
  "Add type"        = mod2_std,
  "Add interaction" = mod3_std)

ggcoef_compare(models) +
  labs(x = "Standarized Coefficient")
```

More meaningful units

Standardizing the variables makes the coefficients directly comparable, but it may be harder to understand what they mean in terms of the variables. For example, the coefficient of `income` in `mod2_std` is 0.25. A literal interpretation is that occupational prestige is expected to increase 0.25 standard deviations for each standard deviation increase in income, but it may be difficult to appreciate what this means.

A better substantive comparison of the coefficients could use understandable scales for the predictors, e.g., months of education, \$100,000 of income or 10% of women's participation. Note that the effect of this is just to multiply the coefficients and their standard errors by a factor. The statistical conclusions of significance are unchanged.

For simplicity, I do this just for Model 1.

```
Prestige_scaled <- Prestige |>
  mutate(education = 12 * education,
        income = income / 100,
        women = women / 10)

mod1_scaled <- lm(prestige ~ education + income + women,
                  data=Prestige_scaled)
```

When we plot this with `ggcoef_model()`, there are many options to control how variables are labeled and other details.

```
ggcoef_model(mod1_scaled,
             signif_stars = FALSE,
```

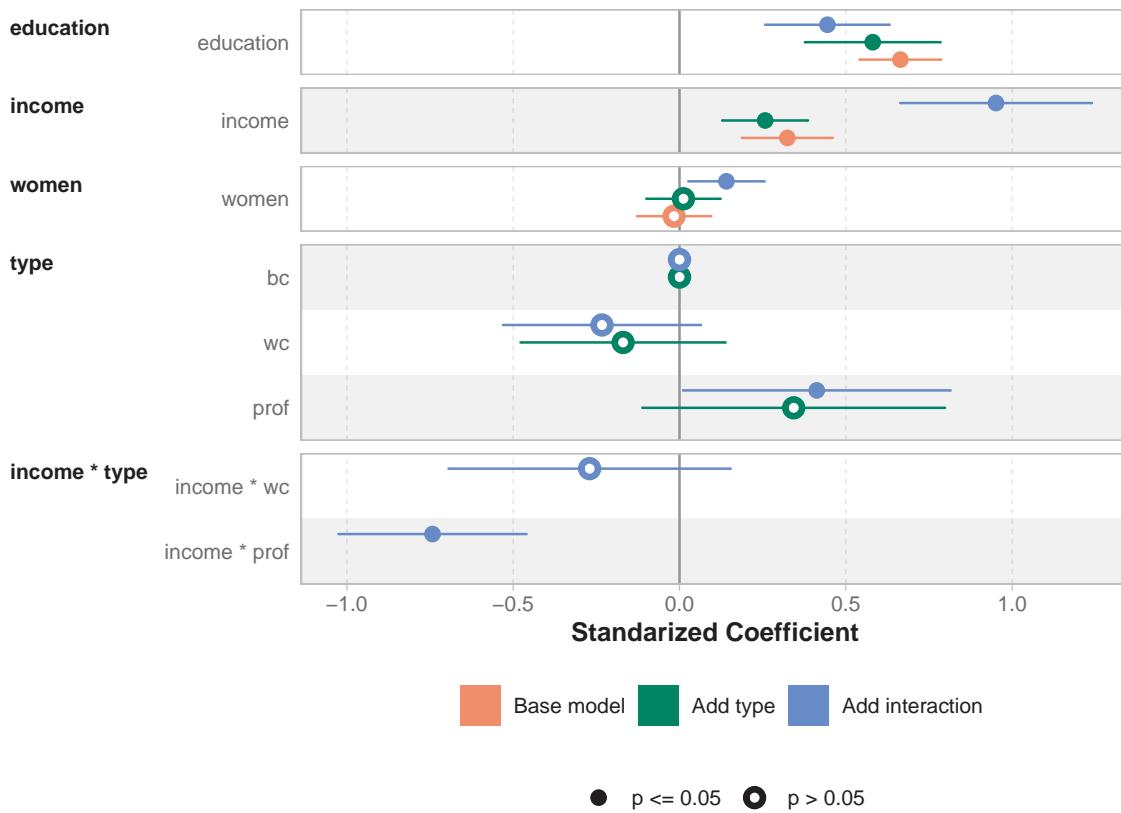


Figure 6.8: Model comparison plot from `ggcoef_compare()`

```
variable_labels = c(education = "education\n(months)",
                  income = "income\n(/$100K)",
                  women = "women\n(/10%)") +
  xlab("Coefficients for prestige with scaled predictors")
```

So, on average, each additional month of education increases the prestige rating by 0.34 units, while an additional \$100,000 of income increases it by 0.13 units. While these are significant effects, they are not large in relation to the scale of `prestige` which ranges 14.8—87.2.

6.4 Added-variable and related plots

In multiple regression problems, it is most often useful to construct a scatterplot matrix and examine the plot of the response vs. each of the predictors as well as those of the predictors against each other. However, the simple, *marginal* scatterplots of a response y against *each* of several predictors x_1, x_2, \dots can be misleading because each one ignores the other predictors.

To see this consider a toy dataset, `coffee`, giving measures of coffee consumption, occupational stress and an index of heart problems in a sample of $n = 20$ graduate students and professors.

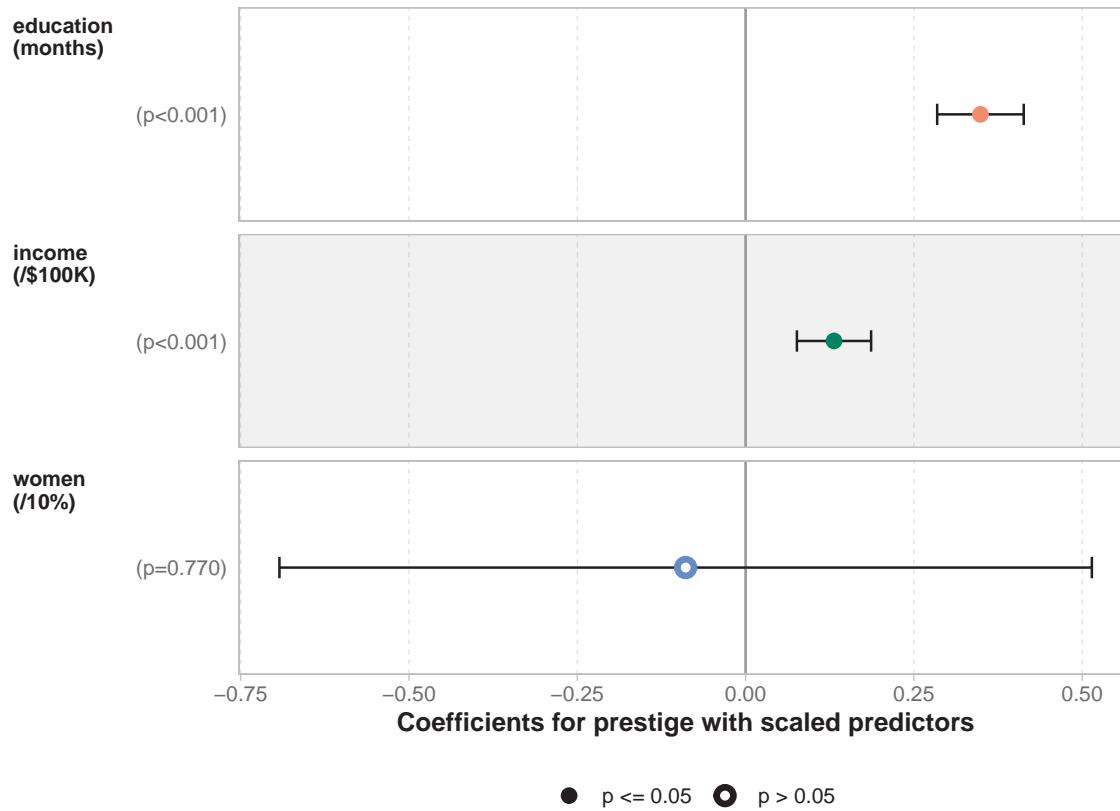


Figure 6.9: Plot of coefficients for prestige with scaled predictors for Model 1.

```
data(coffee, package="matlib")

scatterplotMatrix(~ Heart + Coffee + Stress,
  data=coffee,
  smooth = FALSE,
  ellipse = list(levels=0.68, fill.alpha = 0.1),
  pch = 19, cex.labels = 2.5)
```

The message from these marginal plots in Figure 6.10 seems to be that coffee is bad for your heart, stress is bad for your heart, and stress is also strongly related to coffee consumption. Yet, when we fit a model with both variables together, we get the following results:

```
fit.both <- lm(Heart ~ Coffee + Stress, data=coffee)
lmtest::coeftest(fit.both)
#>
#> t test of coefficients:
#>
#>            Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -7.794      5.793   -1.35    0.20
#> Coffee       -0.409      0.292   -1.40    0.18
#> Stress        1.199      0.224    5.34 5.4e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

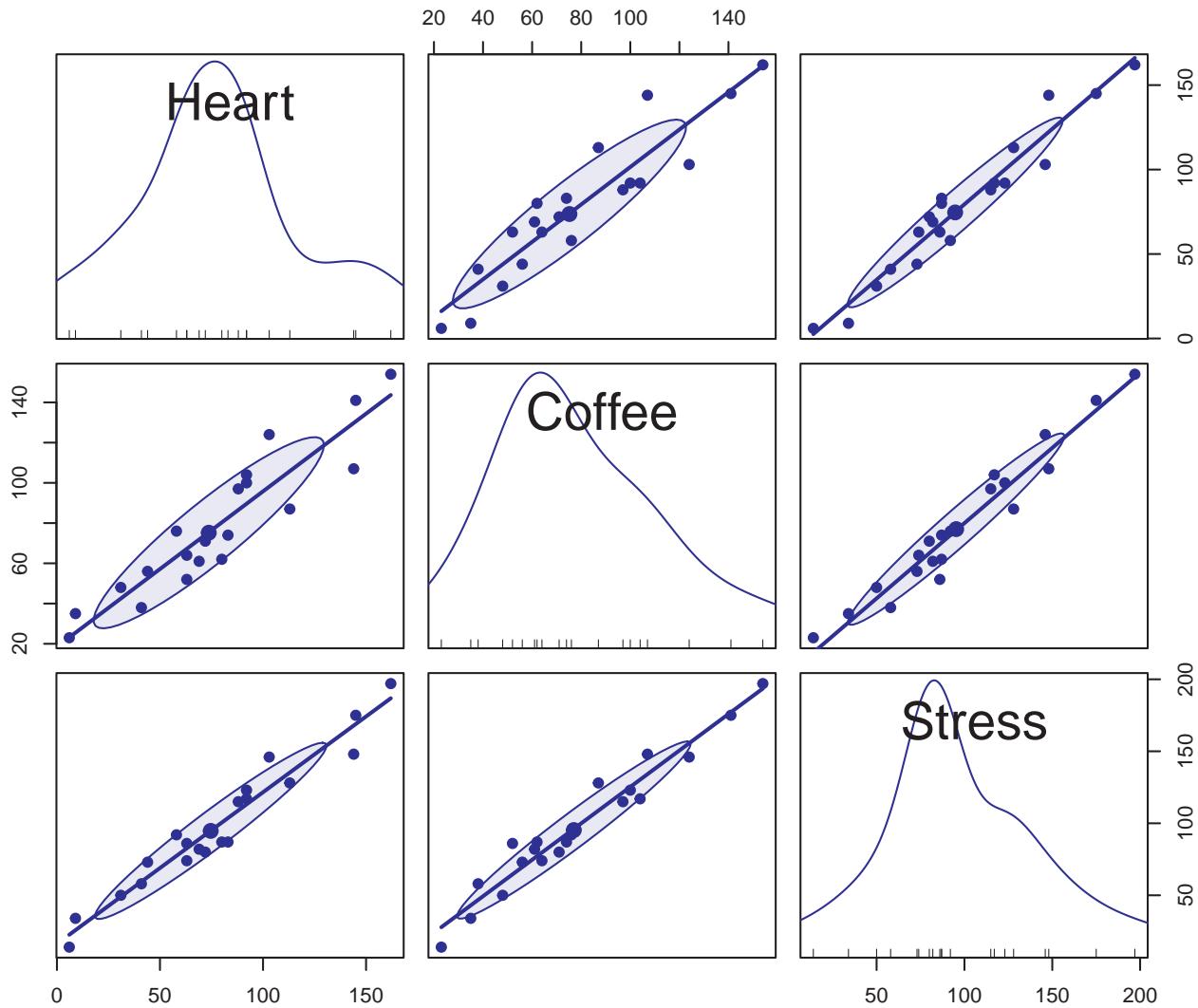


Figure 6.10: Scatterplot matrix showing pairwise relations among Heart (y), Coffee consumption (x_1) and Stress (x_2), with linear regression lines and 68% data ellipses for the bivariate relations

The coefficients suggest that stress is indeed bad for your heart, but the negative (though non-significant) coefficient for coffee suggests that coffee is good for you. How can this be? Does that mean I should drink more coffee, while avoiding stress?

The reason for this apparent paradox is that the general linear model fit by `lm()` estimates all effects together and so the coefficients pertain to the *partial* effect of a given predictor, *adjusting* for the effects of all others. That is, the coefficient for coffee ($\beta_{\text{Coffee}} = -0.41$) estimates the effect of coffee for people with *same* level of stress. In the marginal scatterplot, the positive slope for coffee (1.10) ignores the correlation of coffee and stress.

This is an example of *confounding* in regression when an important predictor is omitted. Stress is positively associated with both coffee consumption and heart damage. When stress is omitted, the coefficient for coffee is biased because it “picks up” the relation with the omitted variable.

A solution to this problem is the *added-variable* plot (“AV plot”, also called *partial regression* plot, MostellerTukey-1977). This is a multivariate analog of a simple marginal scatterplot, designed to visualize directly the partial relation between y and the predictors x_1, x_2, \dots in a multiple regression model.

You can think of this as a magic window that hides the relations of all other variables with each of the y and x_i shown in a given added-variable plot. This gives an unobstructed view of the net relation between y and x_i with the effect of all other variables removed. In effect, it reduces the problem of viewing the complete model in p -dimensional space to a sequence of p 2D plots, each of which tells the story of one predictor, unentangled from the others. This is essentially the same idea as the partial variables plot (Section 3.15.3) used to understand partial correlations.

The construction of an AV plot is conceptually very simple. For variable x_i , imagine that we fit two supplementary regressions:

- Regress \mathbf{y} on $\mathbf{X}_{(-i)}$, the model matrix of all of the regressors except x_i . By definition, the residuals from this regression, $\mathbf{y}^* \equiv \mathbf{y} | \text{others} = \mathbf{y} - \hat{\mathbf{y}} | \mathbf{X}_{(-i)}$, are the part of \mathbf{y} that cannot be explained by all the other regression terms. These residuals are necessarily uncorrelated with the other predictors.
- Regress x_i on the other predictors, $\mathbf{X}_{(-i)}$ and again obtain the residuals. These residuals, $\mathbf{x}_i^* \equiv \mathbf{x}_i | \text{others} = \mathbf{x}_i - \hat{\mathbf{x}}_i | \mathbf{X}_{(-i)}$ give the part of x_i that cannot be explained by the others, and so are uncorrelated with them.

The AV plot is then just a simple scatterplot of these residuals, \mathbf{y}^* on the vertical axis, and \mathbf{x}^* on the horizontal. In practice, it is unnecessary to run the auxilliary regressions this way (Velleman & Welsh, 1981). Both can be calculated using `stats::lsfit()` roughly as follows:

```
AVcalc <- function(model, variable)
  X <- model.matrix(model)
  response <- model.response(model)
  x <- X[, -variable]
  y <- cbind(X[, variable], response)
  fit <- lsfit(x, y, intercept = FALSE)
  resids <- residuals(fit)
  return(resids)
```

Note that \mathbf{y} here contains both the current predictor, \mathbf{x}_i and the response \mathbf{y} , so the residuals `resids` have two columns, one for $x_i | \text{others}$ and one for $y | \text{others}$.

Added-variable plots are produced using `car::avPlot()` for one predictor or `avPlots()` for any number of model terms. The `id` argument controls which points are identified in the plots; `n=2` labels the two points that are furthest from the mean on the horizontal axis *and* the two with the largest absolute residuals. For instance, in Figure 6.11, observations 5 and 13 are flagged because their conditional \mathbf{x}_i^* values are extreme; observation 17 has a large absolute residual, $\mathbf{y}^* = \text{Heart} | \text{others}$.

```
avPlots(fit.both,
  ellipse = list(levels = 0.68, fill=TRUE, fill.alpha = 0.1),
  pch = 19,
  id = list(n = 2),
  cex.lab = 1.5,
  main = "Added-variable plots for Coffee data")
```

The data ellipses for \mathbf{x}_i^* and \mathbf{y}^* summarize the conditional (or partial) relations of the response to each predictor controlling for all other predictors in each plot. The essential idea is that the data ellipse for $(\mathbf{x}_i^*, \mathbf{y}^*)$ has the identical relation to the estimate \hat{b}_i in a multiple regression as the data ellipse of (\mathbf{x}, \mathbf{y}) has to the slope in a simple regression.

6.4.1 Properties of AV plots

AV plots are particularly interesting and useful for the following noteworthy properties:

Added-variable plots for Coffee data

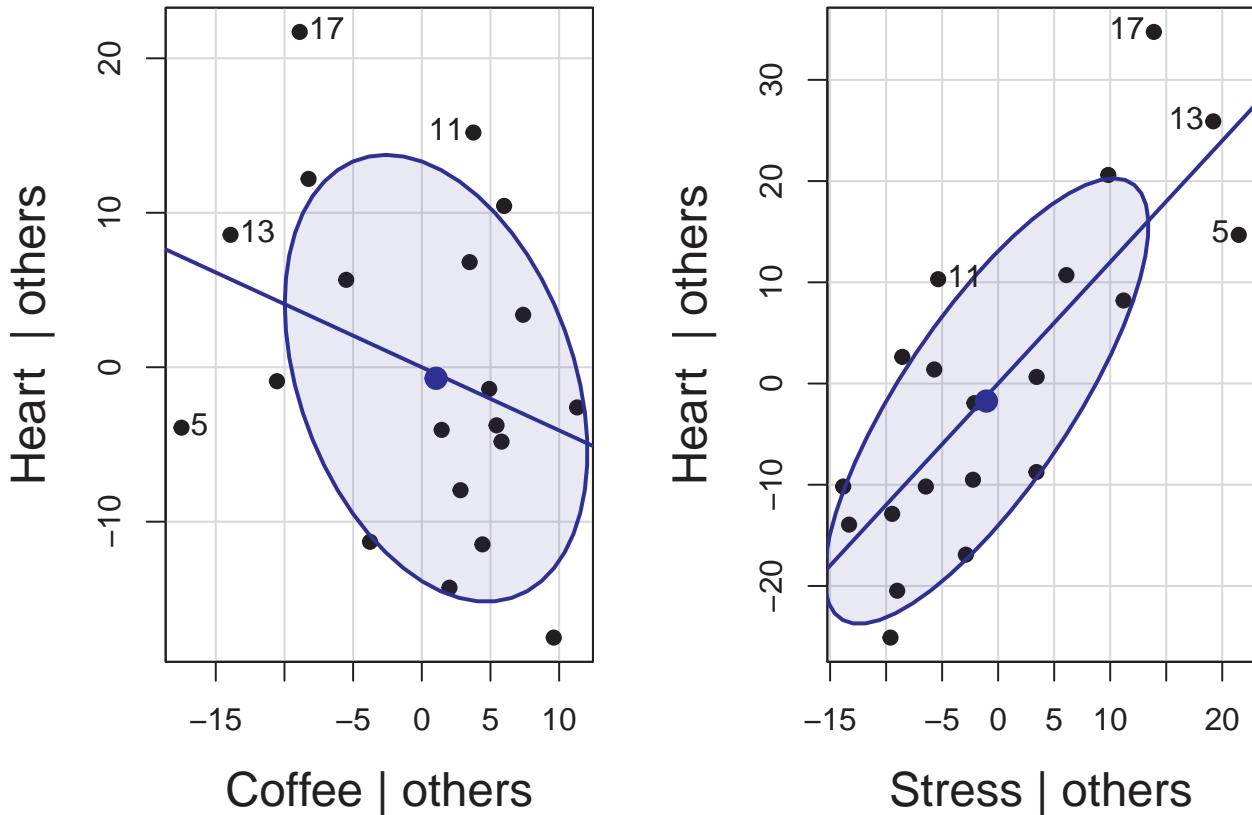


Figure 6.11: Added-variable plots for Coffee and Stress in the multiple regression model

- The slope of the simple regression in the AV plot for variable x_i is identical to the slope b_i for that variable in the full multiple regression model.
- The residuals in this plot are the same as the residuals using all predictors. This means you can see the degree of fit for observations directly in relation to the various predictors, which is not the case for marginal scatterplots.
- Consequentially, the standard deviation of the (vertical) residuals in the AV plot is the same as $s = \sqrt{MSE}$ in the full model and the standard error of a coefficient is $SE(b_i) = s/\sqrt{\sum(\mathbf{x}_i^*)^2}$. This is shown by the size of the shadow of the data ellipses on the vertical axis in Figure 6.11.
- The horizontal positions, \mathbf{x}_i^* , of points adjust for all other predictors, and so we can see points at the extreme left and right as unusual in relation to the others. If these points are also badly fitted (large residuals), we can see their influence on the fitted relation in the full model. AV plots thus provide visual displays of (partial) *leverage* and *influence* on each of the regression coefficients.
- The correlation of \mathbf{x}_i^* and \mathbf{y}^* shown by the shape of the data ellipses is the *partial correlation* between \mathbf{x}_i and \mathbf{y}_i with other predictors partialled out.

6.4.2 Marginal - conditional plots

The relation of the *conditional* data ellipses in AV plots to those in *marginal* plots of the same variables provides further insight into what it means to “control for” other variables. Figure 6.12 shows the same added-variable plots for Heart disease on Stress and Coffee as in Figure 6.11 (with a zoomed-out scaling), but

here we also overlay the marginal data ellipses for $(\mathbf{x}_i, \mathbf{y})$ (centered at the means), and marginal regression lines for Stress and Coffee separately. Drawing arrows connecting the original data points to their positions in the AV plot shows what happens when we condition on or partial out the other variable.

These *marginal - conditional plots* are produced by `car:::mcPlot()` (for one regressor) and `car:::mcPlots()` (for several). The plots for the marginal and conditional relations can be compared separately using the same scales for both, or overlaid as shown here. The points labeled here are only those with large absolute residuals y^* in the vertical direction.

```
mcPlots(fit.both,
  ellipse = list(levels=0.68, fill=TRUE, fill.alpha=0.2),
  id = list(n=2),
  pch = c(16, 16),
  col.marginal = "red", col.conditional = "blue",
  col.arrows = "black",
  cex.lab = 1.5)
```

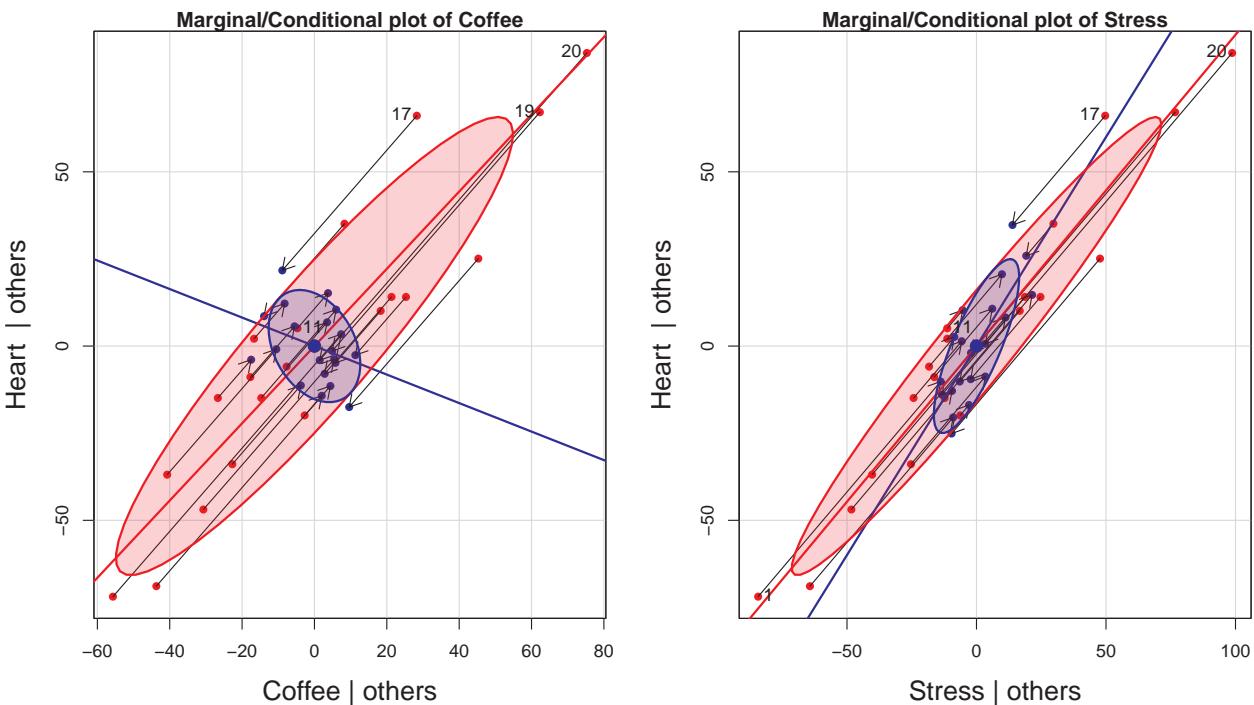


Figure 6.12: Marginal + conditional (added-variable) plots for Coffee and Stress in the multiple regression predicting Heart disease. Each panel shows the 68% conditional data ellipse for x_i^*, y^* residuals (shaded, blue) as well as the marginal 68% data ellipse for the (x_i, y) variables, shifted to the origin. Arrows connect the mean-centered marginal points (red) to the residual points (blue).

The most obvious feature of Figure 6.12 is that Coffee has a negative slope in the conditional AV plot but a positive slope in the marginal plot. This is an example of Simpson's paradox in a regression context: marginal and conditional relations can have opposite signs.

Less obvious is the relation between the marginal and AVP ellipses. In 3D, the marginal data ellipse is the shadow of the ellipsoid for $(\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2)$ on one of the coordinate planes, while the AV plot is a slice through the ellipsoid where either \mathbf{x}_1 or \mathbf{x}_2 is held constant. Thus, the AVP ellipse must be contained in the marginal ellipse, as we can see in Figure 6.12. If there are only two xs , then the AVP ellipse must touch the marginal ellipse at two points.

Finally, Figure 6.12 also shows how conditioning on other predictors works for individual observations, where each point of $(\mathbf{x}_i^*, \mathbf{y}^*)$ is the image of $(\mathbf{x}_i, \mathbf{y})$ along the path of the marginal regression. The variability in the response and in the focal predictor are both reduced, leaving only the uncontaminated relation of \mathbf{y} with \mathbf{x}_i .

These plots are similar in spirit to the ARES plot (“Adding REgressors Smoothly”) proposed by R. D. Cook & Weisberg (1994), but their idea was an interactive animation, displaying a smooth transition between the fit of a marginal model and the fit of a larger model. They used linear interpolation,

$$(\mathbf{x}_i, \mathbf{y}_i)_{\text{interp}} = (\mathbf{x}_i, \mathbf{y}_i) + \lambda[(\mathbf{x}_i^*, \mathbf{y}_i^*) - (\mathbf{x}_i, \mathbf{y}_i)],$$

controlled by a slider whose value, $\lambda \in [0, 1]$, was the weight given to the smaller marginal model. See [this animation](#) for an example using the Duncan data.

6.4.3 Prestige data

For a substantive example, let’s return to the model for income, education and women in the `Prestige` data. The plot in Figure 6.13 shows the strong positive relations of income and education to prestige in the full model, and the negligible relation of percent women. But, in the plot for income, two occupations (physicians and general managers) with high income strongly pull the regression line down from what can be seen in the orientation of the conditional data ellipse.

```
prestige.mod1 <- lm(prestige ~ education + income + women,
                      data=Prestige)

avPlots(prestige.mod1,
        ellipse = list(levels = 0.68),
        id = list(n = 2, cex = 1.2),
        pch = 19,
        cex.lab = 1.5,
        main = "Added-variable plots for prestige")
```

The influential points for physicians and general managers could just be unusual, or suggest that the relation of income to prestige is nonlinear. A rough test of this is to fit a smoothed curve through the points in the AV plot as shown in Figure 6.14.

```
op <- par(mar=c(4, 4, 1, 0) + 0.5)
res <- avPlot(prestige.mod1, "income",
              ellipse = list(levels = 0.68),
              pch = 19,
              cex.lab = 1.5)
smooth <- loess.smooth(res[,1], res[,2])
lines(smooth, col = "red", lwd = 2.5)
```

However, this use of AV plots to diagnose nonlinearity or suggest transformations can be misleading (R. D. Cook, 1996). Curvature in these plots is an indication of *some* model deficiency, but unless the predictors are uncorrelated, they cannot determine the form of a possible transformation of the predictors.

6.4.4 Component + Residual plots

A related method, the *component + residual plot* (“C+R plot”, also called *partial residual plot*, Larsen & McCleary (1972); R. D. Cook (1993)) gives a plot more suited to detecting the need to transform a predictor \mathbf{x}_i to a form $f(\mathbf{x}_i)$ to make it’s relationship with the response \mathbf{y} more nearly linear. This plot displays the partial residual $\mathbf{e} + \hat{b}_i \mathbf{x}_i$ on the vertical axis against \mathbf{x}_i on the horizontal, where \mathbf{e} are the residuals from the full model. A smoothed curve through the points will often suggest the form of the transformation $f()$. The fact that the horizontal axis is \mathbf{x}_i itself rather than \mathbf{x}_i^* makes it easier to see the functional form.

Added-variable plots for prestige

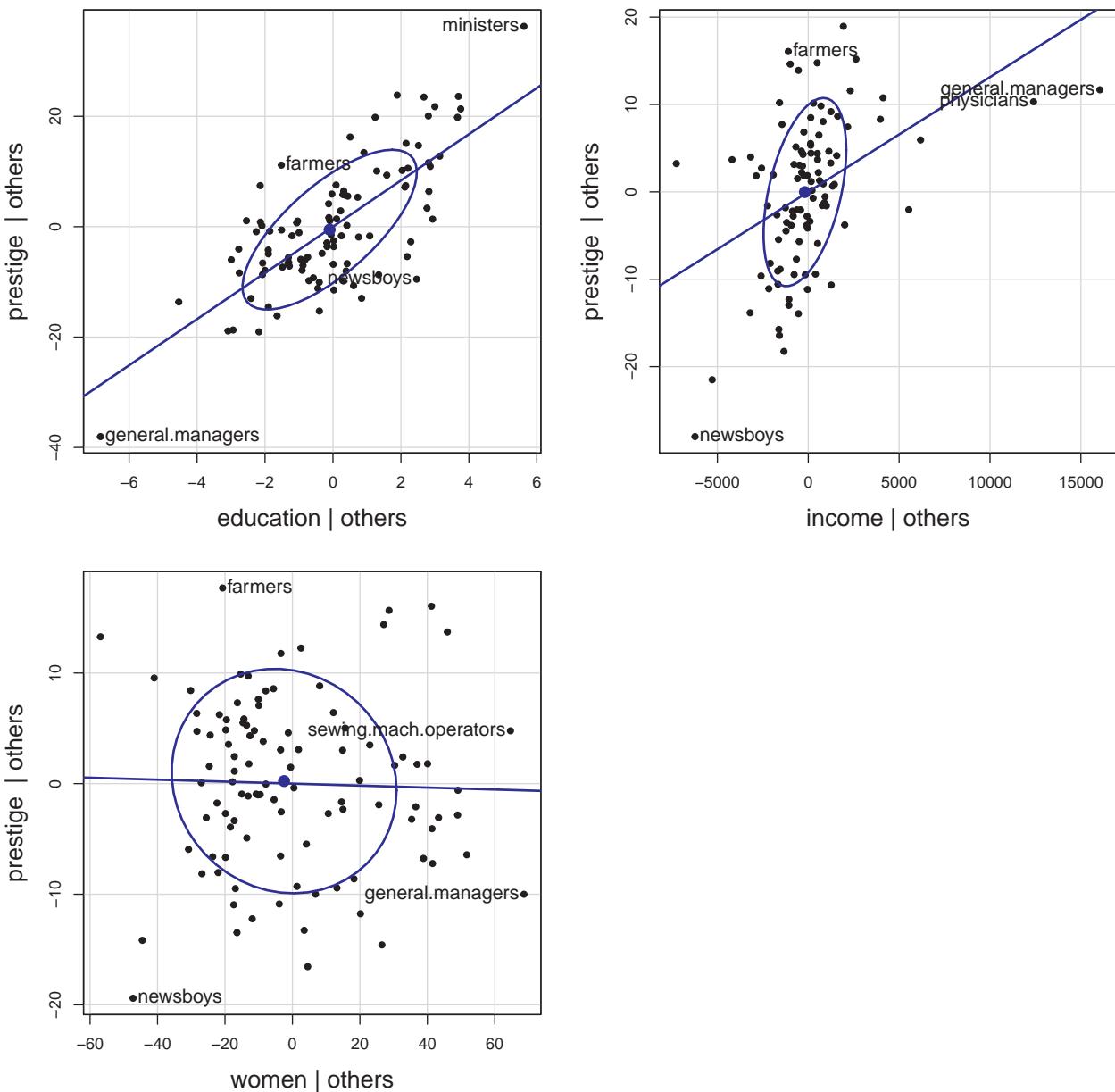


Figure 6.13: Added-variable plot for the quantitative predictors in the `Prestige` data.

The C+R plot has the same desirable properties as the AV plot: The slope \hat{b}_i and residuals e in this plot are the same as those in the full model.

C+R plots are produced by `car::crPlots()` and `car::crPlot()`. Figure 6.15 shows this just for income in the model `prestige.mod1`. (These plots for education and women show no strong evidence of curvilinearity.) The dashed blue line is the linear *partial fit*, $\hat{b}_i \mathbf{x}_i$, whose slope $\hat{b}_2 = 0.0013$ is the same as that for income in `prestige.mod1`. The solid red curve is the loess smooth through the points. The same points are identified as noteworthy as in AV plot in Figure 6.14.

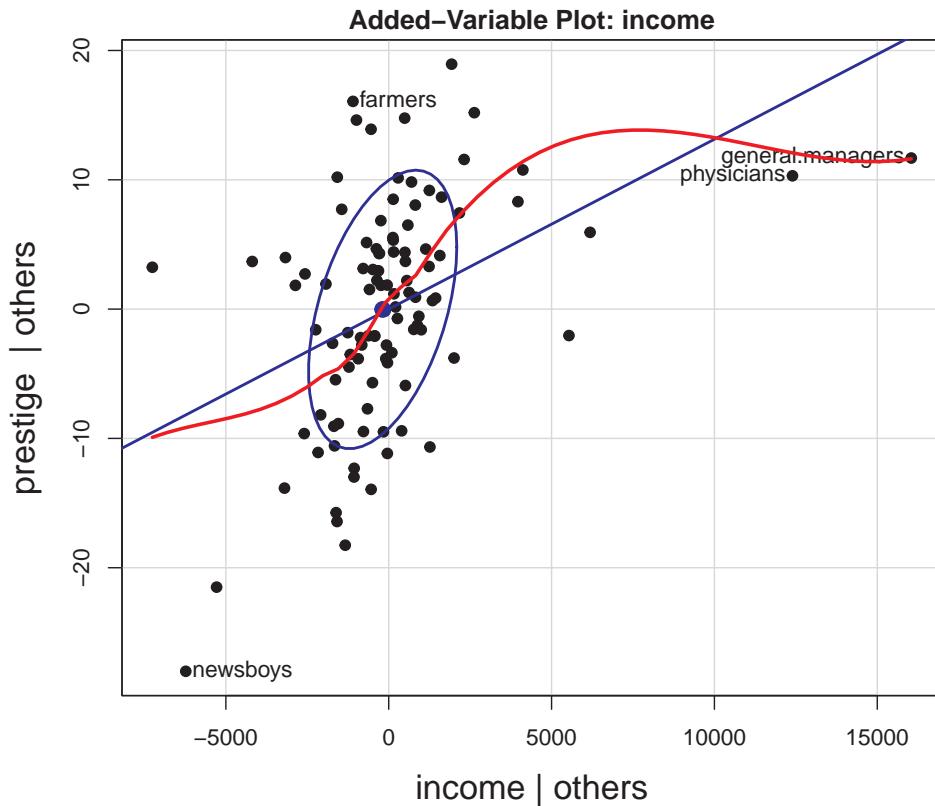


Figure 6.14: Added-variable plot for income, with a loess smooth.

```
crPlot(prestige.mod1, "income",
       smooth = TRUE,
       order = 2,
       pch = 19,
       col.lines = c("blue", "red"),
       id = list(n=2, cex = 1.2),
       cex.lab = 1.5)
```

The partial relation between prestige and income is clearly curved, so it would be appropriate to transform income or to include a polynomial (quadratic) term and refit the model. As suggested earlier (Example 3.2) it makes sense statistically and substantively to model the effect of income on a log scale, so then the slope for `log(income)` would measure the increment in prestige for a constant *percentage increase* in income.

The effect of percent women on prestige seen in Figure 6.13 appears very small and essentially linear. However, if we wished to examine this more closely, we could use the C+R plot in Figure 6.16.

```
crPlot(prestige.mod1, "women",
       pch = 19,
       col.lines = c("blue", "red"),
       id = list(n=2, cex = 1.2),
       cex.lab = 1.5)
```

This shows a slight degree of curvature, with modestly larger values in the extremes. If we wished to test this statistically, we could fit a model with a quadratic effect of women, and compare that to the linear-only effect using `anova()`.

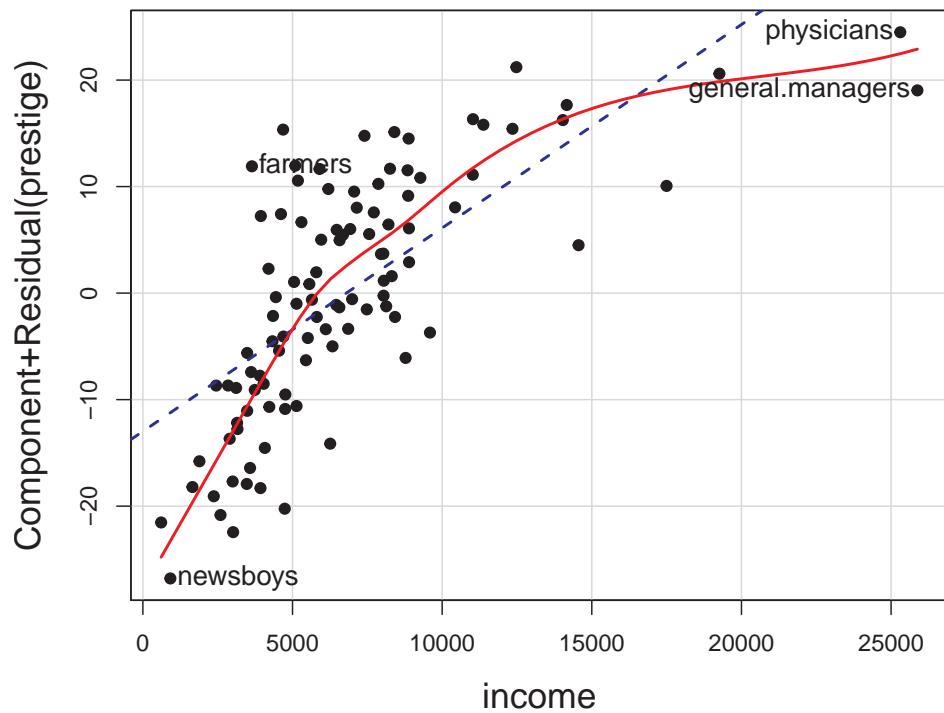


Figure 6.15: Component + residual plot for income in the model for the quantitative predictors of prestige. The dashed blue line is the partial linear fit for income. The solid red curve is the loess smooth.

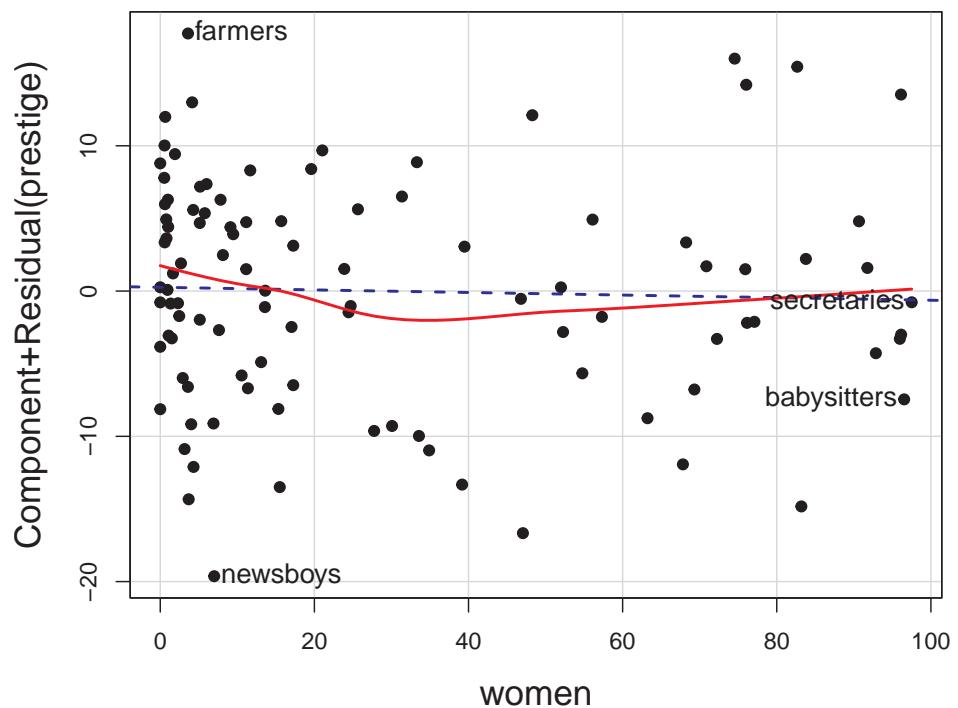


Figure 6.16: Component + residual plot for women in the model for the quantitative predictors of prestige.

```
prestige.mod2 <- lm(prestige ~ education + income + poly(women,2),
```

```

data=Prestige)

anova(prestige.mod1, prestige.mod2)
#> Analysis of Variance Table
#>
#> Model 1: prestige ~ education + income + women
#> Model 2: prestige ~ education + income + poly(women, 2)
#>   Res.Df RSS Df Sum of Sq   F Pr(>F)
#> 1     98 6034
#> 2     97 5907  1      127 2.08  0.15

```

This model ignores the `type` of occupation (“bc”, “wc”, “prof”) as well as any possible interactions of `type` with other predictors. We examine this next, using effect displays.

6.5 Effect displays

For two predictors it is possible, even if awkward, to display the fitted response surface in a 3D plot or faceted 2D views in what I call a *full model plot*. For more than two predictors such displays become cumbersome if not impractical, particularly when there are interactions in the model, when some effects are curvilinear, or when the main substantive interest is focused understanding on one or more main effects or interaction terms in the presence of others. The method of *effect displays*, largely introduced by John Fox (Fox, 1987, 2003; Fox & Weisberg, 2018b) is a generally useful solution to this problem.² These plots are nearly always easier to understand than tables of coefficients.

The idea of effect displays is quite simple, but very general and handles models of arbitrary complexity. Imagine that in a model we have a particular subset of predictors (*focal predictors*) whose effects on the response variable we wish to visualize. The essence of an effect display is that we calculate the predicted values (and standard errors) of the response for the model term(s) involving the focal predictors (and all low-order relatives, e.g, main effects that are marginal to an interaction) as those predictors are allowed to vary over a grid covering their range.

For a given plot, the other, non-focal variables are “controlled” by being fixed at typical values. For example, a quantitative predictor could be fixed at its mean, median or some representative value. A factor could be fixed at equal proportions of its levels or its proportions in the data. The result, when plotted, shows the predicted effects of the focal variables, either with multiple lines or in a faceted display, but with all the other variables controlled, adjusted for or averaged over. For interaction effects all low-order relatives are typically included in the fitted values for the term being graphed.

In practical terms, a scoring matrix \mathbf{X}^* is defined by the focal variables varied over their ranges and the other variables held fixed. The fitted values for a model term are then calculated as $\hat{\mathbf{y}}^* = \mathbf{X}^* \mathbf{b}$ using the equivalent of:

```

predict(model, newdata = X, se.fit = TRUE)

```

which also calculates the standard errors as the square roots of $\text{diag}(\widehat{\text{Var}}(\mathbf{b})\mathbf{X}^{*\top})$ where $\widehat{\text{Var}}(\mathbf{b})$ is the estimated covariance matrix of the coefficients. Consequently, predictor effect values can be obtained for *any* modelling function that has `predict()` and `vcov()` methods. To date, effect displays are available for over 100 different model types in various packages.

²Earlier, but less general expression of these ideas go back to the use of **adjusted means** in analysis of covariance (Fisher, 1925a) or **least squares means** or **population marginal means** in analysis of variance (Searle et al., 1980)

To illustrate the mechanics for the effect of education in the `prestige.mod1` model, construct a data frame varying education, but fixing income and women at their means:

```
X <- expand.grid(
  education = seq(8, 16, 2),
  income = mean(Prestige$income),
  women = mean(Prestige$women)) |>
  print(digits = 3)
#>   education income women
#> 1         8    6798    29
#> 2        10    6798    29
#> 3        12    6798    29
#> 4        14    6798    29
#> 5        16    6798    29
```

`predict()` then gives the fitted values for a simple effect plot of prestige against education. `predict.lm()` returns list, so it is necessary to massage this to a data frame for graphing.

```
pred <- predict(prestige.mod1, newdata=X, se.fit = TRUE)
cbind(X, fit = pred$fit, se = pred$se.fit) |>
  print(digits=3)
#>   education income women  fit     se
#> 1         8    6798    29 35.4 1.318
#> 2        10    6798    29 43.7 0.828
#> 3        12    6798    29 52.1 0.919
#> 4        14    6798    29 60.5 1.487
#> 5        16    6798    29 68.9 2.188
```

As Fox & Weisberg (2018b) note, effect displays can be combined with partial residuals to visualize *both* fit and potential lack of fit simultaneously, by plotting residuals from a model around 2D slices of the fitted response surface. This adds the benefits of C+R plots, in that we can see the impact of unmodeled curvilinearity and interactions in addition to those of predictor effect displays.

There are several implementations of effect displays in R, whose details, terminology and ease of use vary. Among these, `ggeffects` (Lüdecke, 2025) calculates adjusted predicted values under several methods for conditioning. `marginaleffects` (Arel-Bundock, 2025a) is similar and also provides estimation of marginal slopes, contrasts, odds ratios, etc. Both have `plot()` methods based on `ggplot2`. My favorite is the `effects` (Fox et al., 2025) package, which alone provides partial residuals, and is somewhat easier to use, though it uses `lattice` graphics. See the vignette [Predictor Effects Graphics Gallery](#) for details of the computations for effect displays.

The main functions for computing fitted effects are `predictorEffect()` (for one predictor) and `predictorEffects()` (for one or more). For a model `mod` with formula `y ~ x1 + x2 + x3 + x1:x2`, the call to `predictorEffects(mod, ~ x1)` recognizes that an interaction is present and calculates the fitted values for combinations of `x1` and `x2`, holding `x3` fixed at its average value. This returns an object of class "eff" which can be graphed using the `plot.eff()` method.

The effect displays for several predictors can be plotted together, as with `avplots()` (Figure 6.13) by including them in the plot formula, e.g., `predictorEffects(mod, ~ x1 + x3)`. Another function, `allEffects()` calculates the effects for each high-order term in the model, so `allEffects(mod) |> plot()` is handy for getting a visual overview of a fitted model.

6.5.1 Prestige data

To illustrate effect plots, I consider a more complex model, allowing a quadratic effect of women, representing income on a \log_{10} scale, and allowing this to interact with type of occupation. `Anova()` provides the Type II tests of each of the model terms.

```
prestige.mod3 <- lm(prestige ~ education + poly(women, 2) +
                      log10(income)*type, data=Prestige)

# test model terms
Anova(prestige.mod3)
#> Anova Table (Type II tests)
#>
#> Response: prestige
#>             Sum Sq Df F value    Pr(>F)
#> education          994  1 25.87 2.0e-06 ***
#> poly(women, 2)      414  2   5.38 0.00620 **
#> log10(income)     1523  1 39.63 1.1e-08 ***
#> type                589  2   7.66 0.00085 ***
#> log10(income):type  221  2   2.88 0.06133 .
#> Residuals         3420 89
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The fitted coefficients, standard errors and t -tests from `coeftest()` are shown below. The coefficient for education means that an increase of one year of education, holding other predictors fixed, gives an expected increase of 2.96 in prestige. The other coefficients are more difficult to understand. For example, the effect of women is represented by two coefficients for the linear and quadratic components of `poly(women, 2)`. The interpretation of coefficients of terms involving `type` depend on the contrasts used. Here, with the default treatment contrasts, they represent comparisons with `type = "bc"` as the reference level. It is not obvious how to understand the interaction effects like `log10(income):typeprof`.

```
lmtest::coeftest(prestige.mod3)
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -137.500    23.522  -5.85  8.2e-08 ***
#> education      2.959     0.582   5.09  2.0e-06 ***
#> poly(women, 2)1 28.339    10.190   2.78  0.0066 **
#> poly(women, 2)2 12.566     7.095   1.77  0.0800 .
#> log10(income)  40.326     6.714   6.01  4.1e-08 ***
#> typewc          0.969    39.495   0.02  0.9805
#> typeprof        74.276    30.736   2.42  0.0177 *
#> log10(income):typewc -1.073   10.638  -0.10  0.9199
#> log10(income):typeprof -17.725    7.947  -2.23  0.0282 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It is easiest to produce effect displays for all terms in the model using `allEffects()`, accepting all defaults. This gives (Figure 6.17) effect plots for the main effects of education and income and the interaction of income with type, with the non-focal variables held fixed. Each plot shows the fitted regression relation and a default

95% pointwise confidence band using the standard errors. Rug plots at the bottom show the locations of observations for the horizontal focal variable, which is useful when the observations are not otherwise plotted.

```
allEffects(prestige.mod3) |>
  plot()
```

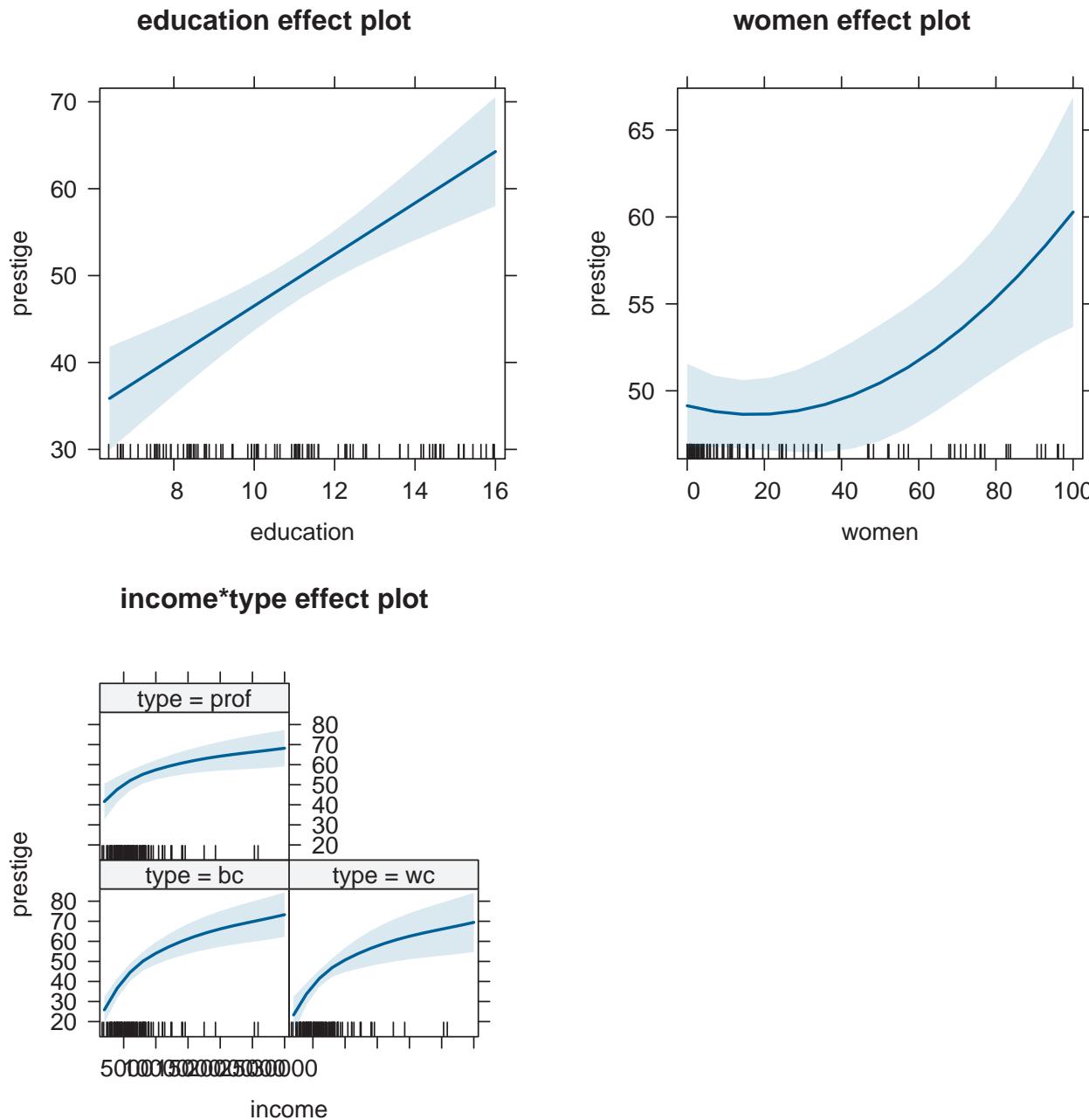


Figure 6.17: Predictor effect plot for all terms in the model with 95% confidence bands.

The effect for women, holding education, income and type constant looks to be quite strong and curved upwards. But note that these plots use different vertical scales for prestige in each plot and the range in the plot for women is much smaller than in the others. The interaction is graphed showing separate curves for the three levels of type.

For a more detailed look, it is useful to make separate plots for the predictors in the model, which allows customizing options for calculation and display. Partial residuals for the observations are computed by using `residuals = TRUE` in the call to `predictorEffects()`. The slope of the fitted line (in blue) is exactly coefficient for education in the full model. As with C+R plots, a smooth loess curve (in red) gives a visual assessment of linearity for a given predictor. A wide variety of graphing options are available in the call to `plot()`. Figure 6.18 shows the effect display for education with partial residuals and point identification of those points with the largest Mahalanobis distances from the centroid.

```
lattice:::trellis.par.set(par.xlab.text=list(cex=1.5),
                          par.ylab.text=list(cex=1.5))

predictorEffects(prestige.mod3, ~ education,
                 residuals = TRUE) |>
  plot(partial.residuals = list(pch = 16, col="blue"),
       id=list(n=4, col="black"))
```

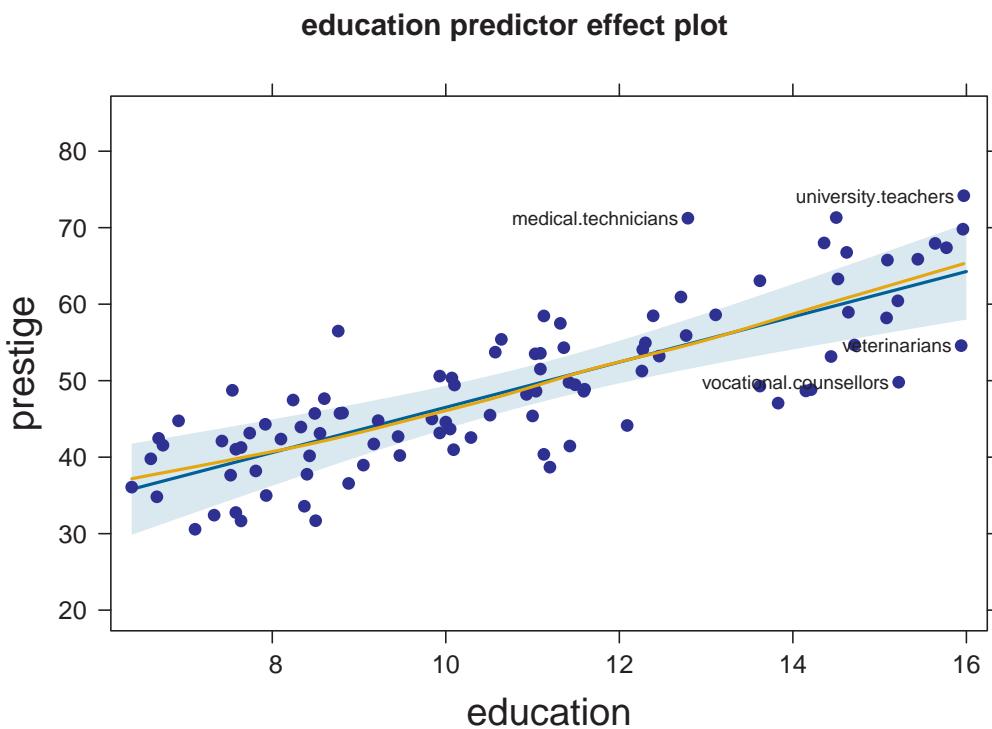


Figure 6.18: Predictor effect plot for education displaying partial residuals. The blue line shows the slice of the fitted regression surface where other variables are held fixed. The red curve shows a loess smooth of the partial residuals.

The effect plot for women in this model is shown in Figure 6.19. This uses the same vertical scale as in Figure 6.18, showing a more modest effect of percent women.

```
predictorEffects(prestige.mod3, ~women,
                 residuals = TRUE) |>
  plot(partial.residuals = list(pch = 16, col="blue", cex=0.8),
       id=list(n=4, col="black"))
```

Because of the interaction with `type`, the fitted effects for `income` are calculated for the three types of

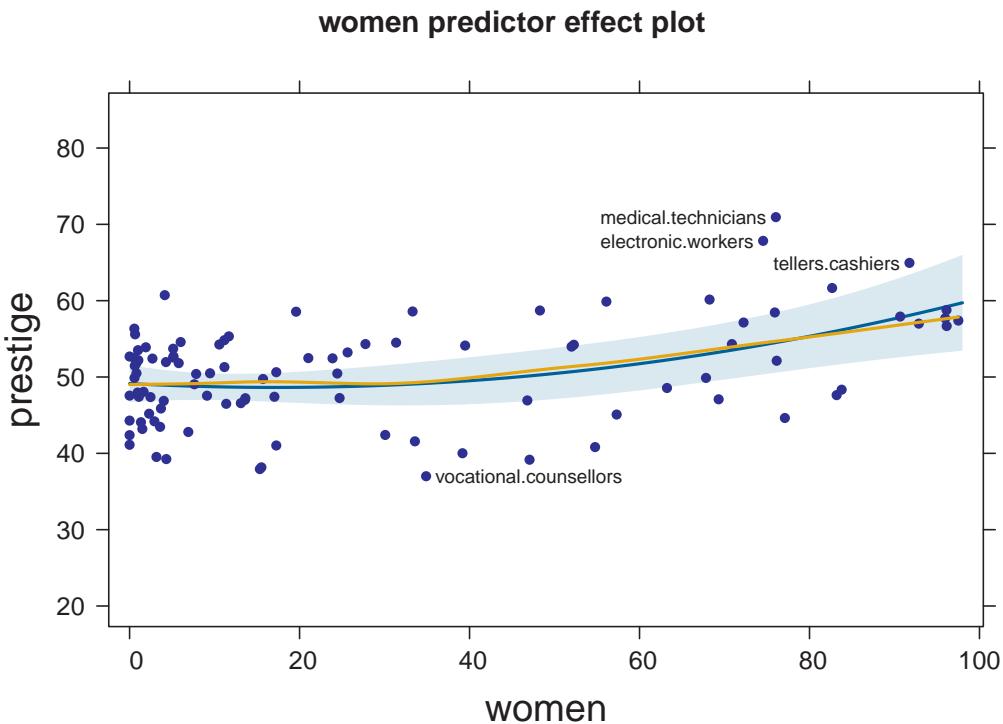


Figure 6.19: Predictor effect plot for women with partial residuals

occupation. It is easiest to compare these in the a single plot (using `multiline = TRUE`), rather than in separate panels as in Figure 6.17. Income is represented as `log10(income)` in the model `prestige.mod3`, and it is also easier to understand the interaction by plotting income on a log scale, using the `axes` argument to specify a transformation of the x axis. I use 68% confidence bands here to make the differences among type more apparent.

```
predictorEffects(prestige.mod3, ~ income,
                 confidence.level = 0.68) |>
  plot(lines=list(multiline=TRUE, lwd=3),
       confint=list(style="bands"),
       axes=list(
         x=list(income=list(transform=list(trans=log, inverse=exp))),
         key.args = list(x=.7, y=.35))
```

Figure 6.20 provides a clear interpretation of the interaction, represented by the coefficients shown above for `log10(income):typewc` and `log10(income):typeprof` in the model. Averaging over three occupation types, prestige increases linearly with log income with a coefficient of 40.33. This means that increasing income by 10% (say) gives an increase of $40.33/10 = 4.033$ in prestige. The slope for professional workers is less steep: the coefficient for `log10(income):typeprof` is -17.725. For these workers compared with blue collar jobs, prestige increases 1.77 less with a 10% increase in income. The difference in slopes for blue collar and white collar jobs is negligible.

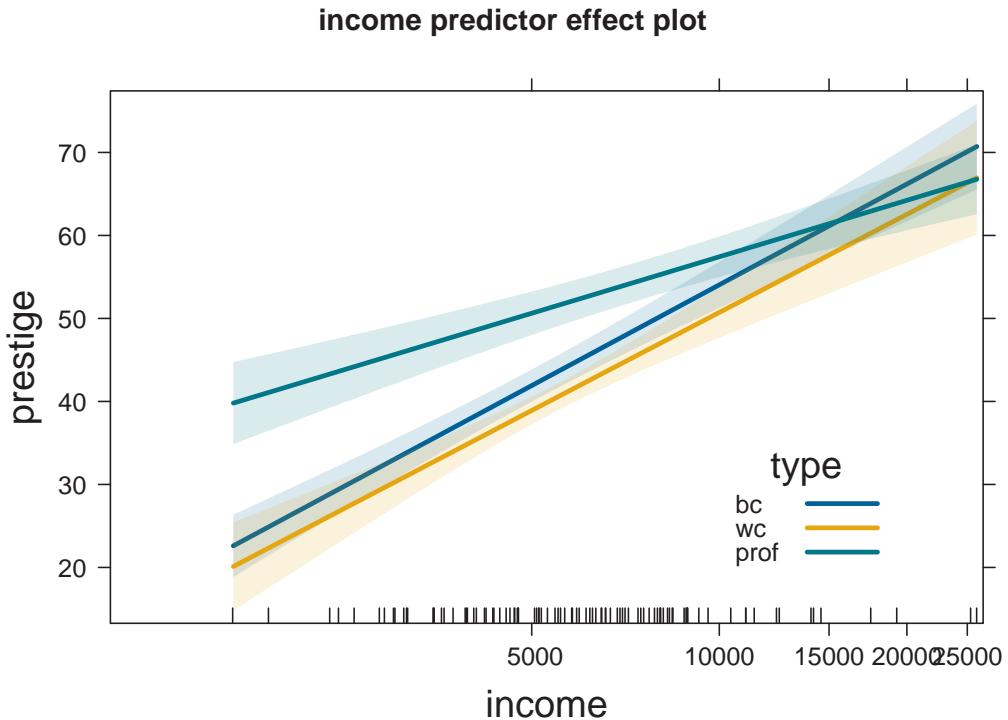


Figure 6.20: Predictor effect plot for income, plotted on a log scale.

6.6 Outliers, leverage and influence

In small to moderate samples, “unusual” observations can have dramatic effects on a fitted regression model, as we saw in the analysis of Davis’s data on reported and measured weight (Section 2.1.2) where one erroneous observation hugely altered the fitted line. As well, it turns out that two observations in Duncan’s data are unusual enough that removing them alters his conclusion that income and education have nearly equal effects on occupational prestige.

An observation can be unusual in three archetypal ways, with different consequences:

- Unusual in the response y , but typical in the predictor(s), \mathbf{x} — a badly fitted case with a large absolute residual, but with x not far from the mean, as in Figure 2.4. This case does not do much harm to the fitted model.
- Unusual in the predictor(s) \mathbf{x} , but typical in y — an otherwise well-fitted point. This case also does little harm, and in fact can be considered to improve precision, a “good leverage” point.
- Unusual in **both** \mathbf{x} and y — This is the case, a “bad leverage” point, revealed in the analysis of Davis’s data, Figure 2.3, where the one erroneous point for women was highly influential, pulling the regression line towards it and affecting the estimated coefficient as well as all the fitted values. In addition, subsets of observations can be *jointly* influential, in that their effects combine, or can mask each other’s influence.

Influential cases are the ones that matter most. As suggested above, to be influential an observation must be unusual in **both** \mathbf{x} and y , and affects the estimated coefficients, thereby also altering the predicted values for all observations. A heuristic formula capturing the relations among leverage, “outlyingness” on y and influence is

$$\text{Influence}_{\text{coefficients}} = X_{\text{leverage}} \times Y_{\text{residual}}$$

As described below, leverage is proportional to the squared distance $(x_i - \bar{x})^2$ of an observation x_i from its mean in simple regression and to the squared Mahalanobis distance in the general case. The Y_{residual} is best measured by a *studentized* residual, obtained by omitting each case i in turn and calculating its residual from the coefficients obtained from the remaining cases.

6.6.1 The leverage-influence quartet

These ideas can be illustrated in the “leverage-influence quartet” by considering a standard simple linear regression for a sample and then adding one additional point reflecting the three situations described above. Below, I generate a sample of $N = 15$ points with x uniformly distributed between (40, 60) and $y \sim 10 + 0.75x + \mathcal{N}(0, 1.25^2)$, duplicated four times.

```
library(tidyverse)
library(car)
set.seed(42)
N <- 15
case_labels <- paste(1:4, c("OK", "Outlier", "Leverage", "Influence"))
levdemo <- tibble(
  case = rep(case_labels,
             each = N),
  x = rep(round(40 + 20 * runif(N), 1), 4),
  y = rep(round(10 + .75 * x + rnorm(N, 0, 1.25), 4)),
  id = " "
)
mod <- lm(y ~ x, data=levdemo)
coef(mod)
#> (Intercept)           x
#>     13.332        0.697
```

The additional points, one for each situation are set to the values below.

- **Outlier:** (52, 60) a low leverage point, but an outlier (0) with a large residual
- **Leverage:** (75, 65) a “good” high leverage point (L) that fits well with the regression line
- **Influence:** (70, 40) a “bad” high leverage point (OL) with a large residual.

```
extra <- tibble(
  case = case_labels,
  x = c(65, 52, 75, 70),
  y = c(NA, 65, 65, 40),
  id = c(" ", "O", "L", "OL")
)

#' Join these to the data
both <- bind_rows(levdemo, extra) |>
  mutate(case = factor(case))
```

We can plot these four situations with `ggplot2` in panels faceted by `case` as shown below. The standard version of this plot shows the regression line for the `original data` and that for the `ammended data` with the additional point. Note that we use the `levdemo` dataset in `geom_smooth()` for the regression line with the original data, but specify `data = both` for that with the additional point.

```
ggplot(levdemo, aes(x = x, y = y)) +
  geom_point(color = "blue", size = 2) +
  geom_smooth(data = both,
              method = "lm", formula = y ~ x, se = FALSE,
              color = "red", linewidth = 1.3, linetype = 1) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE,
              color = "blue", linewidth = 1, linetype = "longdash") +
  stat_ellipse(data = both, level = 0.5, color="blue", type="norm", linewidth = 1.4) +
  geom_point(data=extra, color = "red", size = 4) +
  geom_text(data=extra, aes(label = id), nudge_x = -2, size = 5) +
  facet_wrap(~case, labeller = label_both) +
  theme_bw(base_size = 14)
```

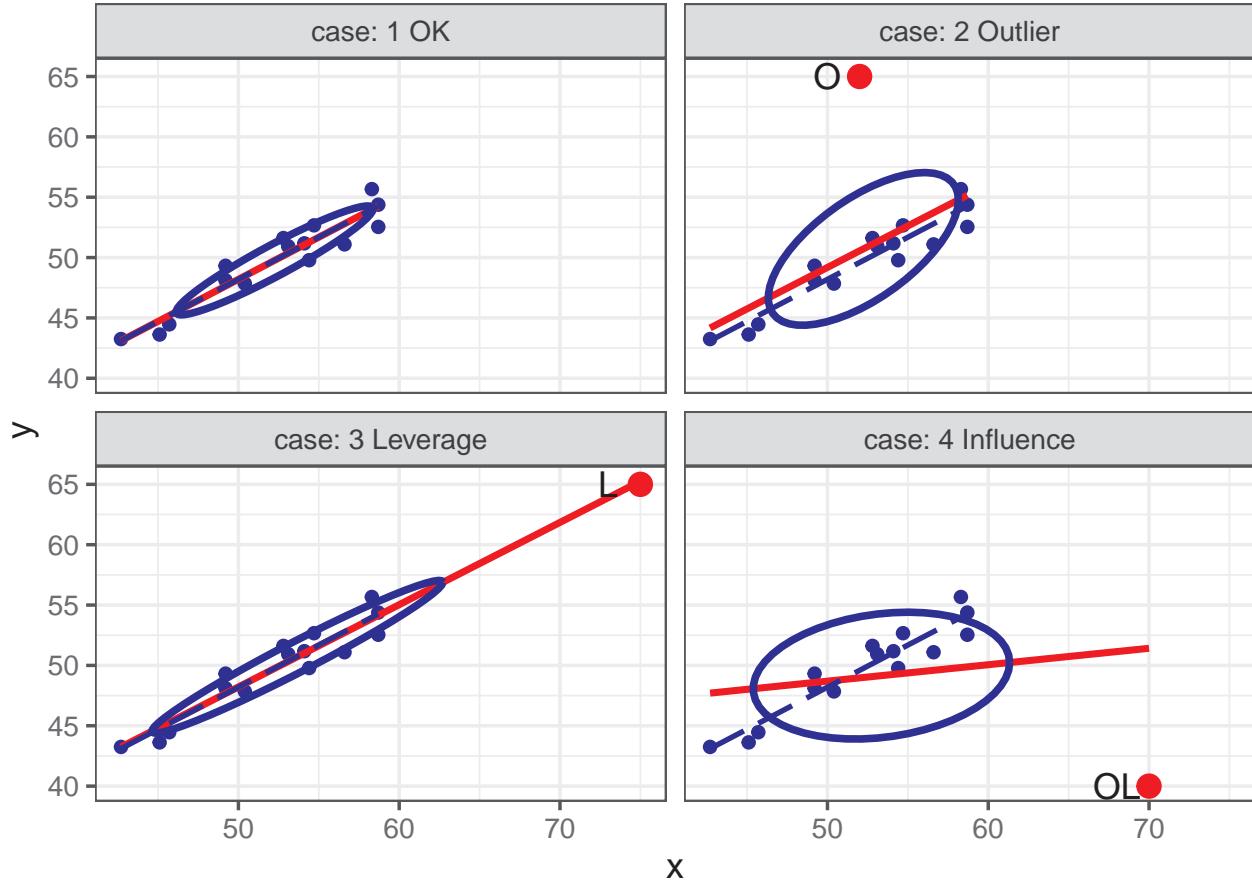


Figure 6.21: Leverage influence quartet with data 50% ellipses. Case (1) original data; (2) adding one low-leverage outlier, “O”; (3) adding one “good” leverage point, “L”; (4) adding one “bad” leverage point, “OL”. The dashed blue line is the fitted line for the original data, while the solid red line reflects the additional point. The data ellipses show the effect of the additional point on precision.

The standard version of this graph shows only the fitted regression lines in each panel. As can be seen, the fitted line doesn’t change very much in panels (2) and (3); only the bad leverage point, “OL” in panel (4) is harmful. Adding data ellipses to each panel immediately makes it clear that there is another part to this story—the effect of the unusual point on *precision* (standard errors) of our estimates of the coefficients.

Now, we see *directly* that there is a big difference in impact between the low-leverage outlier [panel (2)] and the high-leverage, small-residual case [panel (3)], even though their effect on coefficient estimates is negligible.

In panel (2), the single outlier inflates the estimate of residual variance (the size of the vertical slice of the data ellipse at \bar{x}), while in panel (3) this is decreased.

To allow direct comparison and make the added value of the data ellipse more apparent, we overlay the data ellipses from Figure 6.21 in a single graph, shown in Figure 6.22. Here, we can also see why the high-leverage point “L” (added in panel (c) of Figure 6.21) is called a “good leverage” point. By increasing the standard deviation of x , it makes the data ellipse somewhat more elongated, giving increased precision of our estimates of β .

```
colors <- c("black", "blue", "darkgreen", "red")
with(both,
  {dataEllipse(x, y, groups = case,
    levels = 0.68,
    plot.points = FALSE, add = FALSE,
    center.pch = "+",
    col = colors,
    fill = TRUE, fill.alpha = 0.1)
})

case1 <- both |> filter(case == "1 OK")
points(case1[, c("x", "y")], cex=1)

points(extra[, c("x", "y")],
  col = colors,
  pch = 16, cex = 2)

text(extra[, c("x", "y")],
  labels = extra$id,
  col = colors, pos = 2, offset = 0.5)
```

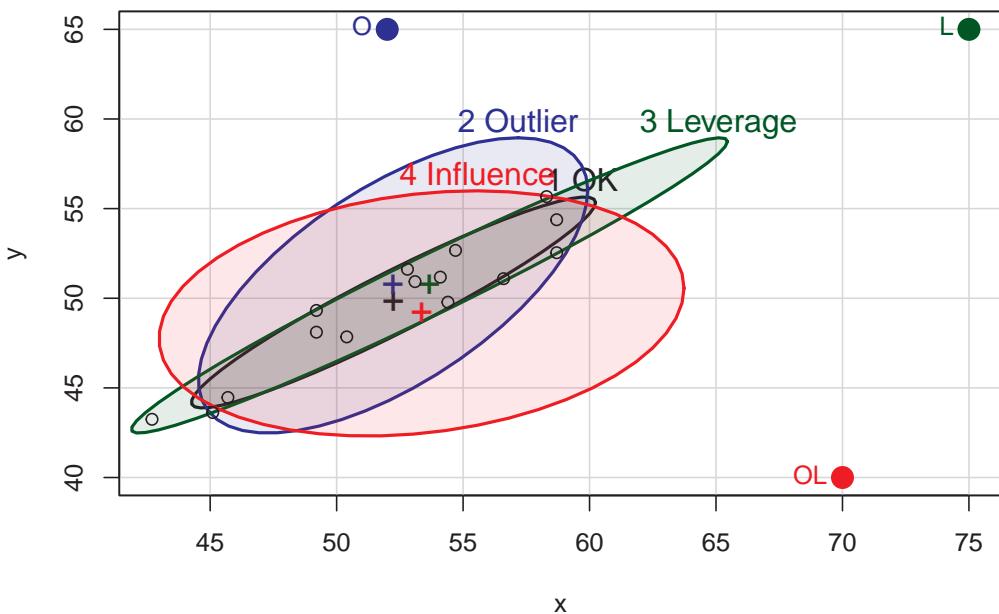


Figure 6.22: Data ellipses in the Leverage-influence quartet. This graph overlays the data ellipses and additional points from the four panels of Figure 6.22. It can be seen that only the OL point affects the slope, while the O and L points affect precision of the estimates in opposite directions.

6.6.1.1 Measuring leverage

Leverage is thus an index of the *potential* impact of an observation on the model due to its' atypical value in the X space of the predictor(s). It is commonly measured by the “hat” value, h_i , so called because it puts the hat ($\widehat{\bullet}$) on \mathbf{y} , i.e., the vector of fitted values can be expressed as

$$\begin{aligned}\hat{\mathbf{y}} &= \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\mathbf{H}} \mathbf{y} \\ &= \mathbf{H} \mathbf{y}.\end{aligned}$$

Here, $h_i \equiv h_{ii}$ are the diagonal elements of the Hat matrix \mathbf{H} . In simple regression, hat values are proportional to the squared distance of the observation x_i from the mean,

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_i (x_i - \bar{x})^2}, \quad (6.1)$$

and range from $1/n$ to 1, with an average value $\bar{h} = 2/n$. Consequently, observations with h_i greater than $2\bar{h}$ or $3\bar{h}$ are commonly considered to be of high leverage.

With $p \geq 2$ predictors, an analogous relationship holds, but the correlations among the predictors must be taken into account. It is demonstrated below that $h_i \propto D^2(\mathbf{x} - \bar{\mathbf{x}})$, the Mahalanobis squared distance of \mathbf{x} from the centroid $\bar{\mathbf{x}}$ ³.

The generalized version of Equation 6.1 is

$$h_i = \frac{1}{n} + \frac{1}{n-1} D^2(\mathbf{x} - \bar{\mathbf{x}}), \quad (6.2)$$

where $D^2(\mathbf{x} - \bar{\mathbf{x}}) = (\mathbf{x} - \bar{\mathbf{x}})^\top \mathbf{S}_X^{-1} (\mathbf{x} - \bar{\mathbf{x}})$. From Section 3.2, it follows that contours of constant leverage correspond to data ellipses or ellipsoids of the predictors in \mathbf{x} , whose boundaries, assuming normality, correspond to quantiles of the χ_p^2 distribution

Example:

To illustrate Equation 6.2, I generate $N = 100$ points from a bivariate normal distribution with means $\mu = (30, 30)$, variances = 10, and a correlation $\rho = 0.7$ and add two noteworthy points that show an apparently paradoxical result.

```
set.seed(421)
N <- 100
r <- 0.7
mu <- c(30, 30)
cov <- matrix(c(10, 10*r,
               10*r, 10), ncol=2)

X <- MASS::mvrnorm(N, mu, cov) |> as.data.frame()
colnames(X) <- c("x1", "x2")

# add 2 points
X <- rbind(X,
            data.frame(x1 = c(28, 38),
                       x2 = c(42, 35)))
```

³See this [Stats StackExchange discussion](#) for a proof.

The Mahalanobis squared distances of these points can be calculated using `heplots::Mahalanobis()`, and their corresponding hatvalues found using `hatvalues()` for any linear model using both `x1` and `x2`.

```
X <- X |>
  mutate(Dsq = heplots::Mahalanobis(X)) |>
  mutate(y = 2*x1 + 3*x2 + rnorm(nrow(X), 0, 5),
         hat = hatvalues(lm(y ~ x1 + x2)))
```

Plotting `x1` and `x2` with data ellipses shows the relation of leverage to squared distance from the mean. The blue point looks to be farther from the mean, but the red point is actually very much further by Mahalanobis squared distance, which takes the correlation into account; it thus has much greater leverage.

```
dataEllipse(X$x1, X$x2,
            levels = c(0.40, 0.68, 0.95),
            fill = TRUE, fill.alpha = 0.05,
            col = "darkgreen",
            xlab = "X1", ylab = "X2")
points(X[1:nrow(X) > N, 1:2], pch = 16,
       col=c("red", "blue"), cex = 2)
X |> slice_tail(n = 2) |>      # last two rows
points(pch = 16, col=c("red", "blue"), cex = 2)
```

The fact that hatvalues are proportional to leverage can be seen by plotting one against the other. I highlight the two noteworthy points in their colors from Figure 6.23 to illustrate how much greater leverage the red point has compared to the blue point.

```
plot(hat ~ Dsq, data = X,
      cex = c(rep(1, N), rep(2, 2)),
      col = c(rep("black", N), "red", "blue"),
      pch = 16,
      ylab = "Hatvalue",
      xlab = "Mahalanobis Dsq")
```

Look back at these two points in Figure 6.23. Can you guess how much further the red point is from the mean than the blue point? You might be surprised that its' D^2 and leverage are about five times as great!

```
X |> slice_tail(n=2)
#>   x1  x2   Dsq    y    hat
#> 1 28  42 25.65 179  0.2638
#> 2 38  35  4.95 175  0.0588
```

6.6.1.2 Outliers: Measuring residuals

From the discussion in Section 6.6, outliers for the response y are those observations for which the residual $e_i = y_i - \hat{y}_i$ are unusually large in magnitude. However, as demonstrated in Figure 6.21, a high-leverage point will pull the fitted line towards it, reducing its' residual and thus making them look less unusual.

The standard approach (R. D. Cook & Weisberg, 1982; Hoaglin & Welsch, 1978) is to consider a *deleted residual* $e_{(-i)}$, conceptually as that obtained by re-fitting the model with observation i omitted and obtaining the fitted value $\hat{y}_{(-i)}$ from the remaining $n - 1$ observations,

$$e_{(-i)} = y_i - \hat{y}_{(-i)}.$$

The (externally) *studentized residual* is then obtained by dividing $e_{(-i)}$ by its estimated standard error,

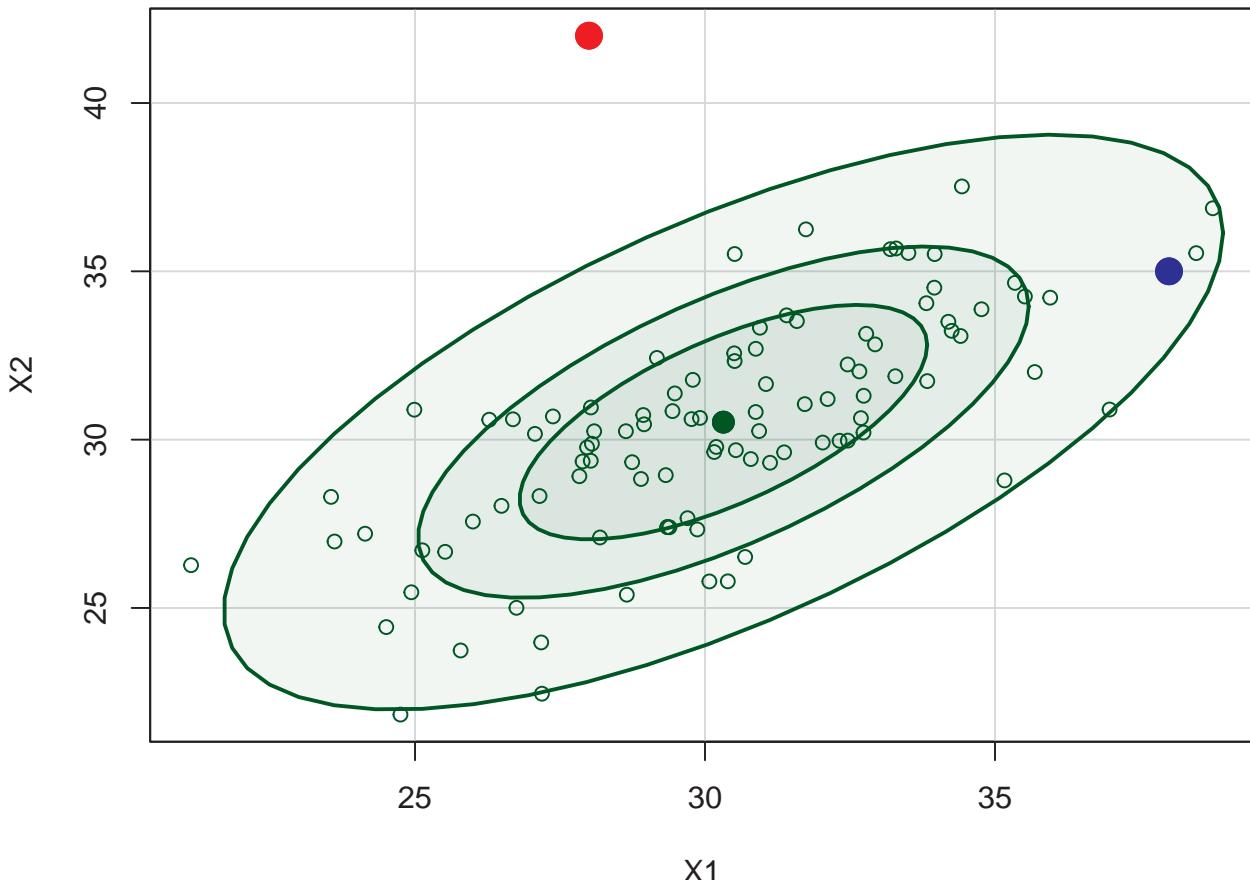


Figure 6.23: Data ellipses for a bivariate normal sample with correlation 0.7, and two additional noteworthy points. The blue point looks to be farther from the mean, but the red point is actually more than 5 times further by Mahalanobis squared distance, and thus has much greater leverage.

giving

$$e_{(-i)}^* = \frac{e_{(-i)}}{\text{sd}(e_{(-i)})} = \frac{e_i}{\sqrt{\text{MSE}_{(-i)} (1 - h_i)}} .$$

This is just the ordinary residual e_i divided by a factor that increases with the residual variance but decreases with leverage. It can be shown that these studentized residuals follow a t distribution with $n - p - 2$ degrees of freedom, so a value $|e_{(-i)}^*| > 2$ can be considered large enough to pay attention to.

In practice for classical linear models, it is unnecessary to actually re-fit the model n times. Velleman & Welsh (1981) show that all these leave-one-out quantities can be calculated from the model fitted to the full data set and the hat (projection) matrix $\mathbf{H} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ from which $\hat{\mathbf{b}} = \mathbf{Hy}$.

6.6.1.3 Measuring influence

As described at the start of this section, the actual influence of a given case depends multiplicatively on its' leverage and residual. But how can we measure it?

The essential idea introduced above, is to delete the observations one at a time, each time refitting the regression model on the remaining $n-1$ observations. Then, for observation i compare the results using all n observations to those with the i^{th} observation deleted to see how much influence the observation has on the analysis.

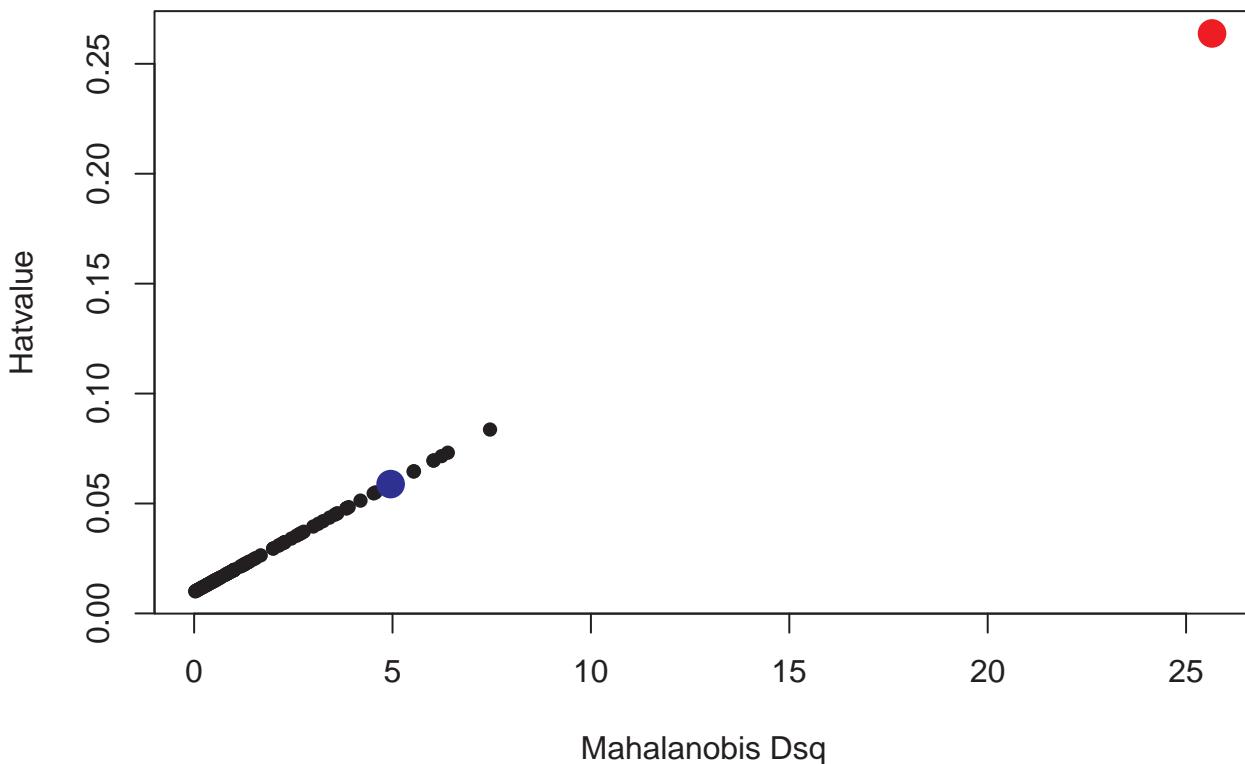


Figure 6.24: Hat values are proportional to squared Mahalanobis distances from the mean.

The simplest such measure, called DFFITS, compares the predicted value for case i with what would be obtained when that observation is excluded.

$$\begin{aligned} \text{DFFITS}_i &= \frac{\hat{y}_i - \hat{y}_{(-i)}}{\sqrt{\text{MSE}_{(-i)} h_i}} \\ &= e_{(-i)}^* \times \sqrt{\frac{h_i}{1 - h_i}} . \end{aligned}$$

The first equation gives the signed difference in fitted values in units of the standard deviation of that difference weighted by leverage; the second version (Belsley et al., 1980) represents that as a product of residual and leverage. A rule of thumb is that an observation is deemed to be influential if $|\text{DFFITS}_i| > 2\sqrt{(p+1)/n}$.

Influence can also be assessed in terms of the change in the estimated coefficients $\mathbf{b} = \hat{\beta}$ versus their values $\mathbf{b}_{(-i)}$ when case i is removed. Cook's distance, D_i , summarizes the size of the difference as a weighted sum of squares of the differences $\mathbf{d} = \mathbf{b} - \mathbf{b}_{(-i)}$ (R. D. Cook, 1977).

$$D_i = \mathbf{d}^\top (\mathbf{X}^\top \mathbf{X}) \mathbf{d} / (p+1)\hat{\sigma}^2$$

This can be re-expressed in terms of the components of residual and leverage

$$D_i = \frac{e_{(-i)}^{*2}}{p+1} \times \frac{h_i}{(1-h_i)} \quad (6.3)$$

Cook's distance is in the metric of an F distribution with p and np degrees of freedom, so values $D_i > 4/n$ are considered large.

6.6.2 Influence plots

The most common plot to detect influence is a bubble plot of the studentized residuals versus hat values, with the size (area) of the plotting symbol proportional to Cook's D . These plots are constructed using `car::influencePlot()` which also fills the bubble symbols with color whose opacity is proportional to Cook's D .

This is shown in Figure 6.25 for the demonstration dataset constructed in Section 6.6.1. In this plot, notable cutoffs for hatvalues at $2\bar{h}$ and $3\bar{h}$ are shown by dashed vertical lines and horizontal cutoffs for studentized residuals are shown at values of ± 2 .

The demonstration data of Section 6.6.1 has four copies of the same (x, y) data, three of which have an unusual observation. The influence plot in Figure 6.25 subsets the data to give the $19 = 15 + 4$ unique observations, including the three unusual cases. As can be seen, the high "Leverage" point has less influence than the point labeled "Influence", which has moderate leverage but a large absolute residual.

```
once <- both[c(1:16, 62, 63, 64),]      # unique observations
once.mod <- lm(y ~ x, data=once)
inf <- influencePlot(once.mod,
                      id = list(cex = 0.01),
                      fill.alpha = 0.5,
                      cex.lab = 1.5)

# custom labels
unusual <- bind_cols(once[17:19,], inf) |>
  print(digits=3)
#> # A tibble: 3 x 7
#>   case       x     y id   StudRes   Hat CookD
#> * <fct>    <dbl> <dbl> <chr>  <dbl> <dbl> <dbl>
#> 1 2 Outlier    52    65  0     3.11  0.0591 0.201
#> 2 3 Leverage    75    65  L     1.52  0.422  0.784
#> 3 4 Influence   70    40  0L    -4.93  0.262  1.82
with(unusual, {
  casetype <- gsub("\d ", "", case)
  text(Hat, StudRes, label = casetype,
       pos = c(4, 2, 3), cex=1.5)
})
```

6.6.3 Duncan data

Let's return to the `Duncan` data used as an example in Section 6.1.1 where a few points stood out as unusual in the basic diagnostic plots (Figure 6.2). The influence plot in Figure 6.26 helps to make sense of these noteworthy observations. The default method for identifying points in `influencePlot()` labels points with any of large studentized residuals, hat-values or Cook's distances.

```
inf <- influencePlot(duncan.mod, id = list(n=3),
                      cex.lab = 1.5)
```

`influencePlot()` returns (invisibly) the influence diagnostics for the cases identified in the plot. It is often useful to look at data values for these cases to understand why each of these was flagged.

```
merge(Duncan, inf, by="row.names", all.x = FALSE) |>
  arrange(desc(CookD)) |>
  print(digits=3)
```

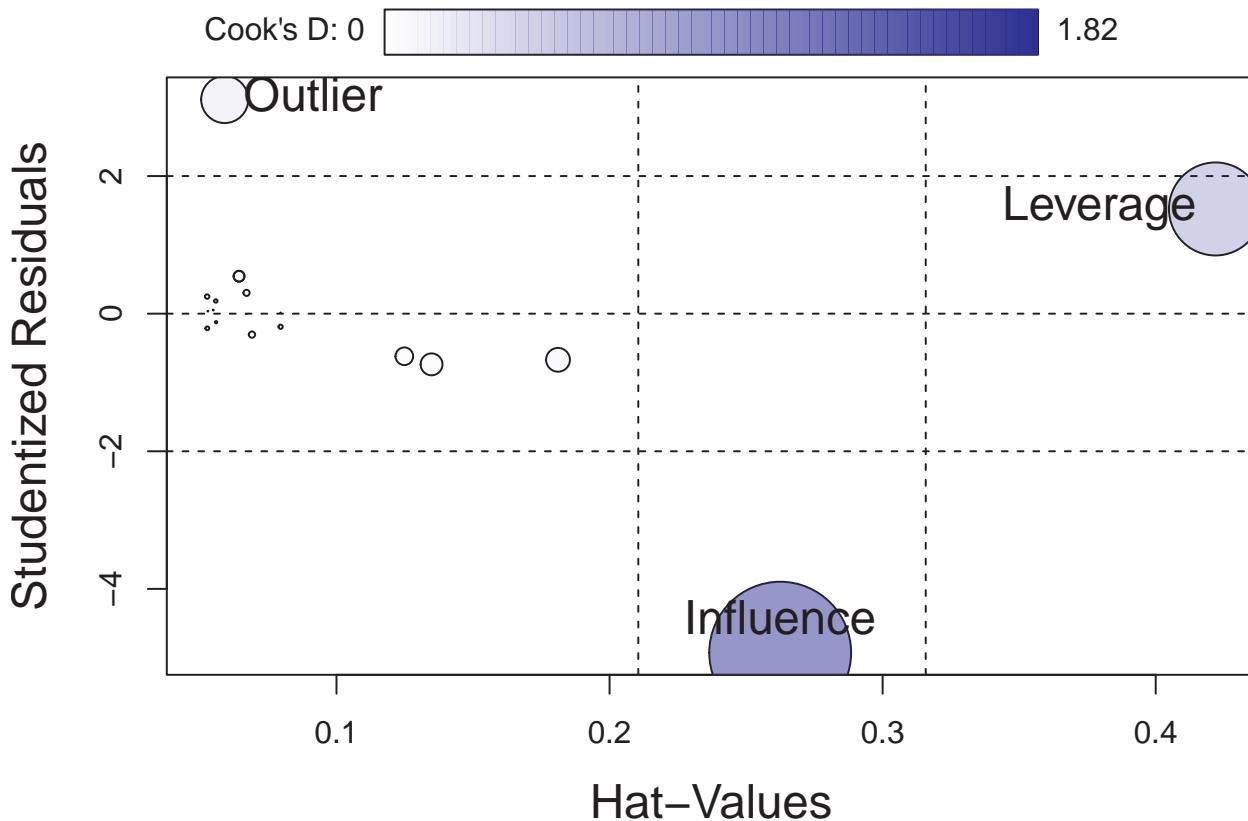


Figure 6.25: Influence plot for the demonstration data. The areas of the bubble symbols are proportional to Cook's D . The impact of the three unusual points on Cook's D is clearly seen.

```
#>      Row.names type income education prestige StudRes     Hat   CookD
#> 1    minister prof    21       84      87  3.135 0.1731 0.5664
#> 2 conductor wc     76       34      38 -1.704 0.1945 0.2236
#> 3 reporter  wc     67       87      52 -2.397 0.0544 0.0990
#> 4 RR.engineer bc     81       28      67  0.809 0.2691 0.0810
#> 5 contractor prof   53       45      76  2.044 0.0433 0.0585
```

- *minister* has by far the largest influence, because it has an extremely positive residual and a large hat value. Looking at the data, we see that ministers have very low income, so their prestige is under-predicted. The large hat value reflects the fact that ministers have low income combined with very high education.
- *conductor* has the next largest Cook's D . It has a large hat value because its combination of relatively high income and low education is unusual in the data.
- Among the others, *reporter* has a relatively large negative residual—its prestige is far less than the model predicts—but its low leverage make it not highly influential. *railroad engineer* has an extremely large hat value because its income is very high in relation to education. But this case is well-predicted and has a small residual, so its leverage is not large.

6.6.4 Influence in added-variable plots

The properties of added-variable plots discussed in Section 6.4 make them also useful for understanding why cases are influential because they control for other predictors in each plot, and therefore show the *partial*

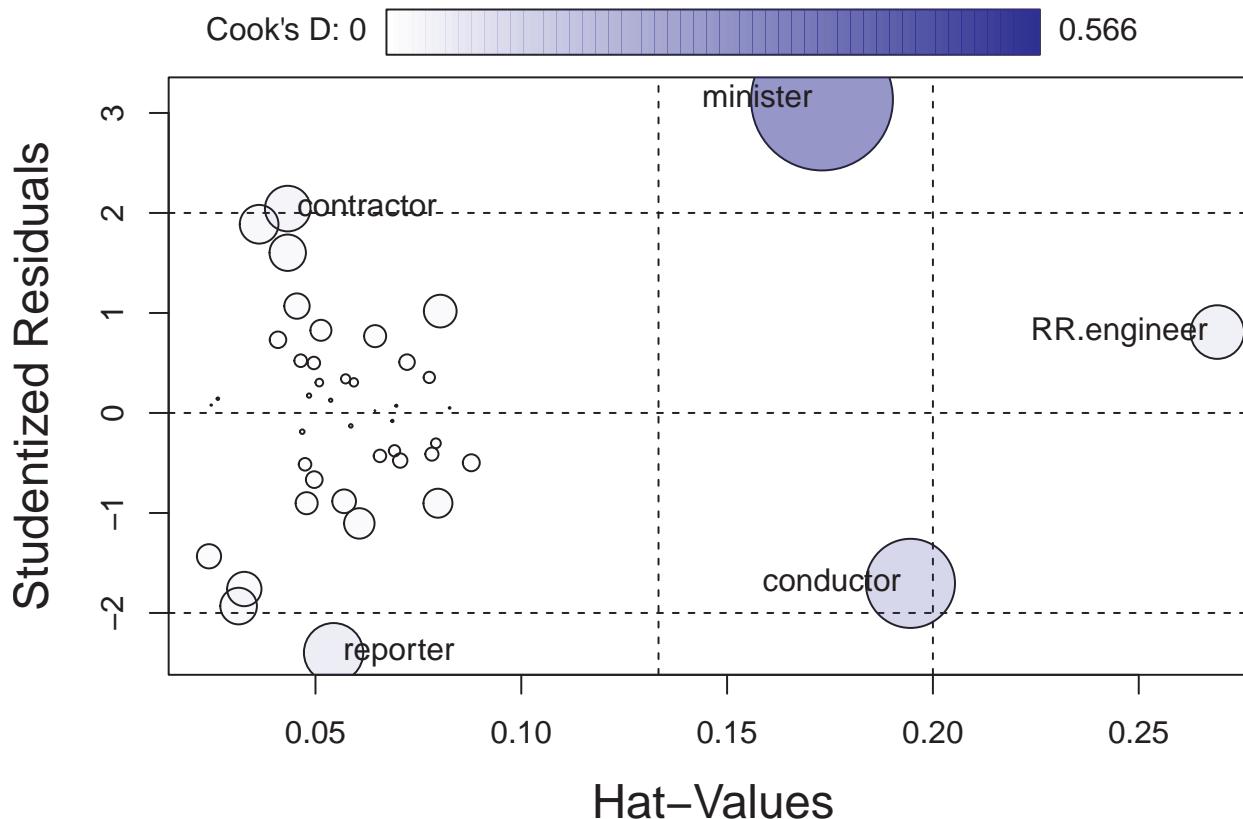


Figure 6.26: Influence plot for the model predicting occupational prestige in Duncan's data. Cases with large studentized residuals, hat-values or Cook's distances are labeled.

contributions of each observation to hat values and residuals. As a consequence, we can see directly the how individual cases become individually or jointly influential.

The Duncan data provides a particularly instructive example of this. Figure 6.27 shows the AV plots for both income and education in the model `duncan.mod`, with some annotations added. I want to focus here on the *joint* influence of the occupations minister and conductor which were seen to be the most influential in Figure 6.26. The green vertical lines show their residuals in each panel and the red lines show the regressions when these two observations are deleted.

The basic AV plots are produced using the call to `avPlots()` below. To avoid clutter, I use the argument `id = list(method = "mahal", n=3)` so that only the three points with the greatest Mahalanobis distances from the centroid in each plot are labeled. These are the cases with the largest leverage seen in Figure 6.26.

```
avPlots(duncan.mod,
  ellipse = list(levels = 0.68, fill = TRUE, fill.alpha = 0.1),
  id = list(method = "mahal", n=3),
  pch = 16, cex = 0.9,
  cex.lab = 1.5)
```

The two cases—minister and conductor—are the most highly influential, but as we can see in Figure 6.27 their influence combines because they are at opposite sides of the horizontal axis and their residuals are of opposite signs. They act together to decrease the slope for income and increase that for education.

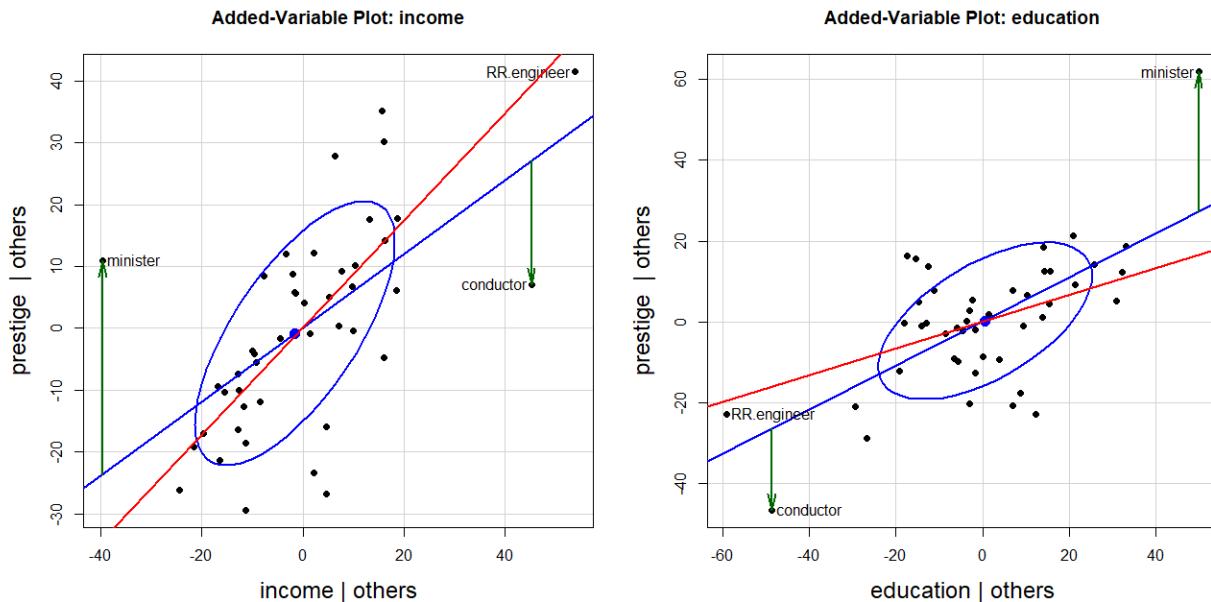


Figure 6.27: Added variable plots for the Duncan model, highlighting the impact of the observations for minister and conductor in each plot. The green lines show the residuals for these observations. The red line in each panel shows the regression line omitting these observations.

```

res <- avPlot(duncan.mod, "income",
               ellipse = list(levels = 0.68),
               id = list(method = "mahal", n=3),
               pch = 16,
               cex.lab = 1.5) |>
  as.data.frame()
fit <- lm(prestige ~ income, data = res)
info <- cbind(res, fitted = fitted(fit),
              resid = residuals(fit),
              hat = hatvalues(fit),
              cookd = cooks.distance(fit))

# noteworthy points in this plot
big <- which(info$cookd > .20)
with(info, {
  arrows(income[big], fitted[big], income[big], prestige[big],
         angle = 12, length = .18, lwd = 2, col = "darkgreen")
})

# line w/o the unusual points
duncan.mod2 <- update(duncan.mod, subset = -c(6, 16))
bs <- coef(duncan.mod2)["income"]
abline(a=0, b=bs, col = "red", lwd=2)

```

Duncan's hypothesis that the slopes for income and education were equal thus fails when these two observations are deleted. The slope for income then becomes 2.6 times that of education.

```
duncan.mod2 <- update(duncan.mod, subset = -c(6, 16))
coef(duncan.mod2)
#> (Intercept)      income   education
#>     -6.409       0.867      0.332
```

6.7 What have we learned?

This chapter unveils the sophisticated visual toolkit that transforms regression models from black boxes into transparent interpretable analyses. Here are the essential insights that will revolutionize how you understand and communicate model results:

- **The regression quartet reveals model reality beyond the summary statistics:** The four diagnostic plots (residuals vs. fitted, Q-Q plot, scale-location, and leverage plots) form a comprehensive health check for your regression models. Like a medical examination, each plot diagnoses different potential problems—linearity violations, non-normality, heteroscedasticity, and influential observations. The enhanced versions in R packages like `car` and `performance` don't just show problems; they label the troublemakers and guide you toward solutions.
- **Coefficient plots make model comparisons effortless and honest:** Raw coefficient tables hide the forest for the trees, especially when predictors have wildly different scales (years vs. dollars vs. percentages). Standardized coefficient plots and meaningful rescaling transform incomprehensible tables into intuitive visual comparisons. When Duncan's income coefficient (0.0013) appears invisible next to education's coefficient (4.19), the plot reveals they're actually comparable forces shaping occupational prestige.
- **Added-variable plots expose the truth behind “controlling for other variables”:** These magical plots perform visual surgery, removing the effects of all other predictors to show the pure relationship between response and focal predictor. They reveal Simpson's paradox in action—how coffee can appear harmful marginally but beneficial conditionally when stress is controlled. The relationship between marginal and conditional ellipses tells the complete story of confounding and adjustment.
- **Effect displays translate complex models into understandable stories:** When interactions, transformations, and multiple predictors make coefficient interpretation impossible, effect displays come to the rescue. They show exactly how prestige changes with income for different occupation types, or how the effects of education varies across meaningful ranges, controlling for other variables. These plots transform “significant at $p < 0.05$ ” into “here's exactly what this means in practice.”
- **Influence diagnostics separate the signal from the statistical noise:** The leverage-influence framework reveals which observations affect your model the most versus which are merely unusual. Through Cook's distance, hat values, and studentized residuals, we learned that `minister` and `conductor` in Duncan's data weren't just outliers—they were the puppet masters controlling his key findings about the equal effects of income and education. Remove them, and the entire conclusion changes.

The chapter's profound message is that regression modeling without visualization is like navigating without a compass. In our R-powered world of complex models and large datasets, sophisticated plotting techniques aren't just helpful—they're essential for understanding what your models actually say about the world.

Topics in Linear Models

The geometric and graphical approach of earlier chapters has already introduced some new ideas for thinking about multivariate data, models for explaining them, and graphical methods for understanding their results. These can be applied to better understand common problems that arise in data analysis.

Packages

In this chapter I use the following packages. Load them now:

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(patchwork)
library(car)
library(matlib)
library(forcats)
library(gganimate)
```

7.1 Ellipsoids in data space and β space

It is most common to look at data and fitted models in “data space,” where axes correspond to variables, points represent observations, and fitted models are plotted as lines (or planes) in this space. As we’ve suggested, data ellipsoids provide informative summaries of relationships in data space. For linear models, particularly regression models with quantitative predictors, there is another space—“ β space”—that provides deeper views of models and the relationships among them. This discussion extends Friendly et al. (2013), Sec. 4.6.

In β space, the axes pertain to coefficients, for example (β_0, β_1) in a simple linear regression. Points in this space are models (true, hypothesized, fitted) whose coordinates represent values of these parameters. For example, one point $\hat{\beta}_{OLS} = (\hat{\beta}_0, \hat{\beta}_1)$ represents the least squares estimate; other points, $\hat{\beta}_{WLS}$ and $\hat{\beta}_{ML}$ would give weighted least squares and maximum likelihood estimates, and the line $\beta_1 = 0$ represents the null hypothesis that the slope is zero.

In the sense described below, data space and β space are each *dual* to the other. In simple linear regression, for example:

- each line, like $\mathbf{y} = \beta_0 + \beta_1 \mathbf{x}$ with intercept β_0 and slope β_1 in data space corresponds to a point (β_0, β_1) in β space, and conversely;
- the set of points on any line $\beta_1 = x + y\beta_0$ in β space corresponds to a set of lines through a given point (x, y) in data space, and conversely;
- the geometric proposition that every pair of points defines a line in one space corresponds to the proposition that every two lines intersect in a point in the other space.

This is illustrated in Figure 7.1. The left panel shows three lines in data space, which can be expressed as

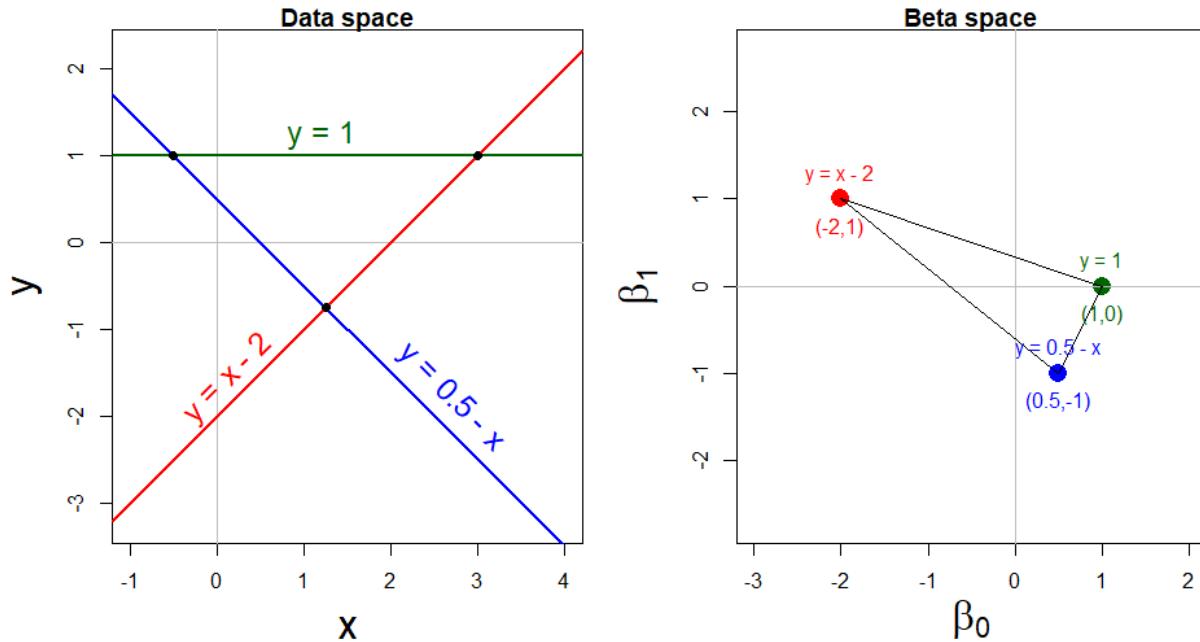


Figure 7.1: Duality of (x, y) lines in data space (left) and points in β -space (right). Each line in data space corresponds to a point, whose intercept and slope are shown in β -space.

linear equations in $\mathbf{z} = (x, y)$ of the form $\mathbf{A}\mathbf{z} = \mathbf{d}$. `matplotlib::showEqn(A, d)` prints these as equations in x and y .

```
A <- matrix(c( 1, 1, 0,
             -1, 1, 1), 3, 2)
d <- c(2, 1/2, 1)
showEqn(A, d, vars = c("x", "y"), simplify = TRUE)
#>   x - 1*y = 2
#>   x + y = 0.5
#> 0*x + y = 1
```

The first equation, $x - y = 2$ can be expressed as the line $y = x - 2$ and corresponds to the point $(\beta_0, \beta_1) = (-2, 1)$ in β space, and similarly for the other two equations. The second equation, $x + y = \frac{1}{2}$, or $y = 0.5 - x$ intersects the first at the point $(x, y) = (1.25, 0.75)$; this corresponds to the line connecting $(-2, 1)$ and $(0.5, -1)$ in β space.

This lovely duality is an example of an [important principle](#) in modern mathematics which translates concepts and structures from one perspective to another and back again. We get two views of the same thing, whose dual nature provides greater insight.

We have seen (Section 3.2) how ellipsoids in data space summarize variance (lack of precision) and correlation of our data. For the purpose of understanding linear models, ellipsoids in β space do the same thing for the estimates of parameters. These ellipsoids are dual and inversely related to each other, a point first made clear by Dempster (1969, Ch. 6):

- In data space, joint confidence intervals for the mean vector or joint prediction regions for the data are given by the ellipsoids $(\bar{x}_1, \bar{x}_2)^T \oplus c\sqrt{\mathbf{S}_{\mathbf{X}}}$, where the covariance matrix $\mathbf{S}_{\mathbf{X}}$ depends on $\mathbf{X}^T \mathbf{X}$ (\oplus here shifts the ellipsoid to one centered at (\bar{x}_1, \bar{x}_2) here, as in Equation 3.2).

- In the dual β space, joint confidence regions for the coefficients of a response variable y on (x_1, x_2) are given by ellipsoids of the form $\hat{\beta} \oplus c\sqrt{\mathbf{S}_\mathbf{X}^{-1}}$, and depend on $(\mathbf{X}^\top \mathbf{X})^{-1}$.

It is useful to understand the underlying geometry here connecting the ellipses for a matrix and its inverse. This can be seen in Figure 7.2, which shows an ellipse for a covariance matrix \mathbf{S} , whose axes, as we saw in Chapter 4 are the eigenvectors \mathbf{v}_i of \mathbf{S} and whose radii are the square roots $\sqrt{\lambda_i}$ of the corresponding eigenvalues. The comparable ellipse for $2\mathbf{S}$ has radii multiplied by $\sqrt{2}$.

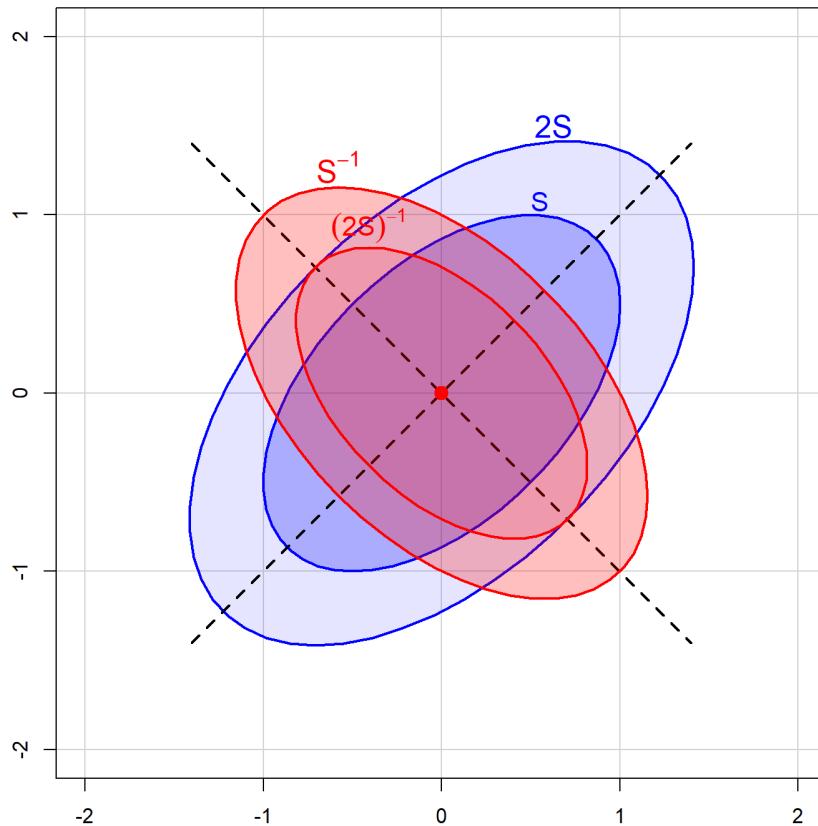


Figure 7.2: Geometric properties of an ellipse \mathbf{S} and its inverse, \mathbf{S}^{-1} . The principal axes (dotted lines) are given by the eigenvectors, which are the same for \mathbf{S} and \mathbf{S}^{-1} . Multiplying \mathbf{S} by 2 makes it's ellipse larger by $\sqrt{2}$, while the same factor makes the ellipse for $(2\mathbf{S})^{-1}$ smaller by the same factor.

As long as \mathbf{S} is of full rank, the eigenvectors of \mathbf{S}^{-1} are identical, while the eigenvalues are $1/\lambda_i$, so the radii are the reciprocals $1/\sqrt{\lambda_i}$. The analogous ellipse for $(2\mathbf{S}^{-1})$ is smaller by a factor of $\sqrt{2}$.

Thus, in two dimensions, the ellipse for \mathbf{S}^{-1} is a 90° rotation of that for \mathbf{S} . It is small in directions where the ellipse for \mathbf{S} is large, and vice-versa. In our statistical applications, this translates as: parameter estimates in β space are more precise (have less variance) in the directions where the data are more widely dispersed, having more information about the relationship.

We illustrate these ideas in the example below.

7.1.1 Coffee, stress and heart disease

Consider the dataset `coffee`, giving measures of `Heart` (y), an index of cardiac damage, `Coffee` (x_1), a measure of daily coffee consumption, and `Stress` (x_2), a measure of occupational stress, in a contrived sample

of $n = 20$ university people.¹ For the sake of the example we assume that the main goal is to determine whether or not coffee is good or bad for your heart, and stress represents one potential confounding variable among others (age, smoking, etc.) that might be useful to control statistically.

```
set.seed(1234)
data(coffee, package="matlib")
coffee |> dplyr::sample_n(6)
#>      Group Coffee Stress Heart
#> 1 Grad_Student    104    117    92
#> 2     Student      52     86    63
#> 3 Grad_Student    76     92    58
#> 4     Student     100    123    92
#> 5     Student      64     74    63
#> 6 Professor      141    175   145
```

Figure 7.3 shows the scatterplot matrix, giving the marginal relations between all pairs of variables. The marginal message seems to be that coffee is bad for your heart, stress is bad for your heart and coffee consumption is also related to occupational stress.

```
scatterplotMatrix(~ Heart + Coffee + Stress, data=coffee,
  smooth = FALSE,
  pch = 16, col = "brown",
  cex.axis = 1.3, cex.labels = 3,
  ellipse = list(levels = 0.68, fill.alpha = 0.1))
```

Yet, when we fit both variables together, we obtain the following results, suggesting that coffee is good for you—the coefficient for coffee is now negative, though non-significant. How can this be?

```
coffee.mod <- lm(Heart ~ Coffee + Stress, data=coffee)
broom::tidy(coffee.mod)
#> # A tibble: 3 x 5
#>   term       estimate std.error statistic  p.value
#>   <chr>     <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept) -7.79      5.79     -1.35  0.196
#> 2 Coffee      -0.409     0.292     -1.40  0.179
#> 3 Stress       1.20      0.224      5.34  0.0000536
```

The answer is that the marginal plots of `Heart` vs. `Coffee` and `Stress` in the first row of Figure 7.3 each ignore the other predictor. In contrast, the coefficients for coffee and stress in the multiple regression model `coffee.mod` are *partial* coefficients, giving the estimated change in heart damage for a unit change in each predictor, but **adjusting for** (controlling for, or holding constant) the other predictor.

We can see these effects directly in **added variable plots** (Section 6.4), but here I consider the relationship of coffee and stress in data space and beta space and how their ellipses relate to each other and to hypothesis tests.

The left panel in Figure 7.4 is the same as that in the (3,2) cell of Figure 7.3 for the relation `Stress ~ Coffee` but with data ellipses of 40% and 60% coverage. The shadows of the 40% ellipse on any axis give univariate intervals of the mean $\bar{x} \pm 1s_x$ (standard deviation) shown by the thick red lines; the shadow of the 68% ellipse corresponds to an interval $\bar{x} \pm 1.5s_x$.

The right panel shows the joint 95% confidence region for the coefficients $(\beta_{\text{Coffee}}, \beta_{\text{Stress}})$ and individual confidence intervals in β space. These are determined as

¹This example was developed by Georges Monette.

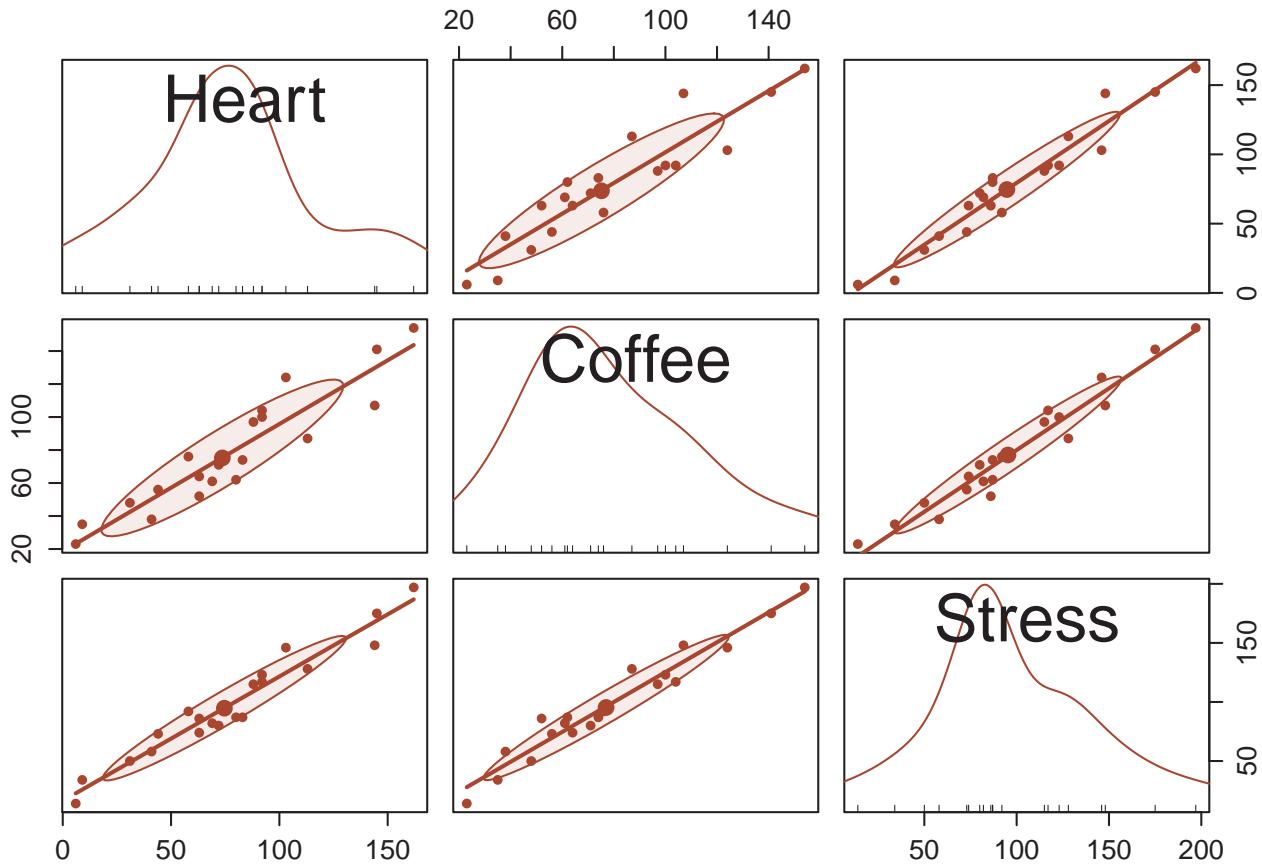


Figure 7.3: Scatterplot matrix for the `coffee` data showing the pairwise relationships among Heart damage (y), Coffee consumption (x_1), and Stress (x_2), with linear regression lines and 68% data ellipses.

$$\hat{\beta} \oplus \sqrt{dF_{d,\nu}^{.95}} \times s_e \times \mathbf{S}_X^{-1/2} .$$

where d is the number of dimensions for which we want coverage, ν is the residual degrees of freedom for s_e , and \mathbf{S}_X is the covariance matrix of the predictors.

Thus, the blue ellipse in Figure 7.4 (right) is the ellipse of **joint** 95% coverage, using the factor $\sqrt{2F_{2,\nu}^{.95}}$, which covers the true values of $(\beta_{\text{Stress}}, \beta_{\text{Coffee}})$ in 95% of samples. Moreover:

- Any *joint* hypothesis (e.g., $\mathcal{H}_0 : \beta_{\text{Stress}} = 0, \beta_{\text{Coffee}} = 0$) can be tested visually, simply by observing whether the hypothesized point, $(0, 0)$ here, lies inside or outside the joint confidence ellipse. That hypothesis is rejected
- The shadows of this ellipse on the horizontal and vertical axes give Scheff'e joint 95% confidence intervals for the parameters, with protection for simultaneous inference ("fishing") in a 2-dimensional space.
- Similarly, using the factor $\sqrt{F_{1,\nu}^{1-\alpha/d}} = t_{\nu}^{1-\alpha/2d}$ would give an ellipse whose 1D shadows are $1 - \alpha$ Bonferroni confidence intervals for d posterior hypotheses.

Visual hypothesis tests and $d = 1$ confidence intervals for the parameters *separately* are obtained from the red ellipse in Figure 7.4, which is scaled by $\sqrt{F_{1,\nu}^{.95}} = t_{\nu}^{.975}$. We call this the *confidence-interval generating ellipse* (or, more compactly, the "confidence-interval ellipse"). The shadows of the confidence-interval ellipse on the axes (thick red lines) give the corresponding individual 95% confidence intervals, which are equivalent to the (partial, Type III) t -tests for each coefficient given in the standard multiple regression output shown above.

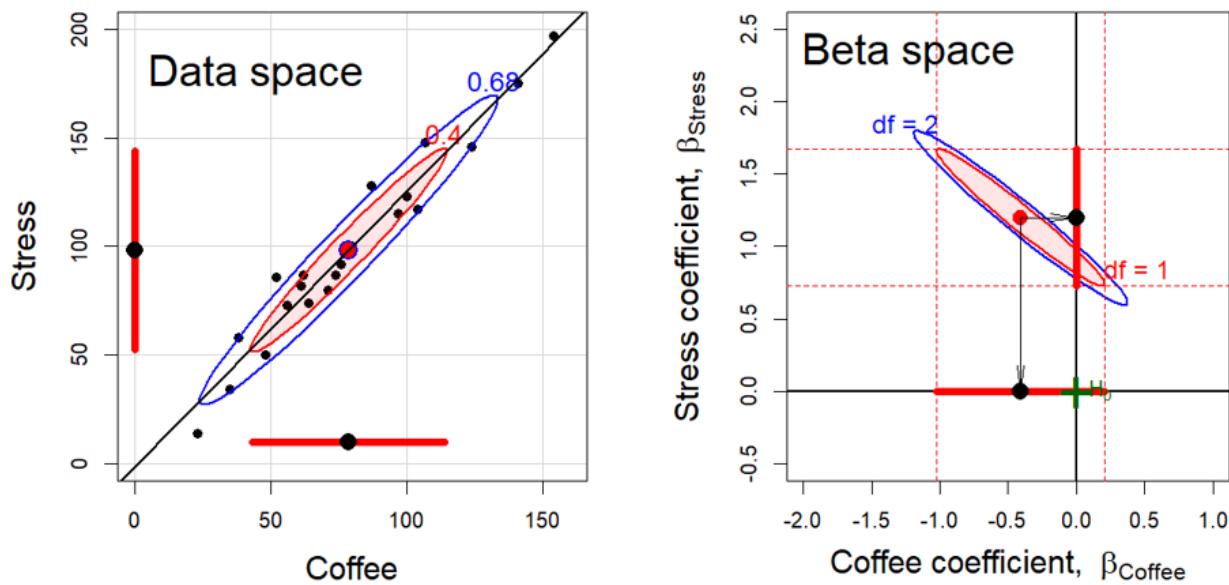


Figure 7.4: Data space and β space representations of Coffee and Stress. Left: 40% and 68% data ellipses. Right: Joint 95% confidence ellipse (blue) for $(\beta_{\text{Coffee}}, \beta_{\text{Stress}})$, confidence interval generating ellipse (red) with 95% univariate shadows. H_0 marks the joint hypothesis that both coefficients equal zero.

Thus, controlling for Stress, the confidence interval for the slope for Coffee includes 0, so we cannot reject the hypothesis that $\beta_{\text{Coffee}} = 0$ in the multiple regression model, as we saw above in the numerical output. On the other hand, the interval for the slope for Stress excludes the origin, so we reject the null hypothesis that $\beta_{\text{Stress}} = 0$, controlling for Coffee consumption.

Finally, consider the relationship between the data ellipse and the confidence ellipse. These have exactly the same shape, but (with equal coordinate scaling of the axes), the confidence ellipse is exactly a 90° rotation and rescaling of the data ellipse. In directions in data space where the slice of the data ellipse is wide—where we have more information about the relationship between Coffee and Stress—the projection of the confidence ellipse is narrow, reflecting greater precision of the estimates of coefficients. Conversely, where slice of the data ellipse is narrow (less information), the projection of the confidence ellipse is wide (less precision).

Confidence ellipses are drawn using `car::confidenceEllipse()`. Click the button to show the code.

```
confidenceEllipse(coffee.mod,
  grid = FALSE,
  xlim = c(-2, 1), ylim = c(-0.5, 2.5),
  xlab = expression(paste("Coffee coefficient, ", beta["Coffee"])),
  ylab = expression(paste("Stress coefficient, ", beta["Stress"])),
  cex.lab = 1.5)
confidenceEllipse(coffee.mod, add=TRUE, draw = TRUE,
  col = "red", fill = TRUE, fill.alpha = 0.1,
  dfn = 1)
abline(h = 0, v = 0, lwd = 2)

# confidence intervals
beta <- coef( coffee.mod )[-1]
CI <- confint(coffee.mod)
lines( y = c(0,0), x = CI["Coffee",] , lwd = 6, col = 'red')
```

```

lines( x = c(0,0), y = CI["Stress"], lwd = 6, col = 'red')
points( diag( beta ), col = 'black', pch = 16, cex=1.8)

abline(v = CI["Coffee"], col = "red", lty = 2)
abline(h = CI["Stress"], col = "red", lty = 2)

text(-2.1, 2.35, "Beta space", cex=2, pos = 4)
arrows(beta[1], beta[2], beta[1], 0, angle=8, len=0.2)
arrows(beta[1], beta[2], 0, beta[2], angle=8, len=0.2)

text( -1.5, 1.85, "df = 2", col = 'blue', adj = 0, cex=1.2)
text( 0.2, .85, "df = 1", col = 'red', adj = 0, cex=1.2)

heplots::mark.H0(col = "darkgreen", pch = "+", lty = 0, pos = 4, cex = 3)

```

7.2 Measurement error

7.2.1 OLS is BLUE

In classical linear models, the predictors are often considered to be fixed variables, or, if random, to be measured without error and independent of the regression errors. Either condition, along with the assumption of linearity, guarantees that the standard OLS estimators are *unbiased*. That is, in a simple linear regression, $y = \beta_0 + \beta_1 x + \epsilon$, the estimated slope $\hat{\beta}_1$ will have an average, expected value $\mathcal{E}(\hat{\beta}_1)$ equal to the true population value β_1 over repeated samples.

Not only this, but the Gauss-Markov theorem guarantees that the OLS estimator is also the most *efficient* because it has the least variance among all linear and unbiased estimators. The classical OLS estimator is said to be **BLUE**: **B**est (lowest variance), **L**inear (among linear estimators), **U**nbiased, **E**stimator.

7.2.2 Errors in predictors

Errors in the response y are accounted for in the model and measured by the mean squared error, $MSE = \hat{\sigma}_\epsilon^2$. But in practice, of course, predictor variables are often also observed indicators, subject to their own error. Indeed, in the behavioral sciences it is rare that predictors are perfectly reliable and measured exactly. This fact that is recognized in errors-in-variables regression models (Fuller, 2006) and in more general structural equation models, but often ignored otherwise. Ellipsoids in data space and β space are well suited to showing the effect of measurement error in predictors on OLS estimates.

The statistical facts are well known, though perhaps counter-intuitive in certain details: measurement error in a predictor biases regression coefficients (towards 0), while error in the measurement in y increases the MSE and thus standard errors of the regression coefficients but does not introduce bias in the coefficients.

7.2.2.1 Example

An illuminating example can be constructed by starting with the simple linear regression

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i ,$$

where x_i is the *true*, fully reliable predictor and y is the response, with error variance σ_ϵ^2 . Now consider that we don't measure x_i exactly, but instead observe x_i^* .

$$x_i^* = x_i + \eta_i ,$$

where the measurement error η_i is independent of the true x_i with variance σ_η^2 . We can extend this example to also consider the effect of adding additional, independent error variance to y , so that instead of y_i we observe

$$y_i^* = y_i + \nu_i$$

with variance σ_ν^2 .

Let's simulate an example where the true relation is $y = 0.2 + 0.3x$ with error standard deviation $\sigma = 0.5$. I'll take x to be uniformly distributed in $[0, 10]$ and calculate y as normally distributed around that linear relation.

```
set.seed(123)
n <- 300

a <- 0.2    # true intercept
b <- 0.3    # true slope
sigma <- 0.5 # baseline error standard deviation

x <- runif(n, 0, 10)
y <- rnorm(n, a + b*x, sigma)
demo <- data.frame(x,y)
```

Then, generate alternative values x^* and y^* with additional error standard deviations around x given by $\sigma_\eta = 4$ and around y given by $\sigma_\nu = 1$.

```
err_y <- 1    # additional error stdev for y
err_x <- 4    # additional error stdev for x
demo <- demo |>
  mutate(y_star = rnorm(n, y, err_y),
        x_star = rnorm(n, x, err_x))
```

There are four possible models we could fit and compare, using the combinations of (x, x^*) and (y, y^*)

```
fit_1 <- lm(y ~ x,           data = demo)    # no additional error
fit_2 <- lm(y_star ~ x,      data = demo)    # error in y
fit_3 <- lm(y ~ x_star,     data = demo)    # error in x
fit_4 <- lm(y_star ~ x_star, data = demo)    # error in x and y
```

However, to show the differences visually, we can simply plot the data for each pair and show the regression lines (with confidence bands) and the data ellipses. To do this efficiently with `ggplot2`, it is necessary to transform the `demo` data to long format with columns `x` and `y`, distinguished by `name` for the four combinations.

```
# make the demo dataset long, with names for the four conditions
df <- bind_rows(
  data.frame(x=demo$x,       y=demo$y,       name="No measurement error"),
  data.frame(x=demo$x,       y=demo$y_star,   name="Measurement error on y"),
  data.frame(x=demo$x_star,  y=demo$y,       name="Measurement error on x"),
  data.frame(x=demo$x_star,  y=demo$y_star,   name="Measurement error on x and y")) |>
  mutate(name = fct_inorder(name))
```

Then, we can plot the data in `df` with points, regression lines and a data ellipse, facetting by `name` to give the **measurement error quartet**.

```
ggplot(df, aes(x, y)) +
  geom_point(alpha = 0.2) +
  stat_ellipse(geom = "polygon",
               color = "blue", fill= "blue",
               alpha=0.05, linewidth = 1.1) +
  geom_smooth(method="lm", formula = y~x, fullrange=TRUE, level=0.995,
              color = "red", fill = "red", alpha = 0.2) +
  facet_wrap(~name)
```

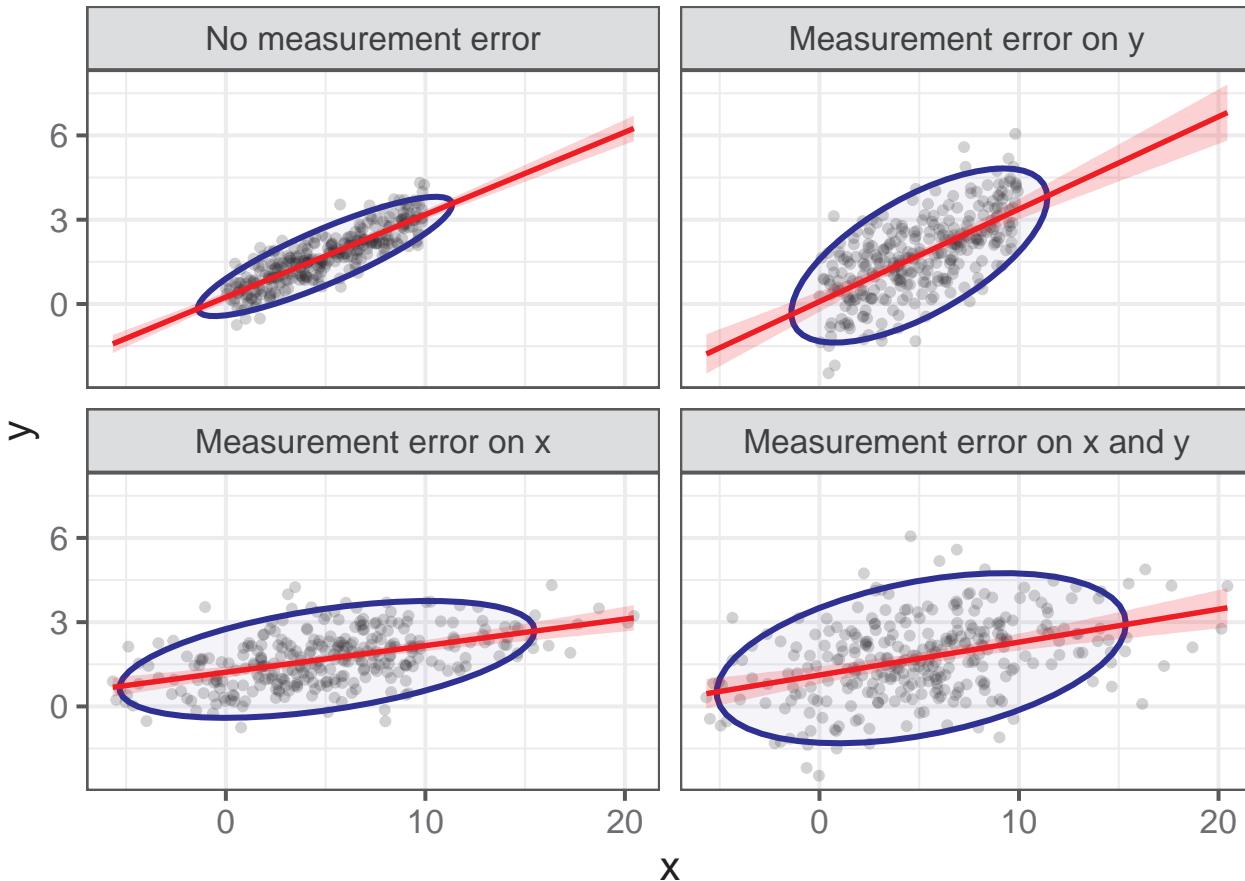


Figure 7.5: The measurement error quartet: Each plot shows the linear regression of y on x , but where additional error variance has been added to y or x or both. The widths of the confidence bands and the vertical extent of the data ellipses show the effect on precision.

Comparing the plots in the first row, you can see that when additional error is added to y , the regression slope remains essentially unchanged, illustrating that the estimate is unbiased. However, the confidence bounds on the regression line become wider, and the data ellipse becomes fatter in the y direction, illustrating the loss of precision.

The effect of error in x is less kind. Comparing the first row of plots with the second row, you can see that the estimated slope decreases when errors are added to x . This is called *attenuation bias*, and it can be shown that

$$\hat{\beta}_{x^*} \longrightarrow \frac{\beta}{1 + \sigma_\eta^2/\sigma_x^2},$$

where β here refers to the regression slope and \longrightarrow means “converges to”, as the sample size gets large. Thus, as σ_η^2 increases, $\hat{\beta}_{x^*}$ becomes less than β .

Beyond plots like Figure 7.5, we can see the effects of error in x or y on the model summary statistics such as the correlation r_{xy} or MSE by extracting these from the fitted models. This is easily done using `dplyr::nest_by(name)` and fitting the regression model to each subset, from which we can obtain the model statistics using `sigma()`, `coef()` and so forth. A bit of `dplyr::mutate()` magic is used to construct indicators `errX` and `errY` giving whether or not error was added to x and/or y .

```
model_stats <- df |>
  dplyr::nest_by(name) |>
  mutate(model = list(lm(y ~ x, data = data)),
         sigma = sigma(model),
         intercept = coef(model)[1],
         slope = coef(model)[2],
         r = sqrt(summary(model)$r.squared)) |>
  mutate(errX = stringr::str_detect(name, " x"),
         errY = stringr::str_detect(name, " y")) |>
  mutate(errX = factor(errX, levels = c("TRUE", "FALSE")),
         errY = factor(errY, levels = c("TRUE", "FALSE"))) |>
  relocate(errX, errY, r, .after = name) |>
  select(-data) |>
  print()
#> # A tibble: 4 x 8
#> # Rowwise:  name
#>   name          errX  errY      r model sigma intercept slope
#>   <fct>        <fct> <fct> <dbl> <lis> <dbl>     <dbl> <dbl>
#> 1 No measurement err~ FALSE FALSE 0.858 <lm>  0.495    0.244  0.294
#> 2 Measurement error ~ FALSE TRUE  0.648 <lm>  1.09     0.0838 0.329
#> 3 Measurement error ~ TRUE  FALSE 0.481 <lm>  0.844    1.22   0.0946
#> 4 Measurement error ~ TRUE  TRUE  0.401 <lm>  1.31     1.12   0.117
```

We plot the model $R = r_{xy}$ and the estimated residual standard error in Figure 7.6 below. The lines connecting the points are approximately parallel, indicating that errors of measurement in x and y have nearly additive effects on model summaries.

```
p1 <- ggplot(data=model_stats,
              aes(x = errX, y = r,
                  group = errY, color = errY,
                  shape = errY, linetype = errY)) +
  geom_point(size = 4) +
  geom_line(linewidth = 1.2) +
  labs(x = "Error on X?",
       y = "Model R",
       color = "Error on Y?",
       shape = "Error on Y?",
       linetype = "Error on Y?")
  legend_inside(c(0.25, 0.8))

p2 <- ggplot(data=model_stats,
              aes(x = errX, y = sigma,
                  group = errY, color = errY,
                  shape = errY, linetype = errY)) +
  geom_point(size = 4) +
  geom_line(linewidth = 1.2) +
```

```

  labs(x = "Error on X?",
       y = "Model residual standard error",
       color = "Error on Y?",
       shape = "Error on Y?",
       linetype = "Error on Y?")

p1 + p2

```

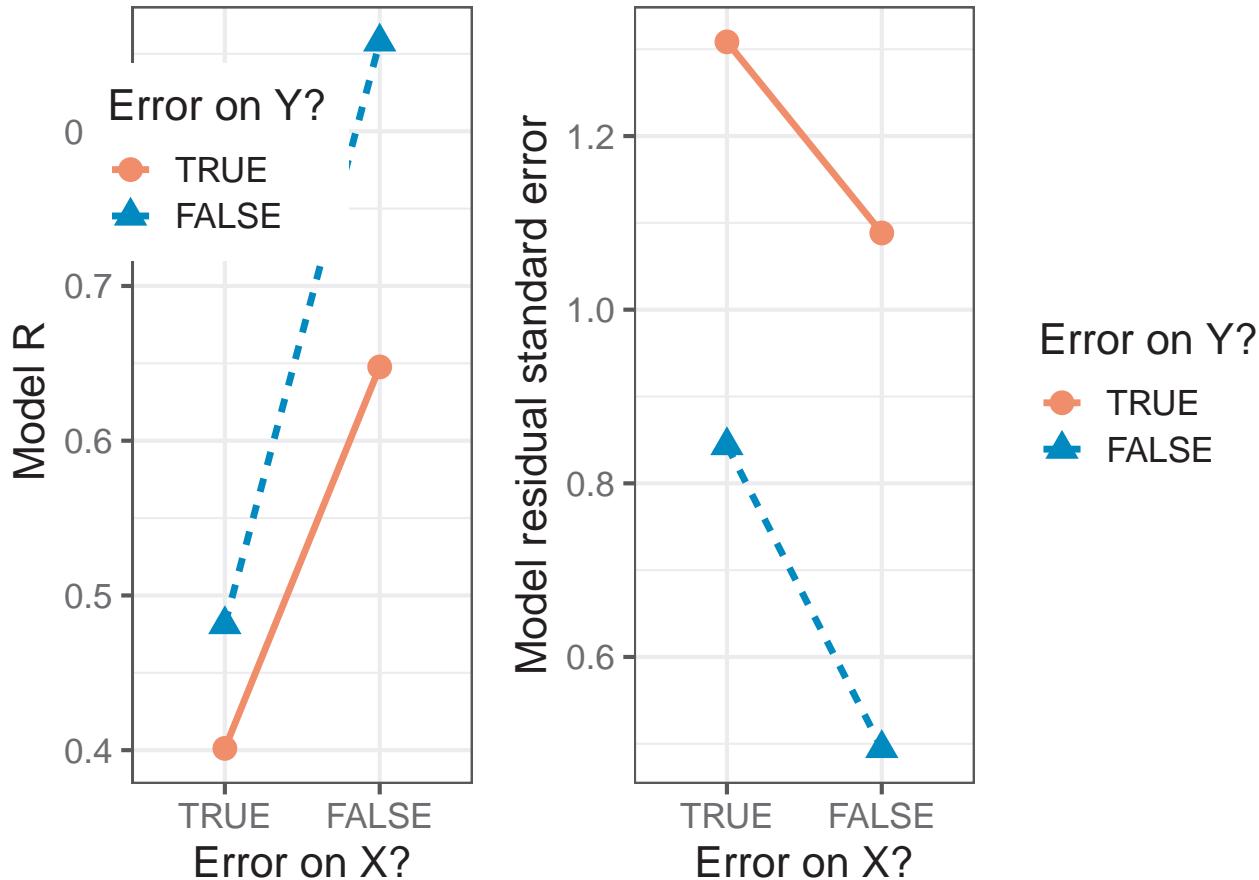


Figure 7.6: Model statistics for the combinations of additional error variance in x or y or both. Left: model R; right: Residual standard error.

7.2.3 Coffee data: β space

In multiple regression the effects of measurement error in a predictor become more complex, because error variance in one predictor, x_1 , say, can affect the coefficients of other terms in the model.

Consider the marginal relation between Heart disease and Stress in the `coffee` data. Figure 7.7 shows this with data ellipses in data space and the corresponding confidence ellipses in β space. Each panel starts with the observed data (the darkest ellipse, marked 0), then adds random normal error, $\mathcal{N}(0, \delta \times \text{SD}_{\text{Stress}})$, with $\delta = \{0.75, 1.0, 1.5\}$, to the value of Stress, while keeping the mean of Stress the same. All of the data ellipses have the same vertical shadows (SD_{Heart}), while the horizontal shadows increase with δ , driving the slope for Stress toward 0.

In β space, it can be seen that the estimated coefficients, $(\beta_0, \beta_{\text{Stress}})$ vary along a line and approach $\beta_{\text{Stress}} = 0$

as δ gets sufficiently large. The shadows of ellipses for $(\beta_0, \beta_{\text{Stress}})$ along the β_{Stress} axis also demonstrate the effects of measurement error on the standard error of β_{Stress} .

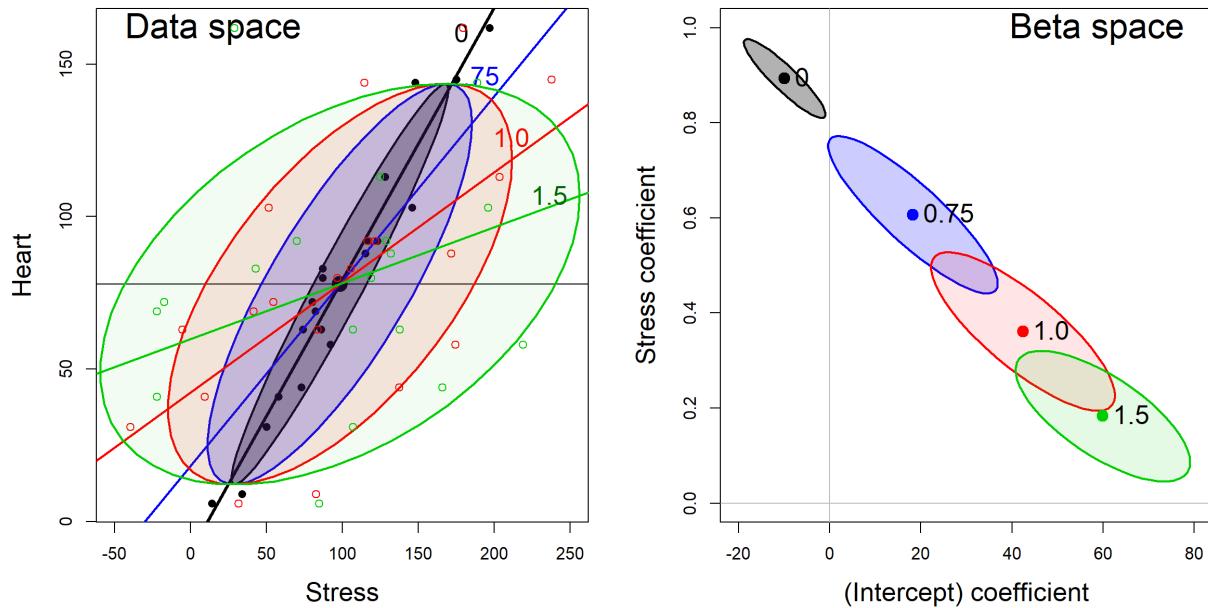


Figure 7.7: Effects of measurement error in Stress on the marginal relationship between Heart disease and Stress. Each panel starts with the observed data ($\delta = 0$), then adds random normal error, $\mathcal{N}(0, \delta \times \text{SD}_{\text{Stress}})$ with standard deviations multiplied by $\delta = 0.75, 1.0, 1.5$, to the value of Stress. Increasing measurement error biases the slope for Stress toward 0. Left: 50% data ellipses; right: 50% confidence ellipses.

Perhaps less well-known, but both more surprising and interesting, is the effect that measurement error in one variable, x_1 , has on the estimate of the coefficient for an *other* variable, x_2 , in a multiple regression model. Figure 7.8 shows the confidence ellipses for $(\beta_{\text{Coffee}}, \beta_{\text{Stress}})$ in the multiple regression predicting Heart disease, adding random normal error $\mathcal{N}(0, \delta \times \text{SD}_{\text{Stress}})$, with $\delta = \{0, 0.2, 0.4, 0.8\}$, to the value of Stress alone. As can be plainly seen, while this measurement error in Stress attenuates its coefficient, it also has the effect of biasing the coefficient for Coffee toward that in the *marginal* regression of Heart disease on Coffee alone.

7.3 What have we learned?

- **Data space and β space are dualities of each other** - While we typically visualize regression models in data space (where points are observations), there's a parallel β space where points represent models and their coefficients. These spaces mirror each other in elegant ways: lines in one space become points in the other, and confidence ellipses in β space are 90° rotations of data ellipses. This duality reveals that we gain precision in estimating coefficients precisely where our data spread the most.
- **Confidence ellipses make hypothesis testing visual and intuitive** - Instead of squinting at p-values in regression output, we can literally see whether hypotheses are supported by checking whether null hypothesis points fall inside or outside confidence ellipses. The shadows of these ellipses automatically give us individual confidence intervals, while the full ellipse captures joint uncertainty about multiple coefficients.
- **Measurement error in predictors is far more dangerous than measurement error in responses** - While errors in your response variable (y) simply inflate standard errors without biasing coefficients, errors in predictors create attenuation bias that systematically pulls slope estimates toward zero. This

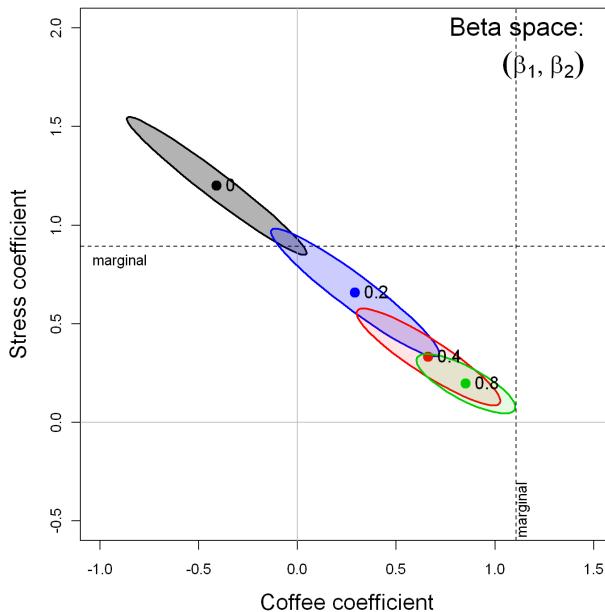


Figure 7.8: Biassing effect of measurement error in one variable (Stress) on on the coefficient of another variable (Coffee) in a multiple regression. The coefficient for Coffee is driven towards its value in the marginal model using Coffee alone, as measurement error in Stress makes it less informative in the joint model.

“errors-in-variables” problem means that unreliable measurements of your predictors can make real effects appear weaker than they actually are.

- **In multiple regression, measurement error in one predictor contaminates estimates of other predictors** - Perhaps most surprisingly, when one predictor in your model suffers from measurement error, it doesn’t just bias its own coefficient—it also distorts the coefficients of other variables in unpredictable ways. This distortion will cascade, meaning that measurement quality affects your entire model, not just individual variables.
- **Ellipses reveal the hidden geometry behind familiar statistical concepts** - Data ellipses, confidence ellipses, and their mathematical relationships provide a geometric foundation for understanding correlation, regression coefficients, confidence intervals, and hypothesis tests. This visual approach transforms abstract statistical concepts into concrete geometric relationships that you can literally see and manipulate.

Collinearity & Ridge Regression

In univariate multiple regression models, we usually hope to have high correlations between the outcome y and each of the predictors, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots]$, but high correlations *among* the predictors can cause problems in estimating and testing their effects. Exactly the same problems can exist in multivariate response models, because they involve only the relations among the predictor variables.

The problem of high correlations among the predictors in a model is called **collinearity** (or multicollinearity), referring to the situation when two or more predictors are very nearly linearly related to each other (collinear). This chapter illustrates the nature of collinearity geometrically, using data and confidence ellipsoids. It describes diagnostic measures to assess these effects, and presents some novel visual tools for these purposes using the `VisCollin` package.

One class of solutions for collinearity involves *regularization methods* such as ridge regression. Another collection of graphical methods, generalized ridge trace plots, implemented in the `genridge` package, sheds further light on what is accomplished by this technique. More generally, the methods of this chapter are further examples of how data and confidence ellipsoids can be used to visualize bias and precision of regression estimates.

Packages

In this chapter I use the following packages. Load them now.

```
library(car)
library(VisCollin)
library(genridge)
library(MASS)
library(dplyr)
library(factoextra)
library(ggrepel)
library(patchwork)
```

8.1 What is collinearity?

Researchers who have studied standard treatments of linear models (e.g, Graybill (1961); Hocking (2013)) are often less than clear about what collinearity is, how to find its sources and how to take steps to resolve them. There are a number of important diagnostic measures that can help, but these are usually presented in a tabular display like Figure 8.1, which prompted this query on an online forum:

Some of my collinearity diagnostics have large values, or small values, or whatever they are *not* supposed to be

- What is bad?
- If bad, what can I do about it?

The trouble with displays like Figure 8.1 is that the important information is hidden in a sea of numbers,

Model	Dimension	Eigenvalue	Condition Index	(Constant)	Variance Proportions					
					x1	x2	x3	x4	x5	x6
1	1	6,257	1,000	,00	,00	,00	,00	,00	,00	,00
2		,265	4,857	,00	,00	,00	,00	,00	,24	,21
3		,232	5,196	,00	,00	,00	,00	,00	,39	,39
4		,187	5,791	,00	,00	,00	,00	,01	,00	,33
5		,058	10,384	,03	,01	,00	,00	,01	,30	,04
6		,001	80,295	,13	,96	,95	,04	,05	,04	,00
7		,001	109,115	,85	,02	,05	,96	,94	,03	,02

a. Dependent Variable: y

Figure 8.1: Collinearity diagnostics for a multiple regression model from SPSS. Source: Arndt Regorzh, How to interpret a Collinearity Diagnostics table in SPSS, <https://bit.ly/3YRB82b>

some of which are bad when *large*, others bad when they are *small* and a large bunch which are irrelevant to interpretation.

In Friendly & Kwan (2009), we liken this problem to that of the reader of Martin Hansford's successful series of books, *Where's Waldo*. These consist of a series of full-page illustrations of hundreds of people and things and a few Waldos—a character wearing a red and white striped shirt and hat, glasses, and carrying a walking stick or other paraphernalia. Waldo was never disguised, yet the complex arrangement of misleading visual cues in the pictures made him very hard to find. Collinearity diagnostics often provide a similar puzzle: where should you look in traditional tabular displays?¹

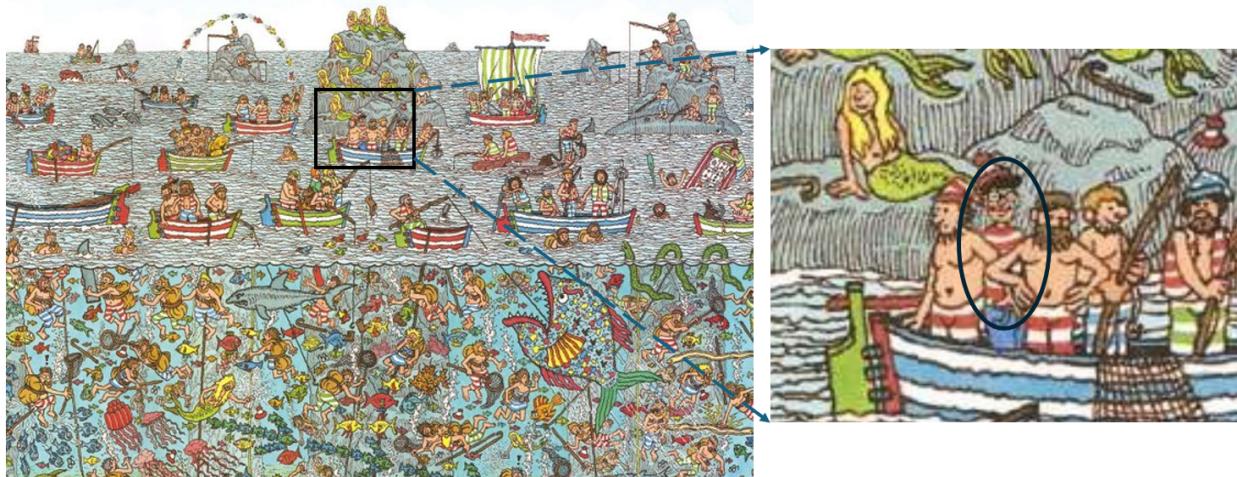


Figure 8.2: A scene from one of the *Where's Waldo* books. Waldo wears a red-striped shirt, but far too many of the other figures in the scene have horizontal red stripes, making it very difficult to find him among all the distractors. This is often the problem with collinearity diagnostics. Source: Modified from <https://bit.ly/48KPcOo>

Recall the standard classical linear model for a response variable y with a collection of predictors in $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p)$

$$\begin{aligned}\mathbf{y} &= \beta_0 + \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \cdots + \beta_p \mathbf{x}_p + \epsilon \\ &= \mathbf{X}\beta + \epsilon,\end{aligned}$$

¹The “Where’s Waldo” problem has attracted attention in machine learning, AI and computational image analysis circles. One approach uses convolutional neural networks. [FindWaldo](#) is one example implemented in Python.

for which the ordinary least squares solution is:

$$\hat{\mathbf{b}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

The sampling variances and covariances of the estimated coefficients is $\text{Var}(\hat{\mathbf{b}}) = \sigma_\epsilon^2 \times (\mathbf{X}^T \mathbf{X})^{-1}$ and σ_ϵ^2 is the variance of the residuals ϵ , estimated by the mean squared error (MSE).

In the limiting case, collinearity becomes particularly problematic when one x_i is *perfectly* predictable from the other xs , i.e., $R^2(x_i | \text{other } x) = 1$. This is problematic because:

- there is no *unique* solution for the regression coefficients $\mathbf{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X} \mathbf{y}$;
- the standard errors $s(b_i)$ of the estimated coefficients are infinite and t statistics $t_i = b_i / s(b_i)$ are 0.

This extreme case reflects a situation when one or more predictors are effectively redundant, for example when you include two variables x and y and their sum $z = x + y$ in a model. For instance, a dataset may include variables for income, expenses, and savings. But income is the sum of expenses and savings, so not all three should be used as predictors.

A more subtle case is the use *ipsatized*, defined as scores that sum to a constant, such as proportions of a total. You might have scores on tests of reading, math, spelling and geography. With ipsatized scores, any one of these is necessarily 1 – sum of the others, i.e., if reading is 0.5, math and geography are both 0.15, then geography must be 0.2. Once three of the four scores are known, the last provides no new information.

More generally, collinearity refers to the case when there are very high **multiple correlations** among the predictors, such as $R^2(x_i | \text{other } x) \geq 0.9$. Note that you can't tell simply by looking at the simple correlations. A large correlation r_{ij} is *sufficient* for collinearity, but not *necessary*—you can have variables x_1, x_2, x_3 for which the pairwise correlation are low, but the multiple correlation is high.

The consequences are:

- The estimated coefficients have large standard errors, $s(\hat{b}_j)$. They are multiplied by the square root of the variance inflation factor, $\sqrt{\text{VIF}}$, discussed below.
- The large standard errors deflate the t -statistics, $t = \hat{b}_j / s(\hat{b}_j)$, by the same factor, so a coefficient that would significant if the predictors were uncorrelated becomes insignificant when collinearity is present.
- Thus you may find a situation where an overall model is highly significant (large F -statistic), while no (or few) of the individual predictors are. This is a puzzlement!
- Beyond this, the least squares solution may have poor numerical accuracy ([Longley, 1967](#)), because the solution depends inversely on the determinant $|\mathbf{X}^T \mathbf{X}|$, which approaches 0 as multiple correlations increase.
- There is an interpretive problem as well. Recall that the coefficients \hat{b} are *partial coefficients*, meaning that they estimate change Δy in y when x changes by one unit Δx , but holding **all other variables constant**. Then, the model may be trying to estimate something that does not occur in the data. (For example: predicting strength from the highly correlated height and weight)

8.1.1 Visualizing collinearity

Collinearity can be illustrated in data space for two predictors in terms of the stability of the regression plane for a linear model $Y = X_1 + X_2$. Figure 8.3 (adapted from [Fox \(2016\)](#), Fig. 13.2) shows three cases as 3D plots of (X_1, X_2, Y) , where the correlation of predictors can be observed in the (X_1, X_2) plane.

- shows a case where X_1 and X_2 are uncorrelated as can be seen in their scatter in the horizontal plane (+ symbols). The gray regression plane is well-supported; a small change in Y for one observation won't make much difference.
- In panel (b), X_1 and X_2 have a perfect correlation, $r(x_1, x_2) = 1.0$. The regression plane is not unique; in fact there are an infinite number of planes that fit the data equally well. Note that, if all we care about is prediction (not the coefficients), we could use X_1 or X_2 , or both, or any weighted sum of them in a model and get the same predicted values.

- (c) Shows a typical case where there is a strong correlation between X_1 and X_2 . The regression plane here is unique, but is not well determined. A small change in Y **can** make quite a difference in the fitted value or coefficients, depending on the values of X_1 and X_2 . Where X_1 and X_2 are far from their near linear relation in the bottom plane, you can imagine that it is easy to tilt the plane substantially by a small change in Y.

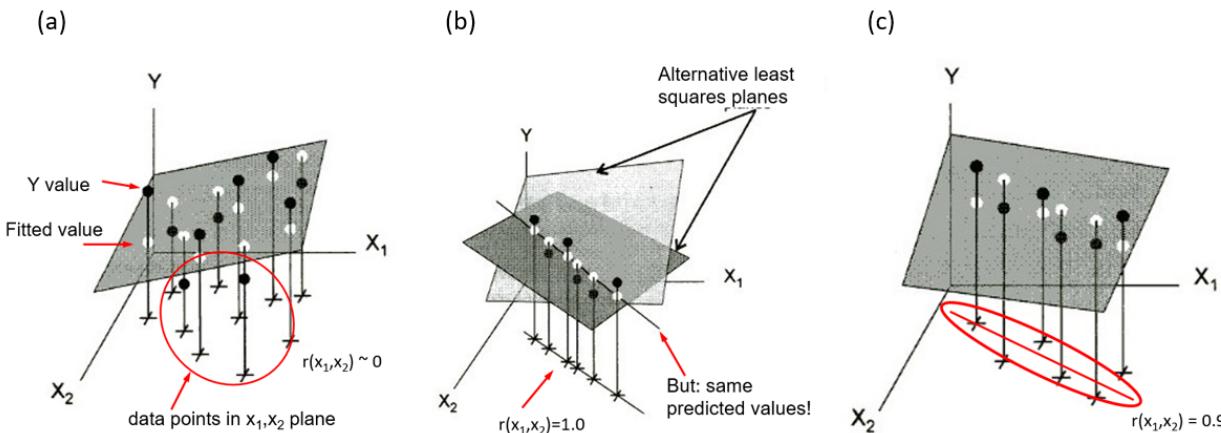


Figure 8.3: Effect of collinearity on the least squares regression plane. (a) Small correlation between predictors; (b) Perfect correlation ; (c) Very strong correlation. The black points show the data Y values, white points are the fitted values in the regression plane, and + signs represent the values of X₁ and X₂. Source: Adapted from Fox (2016), Fig. 13.2

8.1.2 Data space and β space

It is also useful to visualize collinearity by comparing the representation in **data space** with the analogous view of the confidence ellipses for coefficients in **beta space**. To do so in this example, I generate data from a known model $y = 3x_1 + 3x_2 + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 100)$ and various true correlations between x_1 and x_2 , $\rho_{12} = (0, 0.8, 0.97)$ ².

First, I use `MASS:mvrnorm()` to construct a list of three data frames `XY` with the same means and standard deviations, but with different correlations. In each case, the variable y is generated with true coefficients

R file: `R/collin-data-beta.R` β = (3, 3), and the fitted model for that value of `rho` is added to a corresponding list of models, `mods`.

```
library(MASS)
library(car)

set.seed(421)                      # reproducibility
N <- 200                            # sample size
mu <- c(0, 0)                        # means
s <- c(1, 1)                         # standard deviations
rho <- c(0, 0.8, 0.97)               # correlations
beta <- c(3, 3)                      # true coefficients

# Specify a covariance matrix, with standard deviations
# s[1], s[2] and correlation r
Cov <- function(s, r){
  matrix(c(s[1],           r * s[1]*s[2],
           r * s[1]*s[2], s[2]), 2, 2)
}
```

²This example is adapted from one by John Fox (2022), [Collinearity Diagnostics](#)

```

        r * s[1]*s[2], s[2]), nrow = 2, ncol = 2)
}

# Generate a data frame of X, y for each rho
# Fit the model for each
XY <- vector(mode ="list", length = length(rho))
mods <- vector(mode ="list", length = length(rho))
for (i in seq_along(rho)) {
  r <- rho[i]
  X <- mvrnorm(N, mu, Sigma = Cov(s, r))
  colnames(X) <- c("x1", "x2")
  y <- beta[1] * X[,1] + beta[2] * X[,2] + rnorm(N, 0, 10)

  XY[[i]] <- data.frame(X, y=y)
  mods[[i]] <- lm(y ~ x1 + x2, data=XY[[i]])
}

```

The estimated coefficients can then be extracted using `coef()` applied to each model:

```

coefs <- sapply(mods, coef)
colnames(coefs) <- paste0("mod", 1:3, " (rho=", rho, ")")
coefs
#>           mod1 (rho=0) mod2 (rho=0.8) mod3 (rho=0.97)
#> (Intercept)      1.01       -0.0535      0.141
#> x1              3.18        3.4719      3.053
#> x2              1.68        2.9734      2.059

```

Then, I define a function to plot the data ellipse (`car::dataEllipse()`) for each data frame and confidence ellipse (`car::confidenceEllipse()`) for the coefficients in the corresponding fitted model. In the plots in Figure 8.4, I specify the x, y limits for each plot so that the relative sizes of these ellipses are comparable, so that variance inflation can be assessed visually.

```

do_plots <- function(XY, mod, r) {
  X <- as.matrix(XY[, 1:2])
  dataEllipse(X,
    levels= 0.95,
    col = "darkgreen",
    fill = TRUE, fill.alpha = 0.05,
    xlim = c(-3, 3),
    ylim = c(-3, 3), asp = 1)
  text(0, 3, bquote(rho == .(r)), cex = 2, pos = NULL)

  confidenceEllipse(mod,
    col = "red",
    fill = TRUE, fill.alpha = 0.1,
    xlab = expression(paste("x1 coefficient, ", beta[1])),
    ylab = expression(paste("x2 coefficient, ", beta[2])),
    xlim = c(-5, 10),
    ylim = c(-5, 10),
    asp = 1)
  points(beta[1], beta[2], pch = "+", cex=2)
  abline(v=0, h=0, lwd=2)
}

```

```

}

op <- par(mar = c(4,4,1,1)+0.1,
          mfc = c(2, 3),
          cex.lab = 1.5)
for (i in seq_along(rho)) {
  do_plots(XY[[i]], mods[[i]], rho[i])
}
par(op)

```

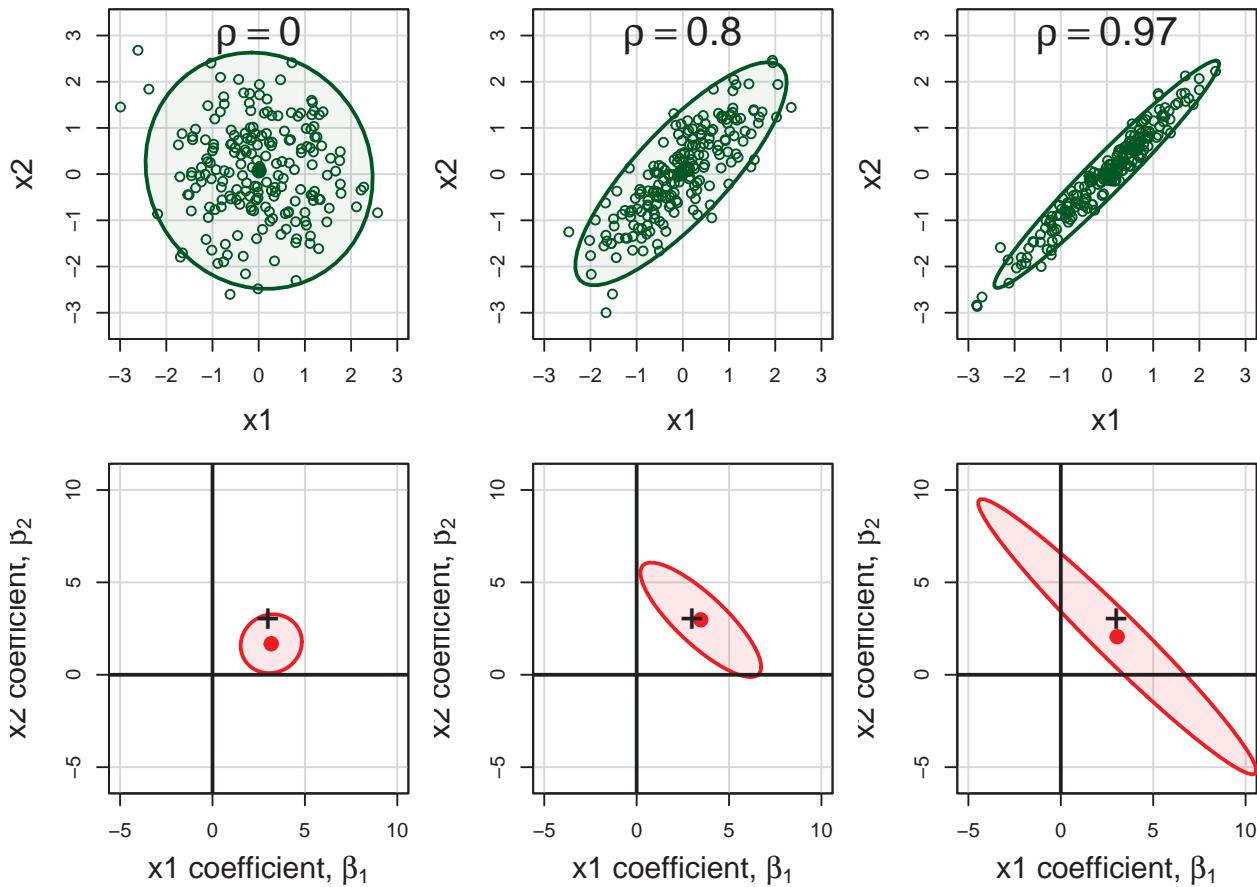


Figure 8.4: 95% Data ellipses for x_1 , x_2 and the corresponding 95% confidence ellipses for their coefficients in the model predicting y . In the confidence ellipse plots, reference lines show the value $(0,0)$ for the null hypothesis and “+” marks the true values for the coefficients. This figure adapts an example by John Fox (2022).

Recall (Section 7.1) that the confidence ellipse for (β_1, β_2) is just a 90 degree rotation (and rescaling) of the data ellipse for (x_1, x_2) : it is wide (more variance) in any direction where the data ellipse is narrow.

The shadows of the confidence ellipses on the coordinate axes in Figure 8.4 represent the standard errors of the coefficients, and get larger with increasing ρ . This is the effect of variance inflation, described in the following section.

8.2 Measuring collinearity

This section first describes the *variance inflation factor* (VIF) used to measure the effect of possible collinearity on each predictor and a collection of diagnostic measures designed to help interpret these. Then I describe some novel graphical methods to make these effects more readily understandable, to answer the “Where’s Waldo” question posed at the outset.

8.2.1 Variance inflation factors

How can we measure the effect of collinearity? The essential idea is to compare, for each predictor the variance $s^2(\hat{b}_j)$ that the coefficient that x_j would have if it was totally unrelated to the other predictors to the actual variance it has in the given model.

For two predictors such as shown in Figure 8.4 the sampling variance of x_1 can be expressed as

$$s^2(\hat{b}_1) = \frac{MSE}{(n - 1) s^2(x_1)} \times \left[\frac{1}{1 - r_{12}^2} \right]$$

The first term here is the variance of b_1 when the two predictors are uncorrelated. The term in brackets represents the **variance inflation factor** (Marquardt, 1970), the amount by which the variance of the coefficient is multiplied as a consequence of the correlation r_{12} of the predictors. As $r_{12} \rightarrow 1$, the variances approaches infinity.

More generally, with any number of predictors, this relation has a similar form, replacing the simple correlation r_{12} with the multiple correlation predicting x_j from all others,

$$s^2(\hat{b}_j) = \frac{MSE}{(n - 1) s^2(x_j)} \times \left[\frac{1}{1 - R_{j|\text{others}}^2} \right]$$

So, we have that the variance inflation factors are:

$$\text{VIF}_j = \frac{1}{1 - R_{j|\text{others}}^2}$$

In practice, it is often easier to think in terms of the square root, $\sqrt{\text{VIF}_j}$ as the multiplier of the standard errors. The denominator, $1 - R_{j|\text{others}}^2$ is sometimes called **tolerance**, a term I don’t find particularly useful, but it is just the proportion of the variance of x_j that is *not* explainable from the others.

For the cases shown in Figure 8.4 the VIFs and their square roots are:

```
vifs <- sapply(mods, car::vif)
colnames(vifs) <- paste("rho:", rho)
vifs
#>   rho: 0 rho: 0.8 rho: 0.97
#> x1      1     3.09     18.6
#> x2      1     3.09     18.6

sqrt(vifs)
#>   rho: 0 rho: 0.8 rho: 0.97
#> x1      1     1.76     4.31
#> x2      1     1.76     4.31
```

Note that when there are terms in the model with more than one degree of freedom, such as education with four levels (and hence 3 df) or a polynomial term specified as `poly(age, 3)`, that variable, education or age

is represented by three separate *xs* in the model matrix, and the standard VIF calculation gives results that vary with how those terms are coded in the model.

To allow for these cases, Fox & Monette (1992) define *generalized*, GVIFs as the inflation in the squared area of the confidence ellipse for the coefficients of such terms, relative to what would be obtained with uncorrelated data. Visually, this can be seen by comparing the areas of the ellipses in the bottom row of Figure 8.4. Because the magnitude of the GVIF increases with the number of degrees of freedom for the set of parameters, Fox & Monette suggest the analog $\sqrt{\text{GVIF}^{1/2\text{df}}}$ as the measure of impact on standard errors. This is what `car::vif()` calculates for a factor or other term with more than 1 df.

Example 8.1. Cars data

This example uses the `cars` dataset in the `VisCollin` package containing various measures of size and performance on 406 models of automobiles from 1982. Interest is focused on predicting gas mileage, `mpg`.

```
data(cars, package = "VisCollin")
str(cars)
#> 'data.frame': 406 obs. of 10 variables:
#> $ make     : Factor w/ 30 levels "amc","audi","bmw",...: 6 4 22 1 12 12 6 22 23 1 ...
#> $ model    : chr "chevelle" "skylark" "satellite" "rebel" ...
#> $ mpg      : num 18 15 18 16 17 15 14 14 14 15 ...
#> $ cylinder: int 8 8 8 8 8 8 8 8 8 ...
#> $ engine   : num 307 350 318 304 302 429 454 440 455 390 ...
#> $ horse    : int 130 165 150 150 140 198 220 215 225 190 ...
#> $ weight   : int 3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
#> $ accel    : num 12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
#> $ year     : int 70 70 70 70 70 70 70 70 70 70 ...
#> $ origin   : Factor w/ 3 levels "Amer","Eur","Japan": 1 1 1 1 1 1 1 1 1 1 ...
```

We fit a model predicting gas mileage (`mpg`) from the number of cylinders, engine displacement, horsepower, weight, time to accelerate from 0 – 60 mph and model year (1970–1982). Perhaps surprisingly, only `weight` and `year` appear to significantly predict gas mileage. What's going on here?

```
cars.mod <- lm (mpg ~ cylinder + engine + horse +
                  weight + accel + year,
                  data=cars)
Anova(cars.mod)
#> Anova Table (Type II tests)
#>
#> Response: mpg
#>           Sum Sq Df F value Pr(>F)
#> cylinder     12   1  0.99  0.32
#> engine        13   1  1.09  0.30
#> horse         0   1  0.00  0.98
#> weight       1214   1 102.84 <2e-16 ***
#> accel         8   1  0.70  0.40
#> year         2419   1 204.99 <2e-16 ***
#> Residuals   4543 385
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We check the variance inflation factors, using `car::vif()`. We see that most predictors have very high VIFs, indicating moderately severe multicollinearity.

```
vif(cars.mod)
#> cylinder engine horse weight accel year
#> 10.63   19.64  9.40  10.73  2.63  1.24

sqrt(vif(cars.mod))
#> cylinder engine horse weight accel year
#> 3.26    4.43   3.07   3.28   1.62   1.12
```

According to \sqrt{VIF} , the standard error of `cylinder` has been multiplied by $\sqrt{10.63} = 3.26$ and it's t -value is divided by this number, compared with the case when all predictors are uncorrelated. `engine`, `horse` and `weight` suffer a similar fate.

If we also included the factor `origin` in the models, we would get the generalized GVIF:

```
cars.mod2 <- lm (mpg ~ cylinder + engine + horse +
                  weight + accel + year + origin,
                  data=cars)
vif(cars.mod2)
#>           GVIF Df GVIF^(1/(2*Df))
#> cylinder 10.74  1      3.28
#> engine   22.94  1      4.79
#> horse     9.96  1      3.16
#> weight   11.07  1      3.33
#> accel     2.63  1      1.62
#> year     1.30  1      1.14
#> origin    2.10  2      1.20
```

💡 Connection with inverse of correlation matrix

In the linear regression model with standardized predictors, the covariance matrix of the estimated intercept-excluding parameter vector \mathbf{b}^* has the simpler form,

$$\mathcal{V}(\mathbf{b}^*) = \frac{\sigma^2}{n-1} \mathbf{R}_X^{-1}.$$

where \mathbf{R}_X is the correlation matrix among the predictors. It can then be seen that the VIF_j are just the diagonal entries of \mathbf{R}_X^{-1} .

More generally, the matrix $\mathbf{R}_X^{-1} = (r^{ij})$, when standardized to a correlation matrix as $-r^{ij}/\sqrt{r^{ii} r^{jj}}$ gives the matrix of all partial correlations, $r_{ij} | \text{others}$. }

8.2.2 Collinearity diagnostics

OK, we now know that large VIF_j indicate predictor coefficients whose estimation is degraded due to large $R_j^2 | \text{others}$. But for this to be useful, we need to determine:

- how many dimensions in the space of the predictors are associated with nearly collinear relations?
- which predictors are most strongly implicated in each of these?

Answers to these questions are provided using measures developed by Belsley and colleagues (Belsley et al., 1980; Belsley, 1991). These measures are based on the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_p$ of the correlation matrix R_X of the predictors (preferably centered and scaled, and not including the constant term for the intercept), and the corresponding eigenvectors in the columns of $\mathbf{V}_{p \times p}$, given by the the eigen decomposition

$$\mathbf{R}_X = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^\top$$

By elementary matrix algebra, the eigen decomposition of \mathbf{R}_{XX}^{-1} is then

$$\mathbf{R}_X^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^T, \quad (8.1)$$

so, \mathbf{R}_X and \mathbf{R}_{XX}^{-1} have the same eigenvectors, and the eigenvalues of \mathbf{R}_X^{-1} are just λ_i^{-1} . Using Equation 8.1, the variance inflation factors may be expressed as

$$\text{VIF}_j = \sum_{k=1}^p \frac{V_{jk}^2}{\lambda_k},$$

which shows that only the *small* eigenvalues contribute to variance inflation, but only for those predictors that have large eigenvector coefficients on those small components. These facts lead to the following diagnostic statistics for collinearity:

- **Condition indices:** The smallest of the eigenvalues, those for which $\lambda_j \approx 0$, indicate collinearity and the number of small values indicates the number of near collinear relations. Because the sum of the eigenvalues, $\Sigma \lambda_i = p$ increases with the number of predictors p , it is useful to scale them all in relation to the largest. This leads to *condition indices*, defined as $\kappa_j = \sqrt{\lambda_1/\lambda_j}$. These have the property that the resulting numbers have common interpretations regardless of the number of predictors.
 - For completely uncorrelated predictors, all $\kappa_j = 1$.
 - $\kappa_j \rightarrow \infty$ as any $\lambda_k \rightarrow 0$.
- **Variance decomposition proportions:** Large VIFs indicate variables that are involved in *some* nearly collinear relations, but they don't indicate *which* other variable(s) each is involved with. For this purpose, Belsley et al. (1980) and Belsley (1991) proposed calculation of the proportions of variance of each variable associated with each principal component as a decomposition of the coefficient variance for each dimension.

These measures can be calculated using `VisCollin::colldiag()`. For the current model, the usual display contains both the condition indices and variance proportions. However, even for a small example, it is often difficult to know what numbers to pay attention to.

```
(cd <- colldiag(cars.mod, center=TRUE))
#> Condition
#> Index      Variance Decomposition Proportions
#>          cylinder engine horse weight accel year
#> 1    1.000  0.005   0.003  0.005  0.004   0.009  0.010
#> 2    2.252  0.004   0.002  0.000  0.007   0.022  0.787
#> 3    2.515  0.004   0.001  0.002  0.010   0.423  0.142
#> 4    5.660  0.309   0.014  0.306  0.087   0.063  0.005
#> 5    8.342  0.115   0.000  0.654  0.715   0.469  0.052
#> 6   10.818  0.563   0.981  0.032  0.176   0.013  0.004
```

Belsley (1991) recommends that the sources of collinearity be diagnosed (a) only for those components with large κ_j , and (b) for those components for which the variance proportion is large (say, ≥ 0.5) on *two* or more predictors. The print method for "colldiag" objects has a `fuzz` argument controlling this.

```
print(cd, fuzz = 0.5)
#> Condition
#> Index      Variance Decomposition Proportions
#>          cylinder engine horse weight accel year
#> 1    1.000   .     .     .     .     .
#> 2    2.252   .     .     .     .     .     0.787
#> 3    2.515   .     .     .     .     .
#> 4    5.660   .     .     .     .     .     .
```

```
#> 5   8.342  .       .      0.654 0.715  .       .
#> 6 10.818 0.563  0.981  .       .      .       .
```

The mystery is solved, if you can read that table with these recommendations in mind. There are two nearly collinear relations among the predictors, corresponding to the two smallest dimensions.

- Dimension 5 reflects the high correlation between horsepower and weight,
- Dimension 6 reflects the high correlation between number of cylinders and engine displacement.

Note that the high variance proportion for `year` (0.787) on the second component creates no problem and should be ignored because (a) the condition index is low and (b) it shares nothing with other predictors.

8.3 Tableplots

The default tabular display of condition indices and variance proportions from `colldiag()` is what triggered the comparison to “Where’s Waldo”. It suffers from the fact that the important information — (a) how many Waldos? (b) where are they hiding — is disguised by being embedded in a sea of mostly irrelevant numbers. The simple option of using a principled `fuzz` factor helps considerably, but not entirely.

The simplified tabular display above can be improved to make the patterns of collinearity more visually apparent and to signify warnings directly to the eyes. A **tableplot** (Kwan et al., 2009) is a semi-graphic display that presents numerical information in a table using shapes proportional to the value in a cell and other visual attributes (shape type, color fill, and so forth) to encode other information.

For collinearity diagnostics, these show:

- the condition indices, using *squares* whose background color is **red** for condition indices > 10 , **brown** for values > 5 and **green** otherwise, reflecting danger, warning and OK respectively. The value of the condition index is encoded within this using a white square whose side is proportional to the value (up to some maximum value, `cond.max` that fills the cell).
- Variance decomposition proportions are shown by filled *circles* whose radius is proportional to those values and are filled (by default) with shades ranging from white through pink to red. Rounded values of those diagnostics are printed in the cells.

The tableplot below (Figure 8.5) encodes all the information from the values of `colldiag()` printed above. To aid perception, it uses `prop.col` color breaks such that variance proportions < 0.3 are shaded white. The visual message is that one should attend to collinearities with large condition indices **and** large variance proportions implicating two or more predictors.

```
tableplot(cd, title = "Tableplot of cars data",
          cond.max = 30 )
```

R file:
R/cars-colldiag.R

8.4 Collinearity biplots

As we have seen, the collinearity diagnostics are all functions of the eigenvalues and eigenvectors of the correlation matrix of the predictors in the regression model, or alternatively, the SVD of the \mathbf{X} matrix in the linear model (excluding the constant). The standard biplot (Gabriel, 1971; Gower & Hand, 1996) (see: Section 4.3) can be regarded as a multivariate analog of a scatterplot, obtained by projecting a multivariate

Tableplot of cars data

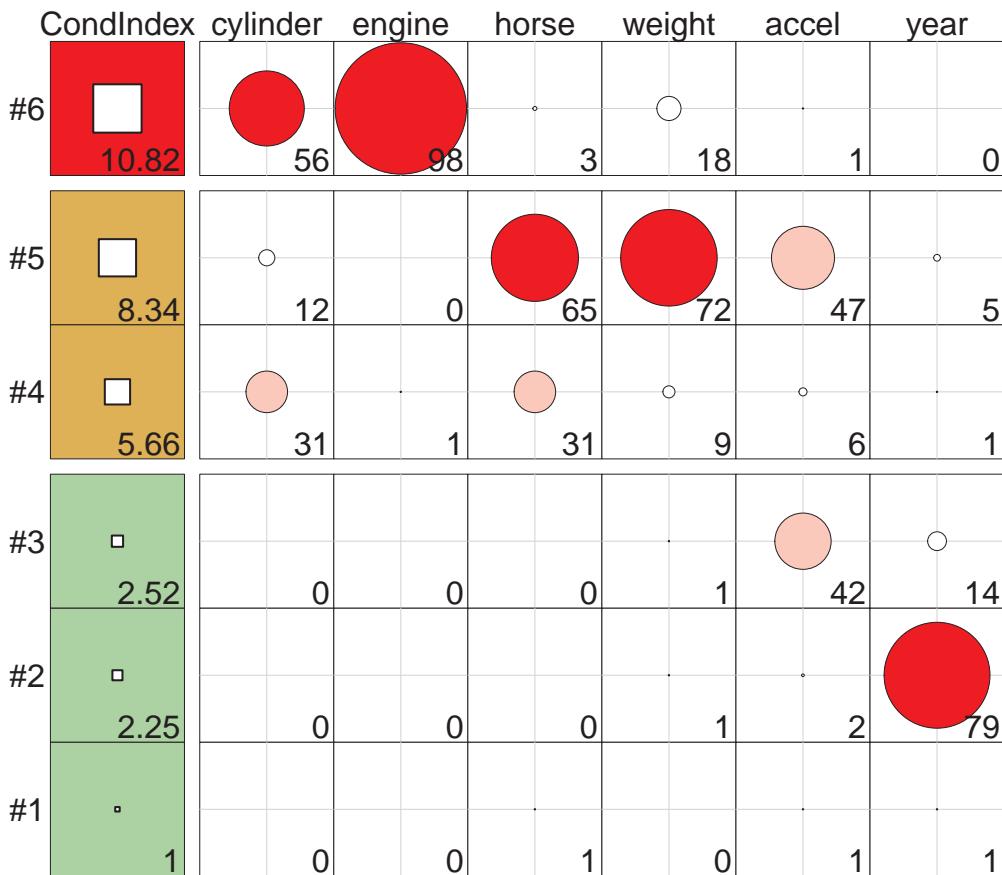


Figure 8.5: Tableplot of condition indices and variance proportions for the `cars` data. In column 1, the square symbols are scaled relative to a maximum condition index of 30. In the remaining columns, variance proportions (times 100) are shown as circles scaled relative to a maximum of 100.

sample into a low-dimensional space (typically of 2 or 3 dimensions) accounting for the greatest variance in the data.

However the standard biplot is less useful for visualizing the relations among the predictors that lead to nearly collinear relations. Instead, biplots of the **smallest dimensions** show these relations directly, and can show other features of the data as well, such as outliers and leverage points. We use `prcomp(X, scale.=TRUE)` to obtain the PCA of the correlation matrix of the predictors:

```

cars.X <- cars |>
  select(where(is.numeric)) |>
  select(-mpg) |>
  tidyverse::drop_na()
cars.pca <- prcomp(cars.X, scale. = TRUE)
cars.pca
#> Standard deviations (1, ..., p=6):
#> [1] 2.070 0.911 0.809 0.367 0.245 0.189
#>
#> Rotation (n x k) = (6 x 6):
#>          PC1     PC2     PC3     PC4     PC5     PC6

```

```
#> cylinder -0.454 -0.1869  0.168 -0.659 -0.2711 -0.4725
#> engine    -0.467 -0.1628  0.134 -0.193 -0.0109  0.8364
#> horse     -0.462 -0.0177 -0.123  0.620 -0.6123 -0.1067
#> weight    -0.444 -0.2598  0.278  0.350  0.6860 -0.2539
#> accel     0.330 -0.2098  0.865  0.143 -0.2774  0.0337
#> year      0.237 -0.9092 -0.335  0.025 -0.0624  0.0142
```

The standard deviations above are the square roots $\sqrt{\lambda_j}$ of the eigenvalues of the correlation matrix, and are returned in the `sdev` component of the "prcomp" object. The eigenvectors are returned in the `rotation` component, whose directions are arbitrary. Because we are interested in seeing the relative magnitude of variable vectors, we are free to multiply them by any constant to make them more visible in relation to the scores for the cars.

```
cars.pca$rotation <- -2.5 * cars.pca$rotation      # reflect & scale the variable vectors

ggp <- fviz_pca_biplot(
  cars.pca,
  axes = 6:5,
  geom = "point",
  col.var = "blue",
  labelsize = 5,
  pointsize = 1.5,
  arrowsize = 1.5,
  addEllipses = TRUE,
  ggtheme = ggplot2::theme_bw(base_size = 14),
  title = "Collinearity biplot for cars data")

# add point labels for outlying points
dsq <- heplots::Mahalanobis(cars.pca$x[, 6:5])
scores <- as.data.frame(cars.pca$x[, 6:5])
scores$name <- rownames(scores)

ggp + geom_text_repel(data = scores[dsq > qchisq(0.95, df = 6), ],
  aes(x = PC6,
      y = PC5,
      label = name),
  vjust = -0.5,
  size = 5)
```

As with the tabular display of variance proportions, Waldo is hiding in the dimensions associated with the smallest eigenvalues (largest condition indices). As well, it turns out that outliers in the predictor space (also high leverage observations) can often be seen as observations far from the centroid in the space of the smallest principal components.

The projections of the variable vectors in Figure 8.6 on the Dimension 5 and Dimension 6 axes are proportional to their variance proportions shown above. The relative lengths of these variable vectors can be considered to indicate the extent to which each variable contributes to collinearity for these two near-singular dimensions.

Thus, we see again that Dimension 6 is largely determined by `engine` size, with a substantial (negative) relation to `cylinder`. Dimension 5 has its' strongest relations to `weight` and `horse`.

Moreover, there is one observation, #20, that stands out as an outlier in predictor space, far from the centroid. It turns out that this vehicle, a Buick Estate wagon, is an early-year (1970) American behemoth, with an

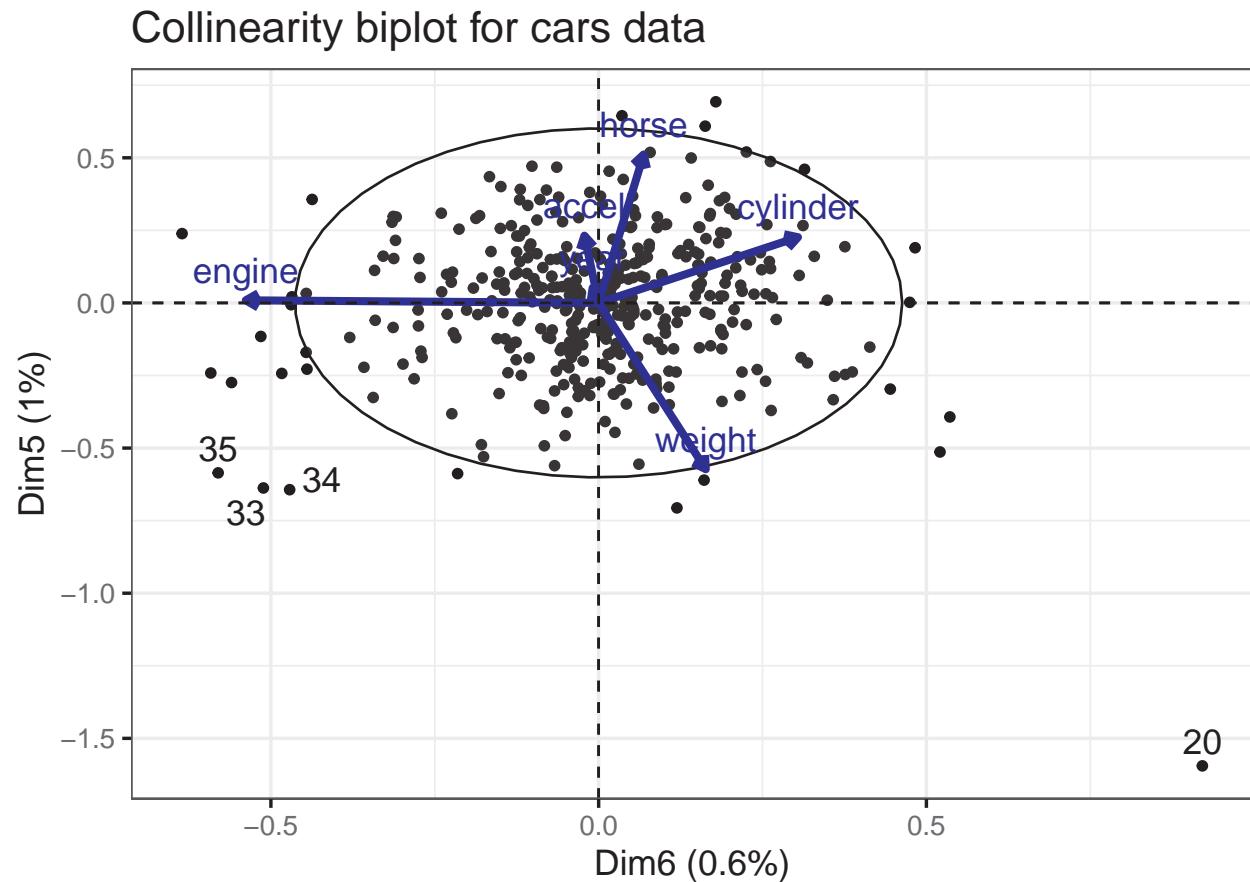


Figure 8.6: Collinearity biplot of the Cars data, showing the last two dimensions. The projections of the variable vectors on the coordinate axes are proportional to their variance proportions. To reduce graphic clutter, only the most outlying observations in predictor space are identified by case labels. An extreme outlier (case 20) appears in the lower right corner.

8-cylinder, 455 cu. in, 225 horse-power engine, and able to go from 0 to 60 mph in 10 sec. (Its MPG is only slightly under-predicted from the regression model, however.)

With PCA and the biplot, we are used to looking at the dimensions that account for the most variation, but the answer to *Where's Waldo?* is that he is hiding in the *smallest* data dimensions, just as he does in Figure 8.2 where the weak signals of his stripped shirt, hat and glasses are embedded in a visual field of noise. As we just saw, outliers hide there also, hoping to escape detection. These small dimensions are also implicated in ridge regression as we will see shortly (Section 8.6).

8.5 Remedies for collinearity: What can I do?

Collinearity is often a **data** problem, for which there is no magic cure. Nevertheless there are some general guidelines and useful techniques to address this problem.

- **Pure prediction:** If we are only interested in predicting / explaining an outcome, and not the model coefficients or which are “significant”, collinearity can be largely ignored. The fitted values are unaffected by collinearity, even in the case of perfect collinearity as shown in Figure 8.3 (b).

- **Structural collinearity:** Sometimes collinearity results from structural relations among the variables that relate to how they have been defined.

- For example, polynomial terms, like x, x^2, x^3 or interaction terms like $x_1, x_2, x_1 * x_2$ are necessarily correlated. A simple cure is to *center* the predictors at their means, using $x - \bar{x}, (x - \bar{x})^2, (x - \bar{x})^3$ or $(x_1 - \bar{x}_1), (x_2 - \bar{x}_2), (x_1 - \bar{x}_1) * (x_2 - \bar{x}_2)$. Centering removes the spurious ill-conditioning, thus reducing the VIFs. Note that in polynomial models, using `y ~ poly(x, 3)` to specify a cubic model generates *orthogonal* (uncorrelated) regressors, whereas in `y ~ x + I(x^2) + I(x^3)` the terms have built-in correlations.
- When some predictors share a common cause, as in GNP or population in time-series or cross-national data, you can reduce collinearity by re-defining predictors to reflect *per capita measures*. In a related example with sports data, when you have cumulative totals (e.g., runs, hits, homeruns in baseball) for players over years, expressing these measures as *per year* will reduce the common effect of longevity on these measures.

- **Model re-specification:**

- Drop one or more regressors that have a high VIF, if they are not deemed to be essential to understanding the model. Care must be taken here to not omit variables which should be controlled or accounted for in interpretation.
- Replace highly correlated regressors with less correlated linear combination(s) of them. For example, two related variables, x_1 and x_2 can be replaced without any loss of information by replacing them with their sum and difference, $z_1 = x_1 + x_2$ and $z_2 = x_1 - x_2$. For instance, in a dataset on fitness, we may have correlated predictors of resting pulse rate and pulse rate while running. Transforming these to average pulse rate and their difference gives new variables which are interpretable and less correlated.

- **Statistical remedies:**

- Transform the predictors \mathbf{X} to uncorrelated principal component scores $\mathbf{Z} = \mathbf{X}\mathbf{V}$, and regress \mathbf{y} on \mathbf{Z} . These will have the identical overall model fit without loss of information. A related technique is *incomplete* principal components regression, where some of the smallest dimensions (those causing collinearity) are omitted from the model. The trade-off is that it may be more difficult to interpret what the model means, but this can be countered with a biplot, showing the projections of the original variables into the reduced space of the principal components.
- Use **regularization methods** such as ridge regression and lasso, which correct for collinearity by introducing shrinking coefficients towards 0, introducing a small amount of bias, . See the [genridge](#) package and its [pkgdown documentation](#) for visualization methods.
- use Bayesian regression; if multicollinearity prevents a regression coefficient from being estimated precisely, then a prior on that coefficient will help to reduce its posterior variance.

Example: Centering

To illustrate the effect of centering a predictor in a polynomial model, we generate a perfect quadratic relationship, $y = x^2$ and consider the correlations of y with x and with $(x - \bar{x})^2$. The correlation of y with x is 0.97, while the correlation of y with $(x - \bar{x})^2$ is zero.

```

x <- 1:20
y1 <- x^2
y2 <- (x - mean(x))^2
XY <- data.frame(x, y1, y2)

(R <- cor(XY))
#>      x     y1     y2

```

```
#> x  1.000 0.971 0.000
#> y1 0.971 1.000 0.238
#> y2 0.000 0.238 1.000
```

The effect of centering here is remove the linear association in what is a purely quadratic relationship, as can be seen by plotting y1 and y2 against x.

```
r1 <- R[1, 2]
r2 <- R[1, 3]

gg1 <-
  ggplot(XY, aes(x = x, y = y1)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", formula = y~x, linewidth = 2, se = FALSE) +
  labs(x = "X", y = "Y") +
  theme_bw(base_size = 16) +
  annotate("text", x = 5, y = 350, size = 6,
          label = paste("X Uncentered\nnr =", round(r1, 3)))

gg2 <-
  ggplot(XY, aes(x = x, y = y2)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", formula = y~x, linewidth = 2, se = FALSE) +
  labs(x = "X", y = "Y") +
  theme_bw(base_size = 16) +
  annotate("text", x = 5, y = 80, size = 6,
          label = paste("X Centered\nnr =", round(r2, 3)))

gg1 + gg2      # show plots side-by-side
```

Example: Interactions

manufacturing process to produce acetylene in relation to reactor temperature (`temp`), the ratio of two components and the contact `time` in the reactor. A naive response surface model might suggest that yield is quadratic in time and there are potential interactions among all pairs of predictors.

```
data(Acetylene, package = "genridge")
acetyl.mod0 <- lm(
  yield ~ temp + ratio + time + I(time^2) +
  temp:time + temp:ratio + time:ratio,
  data=Acetylene)

(acetyl.vif0 <- vif(acetyl.mod0))
#>      temp        ratio        time  I(time^2)  temp:time temp:ratio
#>     383       10555     18080       564       9719      9693
#> ratio:time
#>     225
```

These results are horrible! How much does centering help? I first center all three predictors and then use `update()` to re-fit the same model using the centered data.

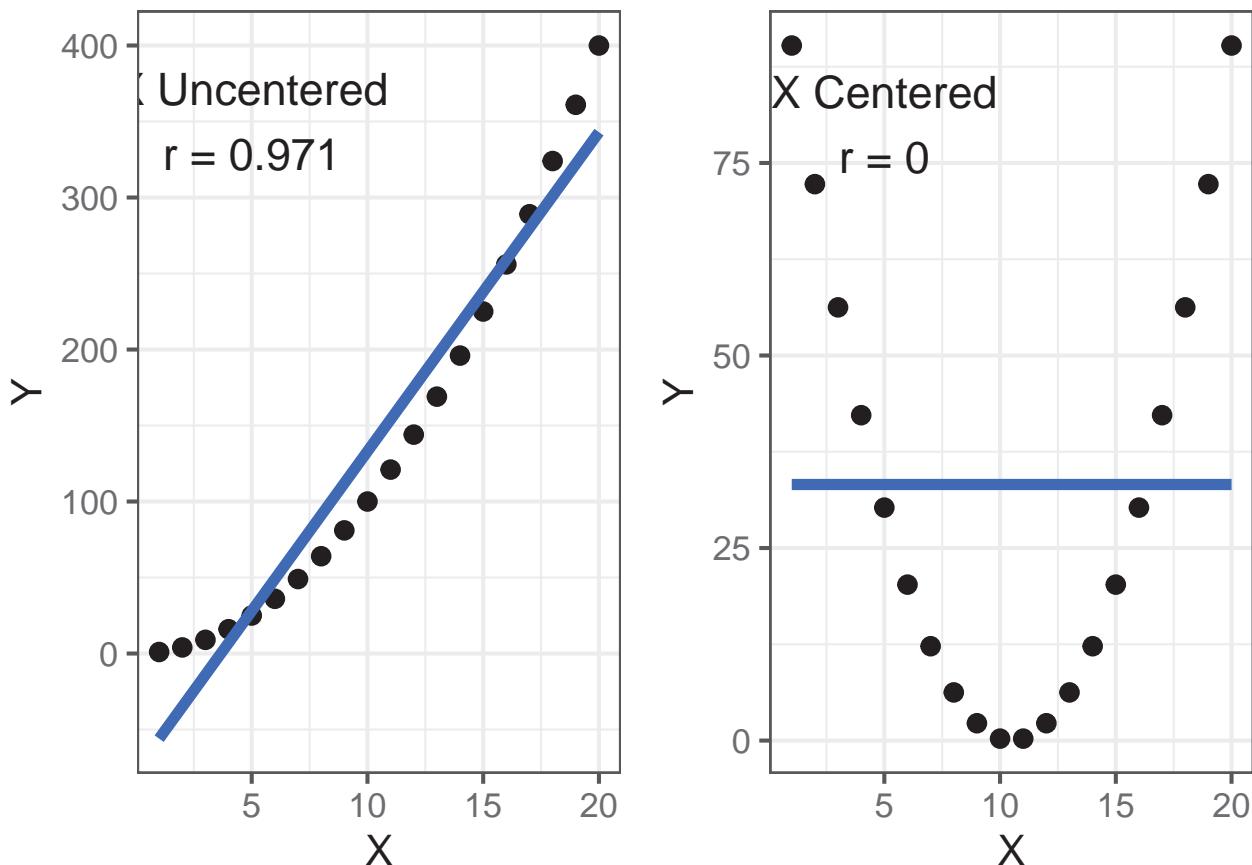


Figure 8.7: Centering a predictor removes the necessary correlation in a quadratic regression

```
Acetylene.centered <-  
  Acetylene |>  
  mutate(temp = temp - mean(temp),  
         time = time - mean(time),  
         ratio = ratio - mean(ratio))  
  
acetyl.mod1 <- update(acetyl.mod0, data=Acetylene.centered)  
(acetyl.vif1 <- vif(acetyl.mod1))  
#>      temp          ratio          time    I(time^2)  temp:time temp:ratio  
#>  57.09        1.09     81.57      51.49      44.67     30.69  
#> ratio:time  
#>      33.33
```

This is far better, although still not great in terms of VIF. But, how much have we improved the situation by the simple act of centering the predictors? The square roots of the ratios of VIFs tell us the impact of centering on the standard errors.

```
sqrt(acetyl.vif0 / acetyl.vif1)  
#>      temp          ratio          time    I(time^2)  temp:time temp:ratio  
#>  2.59        98.24     14.89      3.31      14.75     17.77  
#> ratio:time  
#>      2.60
```

Finally, we use `poly(time, 2)` in the model for the centered data. Because there are multiple degree of freedom terms in the model, `car::vif()` calculates GVIFs here. The final column gives $\sqrt{\text{GVIF}^{1/2\text{df}}}$, the remaining effect of collinearity on the standard errors of terms in this model.

```
acetyl.mod2 <- lm(yield ~ temp + ratio + poly(time, 2) +
                     temp:time + temp:ratio + time:ratio,
                     data=Acetylene.centered)

vif(acetyl.mod2, type = "term")
#>           GVIF Df GVIF^(1/(2*Df))
#> temp      57.09  1    7.56
#> ratio     1.09  1    1.05
#> poly(time, 2) 1733.56  2    6.45
#> temp:time   44.67  1    6.68
#> temp:ratio   30.69  1    5.54
#> ratio:time   33.33  1    5.77
```

8.6 Ridge regression

Ridge regression is an instance of a class of techniques designed to obtain more favorable predictions at the expense of some increase in bias, compared to ordinary least squares (OLS) estimation. These methods began as a way of solving collinearity problems in OLS regression with highly correlated predictors (Hoerl & Kennard, 1970). More recently ridge regression developed to a larger class of model selection methods, of which the LASSO method of Tibshirani (1996) and LAR method of Efron et al. (2004) are well-known instances. See, for example, the reviews in Vinod (1978) and McDonald (2009) for details and context omitted here. The case of ridge regression has also been extended to the case of two or more response variables (P. J. Brown & Zidek, 1980; Haitovsky, 1987).

An essential idea behind these methods is that the OLS estimates are constrained in some way, shrinking them, on average, toward zero, to achieve increased predictive accuracy at the expense of some increase in bias. Another common characteristic is that they involve some tuning parameter (k) or criterion to quantify the tradeoff between bias and variance. In many cases, analytical or computationally intensive methods have been developed to choose an optimal value of the tuning parameter, for example using generalized cross validation, bootstrap methods.

A common means to visualize the effects of shrinkage in these problems is to make what are called *univariate ridge trace plots* (Section 8.7) showing how the estimated coefficients $\hat{\beta}_k$ change as the shrinkage criterion k increases. (An example is shown in Fig XX below.) But this only provides a view of bias. It is the wrong graphic form for a multivariate problem where we want to visualize bias in the coefficients $\hat{\beta}_k$ vs. their precision, as reflected in their estimated variances, $\widehat{\text{Var}}(\hat{\beta}_k)$. A more useful graphic plots the confidence ellipses for the coefficients, showing both bias and precision (Section 8.8). Some of the material below borrows from Friendly (2011) and Friendly (2013).

8.6.1 Properties of ridge regression

To provide some context, I summarize the properties of ridge regression below, comparing the OLS estimates with their ridge counterparts. To avoid unnecessary details related to the intercept, assume the predictors have been centered at their means and the unit vector is omitted from \mathbf{X} . Further, to avoid scaling issues, we standardize the columns of \mathbf{X} to unit length, so that $\mathbf{X}^\top \mathbf{X}$ is a also correlation matrix.

The ordinary least squares estimates of coefficients and their estimated variance covariance matrix take the (hopefully now) familiar form

$$\begin{aligned}\hat{\beta}^{\text{OLS}} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \\ \widehat{\text{Var}}(\hat{\beta}^{\text{OLS}}) &= \hat{\sigma}_\epsilon^2 (\mathbf{X}^\top \mathbf{X})^{-1}.\end{aligned}$$

{#eq-OLS-beta-var}

As we saw earlier, one signal of the problem of collinearity is that the determinant $\det(\mathbf{X}^\top \mathbf{X})$ approaches zero as the predictors become more collinear. The inverse $(\mathbf{X}^\top \mathbf{X})^{-1}$ becomes numerically unstable, or does not exist if the determinant becomes zero in the case of exact dependency of one variable on the others.

Ridge regression uses a trick to avoid this. It adds a constant, k to the diagonal elements, replacing $\mathbf{X}^\top \mathbf{X}$ with $\mathbf{X}^\top \mathbf{X} + k\mathbf{I}$ in ?@eq-OLS-beta-var. This drives the determinant away from zero as k increases. The ridge regression estimates then become,

$$\begin{aligned}\hat{\beta}_k^{\text{RR}} &= (\mathbf{X}^\top \mathbf{X} + k\mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \\ &= \mathbf{G}_k \hat{\beta}^{\text{OLS}}, \\ \widehat{\text{Var}}(\hat{\beta}_k^{\text{RR}}) &= \hat{\sigma}^2 \mathbf{G}_k (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{G}_k^\top,\end{aligned}$$

{#eq-ridge-beta-var}

where $\mathbf{G}_k = [\mathbf{I} + k(\mathbf{X}^\top \mathbf{X})^{-1}]^{-1}$ is the $(p \times p)$ *shrinkage* matrix. Thus, as k increases, \mathbf{G}_k decreases, and drives $\hat{\beta}_k^{\text{RR}}$ toward $\mathbf{0}$ (Hoerl & Kennard, 1970).

Another insight, from the shrinkage literature, is that ridge regression can be formulated as least squares regression, minimizing a residual sum of squares, $\text{RSS}(k)$, which adds a penalty for large coefficients,

$$\text{RSS}(k) = (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta) + k\beta^\top \beta \quad (k \geq 0), \quad (8.2)$$

where the penalty restrict the coefficients to some squared length $\beta^\top \beta = \Sigma \beta_i^2 \leq t(k)$.

The geometry of ridge regression is illustrated in Figure 8.8 for two coefficients $\beta = (\beta_1, \beta_2)$. The blue circles at the origin, having radii $\sqrt{t_k}$, show the constraint that the sum of squares of coefficients, $\beta^\top \beta = \beta_1^2 + \beta_2^2$ be less than k . The red ellipses show contours of the covariance ellipse of $\hat{\beta}^{\text{OLS}}$. As the shrinkage constant k increases, the center of these ellipses travel along the path illustrated toward $\beta = \mathbf{0}$. This path is called the *locus of osculation*, the path along which circles or ellipses first kiss as they expand, like the pattern of ripples from rocks dropped into a pond (Friendly et al., 2013).

?@eq-ridge-beta-var is computationally expensive, potentially numerically unstable for small k , and it is conceptually opaque, in that it sheds little light on the underlying geometry of the data in the column space of \mathbf{X} . An alternative formulation can be given in terms of the singular value decomposition (SVD) of \mathbf{X} ,

$$\mathbf{X} = \mathbf{UDV}^\top$$

where \mathbf{U} and \mathbf{V} are respectively $n \times p$ and $p \times p$ orthonormal matrices, so that $\mathbf{U}^\top \mathbf{U} = \mathbf{V}^\top \mathbf{V} = \mathbf{I}$, and $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_p)$ is the diagonal matrix of ordered singular values, with entries $d_1 \geq d_2 \geq \dots \geq d_p \geq 0$.

Because $\mathbf{X}^\top \mathbf{X} = \mathbf{V} \mathbf{D}^2 \mathbf{V}^\top$, the eigenvalues of $\mathbf{X}^\top \mathbf{X}$ are given by \mathbf{D}^2 and therefore the eigenvalues of \mathbf{G}_k can be shown (Hoerl & Kennard, 1970) to be the diagonal elements of

$$\mathbf{D}(\mathbf{D}^2 + k\mathbf{I})^{-1} \mathbf{D} = \text{diag} \left(\frac{d_i^2}{d_i^2 + k} \right).$$

Noting that the eigenvectors, \mathbf{V} are the principal component vectors, and that $\mathbf{X}\mathbf{V} = \mathbf{UD}$, the ridge estimates can be calculated more simply in terms of \mathbf{U} and \mathbf{D} as

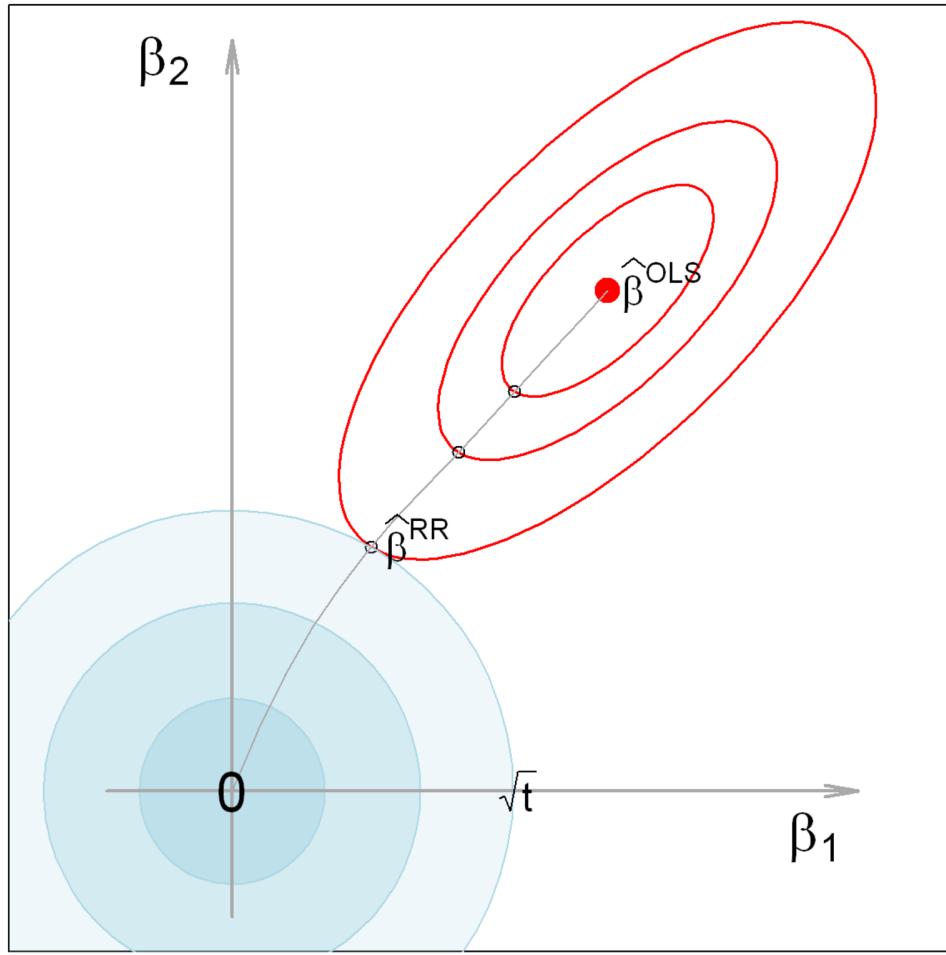


Figure 8.8: Geometric interpretation of ridge regression, using elliptical contours of the $\text{RSS}(k)$ function. The blue circles at the origin show the constraint that the sum of squares of coefficients, $\beta^\top \beta$ be less than k . The red ellipses show the covariance ellipse of two coefficients β . Ridge regression finds the point $\hat{\beta}_k^{\text{RR}}$ where the OLS contours just kiss the constraint region. *Source:* Friendly et al. (2013).

$$\hat{\beta}_k^{\text{RR}} = (\mathbf{D}^2 + k\mathbf{I})^{-1} \mathbf{D} \mathbf{U}^\top \mathbf{y} = \left(\frac{d_i}{d_i^2 + k} \right) \mathbf{u}_i^\top \mathbf{y}, \quad i = 1, \dots, p.$$

The terms $d_i^2/(d_i^2 + k) \leq 1$ are thus the factors by which the coordinates of $\mathbf{u}_i^\top \mathbf{y}$ are shrunk with respect to the orthonormal basis for the column space of \mathbf{X} . The small singular values d_i correspond to the directions which ridge regression shrinks the most. These are the directions which contribute most to collinearity, discussed earlier.

This analysis also provides an alternative and more intuitive characterization of the ridge tuning constant. By analogy with OLS, where the hat matrix, $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ reflects degrees of freedom $\text{df} = \text{tr}(\mathbf{H}) = p$ corresponding to the p parameters, the effective degrees of freedom for ridge regression (Hastie et al., 2009) is

$$\begin{aligned} \text{df}_k &= \text{tr}[\mathbf{X}(\mathbf{X}^T \mathbf{X} + k\mathbf{I})^{-1} \mathbf{X}^T] \\ &= \sum_i^p \text{df}_k(i) = \sum_i^p \left(\frac{d_i^2}{d_i^2 + k} \right) . \end{aligned}$$

{#eq-dfk}

df_k is a monotone decreasing function of k , and hence any set of ridge constants can be specified in terms of equivalent df_k . Greater shrinkage corresponds to fewer coefficients being estimated.

There is a close connection with principal components regression mentioned in Section 8.5. Ridge regression shrinks *all* dimensions in proportion to $\text{df}_k(i)$, so the low variance dimensions are shrunk more. Principal components regression discards the low variance dimensions and leaves the high variance dimensions unchanged.

8.6.2 The genridge package

Ridge regression and other shrinkage methods are available in several packages including **MASS** (the `lm.ridge()` function), **glmnet** (Friedman et al., 2025), and **penalized** (Goeman et al., 2022), but none of these provides insightful graphical displays. `glmnet::glmnet()` also implements a method for multivariate responses with a ‘family=“mgaussian”’.

Here, I focus in the **genridge** package (Friendly, 2024), where the `ridge()` function is the workhorse and `pca.ridge()` transforms these results to PCA/SVD space. `vif.ridge()` calculates VIFs for class “ridge” objects and `precision()` calculates precision and shrinkage measures.

A variety of plotting functions is available for univariate, bivariate and 3D plots:

- `traceplot()` Traditional univariate ridge trace plots
- `plot.ridge()` Bivariate 2D ridge trace plots, showing the covariance ellipse of the estimated coefficients
- `pairs.ridge()` All pairwise bivariate ridge trace plots
- `plot3d.ridge()` 3D ridge trace plots with ellipsoids
- `biplot.ridge()` ridge trace plots in PCA/SVD space

In addition, the `pca()` method for “ridge” objects transforms the coefficients and covariance matrices of a ridge object from predictor space to the equivalent, but more interesting space of the PCA of $\mathbf{X}^T \mathbf{X}$ or the SVD of \mathbf{X} . `biplot.pcaridge()` adds variable vectors to the bivariate plots of coefficients in PCA space

8.7 Univariate ridge trace plots

A classic example for ridge regression is Longley’s (1967) data, consisting of 7 economic variables, observed yearly from 1947 to 1962 ($n=16$), in the dataset `longley`. The goal is to predict Employed from GNP, Unemployed, Armed.Forces, Population, Year, and GNP.deflator.

```
data(longley, package="datasets")
str(longley)
#> 'data.frame':   16 obs. of  7 variables:
#> $ GNP.deflator: num  83 88.5 88.2 89.5 96.2 ...
#> $ GNP          : num  234 259 258 285 329 ...
#> $ Unemployed   : num  236 232 368 335 210 ...
#> $ Armed.Forces: num  159 146 162 165 310 ...
#> $ Population    : num  108 109 110 111 112 ...
```

```
#> $ Year      : int  1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 ...
#> $ Employed  : num  60.3 61.1 60.2 61.2 63.2 ...
```

These data were constructed to illustrate numerical problems in least squares software at the time, and they are (purposely) perverse, in that:

- Each variable is a time series so that there is clearly a lack of independence among predictors. `Year` is at least implicitly correlated with most of the others.
- Worse, there is also some structural collinearity among the variables `GNP`, `Year`, `GNP.deflator`, and `Population`; for example, `GNP.deflator` is a multiplicative factor to account for inflation.

We fit the regression model, and sure enough, there are some extremely large VIFs. The largest, for `GNP` represents a multiplier of $\sqrt{1788.5} = 42.3$ on the standard errors.

```
longley.lm <- lm(Employed ~ GNP + Unemployed + Armed.Forces +
                    Population + Year + GNP.deflator,
                    data=longley)
vif(longley.lm)
#>          GNP    Unemployed Armed.Forces    Population       Year
#> 1788.51        33.62       3.59     399.15     758.98
#> GNP.deflator
#>      135.53
```

Shrinkage values can be specified using k (where $k = 0$ corresponds to OLS) or the equivalent degrees of freedom `$ df_k$` ([?@eq-dfk](#)). (The function uses the notation $\lambda \equiv k$, so the argument is `lambda`.) Among other quantities, `ridge()` returns a matrix containing the coefficients for each predictor for each shrinkage value and other quantities.

```
lambda <- c(0, 0.002, 0.005, 0.01, 0.02, 0.04, 0.08)
lridge <- ridge(Employed ~ GNP + Unemployed + Armed.Forces +
                  Population + Year + GNP.deflator,
                  data=longley, lambda=lambda)
print(lridge, digits = 2)
#> Ridge Coefficients:
#>          GNP    Unemployed Armed.Forces    Population       Year
#> 0.000 -3.447   -1.828    -0.696    -0.344     8.432
#> 0.002 -2.114   -1.644    -0.658    -0.713     7.466
#> 0.005 -1.042   -1.491    -0.623    -0.936     6.567
#> 0.010 -0.180   -1.361    -0.588    -1.003     5.656
#> 0.020  0.499   -1.245    -0.548    -0.868     4.626
#> 0.040  0.906   -1.155    -0.504    -0.523     3.577
#> 0.080  1.091   -1.086    -0.458    -0.086     2.642
#>          GNP.deflator
#> 0.000   0.157
#> 0.002   0.022
#> 0.005  -0.042
#> 0.010  -0.026
#> 0.020   0.098
#> 0.040   0.321
#> 0.080   0.570
```

The standard univariate plot, given by `traceplot()`, simply plots the estimated coefficients for each predictor against the shrinkage factor k .

```
traceplot(lridge,
  X = "lambda",
  xlab = "Ridge constant (k)",
  xlim = c(-0.02, 0.08), cex.lab=1.25)
```

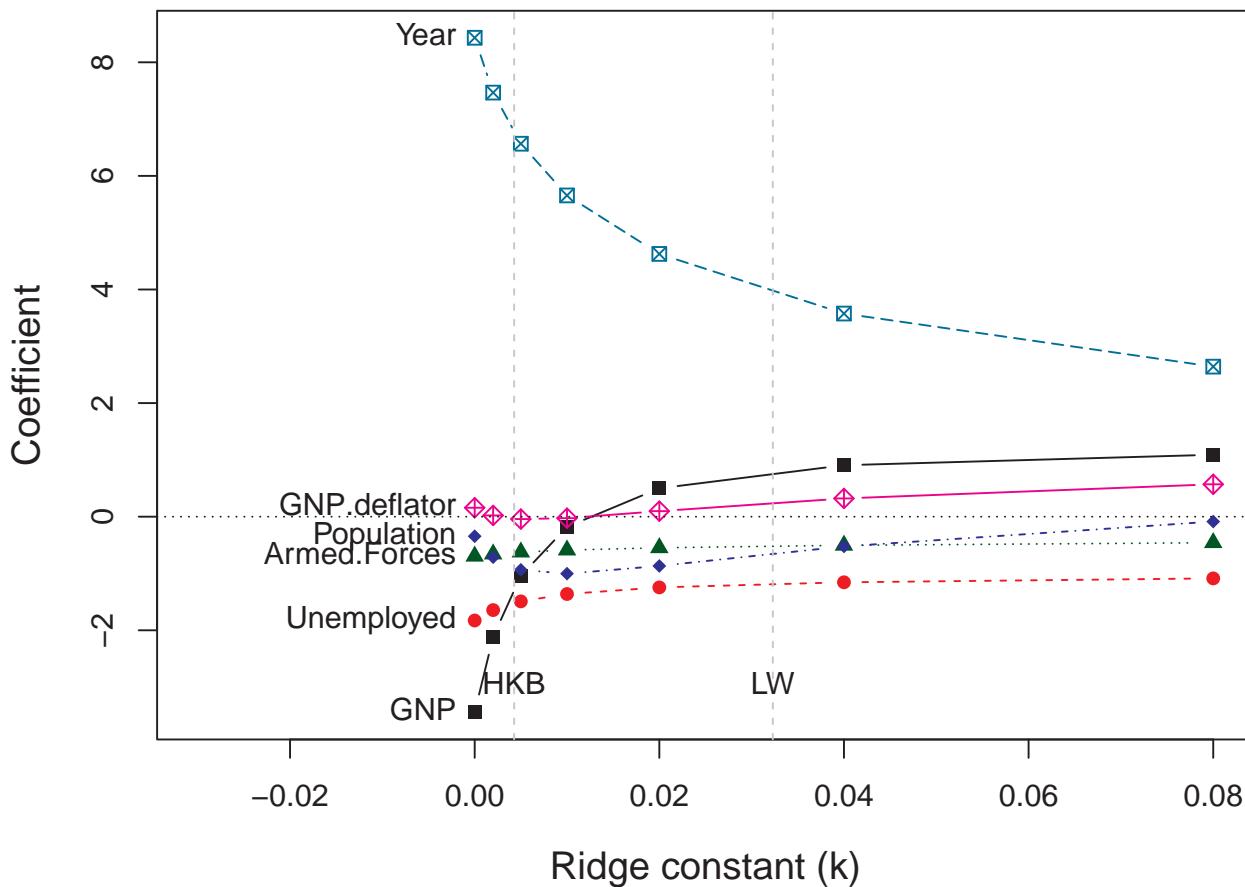


Figure 8.9: Univariate ridge trace plot for the coefficients of predictors of Employment in Longley's data via ridge regression, with ridge constants $k = (0, 0.002, 0.005, 0.01, 0.02, 0.04, 0.08)$. The dotted lines show optimal values for shrinkage by two criteria (HKB, LW).

You can see that the coefficients for Year and GNP are shrunk considerably. Differences from the β value at $k = 0$ represent the bias (smaller $|\beta|$) needed to achieve more stable estimates.

The dotted lines in Figure 8.9 show choices for the ridge constant by two commonly used criteria to balance bias against precision due to Hoerl et al. (1975) (HKB) and Lawless & Wang (1976) (LW). These values (along with a generalized cross-validation value GCV) are also stored in the “ridge” object as a vector `criteria`.

```
lridge$criteria
#>   kHKB      kW      kGCV
#> 0.00428 0.03230 0.00200
```

The shrinkage constant k doesn't have much intrinsic meaning, so it is often easier to interpret the plot when coefficients are plotted against the equivalent degrees of freedom, df_k . OLS corresponds to $df_k = 6$ degrees of freedom in the space of six parameters, and the effect of shrinkage is to decrease the degrees of freedom, as if estimating fewer parameters. This more natural scale also makes the changes in coefficient with shrinkage more nearly linear.

```
traceplot(lridge,
          X = "df",
          xlim = c(4, 6.2), cex.lab=1.25)
```

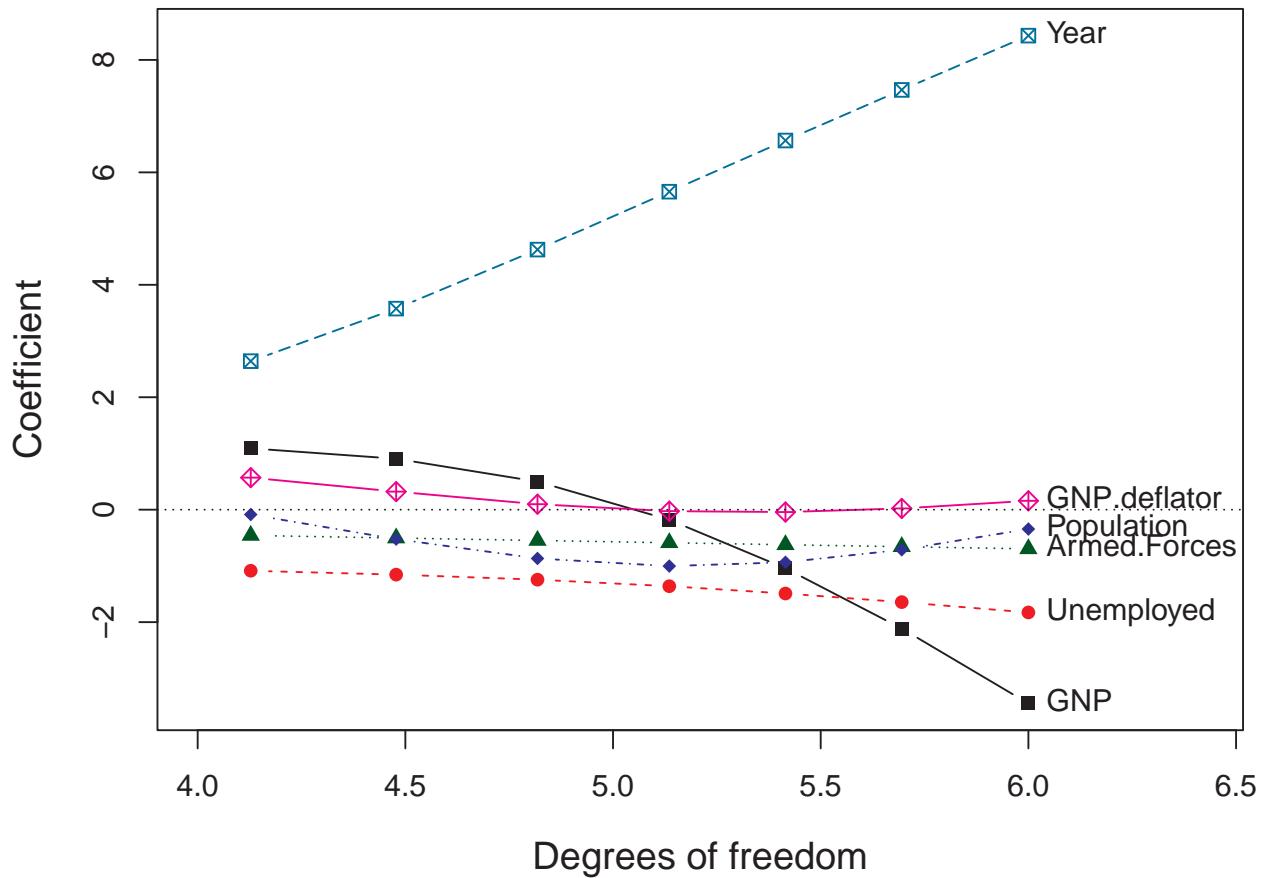


Figure 8.10: Univariate ridge trace plot using equivalent degrees of freedom, df_k to specify shrinkage. This scale is easier to understand and makes the traces of parameters more nearly linear.

But a bigger problem is that these univariate plots are the **wrong kind** of plot! They show the trends in increased bias (toward smaller $|\beta|$) associated with larger k , but they do not show the accompanying increase in *precision* (decrease in variance) achieved by allowing a bit of bias.

For that, we need to consider the variances and covariances of the estimated coefficients. The univariate trace plot is the wrong graphic form for what is essentially a multivariate problem, where we would like to visualize how *both* coefficients and their variances change with k .

8.8 Bivariate ridge trace plots

The bivariate analog of the trace plot suggested by Friendly (2013) plots bivariate confidence ellipses for *pairs* of coefficients. Their centers, $(\hat{\beta}_i, \hat{\beta}_j)$ compared to the OLS values show the bias induced for each coefficient, and also how the change in the ridge estimate for one parameter is related to changes for other parameters.

The size and shapes of the covariance ellipses show directly the effect on precision of the estimates as a

function of the ridge tuning constant. and their size and shape indicate sampling variance, $\widehat{\text{Var}}(\widehat{\beta}_{ij})$. Here, I plot those for GNP against four of the other predictors. The `plot()` method for "ridge" objects plots these ellipses for a pair of variables.

```
clr <- c("black", "red", "brown", "darkgreen", "blue", "cyan4", "magenta")
pch <- c(15:18, 7, 9, 12)
lambdaaf <- c(expression(~widehat(beta)^OLS), as.character(lambda[-1]))

for (i in 2:5) {
  plot(lridge, variables=c(1,i),
    radius=0.5, cex.lab=1.5, col=clr,
    labels=NULL, fill=TRUE, fill.alpha=0.2)
  text(lridge$coef[1,1], lridge$coef[1,i],
    expression(~widehat(beta)^OLS), cex=1.5, pos=4, offset=.1)
  text(lridge$coef[-1,c(1,i)], lambdaaf[-1], pos=3, cex=1.3)
}
```

As can be seen, the coefficients for each pair of predictors trace a path generally in toward the origin (0,0), and the covariance ellipses get smaller, indicating increased precision. Sometimes, these paths are rather direct, but it takes a peculiar curvilinear route in the case of population and GNP.

The `pairs()` method for "ridge" objects shows all pairwise views in scatterplot matrix form. `radius` sets the base size of the ellipse-generating circle for the covariance ellipses.

```
pairs(lridge, radius=0.5, diag.cex = 2,
      fill = TRUE, fill.alpha = 0.1)
```

8.8.1 Visualizing the bias-variance tradeoff

The function `precision()` calculates a number of measures of the effect of shrinkage of the coefficients in relation to the “size” of the covariance matrix $\mathcal{V}_k \equiv \widehat{\text{Var}}(\widehat{\beta}_k^{\text{RR}})$. Larger shrinkage k should lead to a smaller ellipsoid for \mathcal{V}_k , indicating increased precision.

```
pdat <- precision(lridge) |> print()
#>   lambda  df  det  trace max.eig norm.beta norm.diff
#> 0.000  0.000 6.00 -12.9 18.119  15.419     1.000     0.000
#> 0.002  0.002 5.70 -13.6 11.179   8.693     0.857     0.695
#> 0.005  0.005 5.42 -14.4  6.821   4.606     0.741     1.276
#> 0.010  0.010 5.14 -15.4  4.042   2.181     0.637     1.783
#> 0.020  0.020 4.82 -16.8  2.218   1.025     0.528     2.262
#> 0.040  0.040 4.48 -18.7  1.165   0.581     0.423     2.679
#> 0.080  0.080 4.13 -21.1  0.587   0.260     0.337     3.027
```

Here, the first three terms described below are (inverse) measures of precision; the last two quantify shrinkage:

- `det` = $\log |\mathcal{V}_k|$ is an overall measure of variance of the coefficients. It is the (linearized) volume of the covariance ellipsoid and corresponds conceptually to Wilks' Lambda criterion.
- `trace` = $\text{trace}(\mathcal{V}_k)$ is the sum of the variances and also the sum of the eigenvalues of \mathcal{V}_k , conceptually similar to Pillai's trace criterion.
- `max.eig` is the largest eigenvalue measure of size, an analog of Roy's maximum root test.
- `norm.beta` = $\|\beta\| / \max \|\beta\|$ is a summary measure of shrinkage, the normalized root mean square of the estimated coefficients. It starts at 1.0 for $k = 0$ and decreases with the penalty for large coefficients.

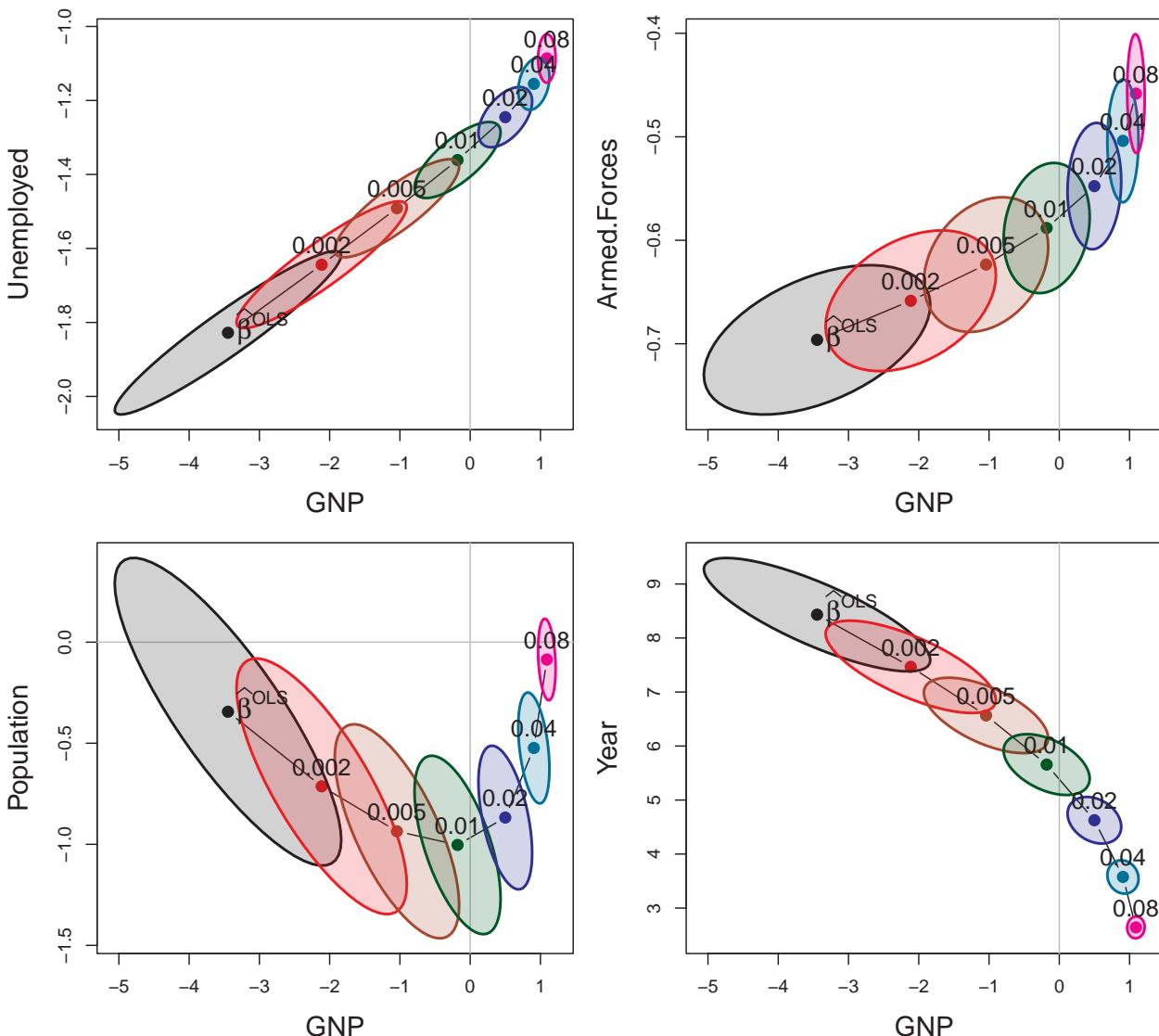


Figure 8.11: Bivariate ridge trace plots for the coefficients of four predictors against the coefficient for GNP in Longley's data, with $k = 0, 0.002, 0.005, 0.01, 0.02, 0.04, 0.08$. In most cases, the coefficients are driven toward zero, but the bivariate plot also makes clear the reduction in variance, as well as the bivariate path of shrinkage.

- `diff.beta` is the root mean square of the difference from the OLS estimate $\|\beta_{OLS} - \beta_k\|$. This measure is inversely related to `norm.beta`.

Plotting shrinkage against a measure of variance gives a direct view of the tradeoff between bias and precision. Here I plot `norm.beta` against `det`, and join the points with a curve. You can see that in this example the HKB criterion prefers a smaller degree of shrinkage, but achieves only a modest decrease in variance. But variance decreases more sharply thereafter and the LW choice achieves greater precision.

```
library(splines)
with(pdat, {
  plot(norm.beta, det, type="b",
    main="Bivariate Ridge Trace Plot for GNP Shrinkage")})
```

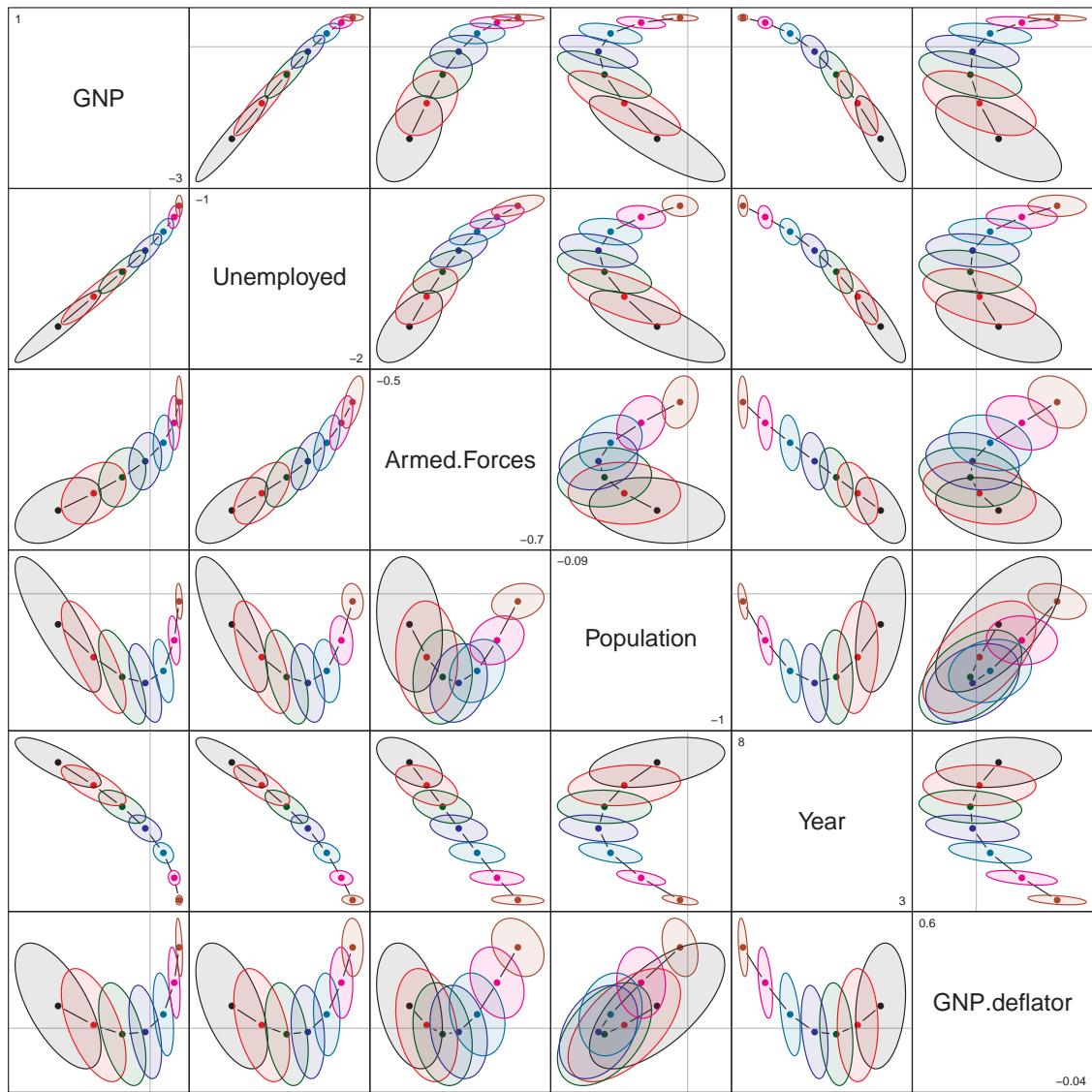


Figure 8.12: Scatterplot matrix of bivariate ridge trace plots. Each panel shows the effect of shrinkage on the covariance ellipse for a pair of predictors.

```

cex.lab=1.25, pch=16,
cex=1.5, col=clr, lwd=2,
xlab='shrinkage: ||b|| / max(||b||)',
ylab='variance: log |Var(b)|')
text(norm.beta, det,
      labels = lambdaf,
      cex = 1.25,
      pos = c(rep(2,length(lambda)-1),4))
text(min(norm.beta), max(det),
      labels = "log |Variance| vs. Shrinkage",
      cex=1.5, pos=4)
})

```

```
# find locations for optimal shrinkage criteria
mod <- lm(cbind(det, norm.beta) ~ bs(lambda, df=5),
           data=pdat)
x <- data.frame(lambda=c(lridge$kHKB,
                         lridge$kLW))
fit <- predict(mod, x)
points(fit[,2:1], pch=15,
       col=gray(.50), cex=1.6)
text(fit[,2:1], c("HKB", "LW"),
     pos=3, cex=1.5, col=gray(.50))
```

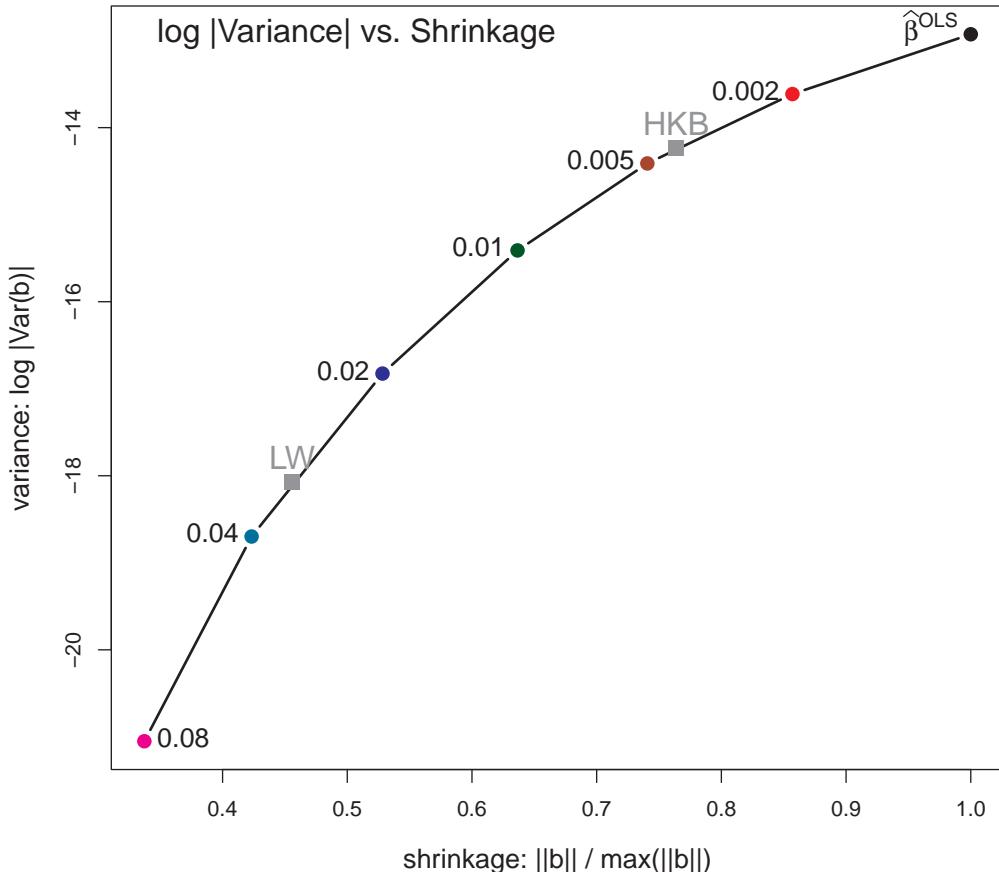


Figure 8.13: The tradeoff between bias and precision. Bias increases as we move away from the OLS solution, but precision increases.

8.9 Low-rank views

Just as principal components analysis gives low-dimensional views of a data set, PCA can be useful to understand ridge regression, just as it did for the problem of collinearity.

The `pca` method transforms a "ridge" object from parameter space, where the estimated coefficients are β_k with covariance matrices \mathbf{V}_k , to the principal component space defined by the right singular vectors, \mathbf{V} , of

the singular value decomposition \mathbf{UDV}^T of the scaled predictor matrix, \mathbf{X} . In PCA space the total variance of the predictors remains the same, but it is distributed among the linear combinations that account for successively greatest variance.

```
plridge <- pca(lridge)
plridge
#> Ridge Coefficients:
#>      dim1     dim2     dim3     dim4     dim5     dim6
#> 0.000  1.51541  0.37939  1.80131  0.34595  5.97391  6.74225
#> 0.002  1.51537  0.37935  1.80021  0.34308  5.69497  5.06243
#> 0.005  1.51531  0.37928  1.79855  0.33886  5.32221  3.68519
#> 0.010  1.51521  0.37918  1.79579  0.33205  4.79871  2.53553
#> 0.020  1.51500  0.37898  1.79031  0.31922  4.00988  1.56135
#> 0.040  1.51459  0.37858  1.77944  0.29633  3.01774  0.88291
#> 0.080  1.51377  0.37778  1.75810  0.25915  2.01876  0.47238
```

Then, a `traceplot()` of the resulting "pcaridge" object shows how the dimensions are affected by shrinkage, shown on the scale of degrees of freedom in Figure 8.14.

```
traceplot(plridge, X="df",
          cex.lab = 1.2, lwd=2)
```

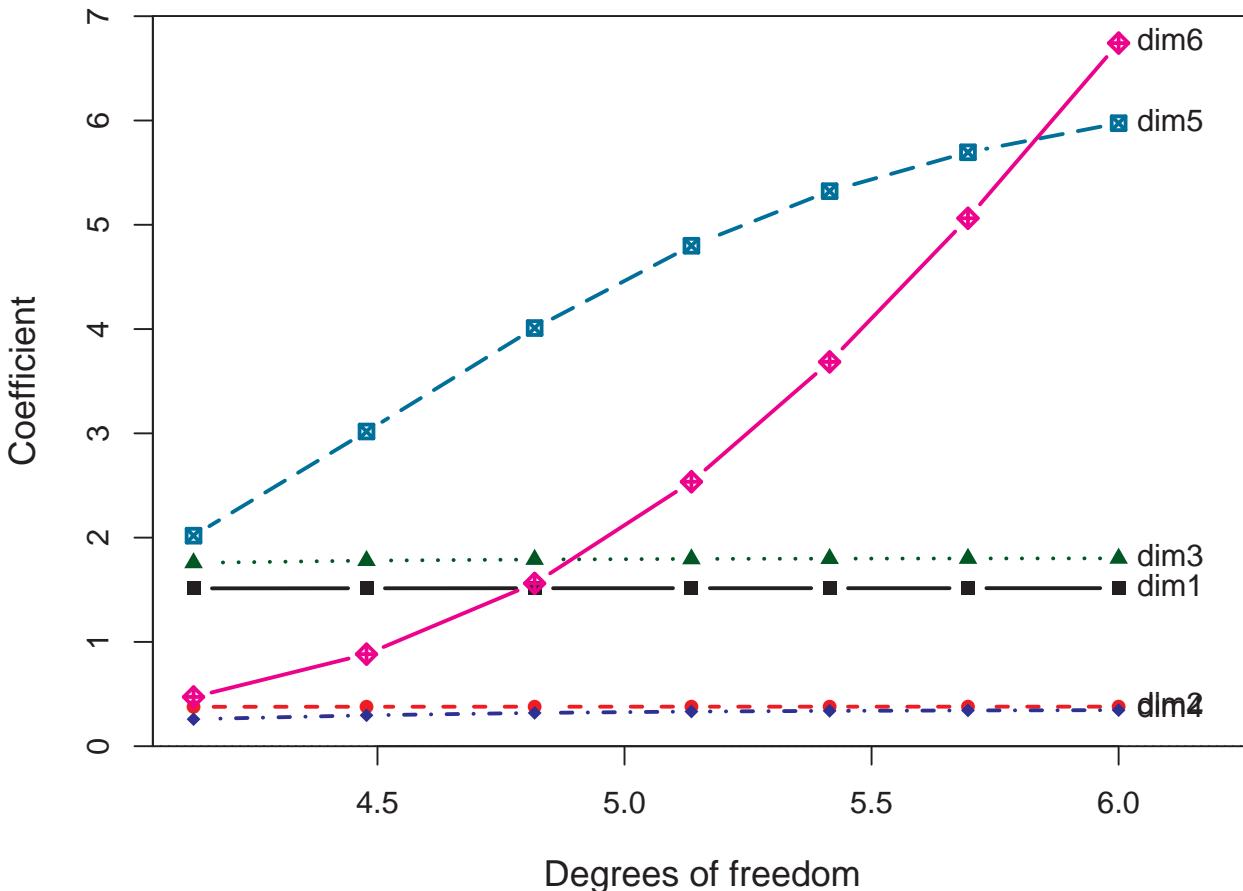


Figure 8.14: Ridge traceplot for the longley regression viewed in PCA space. The dimensions are the linear combinations of the predictors which account for greatest variance.

What may be surprising at first is that the coefficients for the first 4 components are not shrunk at all. These large dimensions are immune to ridge tuning. Rather, the effect of shrinkage is seen only on the *last two dimensions*. But those also are the directions that contribute most to collinearity as we saw earlier.

A `pairs()` plot gives a dramatic representation bivariate effects of shrinkage in PCA space: the principal components of X are uncorrelated, so the ellipses are all aligned with the coordinate axes and the ellipses largely coincide for dimensions 1 to 4. You can see them shrink in one direction in the last two columns and rows.

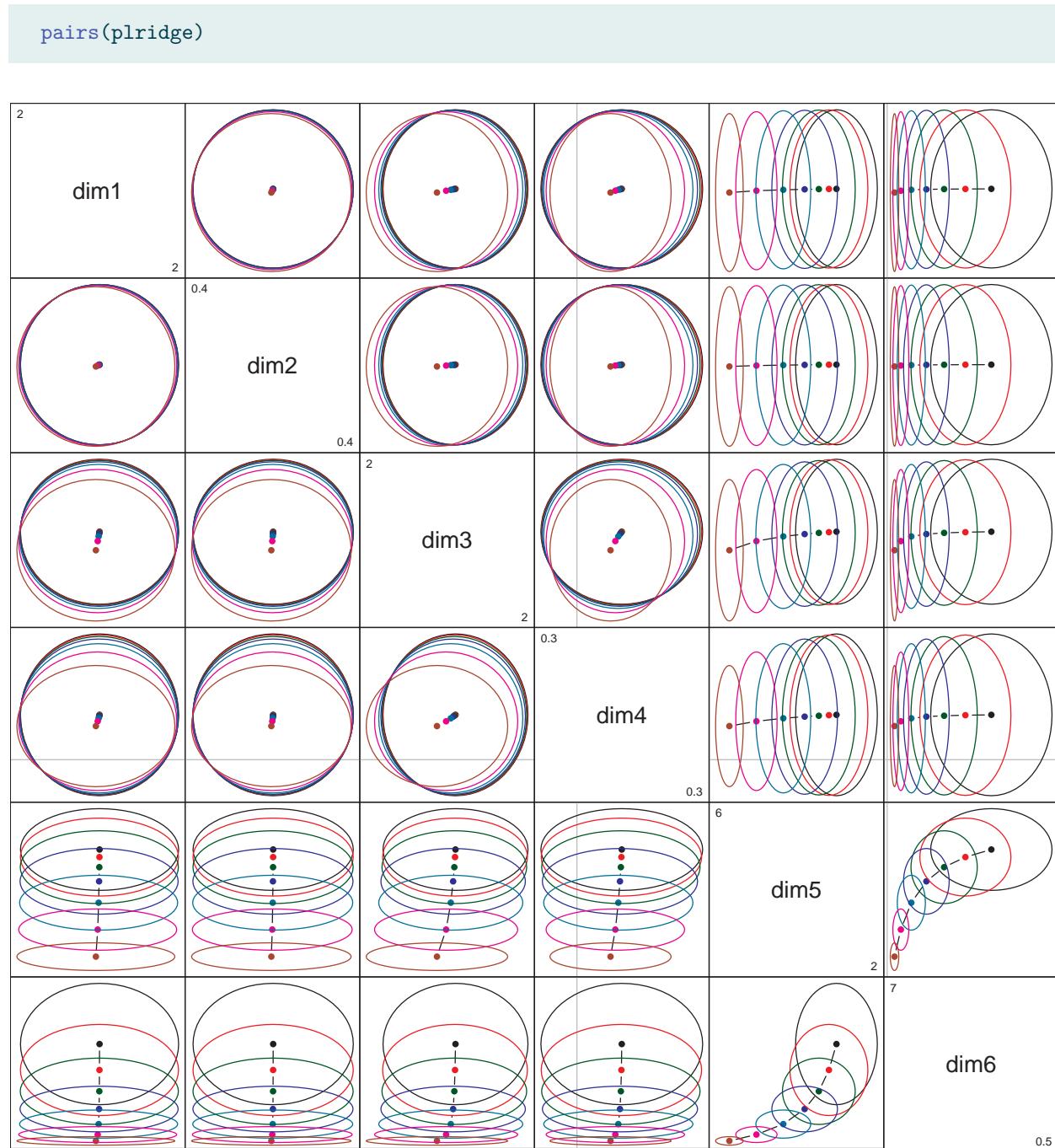


Figure 8.15: All pairwise bivariate ridge plots shown in PCA space.

If we focus on the plot of dimensions 5:6, we can see where all the shrinkage action is in this representation. Generally, the predictors that are related to the smallest dimension (6) are shrunk quickly at first.

```
plot(plridge, variables=5:6,
      fill = TRUE, fill.alpha=0.15, cex.lab = 1.5)
text(plridge$coef[, 5:6],
     label = lambdaef,
     cex=1.5, pos=4, offset=.1)
```

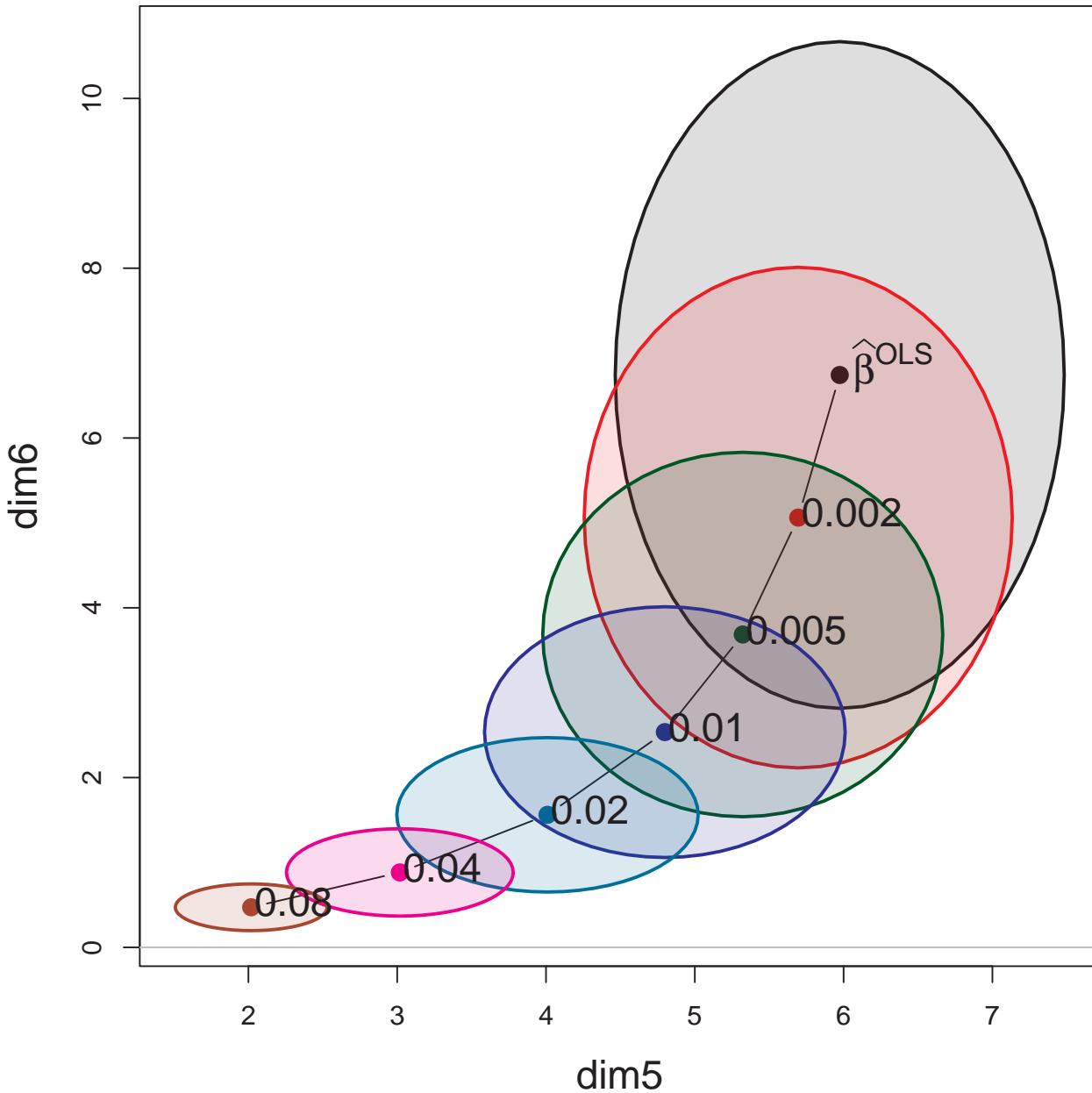


Figure 8.16: Bivariate ridge trace plot for the smallest two dimensions ...

8.9.1 Biplot view

The question arises how to relate this view of shrinkage in PCA space to the original predictors. The biplot is again your friend. You can project variable vectors for the predictor variables into the PCA space of the smallest dimensions, where the shrinkage action mostly occurs to see how the predictor variables relate to these dimensions.

`biplot.pcaridge()` supplements the standard display of the covariance ellipsoids for a ridge regression problem in PCA/SVD space with labeled arrows showing the contributions of the original variables to the dimensions plotted. Recall from Section 4.3 that these reflect the correlations of the variables with the PCA dimensions. The lengths of the arrows reflect the proportion of variance that each predictors shares with the components.

```
biplot(plridge, radius=0.5,
       ref=FALSE, asp=1,
       var.cex=1.15, cex.lab=1.3, col=clr,
       fill=TRUE, fill.alpha=0.15,
       prefix="Dimension ")
#> Vector scale factor set to 5.25
text(plridge$coef[,5:6], lambdaef, pos=2, cex=1.3)
```

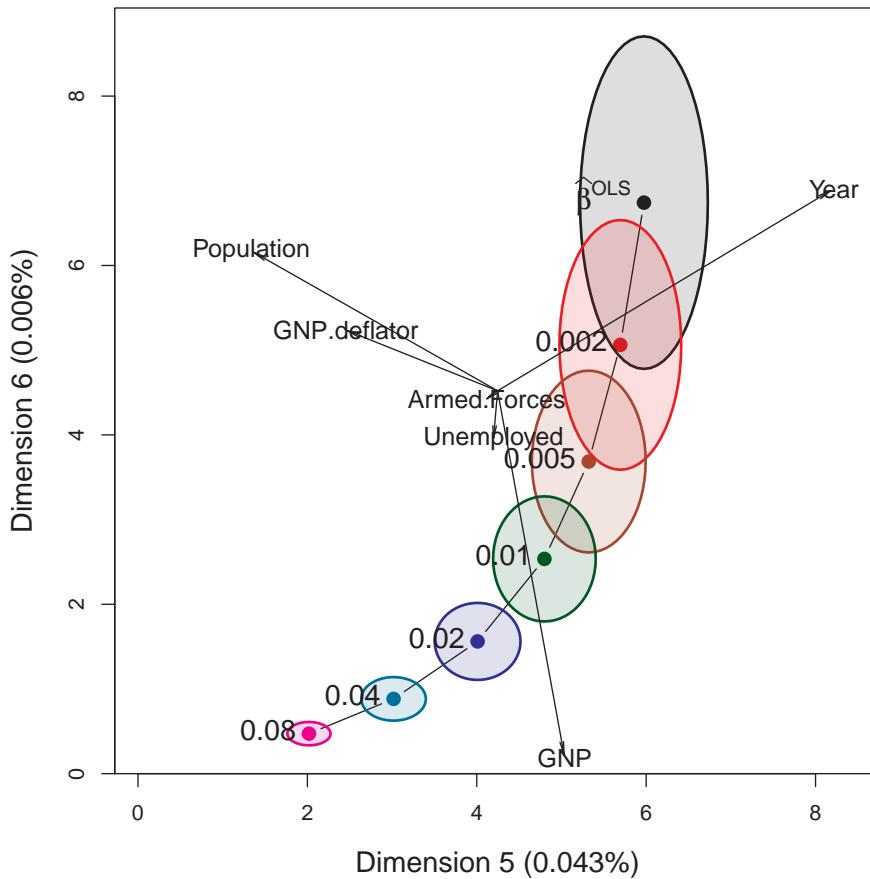


Figure 8.17: Biplot view of the ridge trace plot for the smallest two dimensions, where the effects of shrinkage are most apparent.

The biplot view in Figure 8.17 showing the two smallest dimensions is particularly useful for understanding

how the predictors contribute to shrinkage in ridge regression. Here, Year and Population largely contribute to dimension 5; a contrast between (Year, Population) and GNP contributes to dimension 6.

8.10 What have we learned?

This chapter has considered the problems in regression models which stem from high correlations among the predictors. We saw that collinearity results in unstable estimates of coefficients with larger uncertainty, often dramatically more so than would be the case if the predictors were uncorrelated. Collinearity can be seen as merely a “data problem” which can safely be ignored if we are only interested in prediction. When we want to understand a model, ridge regression can tame the collinearity beast by shrinking the coefficients slightly to gain greater precision in the estimates.

Beyond these statistical considerations, the methods of this chapter highlight the roles of multivariate thinking and visualization in understanding these phenomena and the methods developed for solving them. Data ellipses and confidence ellipses for coefficients again provide tools for visualizing what is concealed in numerical summaries. A perhaps surprising feature of both collinearity and ridge regression is that the important information usually resides in the smallest PCA dimensions and biplots help again to understand these dimensions.

Part IV

Multivariate Linear Models

9

Hotelling's T^2

Just as the one- and two- sample univariate t -test is the gateway drug for understanding analysis of variance, so too Hotelling's T^2 test provides an entry point to multivariate analysis of variance. This simple case provides an entry point to understanding the collection of methods I call the **HE plot framework** for visualizing effects in multivariate linear models, which are a main focus of this book.

The essential idea is that Hotelling's T^2 provides a test of the difference in means between two groups on a *collection* of variables, $\mathbf{x} = x_1, x_2, \dots, x_p$ *simultaneously*, rather than one by one. This has the advantages that it:

- does not require p -value corrections for multiple tests (e.g., Bonferroni);
- combines the evidence from the multiple response variables, and *pools strength*, accumulating support across measures;
- clarifies how the multiple response are *jointly* related to the group effect along a single dimension, the *discriminant axis*;
- in so doing, it reduces the problem of testing differences for two (and potentially more) response variables to testing the difference on a single variable that best captures the multivariable relations.

After describing it's features, I use an example of a two-group T^2 test to illustrate the basic ideas behind multivariate tests and hypothesis error plots. Then, we'll dip our toes into the visual ideas for representing the statistical quantities involved in such tests.

Packages

In this chapter I use the following packages. Load them now.

```
library(car)
library(heplots)
library(Hotelling)
library(ggplot2)
library(dplyr)
library(tidyr)
library(ggbiplot)
library(broom)
```

9.1 T^2 as a generalized t -test

Hotelling's T^2 ([Hotelling, 1931](#)) is an analog of the square of a univariate t statistic, extended to the case of two or more response variables tested together. Consider the basic one-sample t -test, where we wish to test the hypothesis that the mean \bar{x} of a set of N measures on a test of basic math, with standard deviation s does not differ from an assumed mean $\mu_0 = 150$ for a population. The t statistic for testing $H_0 : \mu = \mu_0$ against the two-sided alternative, $H_0 : \mu \neq \mu_0$ is

$$t = \frac{(\bar{x} - \mu_0)}{s/\sqrt{N}} = \frac{(\bar{x} - \mu_0)\sqrt{N}}{s}$$

Squaring this gives

$$t^2 = \frac{N(\bar{x} - \mu_0)^2}{s^2} = N(\bar{x} - \mu_0)(s^2)^{-1}(\bar{x} - \mu_0)$$

Now consider we also have measures on a test of solving word problems for the same sample. Then, a hypothesis test for the means on basic math (BM) and word problems (WP) is the test of the means of these two variables jointly equal some specified values, say, ($\mu_{0,BM} = 150$, $\mu_{0,WP} = 100$):

$$\mathcal{H}_0 : \mu = \mu_0 = \begin{pmatrix} \mu_{0,BM} \\ \mu_{0,WP} \end{pmatrix} = \begin{pmatrix} 150 \\ 100 \end{pmatrix}$$

Hotelling's T^2 is then the analog of t^2 , with the variance-covariance matrix \mathbf{S} of the scores on (BM, WP) replacing the variance of a single score. This is nothing more than the squared Mahalanobis D_M^2 distance between the sample mean vector $(\bar{x}_{BM}, \bar{x}_{WP})^\top$ and the hypothesized means μ_0 , in the metric of \mathbf{S} , as shown in Figure 9.1.

$$\begin{aligned} T^2 &= N(\bar{\mathbf{x}} - \mu_0)^\top \mathbf{S}^{-1} (\bar{\mathbf{x}} - \mu_0) \\ &= ND_M^2(\bar{\mathbf{x}}, \mu_0) \end{aligned}$$

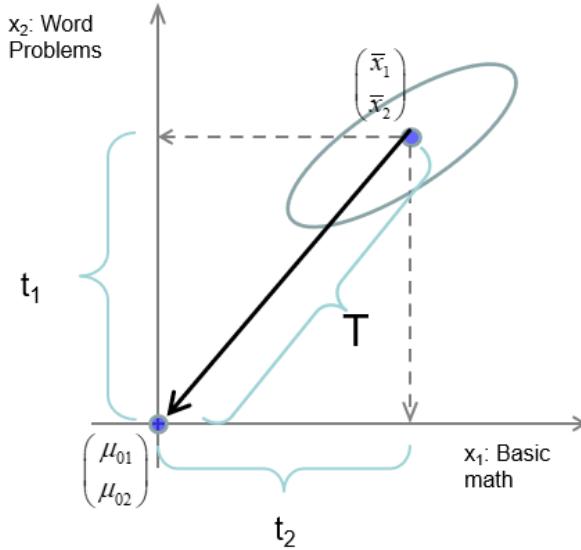


Figure 9.1: Hotelling's T^2 statistic as the squared distance between the sample means and hypothesized means relative to the variance-covariance matrix. *Source:* Author

9.2 T^2 properties

Aside from its elegant geometric interpretation Hotelling's T^2 has simple properties that aid in understanding the extension to more complex multivariate tests.

- **Maximum t^2** : Consider constructing a new variable w as a linear combination of the scores in a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p]$ with weights \mathbf{a} ,

$$w = a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \dots + a_p\mathbf{x}_p = \mathbf{X}\mathbf{a}$$

Hotelling's T^2 is then the maximum value of a univariate $t^2(\mathbf{a})$ over all possible choices of the weights in \mathbf{a} . In this way, Hotellings test reduces a multivariate problem to a univariate one.

- **Eigenvalue** : Hotelling showed that T^2 is the one non-zero eigenvalue (latent root) λ of the matrix $\mathbf{Q}_H = N(\bar{\mathbf{x}} - \mu_0)^T(\bar{\mathbf{x}} - \mu_0)$ relative to $\mathbf{Q}_E = \mathbf{S}$ that solves the equation

$$(\mathbf{Q}_H - \lambda\mathbf{Q}_E)\mathbf{a} = 0 \quad (9.1)$$

In more complex MANOVA problems, there are more than one non-zero latent roots, $\lambda_1, \lambda_2, \dots, \lambda_s$, and test statistics (Wilks' Λ , Pillai and Hotelling-Lawley trace criteria, Roy's maximum root test) are functions of these.

- **Eigenvector** : The corresponding eigenvector is $\mathbf{a} = \mathbf{S}^{-1}(\bar{\mathbf{x}} - \mu_0)$. These are the (raw) *discriminant coefficients*, giving the relative contribution of each variable to T^2 .
- **Critical values** : For a single response, the square of a t statistic with $N - 1$ degrees of freedom is an $F(1, N - 1)$ statistic. But we chose \mathbf{a} to give the *maximum $t^2(\mathbf{a})$* ; this can be taken into account with a transformation of T^2 to give an **exact F** test with the correct sampling distribution:

$$F^* = \frac{N - p}{p(N - 1)} T^2 \sim F(p, N - p) . \quad (9.2)$$

- **Invariance under linear transformation** : Just as a univariate t -test is unchanged if we apply a linear transformation to the variable, $x \rightarrow ax + b$, T^2 is invariant under all linear (*affine*) transformations,

$$\mathbf{x}_{p \times 1} \rightarrow \mathbf{C}_{p \times p}\mathbf{x} + \mathbf{b}$$

So, you get the same results if you convert penguins flipper lengths from millimeters to centimeters or inches. The same is true for all MANOVA tests.

- **Two-sample tests** : With minor variations in notation, everything above applies to the more usual test of equality of multivariate means in a two sample test of $\mathcal{H}_0 : \mu_1 = \mu_2$.

$$T^2 = N(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \mathbf{S}_p^{-1} (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$$

where \mathbf{S}_p is the pooled within-sample variance covariance matrix.

Example: Maths score data

The data set `mathscore` gives (fictitious) scores on a test of basic math skills (BM) and solving word problems (WP) for two groups of $N = 6$ students in an algebra course, each taught by different instructors. The null hypothesis is that the means are equal for both variables, $\mathcal{H}_0 : \mu_{BM} = \mu_{WP}$.

```
data(mathscore, package = "heplots")
str(mathscore)
#> 'data.frame': 12 obs. of 3 variables:
#> $ group: Factor w/ 2 levels "1","2": 1 1 1 1 1 2 2 2 ...
#> $ BM   : int 190 170 180 200 150 180 160 190 150 160 ...
#> $ WP   : int 90 80 80 120 60 70 120 150 90 130 ...
```

You can carry out the test that the means for both variables are jointly equal across groups using either

`Hotelling::hotelling.test()` ([Curran & Hersh, 2021](#)) or `car::Anova()`, but the latter is more generally useful

```
hotelling.test(cbind(BM, WP) ~ group, data=mathscore) |> print()
#> Test stat: 64.174
#> Numerator df: 2
#> Denominator df: 9
#> P-value: 0.0001213

math.mod <- lm(cbind(BM, WP) ~ group, data=mathscore)
Anova(math.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> group    1     0.865     28.9      2     9 0.00012 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What's wrong with just doing the two t -tests (or equivalent F -test with `lm()`)?

```
Anova(mod1 <- lm(BM ~ group, data=mathscore))
#> Anova Table (Type II tests)
#>
#> Response: BM
#>          Sum Sq Df F value Pr(>F)
#> group      1302  1   4.24  0.066 .
#> Residuals  3071 10
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Anova(mod2 <- lm(WP ~ group, data=mathscore))
#> Anova Table (Type II tests)
#>
#> Response: WP
#>          Sum Sq Df F value Pr(>F)
#> group      4408  1   10.4  0.009 **
#> Residuals  4217 10
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From this, we might conclude that the two groups do *not* differ significantly on Basic Math but strongly differ on Word problems. But the two univariate tests do not take the correlation among the mean differences into account.

If you want to just extract the t -tests, here's a handy trick using `broom::tidy.mlm()`, which summarizes the test statistics for each response and each term in a MLM. The mean difference shown below is that for group 2 - group 1.

```
tidy(math.mod) |>
  filter(term != "(Intercept)") |>
  select(-term) |>
  rename(Mean_diff = estimate,
         t = statistic) |>
```

```

  mutate(signif = noquote(gtools::stars.pval(p.value)))
#> # A tibble: 2 x 6
#>   response Mean_diff std.error      t p.value signif
#>   <chr>     <dbl>     <dbl> <dbl>    <dbl> <noquote>
#> 1 BM        -20.8     10.1  -2.06  0.0665  .
#> 2 WP         38.3     11.9   3.23  0.00897 **
```

To see the differences between the groups on both variables together, we draw their data (68%) ellipses, using `heplots::covEllipses()`. (Setting `pooled=FALSE` here omits drawing the ellipse for the pooled covariance matrix \mathbf{S}_p .)

```

colors <- c("darkgreen", "blue")
covEllipses(mathscore[,c("BM", "WP")], mathscore$group,
            pooled = FALSE,
            col = colors,
            fill = TRUE,
            fill.alpha = 0.05,
            cex = 2, cex.lab = 1.5,
            asp = 1,
            xlab="Basic math", ylab="Word problems")
# plot points
pch <- ifelse(mathscore$group==1, 15, 16)
col <- ifelse(mathscore$group==1, colors[1], colors[2])
points(mathscore[,2:3], pch=pch, col=col, cex=1.25)
```

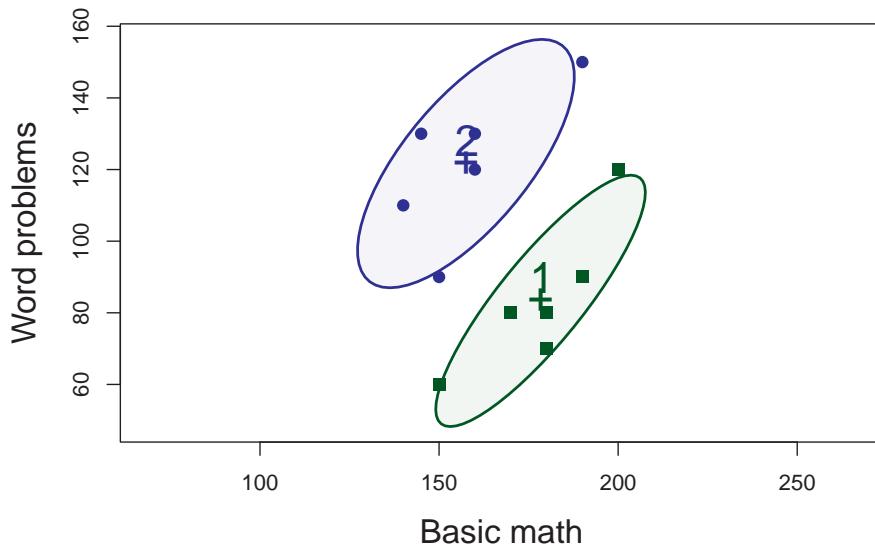


Figure 9.2: Data ellipses for the `mathscore` data, enclosing approximately 68% of the observations in each group

We can see that:

- Group 1 > Group 2 on Basic Math, but worse on Word Problems
- Group 2 > Group 1 on Word Problems, but worse on Basic Math
- Within each group, those who do better on Basic Math also do better on Word Problems

We can also see why the univariate test, at least for Basic math is non-significant: the scores for the two groups overlap considerably on the horizontal axis. They are slightly better separated along the vertical axis

for word problems. The plot also reveals why Hotelling's T^2 reveals such a strongly significant result: the two groups are very widely separated along an approximately 45° line between them.

A relatively simple interpretation is that the groups don't really differ in overall math ability, but perhaps the instructor in Group 1 put more focus on basic math skills, while the instructor for Group 2 placed greater emphasis on solving word problems.

In Hotelling's T^2 , the “size” of the difference between the means (labeled “1” and “2”) is assessed relative to the pooled within-group covariance matrix \mathbf{S}_p , which is just a size-weighted average of the two within-sample matrices, \mathbf{S}_1 and \mathbf{S}_2 ,

$$\mathbf{S}_p = [(n_1 - 1)\mathbf{S}_1 + (n_2 - 1)\mathbf{S}_2]/(n_1 + n_2 - 2) .$$

Visually, imagine sliding the the separate data ellipses to the grand mean, $(\bar{x}_{BM}, \bar{x}_{WP})$ and finding their combined data ellipse. This is just the data ellipse of the sample of deviations of the scores from their group means, or that of the residuals from the model `lm(cbind(BM, WP) ~ group, data=mathscore)`

To see this, we plot \mathbf{S}_1 , \mathbf{S}_2 and \mathbf{S}_p together,

```
covEllipses(mathscore[,c("BM", "WP")], mathscore$group,
            col = c(colors, "red"),
            fill = c(FALSE, FALSE, TRUE),
            fill.alpha = 0.3,
            cex = 2, cex.lab = 1.5,
            asp = 1,
            xlab="Basic math", ylab="Word problems")
```

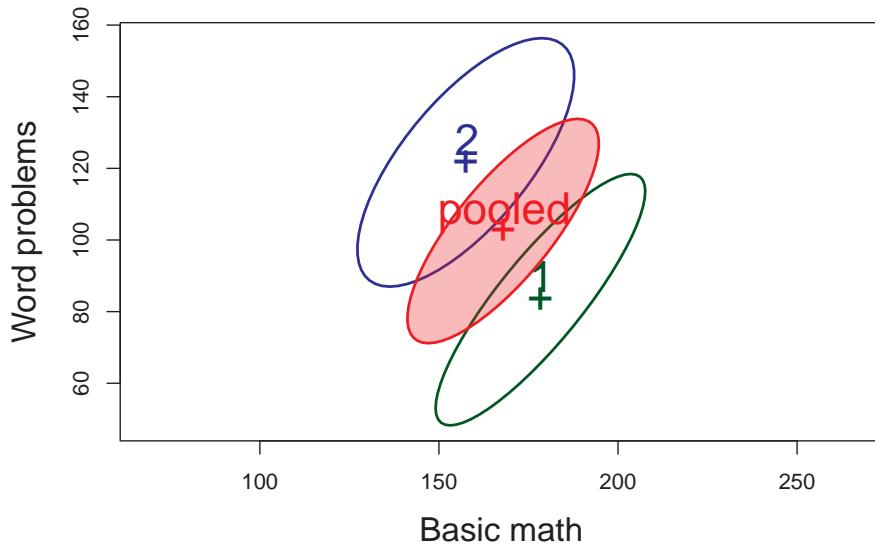


Figure 9.3: Data ellipses and the pooled covariance matrix `mathscore` data.

One of the assumptions of the T^2 test (and of MANOVA) is that the within-group variance covariance matrices, \mathbf{S}_1 and \mathbf{S}_2 , are the same. In Figure 9.3, you can see how the shapes of \mathbf{S}_1 and \mathbf{S}_2 are very similar, differing in that the variance of word Problems is slightly greater for group 2. In Chapter XX we take of the topic of visualizing tests of this assumption, based on Box's M -test.

9.3 HE plot and discriminant axis

As we describe in detail in Chapter 11, all the information relevant to the T^2 test and MANOVA can be captured in the remarkably simple *Hypothesis Error* plot, which shows the relative size of two data ellipses,

- **H:** the data ellipse of the *fitted* values, which are just the group means on the two variables, $\bar{\mathbf{x}}$, corresponding to \mathbf{Q}_H in Equation 9.1. In case of T^2 , the **H** matrix is of rank 1, so the “ellipse” plots as a line.

```
# calculate H directly
fit <- fitted(math.mod)
xbar <- colMeans(mathscore[,2:3])
N <- nrow(mathscore)
crossprod(fit) - N * outer(xbar, xbar)
#>      BM     WP
#> BM  1302 -2396
#> WP -2396  4408

# same as: SSP for group effect from Anova
math.aov <- Anova(math.mod)
(H <- math.aov$SSP)
#> $group
#>      BM     WP
#> BM  1302 -2396
#> WP -2396  4408
```

- **E:** the data ellipse of the *residuals*, the deviations of the scores from the group means, $\mathbf{x} - \bar{\mathbf{x}}$, corresponding to \mathbf{Q}_E .

```
# calculate E directly
resids <- residuals(math.mod)
crossprod(resids)
#>      BM     WP
#> BM 3071 2808
#> WP 2808 4217

# same as: SSPE from Anova
(E <- math.aov$SSPE)
#>      BM     WP
#> BM 3071 2808
#> WP 2808 4217
```

9.3.1 heplot()

`heplots::heplot()` takes the model object, extracts the **H** and **E** matrices (from `summary(Anova(math.mod))`) and plots them. There are many options to control the details.

```
heplot(math.mod,
       fill=TRUE, lwd = 3,
       asp = 1,
```

```
cex=2, cex.lab=1.8,
xlab="Basic math", ylab="Word problems")
```

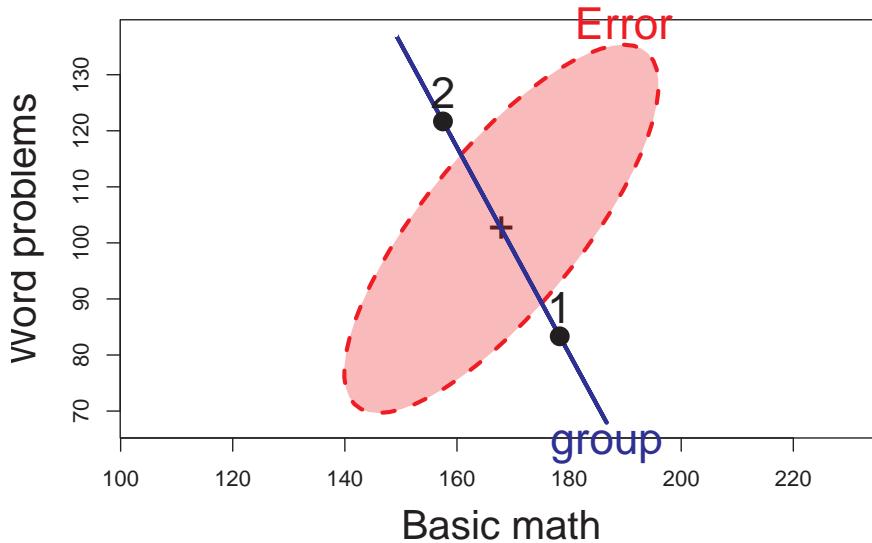


Figure 9.4: Hypothesis error plot of the `mathscore` data. The line through the group means is the **H** ellipse, which plots as a line here. The red ellipse labeled ‘Error’ represents the pooled within-group covariance matrix.

But the HE plot offers more:

- A visual test of significance: the **H** ellipse is scaled so that it projects *anywhere* outside the **E** ellipse, if and only if the test is significant at a given α level ($\alpha = 0.05$ by default)
- The **H** ellipse, which appears as a line, goes through the means of the two groups. This is also the *discriminant axis*, the direction in the space of the variables which maximally discriminates between the groups. That is, if we project the data points onto this line, we get the linear combination w which has the maximum possible univariate t^2 .

You can see how the HE plot relates to the plots of the separate data ellipses by overlaying them in a single figure. We also plot the scores on the discriminant axis, by using this small function to find the orthogonal projection of a point **a** on the line joining two points, **p₁** and **p₂**, which in math is $\mathbf{p}_1 + \frac{\mathbf{d}^\top(\mathbf{a}-\mathbf{p}_1)}{\mathbf{d}^\top\mathbf{d}} \mathbf{d}$, letting $\mathbf{d} = \mathbf{p}_1 - \mathbf{p}_2$.

```
dot <- function(x, y) sum(x*y)      # dot product of two vectors
project_on <- function(a, p1, p2) {
  a <- as.numeric(a)
  p1 <- as.numeric(p1)
  p2 <- as.numeric(p2)
  t <- dot(p2-p1, a-p1) / dot(p2-p1, p2-p1)
  C <- p1 + t*(p2-p1)
  C
}
```

Then, we run the same code as before to plot the data ellipses, and follow this with a call to `heplot()` using the option `add=TRUE` which adds to an existing plot. Following this, we find the group means and draw lines projecting the points on the line between them.

```

covEllipses(mathscore[,c("BM", "WP")], mathscore$group,
            pooled=FALSE,
            col = colors,
            cex=2, cex.lab=1.5,
            asp=1,
            xlab="Basic math", ylab="Word problems"
            )
pch <- ifelse(mathscore$group==1, 15, 16)
col <- ifelse(mathscore$group==1, "red", "blue")
points(mathscore[,2:3], pch=pch, col=col, cex=1.25)

# overlay with HEplot (add = TRUE)
heplot(math.mod,
       fill=TRUE,
       cex=2, cex.lab=1.8,
       fill.alpha=0.2, lwd=c(1,3),
       add = TRUE,
       error.ellipse=TRUE)

# find group means
means <- mathsore |>
  group_by(group) |>
  summarize(BM = mean(BM), WP = mean(WP))

for(i in 1:nrow(mathsore)) {
  gp <- mathsore$group[i]
  pt <- project_on( mathsore[i, 2:3], means[1, 2:3], means[2, 2:3])
  segments(mathsore[i, "BM"], mathsore[i, "WP"], pt[1], pt[2], lwd = 1.2)
}

```

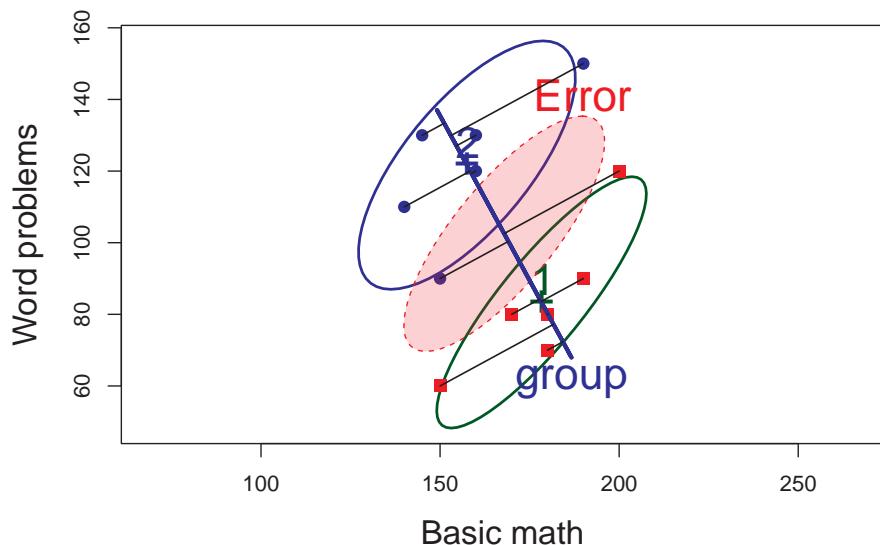


Figure 9.5: HE plot overlaid on top of the within-group data ellipses, with lines showing the projection of each point on the discriminant axis.

9.4 Discriminant analysis

Discriminant analysis for two-group designs or for one-way MANOVA essentially turns the problem around: Instead of asking whether the mean vectors for two or more groups are equal, discriminant analysis tries to find the linear combination w of the response variables that has the greatest separation among the groups, allowing cases to be best classified.

For the maths score data, you can perform the discriminant analysis as follows, using the **MASS** function **lda()**:

```
(math.lda <- MASS::lda(group ~ ., data=mathscore))
#> Call:
#> lda(group ~ ., data = mathscore)
#>
#> Prior probabilities of groups:
#>   1   2
#> 0.5 0.5
#>
#> Group means:
#>   BM    WP
#> 1 178  83.3
#> 2 158 121.7
#>
#> Coefficients of linear discriminants:
#>   LD1
#> BM -0.0835
#> WP  0.0753
```

The coefficients give $w = -0.084 \text{ BM} + 0.075 \text{ WP}$. This is exactly the direction given by the line for the **H** ellipse in Figure 9.5.

To round this out, we can calculate the discriminant scores by multiplying the matrix \mathbf{X} by the vector $\mathbf{a} = \mathbf{S}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$ of the discriminant weights. These were shown in Figure 9.5 as the projections of the data points on the line joining the group means,

```
math.lda$scaling
#>       LD1
#> BM -0.0835
#> WP  0.0753

scores <- cbind(group = mathscore$group,
                 as.matrix(mathscore[, 2:3]) %*% math.lda$scaling) |>
  as.data.frame()
scores |>
  group_by(group) |>
  slice(1:3)
#> # A tibble: 6 x 2
#> # Groups:   group [2]
#>   group   LD1
#>   <dbl> <dbl>
#> 1      1 -9.09
#> 2      1 -8.17
```

```
#> 3     1 -9.01
#> 4     2 -4.33
#> 5     2 -4.58
#> 6     2 -5.75
```

Then a t -test on these scores gives the same value as Hotelling's T ; it is accessed via the **statistic** component of `t.test()`

```
t <- t.test(LD1 ~ group, data=scores)$statistic
c(t, T2 = t^2)
#>      t    T2.t
#> -8.01 64.17
```

Finally, it is instructive to compare violin plots for the three measures, BM, WP and LD1. To do this with `ggplot2` requires reshaping the data from wide to long format so the plots can be faceted.

```
scores <- mathscore |>
  bind_cols(LD1 = scores[, "LD1"])

scores |>
  tidyrr::gather(key = "measure", value ="Score", BM:LD1) |>
  mutate(measure = factor(measure, levels = c("BM", "WP", "LD1"))) |>
  ggplot(aes(x = group, y = Score, color = group, fill = group)) +
  geom_violin(alpha = 0.2) +
  geom_jitter(width = .2, size = 2) +
  facet_wrap(~ measure, scales = "free", labeller = label_both) +
  scale_fill_manual(values = c("darkgreen", "blue")) +
  scale_color_manual(values = c("darkgreen", "blue")) +
  theme_bw(base_size = 14) +
  theme(legend.position = "none")
```

You can readily see how well the groups are separated on the discriminant axes, relative to the two individual variables.

9.5 More variables

The `mathscore` data gave a simple example with two outcomes to explain the essential ideas behind Hotelling's T^2 and multivariate tests. Multivariate methods become increasingly useful as the number of response variables increases because it is harder to show them all together and see how they relate to differences between groups.

A classic example is the dataset `banknote`, containing six size measures made on 100 genuine and 100 counterfeit old-Swiss 1000-franc bank notes (Flury & Riedwyl, 1988). The goal is to see how well the real and fake banknotes can be distinguished. The measures are the `Length` and `Diagonal` lengths of a banknote and the `Left`, `Right`, `Top` and `Bottom` edge margins in mm.

Before considering hypothesis tests, let's look at some exploratory graphics. Figure 9.7 shows univariate violin and boxplots of each of the measures. To make this plot, faceted by measure, I first reshape the data from wide to long and make `measure` a factor with levels in the order of the variables in the data set.

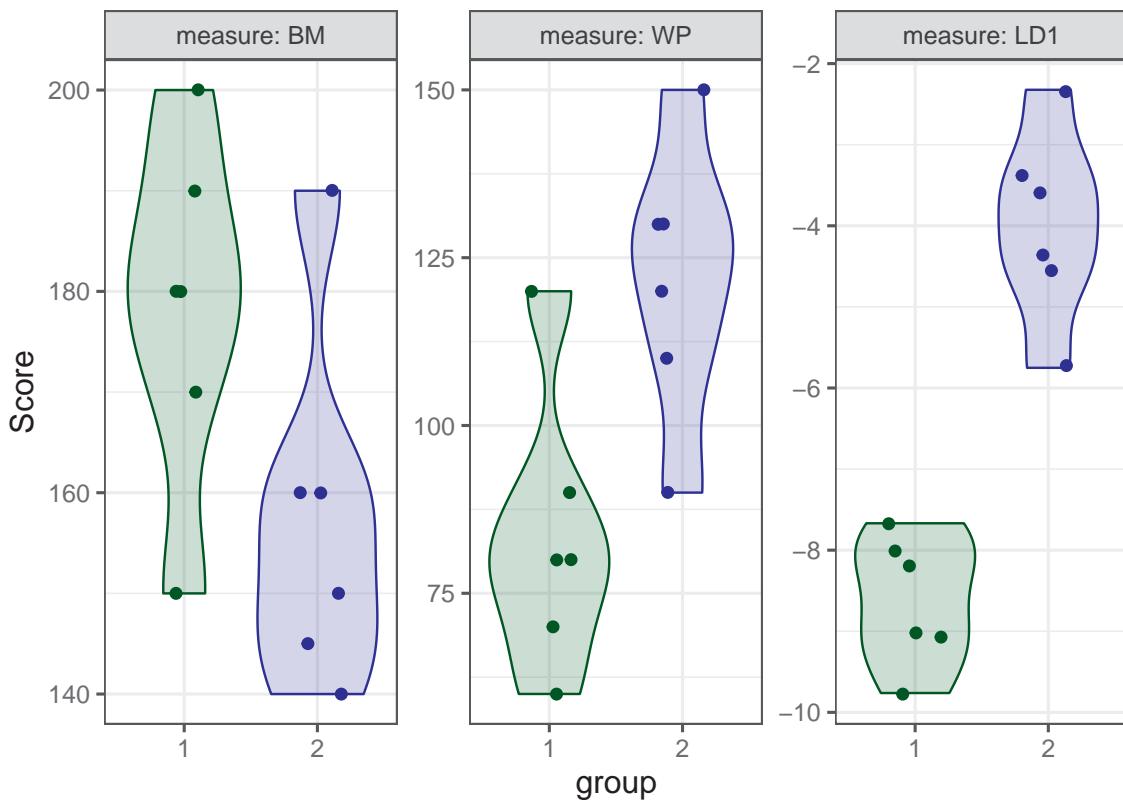


Figure 9.6: Violin plots comparing group 1 and 2 for the two observed measures and the linear discriminant score.

```
data(banknote, package= "mclust")
banknote |>
  tidyverse::gather(key = "measure",
                    value = "Size",
                    Length:Diagonal) |>
  mutate(measure = factor(measure,
                          levels = c(names(banknote)[-1]))) |>

  ggplot(aes(x = Status, y = Size, color = Status)) +
  geom_violin(aes(fill = Status), alpha = 0.2) +          # (1)
  geom_jitter(width = .2, size = 1.2) +                  # (2)
  geom_boxplot(width = 0.25,                             # (3)
                linewidth = 1.1,
                color = "black",
                alpha = 0.5) +
  labs(y = "Size (mm)") +
  facet_wrap(~ measure, scales = "free", labeller = label_both) +
  theme_bw(base_size = 14) +
  theme(legend.position = "top")
```

A quick glance at Figure 9.7 shows that the counterfeit and genuine bills differ in their means on most of the measures, with the counterfeit ones slightly larger on Left, Right, Bottom and Top margins. But univariate plots don't give an overall sense of how these variables are related to one another.

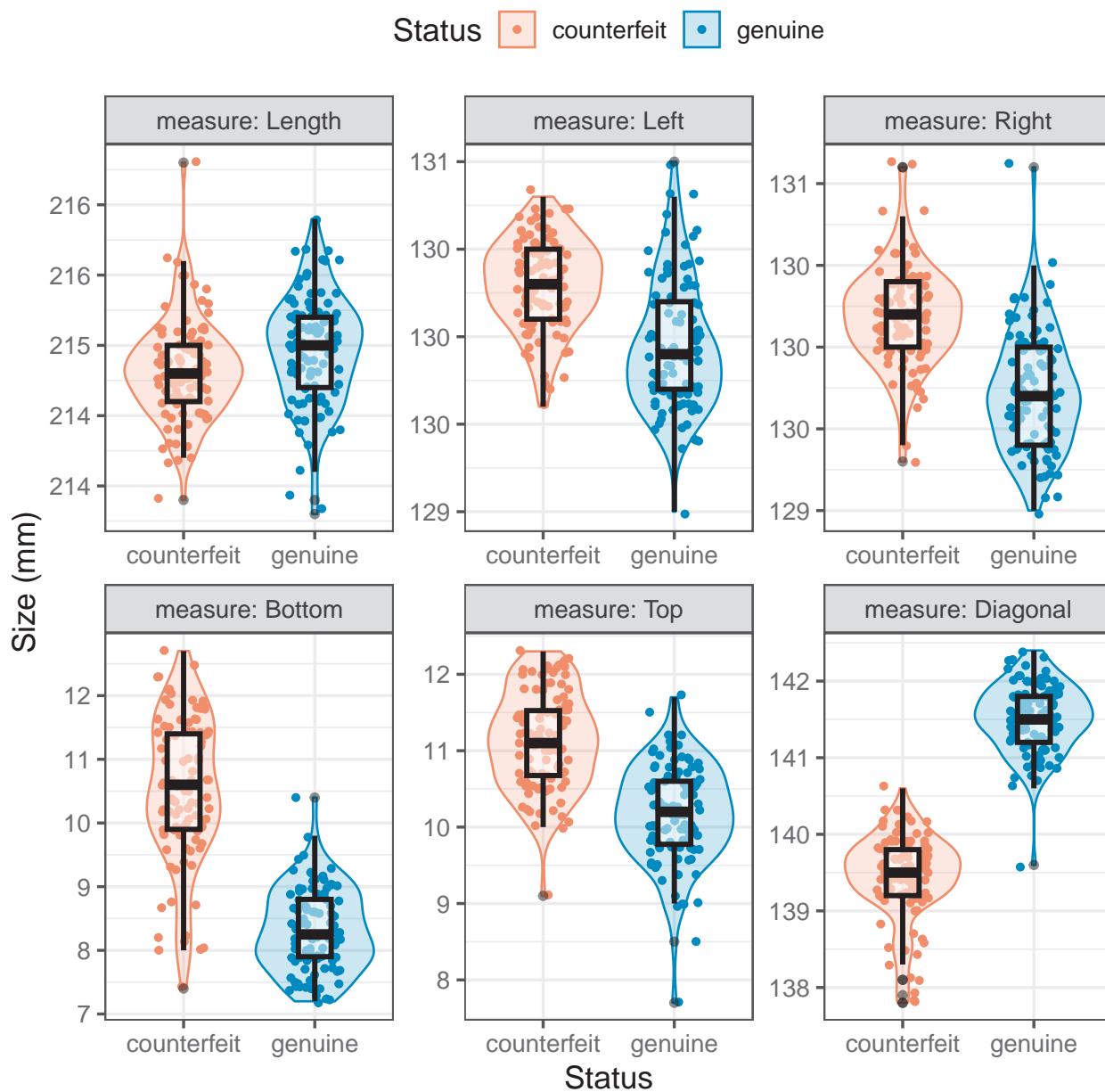


Figure 9.7: Overlaid violin and boxplots of the banknote variables. The violin plots give a sense of the shapes of the distributions, while the boxplots highlight the center and spread.

i Graph craft: Layers and transparency

Figure 9.7 is somewhat complex, so it is useful to understand the steps needed to make this figure show what I wanted. The plot in each panel contains three layers:

- (1) the violin plot based on a density estimate, showing the shape of each distribution;
- (2) the data points, but they are jittered horizontally using `geom_jitter()` because otherwise they would all overlap on the X axis;
- (3) the boxplot, showing the center (median) and spread (IQR) of each distribution.

In composing graphs with layers, order matters, and also does the `alpha` transparency, because each layer adds data ink on top of earlier ones. I plotted these in the order shown because I wanted the violin plot to provide the background, and the boxplot to show a simple univariate summary, not obscured by the other layers. The `alpha` values allow the data ink to be blended for each layer, and in this case, `alpha = 0.5` for the boxplot let the earlier layers show through.

9.5.1 Biplots

Multivariate relations among these six variables could be explored in data space using scatterplots or other methods, but I turn to my trusty multivariate juicer, a biplot, to give a 2D summary. Two dimensions account for 70% of the total variance of all the banknotes, while three would give 85%.

```
banknote.pca <- prcomp(banknote[, -1], scale = TRUE)
summary(banknote.pca)
#> Importance of components:
#>                               PC1    PC2    PC3    PC4    PC5    PC6
#> Standard deviation     1.716 1.131 0.932 0.671 0.5183 0.4346
#> Proportion of Variance 0.491 0.213 0.145 0.075 0.0448 0.0315
#> Cumulative Proportion  0.491 0.704 0.849 0.924 0.9685 1.0000
```

The biplot in Figure 9.8 gives a nicely coherent overview, at least in two dimensions. The first component shows the positive correlations among the measures of the margins, where the counterfeit bills are larger than the real ones and a negative correlation of the Diagonal with the other measures. The length of bills only distinguishes the types of banknotes on the second dimension.

```
banknote.pca <- ggbiplots::reflect(banknote.pca)
ggbiplots(banknote.pca,
          obs.scale = 1, var.scale = 1,
          groups = banknote>Status,
          ellipse = TRUE,
          ellipse.level = 0.5,
          ellipse.alpha = 0.1,
          ellipse.linewidth = 0,
          varname.size = 4,
          varname.color = "black") +
  labs(fill = "Status",
       color = "Status") +
  theme_minimal(base_size = 14) +
  theme(legend.position = 'top')
```

9.5.2 Testing mean differences

As noted above, Hotelling's T^2 is equivalent to a one-way MANOVA, fitting the size measures to the `Status` of the banknotes. `Anova()` reports only the F -statistic based on Pillai's trace criterion.

```
banknote.mlm <- lm(cbind(Length, Left, Right, Bottom, Top, Diagonal) ~ Status,
                     data = banknote)
Anova(banknote.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
```

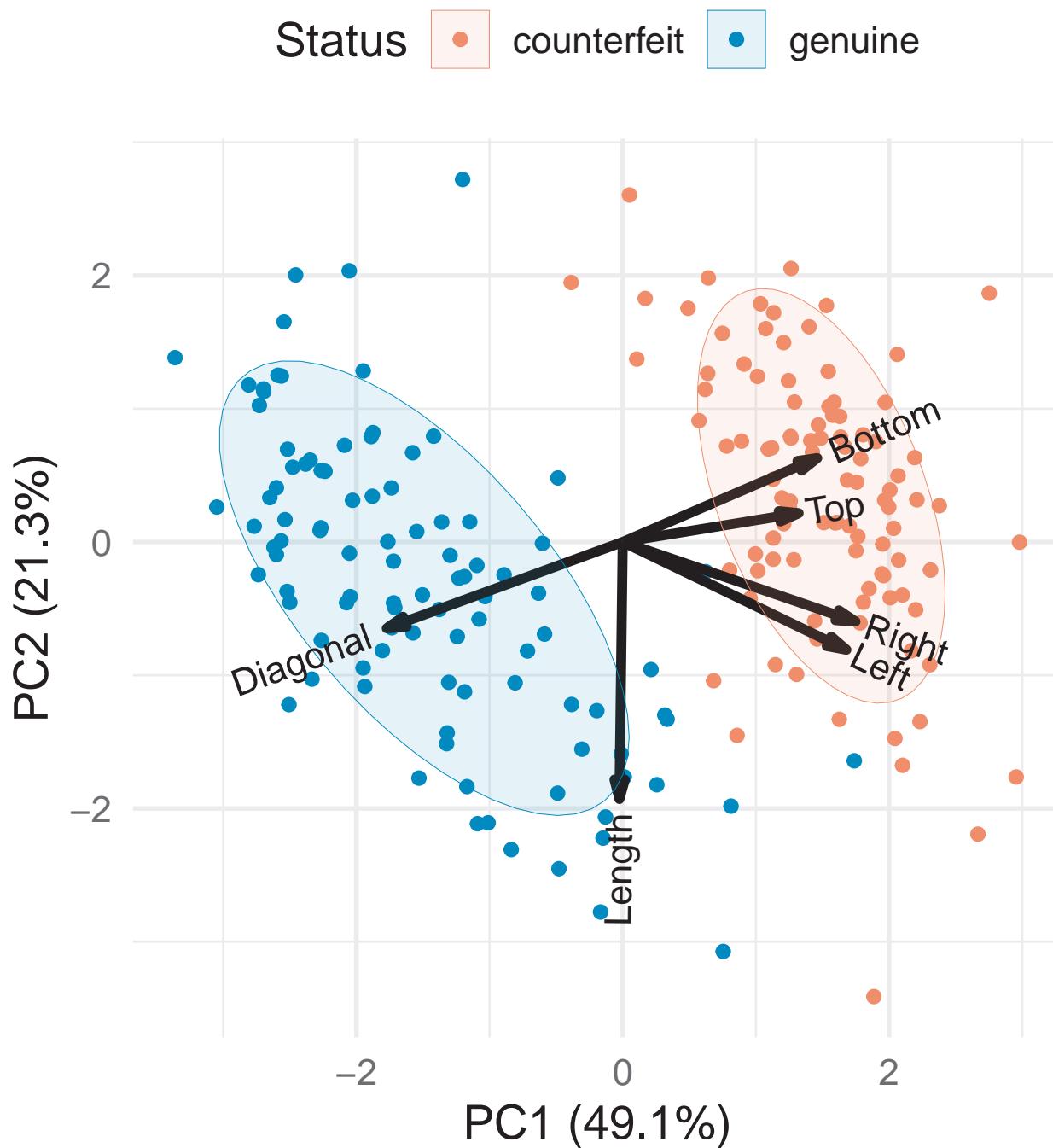


Figure 9.8: Biplot of the banknote variables, showing how the size measurements are related to each other. The points and data ellipses for the component scores are colored by Status, showing how the counterfeit and genuine bills are distinguished by these measures.

```
#>      Df test stat approx F num Df den Df Pr(>F)
#> Status  1    0.924      392     6    193 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You can see all the multivariate test statistics with the `summary()` method for "Anova.mlm" objects. With two groups, and hence a 1 df test, these all translate into identical F -statistics.

```
summary(Anova(banknote.mlm)) |> print(SSP = FALSE)
#>
#> Type II MANOVA Tests:
#>
#> -----
#>
#> Term: Status
#>
#> Multivariate Tests: Status
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1    0.92     392      6    193 <2e-16 ***
#> Wilks       1    0.08     392      6    193 <2e-16 ***
#> Hotelling-Lawley 1   12.18     392      6    193 <2e-16 ***
#> Roy         1   12.18     392      6    193 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If you wish, you can extract the univariate t -tests or equivalent $F = t^2$ statistics from the "mlm" object using `broom::tidy.mlm()`. What is given as the `estimate` is the difference in the mean for the genuine banknotes relative to the counterfeit ones.

```
broom::tidy(banknote.mlm) |>
  filter(term != "(Intercept)") |>
  dplyr::select(-term) |>
  rename(t = statistic) |>
  mutate(F = t^2) |>
  relocate(F, .after = t)
#> # A tibble: 6 x 6
#>   response estimate std.error     t     F p.value
#>   <chr>     <dbl>     <dbl> <dbl> <dbl>   <dbl>
#> 1 Length     0.146    0.0524   2.79   7.77 5.82e- 3
#> 2 Left       -0.357   0.0445  -8.03   64.5 8.50e-14
#> 3 Right      -0.473   0.0464 -10.2    104.  6.84e-20
#> 4 Bottom     -2.22    0.130   -17.1   292.  7.78e-41
#> 5 Top        -0.965   0.0909 -10.6    113.  3.85e-21
#> 6 Diagonal    2.07    0.0715  28.9    836.  5.35e-73
```

The individual $F_{(1,198)}$ statistics can be compared to the $F_{(6,193)} = 392$ value for the overall multivariate test. While all of the individual tests are highly significant, the average of the univariate F s is only 236. The multivariate test gains power by taking the correlations of the size measures into account.

9.6 Variance accounted for: Eta square (η^2)

In a univariate multiple regression model, the coefficient of determination $R^2 = \text{SS}_H/\text{SS}_{\text{Total}}$ gives the proportion of variance accounted for by hypothesized terms in H relative to the total variance. An analog for ANOVA-type models with categorical, group factors as predictors is η^2 (Pearson, 1903), defined as

$$\eta^2 = \frac{SS_{\text{Between groups}}}{SS_{\text{Total}}} .$$

For multivariate response models, the generalization of η^2 uses multivariate analogs of these sums of squares, \mathbf{Q}_H and $\mathbf{Q}_T = \mathbf{Q}_H + \mathbf{Q}_E$, and there are different calculations for a single measure corresponding to the various test statistics (Wilks' Λ , etc.), as described in Chapter 10.

Let's calculate the η^2 for the multivariate model `banknote.mlm` with `Status` as the only predictor, giving $\eta^2 = 0.92$, or 92% of the total variance.

```
heplots::etasq(banknote.mlm)
#>      eta^2
#> Status 0.924
```

This can be compared to the principal components analysis and the biplot in Figure 9.8, where two components (less favorably) accounted for 70% of total variance and it took four PCA dimensions to account for over 90%. The goals of PCA and MANOVA are different, of course, but they are both concerned with accounting for variance of multivariate data. We will meet another multivariate juicer, **canonical discriminant analysis** in Chapter 11.

9.7 What we've learned

TODO: Combine with 'summary/Ch09-summary.qmd

This chapter was designed to illustrate the main ideas for visualizing differences between means on multiple response variables in a two-group design. Hotelling's T^2 is the generalization of a simple univariate t -test and works by combining the responses into a weighted sum that has the maximum possible univariate t for all choices of weights.

Figure 9.9 summarizes what was shown in Section 9.3 and Section 9.4. The data ellipses for the two groups in the `mathscore` data summarize the information about means and within-group variances. In the HE plot, the difference between the means is itself summarized by the line through them, which represents the $\mathbf{H} = \mathbf{Q}_H$ matrix and within-group variation is represented by the "Error" ellipse which is the $\mathbf{E} = \mathbf{S}_p = \mathbf{Q}_E$ matrix.

As we will see later (Chapter 11), the \mathbf{H} ellipse is scaled so that it provides a visual test of significance: it projects somewhere outside the \mathbf{E} ellipse if and only if the means differ significantly. The direction of the line between the means is also the discriminant axis and scores on this axis are weighted sum of the responses that have the greatest possible mean difference.

9.8 Exercises

Exercise 9.1. The value of Hotelling's T^2 found by `hotelling.test()` is 64.17. The value of the equivalent F statistic found by `Anova()` is 28.9. Verify that Equation 9.2 gives this result.

Packages used here:

11 packages used here: broom, car, carData, corpcor, dplyr, ggbio, ggplot2, heplots, Hotelling, knitr, tidyr

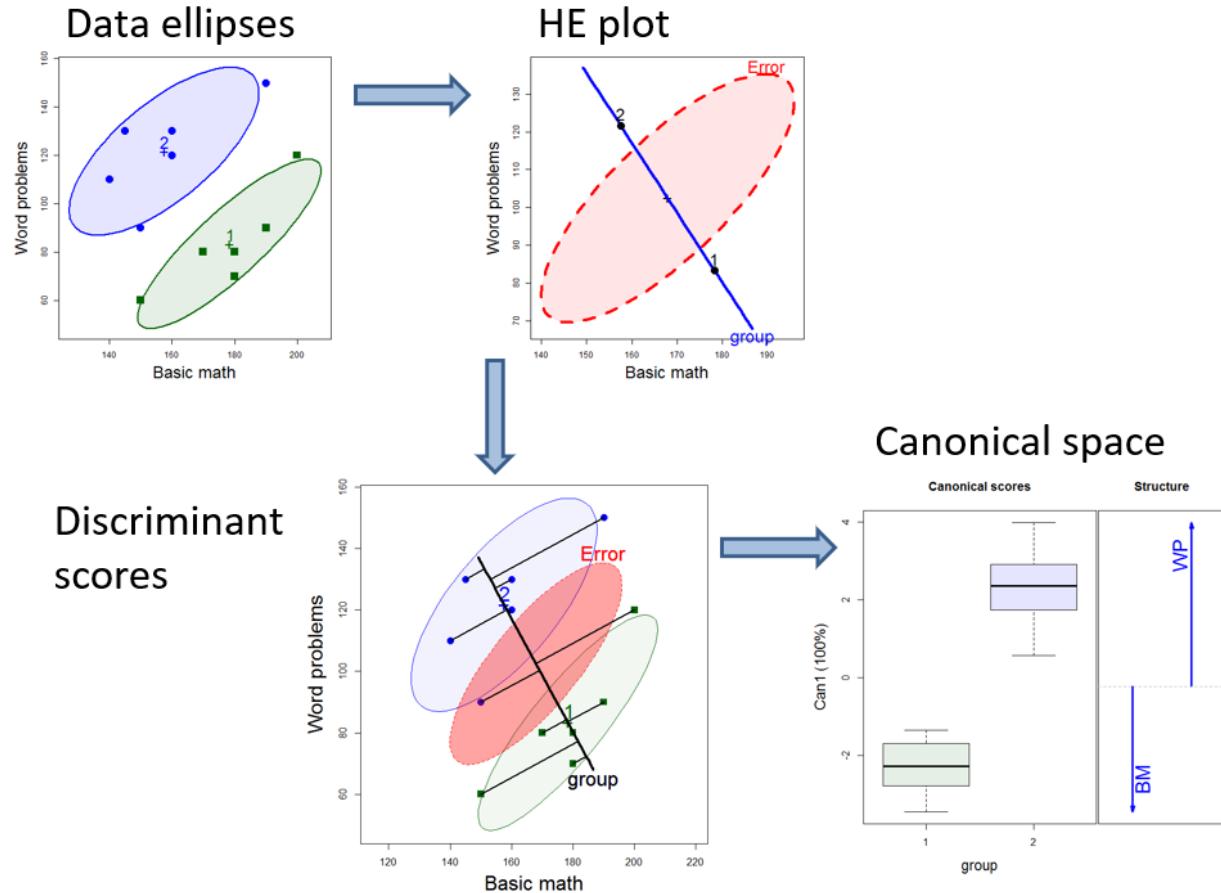


Figure 9.9: The Hypothesis Error plot framework for a two-group design. Above: Data ellipses can be summarized in an HE plot showing the pooled within-group error (\mathbf{E}) ellipse and the \mathbf{H} ‘ellipse’ for the group means. Below: Observations projected on the line joining the means give discriminant scores which correspond to a one-dimensional canonical space, represented by a boxplot of their scores and arrows reflecting the variable weights.

10

Multivariate Linear Models

Chapter 9 introduced the essential ideas of multivariate analysis in the context of a two-group design using Hotelling's T^2 . Through the magical power of multivariate thinking, I extend this here to a "Holy of Holies", inner sanctuary of the Tabernacle, the awed general **Multivariate Linear Model** (MLM).¹

This can be understood as a simple extension of the univariate linear model, with the main difference being that there are multiple response variables considered **together**, instead of just one, analysed alone. (Or, from my perspective, the univariate version is a restricted form of the MLM.) These outcomes might reflect several different *ways* or scales for measuring an underlying theoretical construct, or they might represent different *aspects* of some phenomenon that we hope to better understand when they are studied jointly.

For example, in the case of different measures, there are numerous psychological scales used to assess depression or anxiety and it may be important to include more than one measure to ensure that the construct has been measured adequately. It would add considerably to our understanding to know if the different outcome measures all had essentially the *same* relations to the predictor variables, or if they differ across measures.

In the second case of various aspects, student "aptitude" or "achievement" reflects competency in different various subjects (reading, math, history, science, ...) that are better studied together. We get a better understanding of the factors that influence each of aspects by testing them *jointly*.

Just as in univariate analysis there are variously named techniques (ANOVA, regression) that can be applied to several outcomes, depending on the structure of the predictors at hand. For instance, with one or more continuous predictors and multiple response variables, you could use **multivariate** multiple regression (MMRA) to obtain estimates useful for prediction.

Instead, if the predictors are categorical factors, multivariate analysis of variance (MANOVA) can be applied to test for differences between groups. Again, this is akin to ANOVA in the univariate context—the same underlying model is utilized, but the tests for terms in the model are multivariate ones for the collection of all response variables, rather than univariate ones for a single response.

The main goal of this chapter is to describe the details of the extension of univariate linear models to the case of multiple outcome measures. But the larger goal is to set the stage for the visualization methods using HE plots and low-D views discussed separately in Chapter 11. Some of the example datasets used here will re-appear there, and also in Chapter 12 which concerns some model-diagnostic graphical methods.

However, before considering the details and examples that apply separately to MANOVA and MMRA, it is useful to consider the general features of the multivariate linear model of which these cases are examples.

TODO: Offer defense against Huang (2019) and others here; cite Huberty & Morris (1989). Or, maybe not!

Packages

In this chapter I use the following packages. Load them now:

¹There's a bit of a puzzle here, and therefore a gap in methods and therefore an opportunity. The classical linear models fit by `lm()` extend naturally to non-gaussian data via `glm()` which provides for other families (binary: Bernoulli, count data: Poisson). The standard `lm()` extends quite naturally to a multivariate responses. Yet the combination of these ideas—non-gaussian multivariate models—remains elusive. The **VGAM** (Yee (2015), Yee (2025)) handles the bivariate cases of logistic (and probit) regression, but not much more. See Friendly & Meyer (2016), Sec. 10.4 for an example and graphs of odds ratios in these models. There is a lot more to do on this topic.

```

library(broom)
library(car)
library(dplyr)
library(ggplot2)
library(heplots)
library(patchwork)
library(tidyr)
library(matlib)
library(ggrepel)
library(mvinfluence)
library(MVN)
# set ggplot theme
#ggplot2::theme_set(theme_bw(base_size = 14))

```

10.1 Structure of the MLM

With p response variables, the multivariate linear model is most easily appreciated as the collection of p linear models, one for each response. We have p outcomes, so why not just consider a separate model for each?

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{X}\boldsymbol{\beta}_1 + \boldsymbol{\epsilon}_1 \\ \mathbf{y}_2 &= \mathbf{X}\boldsymbol{\beta}_2 + \boldsymbol{\epsilon}_2 \\ &\vdots \\ \mathbf{y}_p &= \mathbf{X}\boldsymbol{\beta}_p + \boldsymbol{\epsilon}_p \end{aligned}$$

{#eq-mlm-models}

But the problems with fitting separate univariate models are that:

- They don't give *simultaneous* tests for all regressions. The situation is similar to that in a one-way ANOVA, where an *overall* test for group differences is usually applied before testing individual comparisons to avoid problems of multiple testing: g groups gives $g \times (g - 1)/2$ pairwise tests.
- More importantly, fitting separate univariate models does not take *correlations* among the \mathbf{y} s into account.
 - It might be the case that the response variables are all essentially measuring the same thing, but perhaps weakly. If so, the multivariate approach can pool strength across the outcomes to detect their common relations to the predictors, giving greater power.
 - On the other hand, perhaps the responses are related in *different ways* to the predictors. A multivariate approach can help you understand *how many* different ways there are, and characterize each.

The model matrix \mathbf{X} in ?@eq-mlm-models is the same for all responses, but each one gets its own vector $\boldsymbol{\beta}_j$ of coefficients for how the predictors in \mathbf{X} fit a given response \mathbf{y}_j .

Among the beauties of multivariate thinking is that we can put these separate equations together in single equation by joining the responses \mathbf{y}_j as columns in a matrix \mathbf{Y} and similarly arranging the vectors of coefficients $\boldsymbol{\beta}_j$ as columns in a matrix \mathbf{B} .²

TODO Revise notation here, to be explicit about inclusion of $\boldsymbol{\beta}_0$

²A slight hiccup in notation is that the *uppercase* for the Greek Beta (β) is the same as the uppercase Roman \mathbf{B} , so I use $\mathbf{b}_1, \mathbf{b}_2, \dots$ below to refer to its' columns.

The MLM then becomes:

$$\underset{n \times p}{\mathbf{Y}} = \underset{n \times (q+1)}{\mathbf{X}} \underset{(q+1) \times p}{\mathbf{B}} + \underset{n \times p}{\boldsymbol{\varepsilon}}, \quad (10.1)$$

where:

- $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_p)$ is the matrix of n observations on p responses, with typical column \mathbf{y}_j ;
- \mathbf{X} is the model matrix with columns \mathbf{x}_i for q regressors, which typically includes an initial column \mathbf{x}_0 of 1s for the intercept;
- $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_p)$ is a matrix of regression coefficients, one column \mathbf{b}_j for each response variable;
- $\boldsymbol{\varepsilon}$ is a matrix of errors in predicting \mathbf{Y} .

Writing Equation 10.1 in terms of its elements, we have

$$\begin{matrix} \mathbf{Y} \\ \left[\begin{array}{cccc} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \end{array} \right] \end{matrix} = \begin{matrix} \mathbf{X} \\ \left[\begin{array}{cccc} 1 & x_{11} & \cdots & x_{1q} \\ 1 & x_{21} & \cdots & x_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nq} \end{array} \right] \end{matrix} \begin{matrix} \mathbf{B} \\ \left[\begin{array}{cccc} \beta_{01} & \beta_{02} & \cdots & \beta_{0p} \\ \beta_{11} & \beta_{12} & \cdots & \beta_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{q1} & \beta_{q2} & \cdots & \beta_{qp} \end{array} \right] \end{matrix} \\ + \begin{matrix} \boldsymbol{\varepsilon} \\ \left[\begin{array}{cccc} \epsilon_{11} & \epsilon_{12} & \cdots & \epsilon_{1p} \\ \epsilon_{21} & \epsilon_{22} & \cdots & \epsilon_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_{n1} & \epsilon_{n2} & \cdots & \epsilon_{np} \end{array} \right] \end{matrix}$$

The structure of the model matrix \mathbf{X} is exactly the same as the univariate linear model, and may therefore contain,

- **quantitative predictors**, such as `age`, `income`, years of `education`
- **transformed predictors** like $\sqrt{\text{age}}$ or $\log(\text{income})$
- **polynomial terms**: age^2 , age^3, \dots (using `poly(age, k)` in R)
- **categorical predictors** (“factors”), such as treatment (Control, Drug A, drug B), or sex; internally a factor with k levels is transformed to $k-1$ dummy (0, 1) variables, representing comparisons with a reference level, typically the first.
- **interaction terms**, involving either quantitative or categorical predictors, e.g., `age * sex`, `treatment * sex`.

10.1.1 Assumptions

Just as in univariate models, the assumptions of the multivariate linear model almost entirely concern the behavior of the errors (residuals). Let $\boldsymbol{\epsilon}'_i$ represent the i th row of $\boldsymbol{\varepsilon}$. Then it is assumed that:

- **Normality**: The residuals, $\boldsymbol{\epsilon}'_i$ are distributed as multivariate normal, $\mathcal{N}_p(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a non-singular error-covariance matrix.
 - Statistical tests of multivariate normality of the residuals include the Shapiro-Wilk ([Shapiro & Wilk, 1965](#)) and Mardia ([1970, 1974](#)) tests (and others, in the `MVN` package).
 - As in univariate models, the MLM is relatively robust, however this is often better assessed visually using a χ^2 QQ plot of Mahalanobis squared distance against their corresponding χ^2_p values for p degrees of freedom using `heplots::cplot()`.
- **Homoscedasticity**: The error-covariance matrix $\boldsymbol{\Sigma}$ is constant across all observations and grouping factors. Graphical methods to show if this assumption is met are illustrated in Chapter 12.

- **Independence:** ϵ'_i and ϵ'_j are independent for $i \neq j$, so knowing the data for case i gives no information about case j (as would be true if the data consisted of pairs of husbands and wives);
- The predictors, \mathbf{X} , are fixed and measured without error or at least they are independent of the errors, $\boldsymbol{\varepsilon}$.

These statements are simply the multivariate analogs of the assumptions of normality, constant variance and independence of the errors in univariate models. Note that it is unnecessary to assume that the predictors (regressors, columns of \mathbf{X}) are normally distributed.

Implicit in the above is perhaps the most important assumption—that the model has been *correctly specified*. This means:

- **Linearity:** The form of the relations between each \mathbf{y} and the \mathbf{x} s is correct. Typically this means that the relations are *linear*, but if not, we have specified a correct transformation of \mathbf{y} and/or \mathbf{x} .
 - **Completeness:** No relevant predictors have been omitted from the model. For example in the coffee, stress example (Section 7.1.1), omitting stress from the model biases the effect of coffee on heart disease.
 - **Additive effects:** The combined effect of different predictors is the sum of their individual effects.
-

10.2 Fitting the model

The least squares (and also maximum likelihood) solution for the coefficients \mathbf{B} is given by

$$\hat{\mathbf{B}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} .$$

This is precisely the same as fitting the separate responses $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_p$, and placing the estimated coefficients $\hat{\mathbf{b}}_i$ as columns in $\hat{\mathbf{B}}$

$$\hat{\mathbf{B}} = [\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \dots, \hat{\mathbf{b}}_p] .$$

In R, we fit the multivariate linear model with `lm()` simply by giving a collection of response variables `y1`, `y2`, ... on the left-hand side of the model formula, wrapped in `cbind()` which combines them to form a matrix response.

```
lm(cbind(y1, y2, y3) ~ x1 + x2 + ..., data=)
```

In the presence of possible outliers, robust methods are available for univariate linear models (e.g., `MASS::rlm()`). So too, `heplots::robmlm()` provides robust estimation in the multivariate case as illustrated in Section 13.5.

10.2.1 Example: Dog food data

As a toy example to make these ideas concrete, consider the dataset `dogfood`. Here, a dogfood manufacturer wanted to study preference for different dogfood formulas, two of their own (“Old”, “New”) and two from other manufacturers (“Major”, “Alps”).

In a between-dog design, each of $n = 4$ dogs were presented with a bowl of *one* formula and the time to `start` eating and `amount` eaten were recorded. Greater preference would be seen in a shorter delay to start eating and a greater amount, so these responses are expected to be negatively correlated.

```
data(dogfood, package = "heplots")
str(dogfood)
```

```
#> 'data.frame': 16 obs. of 3 variables:
#> $ formula: Factor w/ 4 levels "Old","New","Major",...: 1 1 1 1 2 2 2 2 3 3 ...
#> $ start : int 0 1 1 0 0 1 2 3 1 5 ...
#> $ amount : int 100 97 88 92 95 85 82 89 77 84 ...
```

For this data, boxplots for the two responses provide an initial look, shown in Figure 10.1. Putting these side-by-side makes it easy to see the inverse relation between the medians on the two response variables.

```
dog_long <- dogfood |>
  pivot_longer(c(start, amount),
               names_to = "variable")
ggplot(data = dog_long,
       aes(x=formula, y = value, fill = formula)) +
  geom_boxplot(alpha = 0.2) +
  geom_point(size = 2.5) +
  facet_wrap(~ variable, scales = "free") +
  theme_bw(base_size = 14) +
  theme(legend.position="none")
```

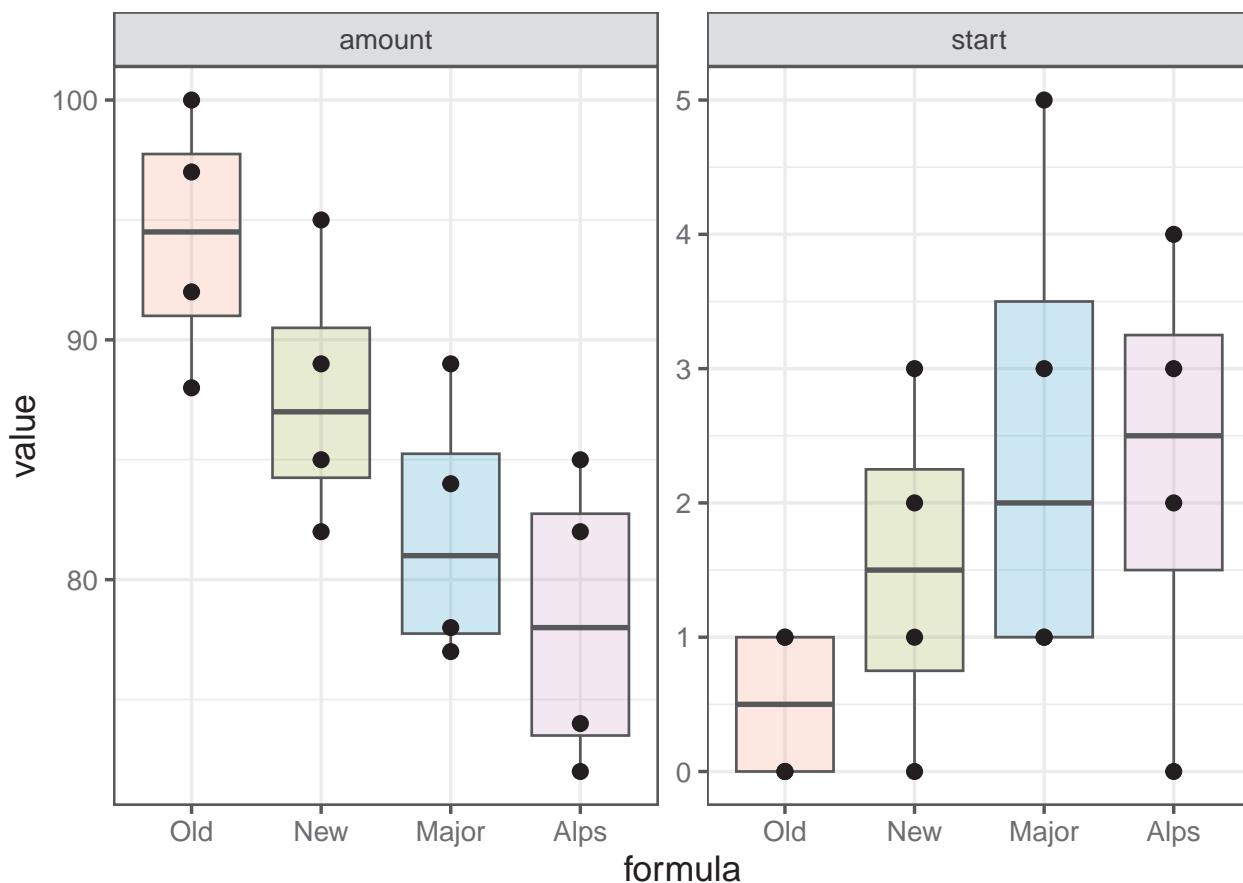


Figure 10.1: Boxplots for time to start eating and amount eaten by dogs given one of four dogfood formulas.

As suggested above, the multivariate model for testing mean differences due to the dogfood `formula` is fit using `lm()` on the matrix `Y` constructed with `cbind(start, amount)`.

```

dogfood.mod <- lm(cbind(start, amount) ~ formula,
                   data=dogfood) |>
  print()
#>
#> Call:
#> lm(formula = cbind(start, amount) ~ formula, data = dogfood)
#>
#> Coefficients:
#>             start     amount
#> (Intercept) 0.50    94.25
#> formulaNew   1.00   -6.50
#> formulaMajor 2.00  -12.25
#> formulaAlps  1.75  -16.00

```

By default, the factor `formula` is represented by three columns in the \mathbf{X} matrix that correspond to treatment contrasts, which are comparisons of the Old formula (a baseline level) with each of the others. The coefficients, for example `formulaNEW`, are the difference in means from those for Old.

Then, the overall multivariate test that means on both variables do not differ is carried out using `car::Anova()`.

```

dogfood.aov <- Anova(dogfood.mod) |>
  print()
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> formula  3    0.702      2.16      6     24   0.083 .
#> ---
#> Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The details of these analysis steps are explained below.

10.2.2 Sums of squares

In univariate response models, statistical tests and model summaries (like R^2) are based on the familiar decomposition of the total sum of squares SS_T into regression or hypothesis (SS_H) and error (SS_E) sums of squares. In the multivariate linear model each of these becomes a $p \times p$ matrix SSP containing sums of squares for the p responses on the diagonal and sums of cross products in the off-diagonal elements. For the MLM this is expressed as:

$$\begin{aligned}
SSP_T &= \mathbf{Y}^\top \mathbf{Y} - n \bar{\mathbf{y}} \bar{\mathbf{y}}^\top \\
&= (\hat{\mathbf{Y}}^\top \hat{\mathbf{Y}} - n \bar{\mathbf{y}} \bar{\mathbf{y}}^\top) + \hat{\varepsilon}^\top \hat{\varepsilon} \\
&= SSP_H + SSP_E \\
&\equiv \mathbf{H} + \mathbf{E} ,
\end{aligned}$$

{#eq-SSP}

where,

- $\bar{\mathbf{y}}$ is the $(p \times 1)$ vector of means for the response variables;
- $\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}}$ is the matrix of fitted values; and
- $\hat{\varepsilon} = \mathbf{Y} - \hat{\mathbf{Y}}$ is the matrix of residuals.

We can visualize this decomposition in the simple case of a two-group design (for the `mathscore` data in

Section 9.2) as shown in Figure 10.2. Let \mathbf{y}_{ij} be the vector of p responses for subject j in group i , $i = 1, \dots, g$ for $j = 1, \dots, n_i$. Then, using \cdot to represent a subscript averaged over, $\mathbf{?@eq-SSP}$ comes from the identity

$$\underbrace{(\mathbf{y}_{ij} - \bar{\mathbf{y}}_{..})}_{T} = \underbrace{(\bar{\mathbf{y}}_{i\cdot} - \bar{\mathbf{y}}_{..})}_{H} + \underbrace{(\mathbf{y}_{ij} - \bar{\mathbf{y}}_{i\cdot})}_{E} \quad (10.2)$$

where each side of Equation 10.2 is squared and summed over observations to give $\mathbf{?@eq-SSP}$. In Figure 10.2,

- The total variance \mathbf{SSP}_T reflects the deviations of the observations \mathbf{y}_{ij} from the grand mean $\bar{\mathbf{y}}_{..}$ and has the data ellipse shown in gray.
- In the middle \mathbf{SSP}_H panel, all the observations are represented at their group means, $\bar{\mathbf{y}}_{i\cdot}$, the fitted values. Their variance and covariance is then reflected by deviations of the group means (weighted for the number of observations per group) around the grand mean.
- The right \mathbf{SSP}_E panel then shows the residual variance, which is the variation of the observations \mathbf{y}_{ij} around their group means, $\bar{\mathbf{y}}_{i\cdot}$. Centering the two data ellipses at the centroid $\bar{\mathbf{y}}_{..}$ then gives the ellipse for the \mathbf{SSP}_E , also called the pooled within-group covariance matrix.

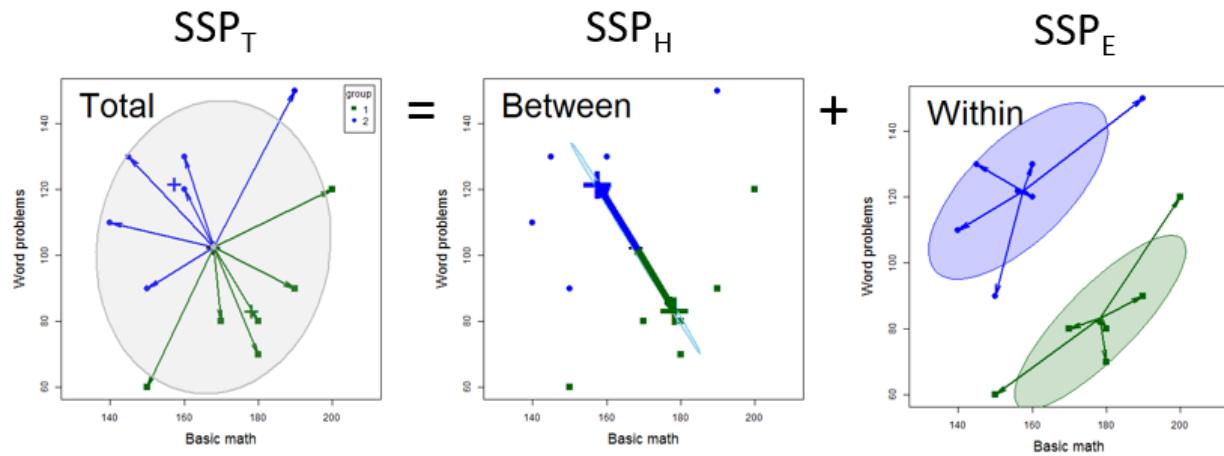


Figure 10.2: Breakdown of the total \mathbf{SSP}_T into sums of squares and products for between-group hypothesis variance (\mathbf{SSP}_H) and within-group, error variance (\mathbf{SSP}_E).

The formulas for these sum of squares and products matrices can be shown explicitly as follows, where the notation \mathbf{zz}^\top generates the $p \times p$ outer product of a vector \mathbf{z} , giving $z_k \times z_\ell$ for all pairs of elements.

$$\mathbf{SSP}_T = \sum_{i=1}^g \sum_{j=1}^{n_i} (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{..}) (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{..})^\top \quad (10.3)$$

$$\mathbf{SSP}_H = \sum_{i=1}^g \mathbf{n}_i (\bar{\mathbf{y}}_{i\cdot} - \bar{\mathbf{y}}_{..}) (\bar{\mathbf{y}}_{i\cdot} - \bar{\mathbf{y}}_{..})^\top \quad (10.4)$$

$$\mathbf{SSP}_E = \sum_{i=1}^g \sum_{j=1}^{n_i} (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{i\cdot}) (\mathbf{y}_{ij} - \bar{\mathbf{y}}_{i\cdot})^\top \quad (10.5)$$

This is the decomposition that we visualize in HE plots, where the size and direction of \mathbf{H} and \mathbf{E} can be represented as ellipsoids.

But first, let's find these results for the example. The easy way is to get them from the result returned by `car:::Anova()`, where the hypothesis \mathbf{SSP}_H for each term in the model is returned as an element in a named list `SSP` and the error \mathbf{SSP}_E is returned as the matrix `SSPE`.

```

SSP_H <- dogfood.aov$SSP |> print()
#> $formula
#>      start amount
#> start    9.69  -70.9
#> amount -70.94  585.7

SSP_E <- dogfood.aov$SSPE |> print()
#>      start amount
#> start   25.8   11.8
#> amount   11.8   390.3

```

You can calculate these directly as shown below. `sweep()` is used to subtract the `colMeans()` from \mathbf{Y} and $\hat{\mathbf{Y}}$ and `crossprod()` premultiplies a matrix by its' transpose.

```

Y <- dogfood[, c("start", "amount")]
Ydev <- sweep(Y, 2, colMeans(Y)) |> as.matrix()
SSP_T <- crossprod(as.matrix(Ydev)) |> print()
#>      start amount
#> start   35.4  -59.2
#> amount -59.2  975.9

fitted <- fitted(dogfood.mod)
Yfit <- sweep(fitted, 2, colMeans(fitted)) |> as.matrix()
SSP_H <- crossprod(Yfit) |> print()
#>      start amount
#> start    9.69  -70.9
#> amount -70.94  585.7

residuals <- residuals(dogfood.mod)
SSP_E <- crossprod(residuals) |> print()
#>      start amount
#> start   25.8   11.8
#> amount   11.8   390.3

```

The decomposition of the total sum of squares and products in ?@eq-SSP can be shown as:

$$\begin{pmatrix} \text{SSP}_T \\ 35.4 & -59.2 \\ -59.2 & 975.9 \end{pmatrix} = \begin{pmatrix} \text{SSP}_H \\ 9.69 & -70.94 \\ -70.94 & 585.69 \end{pmatrix} + \begin{pmatrix} \text{SSP}_E \\ 25.8 & 11.8 \\ 11.8 & 390.3 \end{pmatrix}$$

These numbers are the variances in the diagonal and covariance between start and amount in the off-diagonal, where the sign is important: In SSP_H the negative covariance reflects the fact that for these brands larger time to start eating is negatively related to the amount eaten. In SSP_E the covariance is slightly positive. This might reflect a mild hunger factor: for a given brand, dogs who lunge for their bowls sooner also eat more. But, they're all good boys, right?

10.2.3 How big is SS_H compared to SS_E ?

In a univariate response model, SS_H and SS_E are both scalar numbers and the univariate F test statistic,

$$F = \frac{SS_H/\text{df}_h}{SS_E/\text{df}_e} = \frac{\text{Var}(H)}{\text{Var}(E)}, \quad (10.6)$$

assesses “how big” SS_H is, relative to SS_E , the variance accounted for by a hypothesized model or model

terms relative to error variance. The measure $R^2 = SS_H/(SS_H + SS_E) = SS_H/SS_T$ gives the proportion of total variance accounted for by the model terms.

In the multivariate analog \mathbf{H} and \mathbf{E} are both $p \times p$ matrices, and \mathbf{H} “divided by” \mathbf{E} becomes \mathbf{HE}^{-1} . The answer, “how big” SS_H is compared to SS_E is expressed in terms of the p eigenvalues $\lambda_i, i = 1, 2, \dots, p$ of \mathbf{HE}^{-1} . These are the p values λ which solve the determinant equation

$$\det(\mathbf{HE}^{-1} - \lambda \mathbf{I}) = 0 .$$

The solution also gives the λ_i as the eigenvalues, with vectors \mathbf{v}_i as the corresponding eigenvectors,

$$\mathbf{HE}^{-1} \lambda_i = \lambda_i \mathbf{v}_i . \quad (10.7)$$

This can also be expressed in terms of the size of \mathbf{H} relative to total variation ($\mathbf{H} + \mathbf{E}$) as

$$\mathbf{H}(\mathbf{H} + \mathbf{E})^{-1} \rho_i = \rho_i \mathbf{v}_i , \quad (10.8)$$

which has the same eigenvectors as Equation 10.7 and the eigenvalues are $\rho_i = \lambda_i/(1 + \lambda_i)$.

However, when the hypothesized model terms have df_h degrees of freedom (columns of the \mathbf{X} matrix for that term), \mathbf{H} is of rank df_h , so only $s = \min(p, df_h)$ eigenvalues can be non-zero. For example, a test for a hypothesis about a single quantitative predictor \mathbf{x} , has $df_h = 1$ degree of freedom and $\text{rank}(\mathbf{H}) = 1$; for a factor with g groups, $df_h = \text{rank}(\mathbf{H}) = g - 1$.

For the `dogfood` data, we get the following results:

```
HEinv <- SSP_H %*% solve(SSP_E) |> print()
#>           start amount
#> start    0.466 -0.196
#> amount   -3.488  1.606
eig <- eigen(HEinv)
eig$values
#> [1] 2.0396 0.0317

# as proportions
eig$values / sum(eig$values)
#> [1] 0.9847 0.0153
```

The factor `formula` has four levels and therefore $df_h = 3$ degrees of freedom. But there are only $p = 2$ responses, so there are $s = \min(p, df_h) = 2$ eigenvalues (and corresponding eigenvectors). The eigenvalues tell us that 98.5% of the hypothesis variance due to `formula` can be accounted for by a single dimension.

The overall multivariate test for the model in Equation 10.1 is essentially a test of the hypothesis $\mathcal{H}_0 : \mathbf{B} = 0$ (excluding the row for the intercept). Equivalently, this is a test based on the *incremental* \mathbf{SSP}_H for the hypothesized terms in the model—that is, the difference between the \mathbf{SSP}_H for the full model and the null, intercept-only model. The same idea can be applied to test the difference between any pair of *nested* models—the added contribution of terms in a larger model relative to a smaller model containing a subset of terms.

The eigenvectors \mathbf{v}_i in Equation 10.7 are also important. These are the weights for the variables in a linear combination $v_{i1}\mathbf{y}_1 + v_{i2}\mathbf{y}_2 + \dots + v_{ip}\mathbf{y}_p$ which produces the largest univariate F statistic for the i -th dimension. We exploit this in canonical discriminant analysis and the corresponding canonical HE plots (Section 11.7).

The eigenvectors of \mathbf{HE}^{-1} for the `dogfood` model are shown below:

Table 10.1: Test statistics for multivariate tests combine the size of dimensions of $\mathbf{H}\mathbf{E}^{-1}$ into a single measure.

Criterion	Formula	Partial η^2
Wilks's Λ	$\Lambda = \prod_i^s \frac{1}{1+\lambda_i}$	$\eta^2 = 1 - \Lambda^{1/s}$
Pillai trace	$V = \sum_i^s \frac{\lambda_i}{1+\lambda_i}$	$\eta^2 = \frac{V}{s}$
Hotelling-Lawley trace	$H = \sum_i^s \lambda_i$	$\eta^2 = \frac{H}{H+s}$
Roy maximum root	$R = \lambda_1$	$\eta^2 = \frac{\lambda_1}{1+\lambda_1}$

```
rownames(eig$vectors) <- rownames(HEinv)
colnames(eig$vectors) <- paste("Dim", 1:2)
eig$vectors
#>           Dim 1  Dim 2
#> start    0.123 -0.411
#> amount   -0.992 -0.911
```

The first column corresponds to the weighted sum $0.12 \times \text{start} - 0.99 \times \text{amount}$, which as we saw above accounts for 95.5% of the differences in the group means.

10.3 Multivariate test statistics

In the univariate case, the overall F -test of $\mathcal{H}_0 : \boldsymbol{\beta} = \mathbf{0}$ is the uniformly most powerful invariant test when the assumptions are met. There is nothing better. This is not the case in the MLM.

The reason is that when there are $p > 1$ response variables, and we are testing a hypothesis comprising $\text{df}_h > 1$ coefficients or degrees of freedom, there are $s > 1$ possible dimensions in which \mathbf{H} can be large relative to \mathbf{E} , each measured by the eigenvalue λ_i . There are several test statistics that combine these into a single measure, shown in Table 10.1.

These correspond to different kinds of “means” of the λ_i : geometric (Wilks), arithmetic (Pillai), harmonic (Hotelling-Lawley) and supremum (Roy). See Friendly et al. (2013) for the geometry behind these measures.

Each of these statistics have different sampling distributions under the null hypothesis. But conveniently they can all be converted to F statistics. These conversions are exact when the hypothesis has $s \leq 2$ degrees of freedom, and approximations otherwise.

As well, each has an analog of the R^2 -like partial η^2 measure, giving the partial association accounted for by each term in the MLM. These reflect the proportion of total variation attributable to a given model term, partialling out (excluding) other factors from the total non-error variation. They can be used as a measure of effect size in a MLM.

10.3.1 Testing contrasts and linear hypotheses

Even more generally, these multivariate tests apply to *every* linear hypothesis concerning the coefficients in \mathbf{B} . Suppose we want to test the hypothesis that a subset of rows (predictors) and/or columns (responses) simultaneously have null effects. This can be expressed in the general linear test,

$$\mathcal{H}_0 : \mathbf{C}_{h \times q} \mathbf{B}_{q \times p} = \mathbf{0}_{h \times p},$$

where \mathbf{C} is a full rank $h \leq q$ hypothesis matrix of constants, that selects subsets or linear combinations (contrasts) of the coefficients in \mathbf{B} to be tested in a h degree-of-freedom hypothesis.

In this case, the SSP matrix for the hypothesis has the form

$$\mathbf{H} = (\widehat{\mathbf{CB}})^T [\mathbf{C}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{C}^T]^{-1} (\widehat{\mathbf{CB}}) , \quad (10.9)$$

where there are $s = \min(h, p)$ non-zero eigenvalues of \mathbf{HE}^{-1} . In Equation 10.9, \mathbf{H} measures the (Mahalanobis) squared distances (and cross products) among the linear combinations $\widehat{\mathbf{CB}}$ from the origin under the null hypothesis.

For example, with three responses y_1, y_2, y_3 and three predictors x_1, x_2, x_3 , we can test the hypothesis that neither x_1 nor x_2 contribute at all to predicting the y ’s in terms of the hypothesis that the coefficients for the corresponding rows of \mathbf{B} are zero using a 1-row \mathbf{C} matrix that simply selects those rows:

$$\begin{aligned} \mathcal{H}_0 : \mathbf{CB} &= [0 \ 1 \ 1 \ 0] \begin{pmatrix} \beta_{0,y_1} & \beta_{0,y_2} & \beta_{0,y_3} \\ \beta_{1,y_1} & \beta_{1,y_2} & \beta_{1,y_3} \\ \beta_{2,y_1} & \beta_{2,y_2} & \beta_{2,y_3} \\ \beta_{3,y_1} & \beta_{3,y_2} & \beta_{3,y_3} \end{pmatrix} \\ &= \begin{bmatrix} \beta_{1,y_1} & \beta_{1,y_2} & \beta_{1,y_3} \\ \beta_{2,y_1} & \beta_{2,y_2} & \beta_{2,y_3} \end{bmatrix} = \mathbf{0}_{(2 \times 3)} \end{aligned}$$

In MANOVA designs, it is often desirable to follow up a significant effect for a factor with subsequent tests to determine which groups differ. While you can simply test for all pairwise differences among groups (using Bonferroni or other corrections for multiplicity), a more substantively-driven approach uses planned comparisons or *contrasts* among the factor levels as described in Section 5.1.3.

For a factor with g groups, a contrast is simply a comparison of the mean of one subset of groups against the mean of another subset. This is specified as a weighted sum, L of the means with weights \mathbf{c} that sum to zero,

$$L = \mathbf{c}^T \boldsymbol{\mu} = \sum_i c_i \mu_i \quad \text{such that} \quad \sum c_i = 0$$

Two contrasts, \mathbf{c}_1 and \mathbf{c}_2 are *orthogonal* if the sum of products of their weights is zero, i.e., $\mathbf{c}_1^T \mathbf{c}_2 = \sum c_{1i} c_{2i} = 0$. When contrasts are placed as columns of a matrix \mathbf{C} , they are all *mutually orthogonal* if each pair is orthogonal, which means $\mathbf{C}^T \mathbf{C}$ is a diagonal matrix. Orthogonal contrasts correspond to statistically independent tests. This is nice because they reflect separate, non-overlapping research questions. When these questions are posed *a priori*, in advance of analysis there is no need to correct for multiple testing, on the grounds that you shouldn’t be penalized for having more ideas!

For example, with the $g = 4$ groups for the `dogfood` data, the company might want to test the following comparisons among the formulas Old, New, Major and Alps: (a) Ours vs. Theirs: The average of (Old, New) compared to (Major, Alps); (b) Old vs. New; (c) Major vs. Alps. The contrasts that do this are:

$$\begin{aligned} L_1 &= \frac{1}{2}(\mu_O + \mu_N) - \frac{1}{2}(\mu_M + \mu_A) & \rightarrow \mathbf{c}_1 &= \frac{1}{2}(1 \ 1 \ -1 \ -1) \\ L_2 &= \mu_O - \mu_N & \rightarrow \mathbf{c}_2 &= (1 \ -1 \ 0 \ 0) \\ L_3 &= \mu_M - \mu_A & \rightarrow \mathbf{c}_3 &= (0 \ 0 \ 1 \ -1) \end{aligned}$$

Note that these correspond to *nested dichotomies* among the four groups: first we compare groups (Old and New) against groups (Major and Alps), then subsequently within each of these sets. Nested dicontomy contrasts are always *orthogonal*, and therefore correspond to statistically independent tests. We are effectively taking a three degree-of-freedom question, “do the means differ?” and breaking it down into three separate 1 df tests that answer specific parts of that overall question.

In R, contrasts for a factor are specified as columns of matrix, each of which sums to zero. For this example, we can set this up by creating each as a vector and joining them as columns using `cbind()`:

```

c1 <- c(1, 1, -1, -1)/2    # Old,New vs. Major,Alps
c2 <- c(1, -1, 0, 0)       # Old vs. New
c3 <- c(0, 0, 1, -1)       # Major vs. Alps
C <- cbind(c1,c2,c3)
rownames(C) <- levels(dogfood$formula)

C
#>      c1 c2 c3
#> Old   0.5 1 0
#> New   0.5 -1 0
#> Major -0.5 0 1
#> Alps  -0.5 0 -1

# show they are mutually orthogonal
t(C) %*% C
#>      c1 c2 c3
#> c1  1 0 0
#> c2  0 2 0
#> c3  0 0 2

```

For the `dogfood` data, with `formula` as the group factor, you can set up the analyses to use these contrasts by assigning the matrix `C` to `contrasts()` for that factor in the dataset itself. When the contrasts are changed, it is necessary to refit the model. The estimated coefficients then become the estimated mean differences for the contrasts.

```

contrasts(dogfood$formula) <- C
dogfood.mod <- lm(cbind(start, amount) ~ formula,
                   data=dogfood)
coef(dogfood.mod)
#>           start amount
#> (Intercept) 1.688 85.56
#> formulac1   -1.375 10.88
#> formulac2   -0.500  3.25
#> formulac3    0.125  1.88

```

For example, Ours vs. Theirs estimated by `formulac1` takes 0.69 less time to start eating and eats 5.44 more on average.

For multivariate tests, when all contrasts are pairwise orthogonal, the overall test of a factor with $df_h = g - 1$ degrees of freedom can be broken down into $g - 1$ separate 1 df tests. This gives rise to a set of df_h rank 1 \mathbf{H} matrices that additively decompose the overall hypothesis SSCP matrix,

$$\mathbf{H} = \mathbf{H}_1 + \mathbf{H}_2 + \cdots + \mathbf{H}_{df_h}, \quad (10.10)$$

exactly as the univariate SS_H can be decomposed using orthogonal contrasts in an ANOVA (Section 5.1.3)

You can test such contrasts or any other hypotheses involving linear combinations of the coefficients using `car::linearHypothesis()`. Here, "`formulac1`" refers to the contrast `c1` for the difference between Ours and Theirs. Note that because this is a 1 df test, all four test statistics yield the same F values.

```

hyp <- rownames(coef(dogfood.mod))[-1] |> print()
#> [1] "formulac1" "formulac2" "formulac3"
H1 <- linearHypothesis(dogfood.mod, hyp[1],

```

```

    title="Ours vs. Theirs") |>
print()
#>
#> Sum of squares and products for the hypothesis:
#>      start amount
#> start     7.56 -59.8
#> amount -59.81 473.1
#>
#> Sum of squares and products for error:
#>      start amount
#> start   25.8   11.7
#> amount  11.7   390.3
#>
#> Multivariate Tests: Ours vs. Theirs
#>              Df test stat approx F num Df den Df Pr(>F)
#> Pillai        1   0.625    9.18      2     11 0.0045 ***
#> Wilks         1   0.375    9.18      2     11 0.0045 ***
#> Hotelling-Lawley 1   1.669    9.18      2     11 0.0045 ***
#> Roy           1   1.669    9.18      2     11 0.0045 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Similarly we can test the other two contrasts within these each of these two subsets, but I don't print the results

```

H2 <- linearHypothesis(dogfood.mod, hyp[2],
                        title="Old vs. New")
H3 <- linearHypothesis(dogfood.mod, hyp[3],
                        title="Alps vs. Major")

```

Then, we can illustrate Equation 10.10 by extracting the 1 df **H** matrices (SSPH) from the results of `linearHypothesis`.

$$\begin{pmatrix} 9.7 & \mathbf{H} \\ -70.9 & -70.9 \\ -70.9 & 585.7 \end{pmatrix} = \begin{pmatrix} 7.6 & \mathbf{H}_1 \\ -59.8 & -59.8 \\ -59.8 & 473.1 \end{pmatrix} + \begin{pmatrix} 0.13 & \mathbf{H}_2 \\ 1.88 & 1.88 \\ 1.88 & 28.12 \end{pmatrix} + \begin{pmatrix} 2 & \mathbf{H}_3 \\ -13 & -13 \\ -13 & 84 \end{pmatrix}$$

10.4 ANOVA → MANOVA

Multivariate analysis of variance (MANOVA) generalizes the familiar ANOVA model to situations where there are two or more response variables. Unlike ANOVA, which focuses on discerning statistical differences in one continuous dependent variable influenced by an independent variable (or grouping variable), MANOVA considers several dependent variables at once. It integrates these variables into a single, composite variable through a weighted linear combination, allowing for a comprehensive analysis of how these dependent variables collectively vary with respect to the levels of the independent variable. Essentially, MANOVA investigates whether the grouping variable explains significant variations in the combined dependent variables.

The situation is illustrated in Figure 10.3 where there are two response measures, Y_1 and Y_2 with data collected for three groups. For concreteness, Y_1 might be a score on a math test and Y_2 might be a reading score. Let's also say that group 1 has been studying Shakespeare, while group 2 has concentrated on physics, but group 3 has done nothing beyond the normal curriculum.

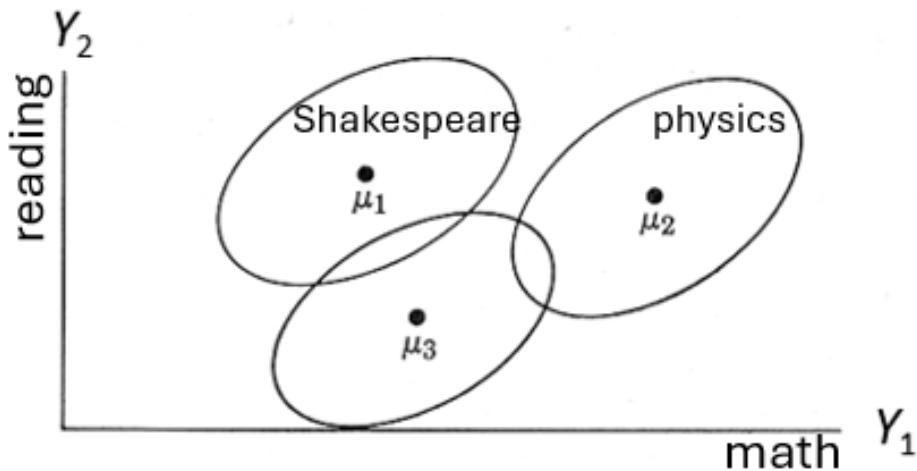


Figure 10.3: Data from simple MANOVA design involving three groups and two response measures, Y_1 and Y_2 , summarized by their data ellipses.

As shown in the figure, the centroids, (μ_{1g}, μ_{2g}) , clearly differ—the data ellipses barely overlap. A multivariate analysis would show a highly difference among groups. From a rough visual inspection, it seems that means differ on the math test Y_1 , with the physics group out-performing the other two. On the reading test Y_2 however it might turn out that the three group means don't differ significantly in an ANOVA, but the Shakespeare and physics groups appear to outperform the normal curriculum group. Doing separate ANOVAs on these variables would miss what is so obvious from Figure 10.3: there is wide separation among the groups in the two tests considered *jointly*.

Figure 10.4 illustrates a second important advantage of performing a multivariate analysis over separate ANOVAS: that of determining the *number of dimensions* or aspects along which groups differ. In the panel on the left, the means of the three groups increase nearly linearly on the combination of Y_1 and Y_2 , so their differences can be ascribed to a single dimension, which simplifies the interpretation: *both* memory and attention scores decrease together with degree of schizophrenia.

For example, the groups here might be patients diagnosed as normal, mild schizophrenia and profound schizophrenia, and the measures could be tests of memory and attention. The obvious multivariate interpretation from the figure is that of increasing impairment of cognitive functioning across the groups, comprised by memory and attention. Note also the positive association *within* each group: those who perform better on the memory task also do better on attention.

In contrast, the right panel of Figure 10.4 shows a situation where the group means have a low correlation. Data like this might arise in a study of parental competency, where there are measures of both the degree of caring (Y_1) and time spent in play (Y_2) by fathers and groups consisting of fathers of children with no disability, or a physical disability or a mental ability.

As can be seen in Figure 10.4 fathers of the disabled children differ from those of the not disabled group in two different directions corresponding to being higher on either Y_1 or Y_2 . The red arrows suggest that the differences among groups could be interpreted in terms of two uncorrelated dimensions, perhaps labeled overall competency and emphasis on physical activity. (The pattern in Figure 10.4 (right) is contrived for the sake of illustration; it does not reflect the data analyzed in the example below.)

10.4.1 Example: Father parenting data

I use a simple example of a three-group multivariate design to illustrate the basic ideas of fitting MLMs in R and testing hypotheses. Visualization methods using HE plots are discussed in Chapter 11.

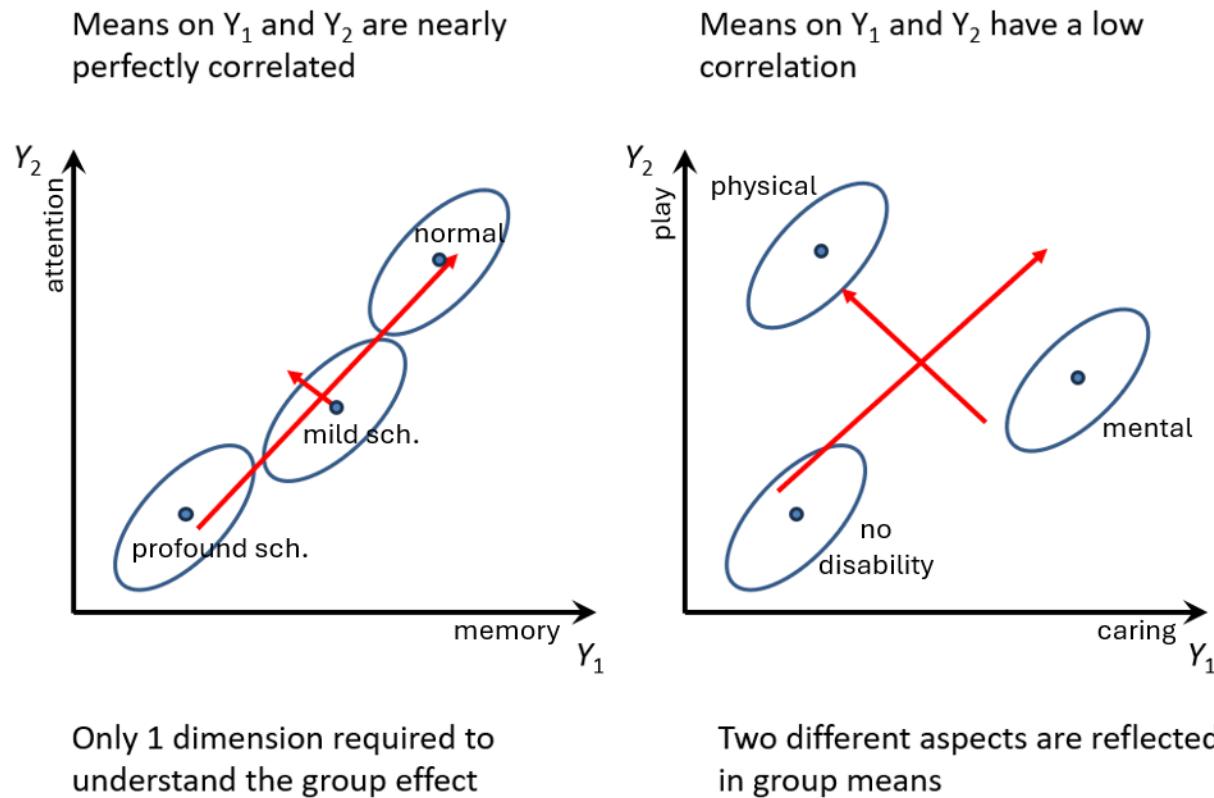


Figure 10.4: A simple MANOVA design involving three groups and two response measures, Y_1 and Y_2 , but with different patterns of the differences among the group means. The red arrows suggest interpretations in terms of dimensions or aspects of the response variables.

The dataset `Parenting` come from an exercise (10B) in Meyers et al. (2006) and are probably contrived, but are modeled on a real study in which fathers were assessed on three subscales of a *Perceived Parenting Competence Scale*,

- `caring`, caretaking responsibilities;
- `emotion`, emotional support provided to the child; and
- `play`, recreational time spent with the child.

The dataset `Parenting` comprises 60 fathers selected from three groups of $n = 20$ each: (a) fathers of a child with *no disabilities* ("Normal"); (b) fathers with a *physically* disabled child; (c) fathers with a *mentally* disabled child. The design is thus a three-group MANOVA, with three response variables.

The main questions concern whether the group means differ on these scales, and the nature of these differences. That is, do the means differ significantly on all three measures? Is there a consistent order of groups across these three aspects of parenting?

More specific questions are: (a) Do the fathers of typical children differ from the other two groups on average? (b) Do the physical and mental groups differ? These questions can be tested using contrasts, and are specified by assigning a matrix to `contrasts(Parenting$group)`; each column is a contrast whose values sum to zero. They are given labels "group1" (normal vs. other) and "group2" (physical vs. mental) in some output.

```
data(Parenting, package="heplots")
C <- matrix(c(1, -.5, -.5,
              0,  1, -1),
```

```

nrow = 3, ncol = 2) |> print()
#>      [,1] [,2]
#> [1,]  1.0   0
#> [2,] -0.5   1
#> [3,] -0.5  -1
contrasts(Parenting$group) <- C

```

Exploratory plots

Before setting up a model and testing, it is well-advised to examine the data graphically. The simplest plots are side-by-side boxplots (or violin plots) for the three responses. With `ggplot2`, this is easily done by reshaping the data to long format and using faceting. In Figure 10.5, I've also plotted the group means with white dots.

```

parenting_long <- Parenting |>
  tidyr::pivot_longer(cols=caring:play,
                      names_to = "variable")

ggplot(parenting_long,
       aes(x=group, y=value, fill=group)) +
  geom_boxplot(outlier.size=2.5,
               alpha=.5,
               outlier.alpha = 0.9) +
  stat_summary(fun=mean,
               color="white",
               geom="point",
               size=2) +
  scale_fill_hue(direction = -1) +      # reverse default colors
  labs(y = "Scale value", x = "Group") +
  facet_wrap(~ variable) +
  theme_bw(base_size = 14) +
  theme(legend.position="top") +
  theme(axis.text.x = element_text(angle = 15,
                                    hjust = 1))

```

In this figure, differences among the groups on `play` are most apparent, with fathers of non-disabled children scoring highest. Differences among the groups on `emotion` are very small, but one high outlier for the fathers of mentally disabled children is apparent. On `caring`, fathers of children with a physical disability stand out as highest.

For exploratory purposes, you might also make a scatterplot matrix. Here, because the MLM assumes homogeneity of the variances and covariance matrices \mathbf{S}_i , I show only the data ellipses in scatterplot matrix format, using `heplots:covEllipses()` (with 50% coverage, for clarity):

```

colors <- scales::hue_pal()(3) |> rev() # match color use in ggplot
covEllipses(cbind(caring, play, emotion) ~ group,
            data=Parenting,
            variables = 1:3,
            fill = TRUE, fill.alpha = 0.2,
            pooled = FALSE,
            level = 0.50,
            col = colors)

```

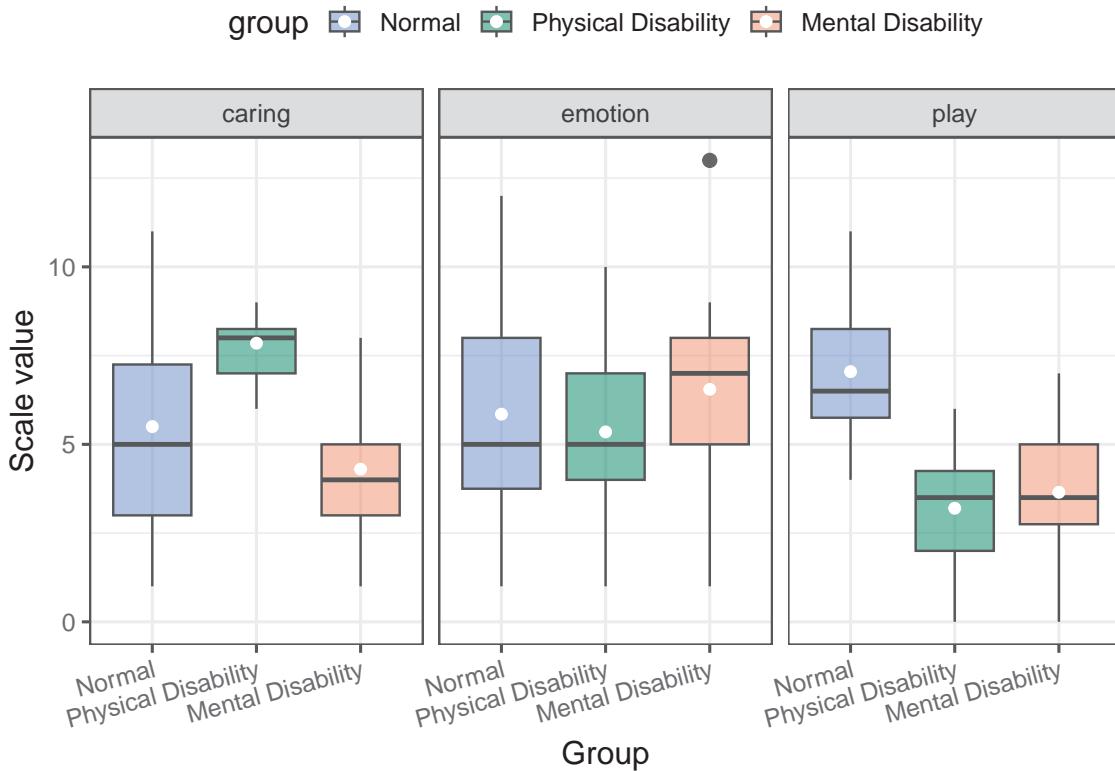


Figure 10.5: Faceted boxplots of scores on the three parenting scales, showing also the mean for each.

If the covariance matrices were all the same, the data ellipses would have roughly the same size and orientation, but that is not the case in Figure 10.6. The normal group shows greater variability overall and the correlations among the measures differ somewhat from group to group. We'll assess later whether this makes a difference in the conclusions that can be drawn (Chapter 12). The group centroids also differ, but the pattern is not particularly clear. We'll see an easier to understand view in HE plots and their canonical discriminant cousins.

10.4.1.1 Testing the model

Let's proceed to fit the multivariate model predicting all three scales from the `group` factor. `lm()` for a multivariate response returns an object of class "`m1m`", for which there are many methods (use `methods(class="m1m")` to find them).

```
parenting.m1m <- lm(cbind(caring, play, emotion) ~ group,
                      data=Parenting) |> print()
#>
#> Call:
#> lm(formula = cbind(caring, play, emotion) ~ group, data = Parenting)
#>
#> Coefficients:
#>            caring     play     emotion
#> (Intercept) 5.8833  4.6333  5.9167
#> group1      -0.3833  2.4167 -0.0667
#> group2       1.7750 -0.2250 -0.6000
```

The coefficients in this model are the values of the contrasts set up above. `group1` is the mean of the typical

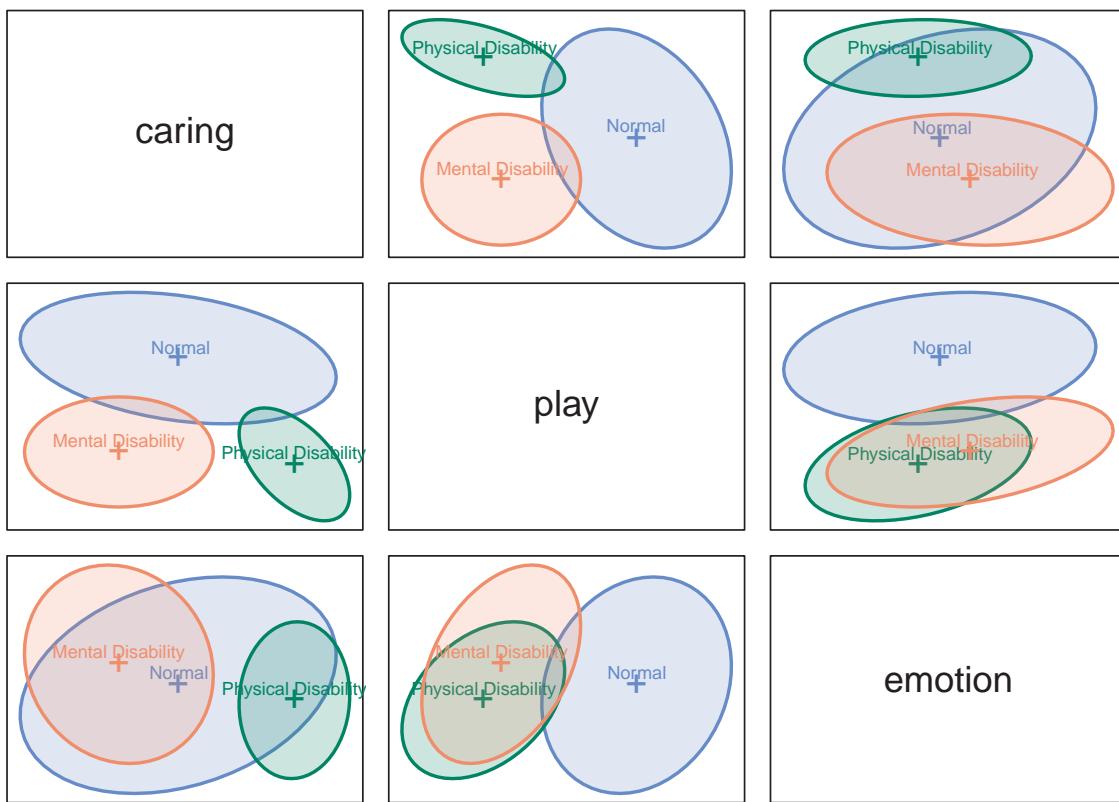


Figure 10.6: Bivariate data ellipses for pairs of the three responses, showing the means, correlations and variances for the three groups.

group minus the average of the other two, which is negative on `caring` and `emotion` but positive for `play`. `group2` is the difference in means for the physical vs. mental groups.

Before doing multivariate tests, it is useful to see what would happen if we ran univariate ANOVAs on each of the responses. These can be extracted from an MLM using `stats:::summary.aov()` and they give tests of the model terms for each response variable separately:

```
summary.aov(parenting.mlm)
#> Response caring :
#>              Df Sum Sq Mean Sq F value Pr(>F)
#> group          2    130    65.2    18.6  6e-07 ***
#> Residuals      57    200     3.5
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Response play :
#>              Df Sum Sq Mean Sq F value Pr(>F)
#> group          2    177    88.6    27.6  4e-09 ***
#> Residuals      57    183     3.2
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Response emotion :
```

```
#>           Df Sum Sq Mean Sq F value Pr(>F)
#> group        2    15   7.27    1.02   0.37
#> Residuals    57   408   7.16
```

For a more condensed summary, you can instead extract the univariate model fit statistics from the "`m1m`" object using the `heplots::glance()` method for a multivariate model object. The code below selects just the R^2 and F -statistic for the overall model for each response, together with the associated p -value.

```
glance(parenting.m1m) |>
  select(response, r.squared, fstatistic, p.value)
#> # A tibble: 3 x 4
#>   response r.squared fstatistic      p.value
#>   <chr>     <dbl>       <dbl>       <dbl>
#> 1 caring     0.395      18.6 0.000000602
#> 2 play       0.492      27.6 0.00000000405
#> 3 emotion    0.0344     1.02 0.369
```

From this, one might conclude that there are differences only in `caring` and `play` and therefore ignore `emotion`, but this would be short-sighted. `car::Anova()`, shown below, gives the overall multivariate test $\mathcal{H}_0 : \mathbf{B} = 0$ of the `group` effect, that the groups don't differ on *any* of the response variables. Note that this has a much smaller p -value than any of the univariate F tests.

```
Anova(parenting.m1m)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> group    2    0.948     16.8      6    112  9e-14 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`Anova()` returns an object of class "Anova.m1m" which has various methods. The `summary()` method for this gives more details, including all four test statistics. With $p = 3$ responses, these tests have $s = \min(p, df_h) = \min(3, 2) = 2$ dimensions and the F approximations are *not* equivalent here. All four tests are highly significant, with Roy's test giving the largest F statistic.

```
parenting.summary <- Anova(parenting.m1m) |>  summary()
print(parenting.summary, SSP=FALSE)
#>
#> Type II MANOVA Tests:
#>
#> -----
#>
#> Term: group
#>
#> Multivariate Tests: group
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai        2    0.948     16.8      6    112 9.0e-14 ***
#> Wilks         2    0.274     16.7      6    110 1.3e-13 ***
#> Hotelling-Lawley 2    1.840     16.6      6    108 1.8e-13 ***
#> Roy          2    1.108     20.7      3     56 3.8e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `summary()` method by default prints the $\text{SSH} = \mathbf{H}$ and $\text{SSE} = \mathbf{E}$ matrices, but I suppressed them above. The data structure returned contains nested elements which can be extracted more easily from the object using `purrr::pluck()`:

```
H <- parenting.summary |>
  purrr::pluck("multivariate.tests", "group", "SSPH") |>
  print()
#>      caring play emotion
#> caring  130.4 -43.767 -41.833
#> play    -43.8 177.233  0.567
#> emotion -41.8   0.567 14.533
E <- parenting.summary |>
  purrr::pluck("multivariate.tests", "group", "SSPE") |>
  print()
#>      caring play emotion
#> caring  199.8 -45.8    35.2
#> play    -45.8 182.7   80.6
#> emotion 35.2   80.6  408.0
```

Linear hypotheses & contrasts

With three or more groups or with a more complex MANOVA design, contrasts provide a way of testing questions of substantive interest regarding differences among group means.

The test of the contrast comparing the typical group to the average of the others is the test using the contrast $c_1 = (1, -\frac{1}{2}, -\frac{1}{2})$ which produces the coefficients labeled "`group1`". The function `car::linearHypothesis()` carries out the multivariate test that this difference is zero. This is a 1 df test, so all four test statistics produce the same F and p -values.

```
coef(parenting.mlm)[["group1",]]
#>      caring play emotion
#> -0.3833 2.4167 -0.0667
linearHypothesis(parenting.mlm, "group1") |>
  print(SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai        1    0.521    19.9     3    55 7.1e-09 ***
#> Wilks         1    0.479    19.9     3    55 7.1e-09 ***
#> Hotelling-Lawley 1    1.088    19.9     3    55 7.1e-09 ***
#> Roy          1    1.088    19.9     3    55 7.1e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Similarly, the difference between the physical and mental groups uses the contrast $c_2 = (0, 1, -1)$ and the test that these means are equal is given by `linearHypothesis()` applied to `group2`.

```
coef(parenting.mlm)[["group2",]]
#>      caring play emotion
#>     1.775 -0.225 -0.600
linearHypothesis(parenting.mlm, "group2") |>
  print(SSP=FALSE)
#>
```

```
#> Multivariate Tests:
#>
#> Df test stat approx F num Df den Df Pr(>F)
#> Pillai 1 0.429 13.8 3 55 8e-07 ***
#> Wilks 1 0.571 13.8 3 55 8e-07 ***
#> Hotelling-Lawley 1 0.752 13.8 3 55 8e-07 ***
#> Roy 1 0.752 13.8 3 55 8e-07 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`linearHypothesis()` is very general. The second argument (`hypothesis.matrix`) corresponds to **C**, and can be specified as numeric matrix giving the linear combinations of coefficients by rows to be tested, or a character vector giving the hypothesis in symbolic form; "`group1`" is equivalent to "`group1 = 0`".

Because the two contrasts used here are orthogonal, they add together to give the overall test of **B = 0**, which implies that the means of the three groups are all equal. The test below gives the same results as `Anova(parenting.mlm)`.

```
linearHypothesis(parenting.mlm, c("group1", "group2")) |>
  print(SSP=FALSE)
#>
#> Multivariate Tests:
#>
#> Df test stat approx F num Df den Df Pr(>F)
#> Pillai 2 0.948 16.8 6 112 9.0e-14 ***
#> Wilks 2 0.274 16.7 6 110 1.3e-13 ***
#> Hotelling-Lawley 2 1.840 16.6 6 108 1.8e-13 ***
#> Roy 2 1.108 20.7 3 56 3.8e-09 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

10.4.2 Ordered factors

When groups are defined by an ordered factor, such as level of physical fitness (rated 1–5) or grade in school, it is tempting to treat that as a numeric variable and use a multivariate regression model. This would assume that the effect of that factor is linear and if not, we might consider adding polynomial terms. A different strategy, often preferable, is to make the group variable an *ordered factor*, for which R assigns *polynomial contrasts*. This gives separate tests of the linear, quadratic, cubic, ... trends of the response, without the need to specify them separately in the model.

10.4.3 Example: Adolescent mental health

The dataset `AddHealth` contains a large cross-sectional sample of participants from grades 7–12 from the National Longitudinal Study of Adolescent Health, described by Warne (2014). It contains responses to two Likert-scale (1–5) items, `anxiety` and `depression`. `grade` is an *ordered factor*, which means that the default contrasts are taken as orthogonal polynomials with linear (`grade.L`), quadratic (`grade.Q`), up to 5th degree (`grade^5`) trends, which decompose the total effect of grade.

```
data(AddHealth, package="heplots")
str(AddHealth)
#> 'data.frame': 4344 obs. of 3 variables:
#> $ grade : Ord.factor w/ 6 levels "7"<"8"<"9"<"10"<...: 5 4 6 1 2 2 2 3 3 3 ...
```

```
#> $ depression: int 0 0 0 0 0 0 0 1 2 ...
#> $ anxiety    : int 0 0 0 1 1 0 0 1 1 0 ...
```

The research questions are:

- How do the means for anxiety and depression vary separately with grade? Is there evidence for linear and nonlinear trends?
- How do anxiety and depression vary *jointly* with grade?
- How does the *association* (correlation, R^2) of anxiety and depression vary with age?

The first question can be answered by fitting separate linear models for each response (e.g., `lm(anxiety ~ grade)`). However the second question is more interesting because it considers the two responses together and takes their correlation into account. This would be fit as the MLM:

$$\mathbf{y} = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 x + \boldsymbol{\beta}_2 x^2 + \cdots \boldsymbol{\beta}_5 x^5 \quad (10.11)$$

or, expressed in terms of the variables,

$$\begin{bmatrix} y_{\text{anx}} \\ y_{\text{dep}} \end{bmatrix} = \begin{bmatrix} \beta_{0,\text{anx}} \\ \beta_{0,\text{dep}} \end{bmatrix} + \begin{bmatrix} \beta_{1,\text{anx}} \\ \beta_{1,\text{dep}} \end{bmatrix} \text{grade} + \begin{bmatrix} \beta_{2,\text{anx}} \\ \beta_{2,\text{dep}} \end{bmatrix} \text{grade}^2 + \cdots + \begin{bmatrix} \beta_{5,\text{anx}} \\ \beta_{5,\text{dep}} \end{bmatrix} \text{grade}^5$$

{#eq-AH-mod2}

With `grade` represented as an ordered factor, the values of x in Equation 10.11 are those of the orthogonal polynomials given by `poly(grade, 5)`.

Exploratory plots

Some exploratory analysis is useful before fitting and visualizing models. As a first step, we find the means, standard deviations, and standard errors of the means.

```
means <- AddHealth |>
  group_by(grade) |>
  summarise(
    n = n(),
    dep_sd = sd(depression, na.rm = TRUE),
    anx_sd = sd(anxiety, na.rm = TRUE),
    dep_se = dep_sd / sqrt(n),
    anx_se = anx_sd / sqrt(n),
    depression = mean(depression),
    anxiety = mean(anxiety) ) |>
  relocate(depression, anxiety, .after = grade) |>
  print()
#> # A tibble: 6 x 8
#>   grade depression anxiety     n dep_sd anx_sd dep_se anx_se
#>   <ord>      <dbl>   <dbl> <int>  <dbl>  <dbl>  <dbl>  <dbl>
#> 1 7          0.881   0.751   622    1.11   1.05  0.0447  0.0420
#> 2 8          1.08    0.804   664    1.19   1.06  0.0461  0.0411
#> 3 9          1.17    0.934   778    1.19   1.08  0.0426  0.0387
#> 4 10         1.27    0.956   817    1.23   1.11  0.0431  0.0388
#> 5 11         1.37    1.12    790    1.20   1.16  0.0428  0.0411
#> 6 12         1.34    1.10    673    1.14   1.11  0.0439  0.0426
```

Now, plot the means with ± 1 error bars. It appears that average level of both depression and anxiety increase steadily with grade, except for grades 11 and 12 which don't differ much. Alternatively, we could describe this as relationships that seem largely linear, with a hint of curvature at the upper end.

```
p1 <- ggplot(data = means, aes(x = grade, y = anxiety)) +
  geom_point(size = 4) +
  geom_line(aes(group = 1), linewidth = 1.2) +
  geom_errorbar(aes(ymin = anxiety - anx_se,
                     ymax = anxiety + anx_se),
                width = .2)

p2 <- ggplot(data = means, aes(x = grade, y = depression)) +
  geom_point(size = 4) +
  geom_line(aes(group = 1), linewidth = 1.2) +
  geom_errorbar(aes(ymin = depression - dep_se,
                     ymax = depression + dep_se),
                width = .2)

p1 + p2
```

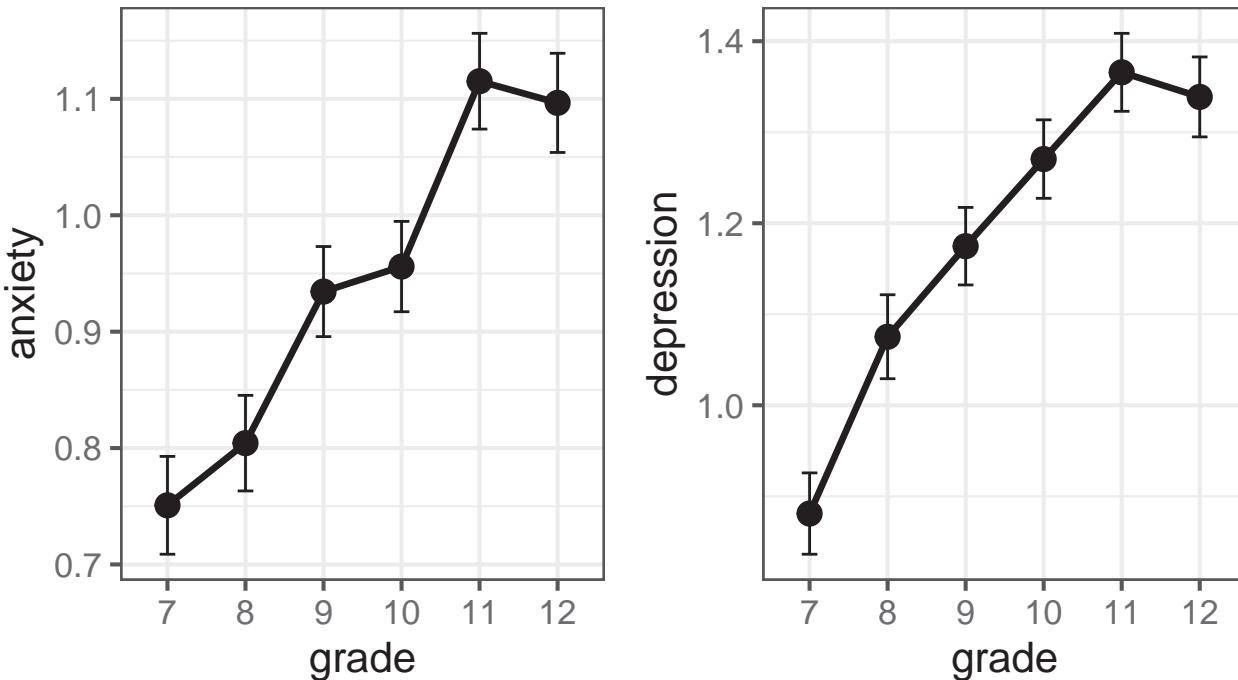


Figure 10.7: Means of anxiety and depression by grade, with ± 1 standard error bars.

It is also useful to within-group correlations using `covEllipses()`, as shown in Figure 10.8. This also plots the bivariate means showing the form of the association , treating anxiety and depression as multivariate outcomes. (Because the variability of the scores within groups is so large compared to the range of the means, I show the data ellipses with coverage of only 10%.)

```
covEllipses(AddHealth[, 3:2], group = AddHealth$grade,
            pooled = FALSE, level = 0.1,
```

```
center.cex = 2.5, cex = 1.5, cex.lab = 1.5,
fill = TRUE, fill.alpha = 0.05)
```

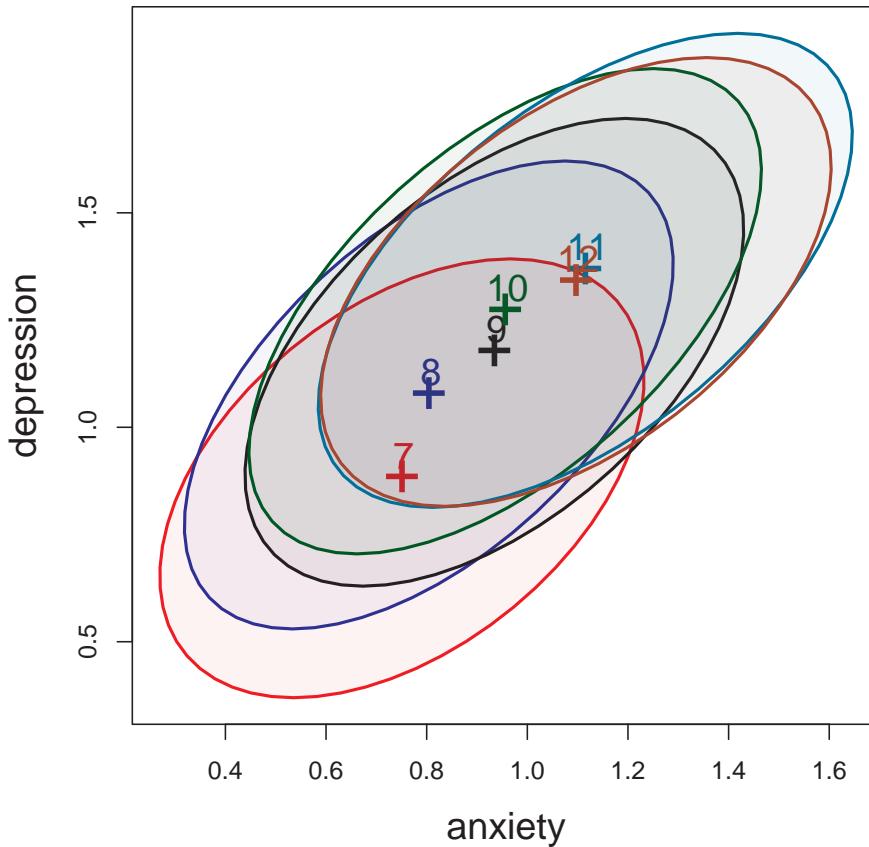


Figure 10.8: Within-group covariance ellipses for the `grade` groups.

Fit the MLM

Now, let's fit the MLM for both responses jointly in relation to `grade`. The null hypothesis is that the means for anxiety and depression are the same at all six grades,

$$\mathcal{H}_0 : \mu_7 = \mu_8 = \dots = \mu_{12} ,$$

or equivalently, that all coefficients except the intercept in the model Equation 10.11 are zero,

$$\mathcal{H}_0 : \beta_1 = \beta_2 = \dots = \beta_5 = \mathbf{0} .$$

We fit the MANOVA model, and test the grade effect using `car::Anova()`. The effect of `grade` is highly significant, as we could tell from Figure 10.7.

```
AH.mlm <- lm(cbind(anxiety, depression) ~ grade, data = AddHealth)

# overall test of `grade`
Anova(AH.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
```

```
#>      Df test stat approx F num Df den Df Pr(>F)
#> grade  5     0.0224    9.83     10    8676 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

However, the overall test, with 5 degrees of freedom is diffuse, in that it can be rejected if any pair of means differ. Given that `grade` is an ordered factor, it makes sense to examine narrower hypotheses of linear and nonlinear trends, `car::linearHypothesis()` on the coefficients of model `AH.mlm`.

```
coef(AH.mlm) |> rownames()
#> [1] "(Intercept)" "grade.L"      "grade.Q"      "grade.C"
#> [5] "grade^4"      "grade^5"
```

The joint test of the linear coefficients $\beta_1 = (\beta_{1,\text{anx}}, \beta_{1,\text{dep}})^T$ for anxiety and depression, $\mathcal{H}_0 : \beta_1 = \mathbf{0}$ is highly significant,

```
## linear effect
linearHypothesis(AH.mlm, "grade.L") |> print(SSP = FALSE)
#>
#> Multivariate Tests:
#>      Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1     0.019    42.5      2    4337 <2e-16 ***
#> Wilks       1     0.981    42.5      2    4337 <2e-16 ***
#> Hotelling-Lawley 1     0.020    42.5      2    4337 <2e-16 ***
#> Roy         1     0.020    42.5      2    4337 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test of the quadratic coefficients $\mathcal{H}_0 : \beta_2 = \mathbf{0}$ indicates significant curvature in trends across grade, as we saw in the plots of their means in Figure 10.7. One interpretation might be that depression and anxiety after increasing steadily up to grade eleven could level off thereafter.

```
## quadratic effect
linearHypothesis(AH.mlm, "grade.Q") |> print(SSP = FALSE)
#>
#> Multivariate Tests:
#>      Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1     0.002    4.24      2    4337  0.014 *
#> Wilks       1     0.998    4.24      2    4337  0.014 *
#> Hotelling-Lawley 1     0.002    4.24      2    4337  0.014 *
#> Roy         1     0.002    4.24      2    4337  0.014 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

An advantage of linear hypotheses is that we can test several terms *jointly*. Of interest here is the hypothesis that all higher order terms beyond the quadratic are zero, $\mathcal{H}_0 : \beta_3 = \beta_4 = \beta_5 = \mathbf{0}$. Using `linearHypothesis` you can supply a vector of coefficient names to be tested for their joint effect when dropped from the model.

```
coefs <- rownames(coef(AH.mlm)) |> print()
#> [1] "(Intercept)" "grade.L"      "grade.Q"      "grade.C"
#> [5] "grade^4"      "grade^5"
```

```

## joint test of all higher terms
linearHypothesis(AH.mlm, coefs[3:5],
                  title = "Higher-order terms") |>
  print(SSP = FALSE)
#>
#> Multivariate Tests: Higher-order terms
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      3   0.002    1.70      6   8676   0.12
#> Wilks       3   0.998    1.70      6   8674   0.12
#> Hotelling-Lawley 3   0.002    1.70      6   8672   0.12
#> Roy         3   0.002    2.98      3   4338   0.03 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

10.4.4 Factorial MANOVA

When there are two or more categorical factors, the general linear model provides a way to investigate the effects (differences in means) of each simultaneously. More importantly, this allows you to determine if factors *interact*, so the effect of one factor varies with the levels of another factor ...

Example: Penguins data

In Chapter 3 we examined the Palmer penguins data graphically, using a mosaic plot (Figure 3.36) of the frequencies of the three factors, `species`, `island` and `sex` and then `ggpairs()` scatterplot matrix (Figure 3.37).

TODO: Complete this

10.5 MRA → MMRA

When all predictor variables are quantitative, the MLM Equation 10.1 becomes the extension of univariate multiple regression analysis (MRA) to the situation where there are p response variables (RA). Just as in univariate models, we might want to test hypotheses about subsets of the predictors, for example when some predictors are meant as controls or things you might want to adjust for in assessing the effects of predictors of main interest.

But first, there are a couple of aspects of statistical practice that should be mentioned.

Model selection is one topic where univariate and multivariate approaches differ. When there are more than a few predictors, approaches like hierarchical regression, LASSO (Tibshirani, 1996) and stepwise selection can be used to eliminate uninformative predictors for each response.³ But this gives a different models for each response, based on the predictors included, each with its own interpretation. In contrast, the multivariate approach considers the outcome variables collectively. You can eliminate predictors that are unimportant, but the mechanics are geared toward removing them from the models for *all* responses.

Overall tests In a one-way ANOVA, to control for multiple testing, it is common practice to carry out an overall F -test to see if the group means differ collectively before testing comparison between specific groups.

³“Automatic” model selection procedures like stepwise regression, while seemingly attractive are dangerous in that can increase false positives or drop variables that should, on logical grounds, be included in the model. See [Stepwise selection of variables in regression is Evil](#) and Harrell ([Harrell, 2015](#)).

Similarly, in univariate multiple regression, researchers sometimes report an overall F -test or test of the R^2 so they can reject the hypothesis that *all* predictors have no effect, before considering them individually.

Similarly, in the case of multivariate linear models, some consider it necessary to reject the multivariate null hypothesis for a predictor term before considering how it contributes to each of the response variables. Some further suggest that the individual univariate models be tested after an overall significant effect. I believe the first of these is wise, but the second might be too much to require when the general goal is to understand the data.

10.5.1 Example: NLSY data

The dataset `NLSY` comes from a small part of the National Longitudinal Survey of Youth, a series of annual surveys conducted by the U.S. Department of Labor to examine the transition of young people into the labor force. This particular subset gives measures of 243 children on mathematics and reading achievement and also measures of behavioral problems (antisocial, hyperactivity). Also available are the yearly income and education of the child's father.

In this analysis the `math` and `read` scores are taken as the outcome variables.⁴ Among the remaining predictors, income and `educ` might be considered as background variables necessary to control for. Interest might then be focused on whether the behavioral variables `antisoc` and `hyperact` contribute beyond that.

```
data(NLSY, package = "heplots")
str(NLSY)
#> 'data.frame': 243 obs. of 6 variables:
#> $ math    : num 50 28.6 50 32.1 21.4 ...
#> $ read    : num 45.2 28.6 53.6 34.5 22.6 ...
#> $ antisoc : int 4 0 2 0 0 1 0 1 1 4 ...
#> $ hyperact: int 3 0 2 2 2 0 1 4 3 5 ...
#> $ income   : num 52.52 42.6 50 6.08 7.41 ...
#> $ educ     : int 14 12 12 12 14 12 12 12 12 9 ...
```

Exploratory plots

To begin, I would examine some scatterplots and univariate displays. I'll start with density plots for all the variables to see the shapes of their distributions, with rug plots at the bottom to show where the observations are located. From Figure 10.9 we see that math and reading scores are positively skewed, anti-social and hyperactivity have distributions highly concentrated in the lower scores. As we would suspect, father's income is quite positively skewed. Father's education is reasonably symmetric, but highly peaked at 12 years of schooling in this sample. The spikes reflect the fact that education is measured in discrete years.

```
NLSY_long <- NLSY |>
  tidyverse::pivot_longer(math:educ, names_to = "variable") |>
  dplyr::mutate(variable = forcats::fct_inorder(variable))

ggplot(NLSY_long, aes(x=value, fill=variable)) +
  geom_density(alpha = 0.5) +
  geom_rug() +
  facet_wrap(~variable, scales="free") +
  theme_bw(base_size = 14) +
  theme(legend.position = "none")
```

⁴Other choices are possible: we could instead try to model the behavioral variables, `antisocial` and `hyperact` first, and then determine if the parental variables add appreciably to this. Modeling choices aren't arbitrary. They should reflect the aims of a study and the story you want to tell about the result.

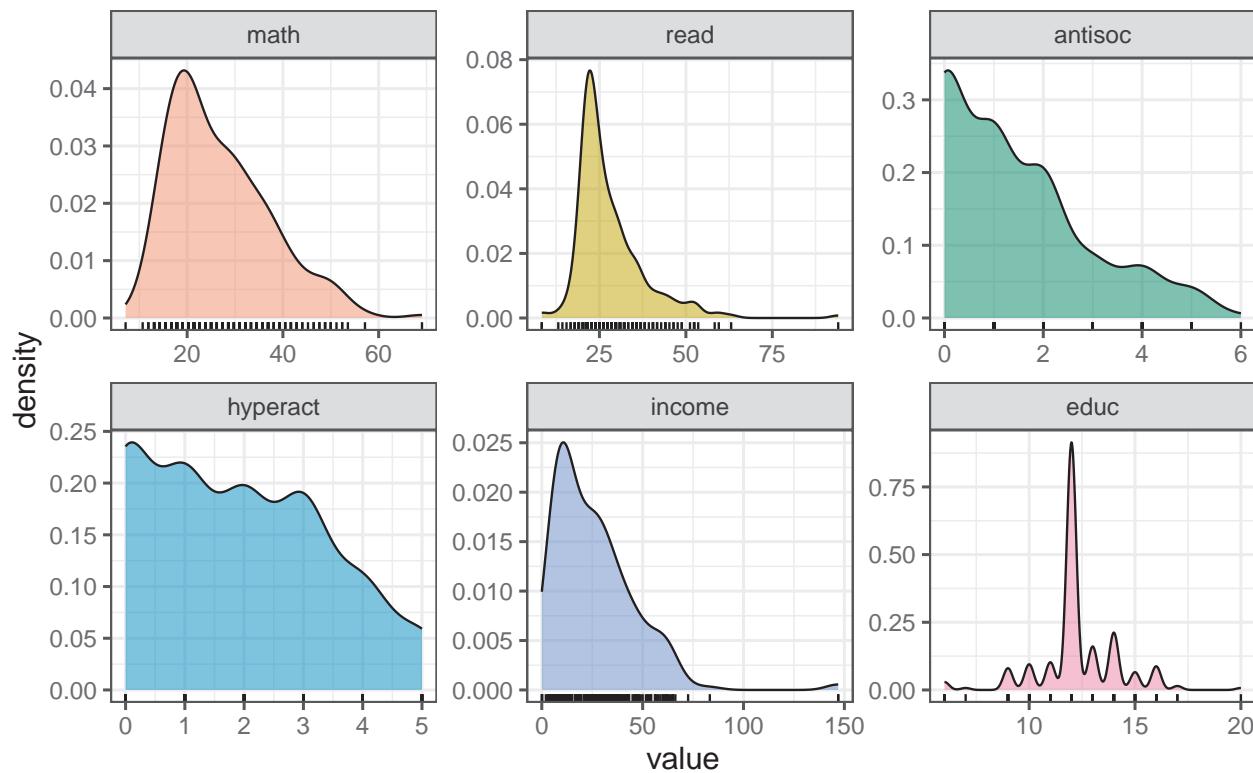


Figure 10.9: Density plots for the variables in the NLSY dataset.

In terms of an analysis focused on `math` and `read` as outcomes, a scatterplot of one against the other is useful, as is collection of scatterplots of each against the remaining variables. The second of these is left as an exercise to the reader.

```
set.seed(47)
ggplot(NLSY, aes(x = read, y = math)) +
  geom_jitter()+
  geom_smooth(method = lm, formula = y~x, fill = "blue", alpha = 0.2) +
  geom_smooth(method = loess, se = FALSE, color = "red", linewidth = 2)
```

The non-linear trend in Figure 10.10 may be due to the sparsity of data in the upper range of reading, and there are also a few unusual points shown in this plot. The function `heplots::noteworthy()` provides a variety of methods to identify such noteworthy points in scatterplots. The default `method` uses Mahalanobis D^2 . The plot below labels the five largest observations.

```
ids <- heplots::noteworthy(NLSY[, 1:2], method = "mahal", n=5)
ggplot(NLSY, aes(x = read, y = math)) +
  geom_jitter()+
  geom_smooth(method = lm, formula = y~x, fill = "blue", alpha = 0.2) +
  geom_text(data = NLSY[ids, ], label = ids, size = 5, nudge_y = 2)
```

Fitting models

We could of course include all of the predictors in a single model, and perhaps be done with it. To develop some model-thinking, it is more useful to proceed in smaller steps to see what we can learn from each. If we

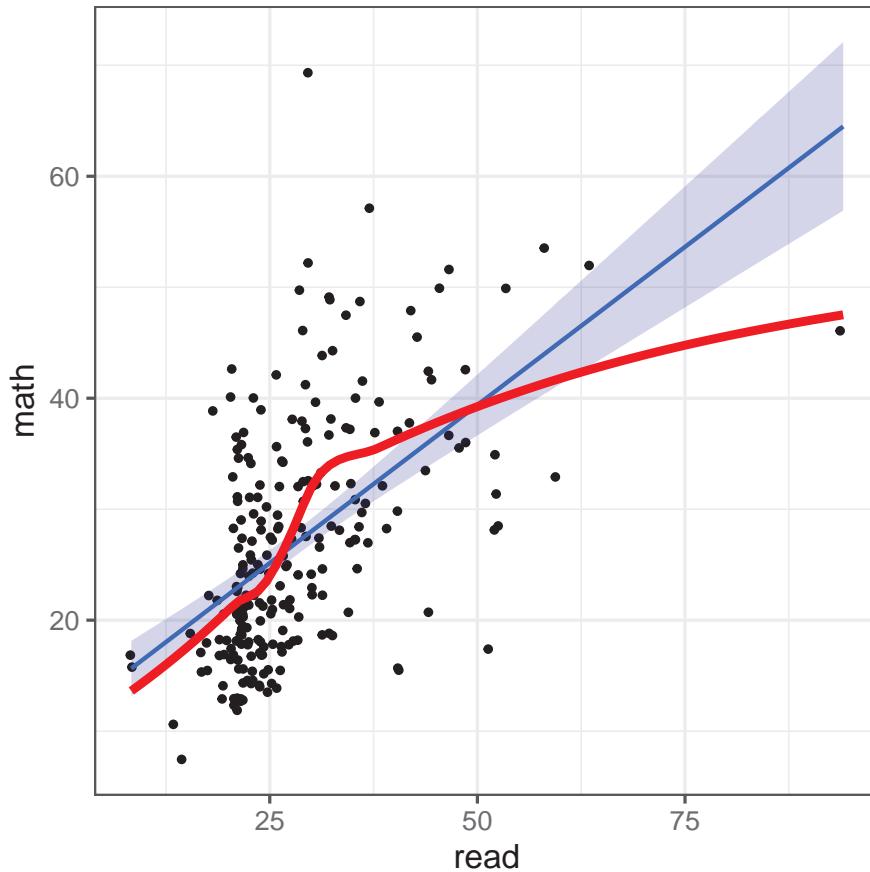


Figure 10.10: Scatterplot of mathematics score against reading score in the NLSY data

view parents' income and education as the most obvious predictors of reading and mathematics scores, those are the variables to fit first.

```
NLSY.mod1 <- lm(cbind(read, math) ~ income + educ,
                  data = NLSY)

Anova(NLSY.mod1) # Type II, partial test
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> income    1   0.0345     4.27      2    239 0.0151 *
#> educ      1   0.0515     6.49      2    239 0.0018 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Overall test

The `Anova()` results above give multivariate tests of the contributions of each predictor *separately* to explaining reading and math and how they vary together. To get an overall test of the global null hypothesis $\mathcal{H}_0 : \mathbf{B} = (\beta_{\text{inc}}, \beta_{\text{educ}}) = \mathbf{0}$ for *all* predictors together, you can use `linearHypothesis()`:

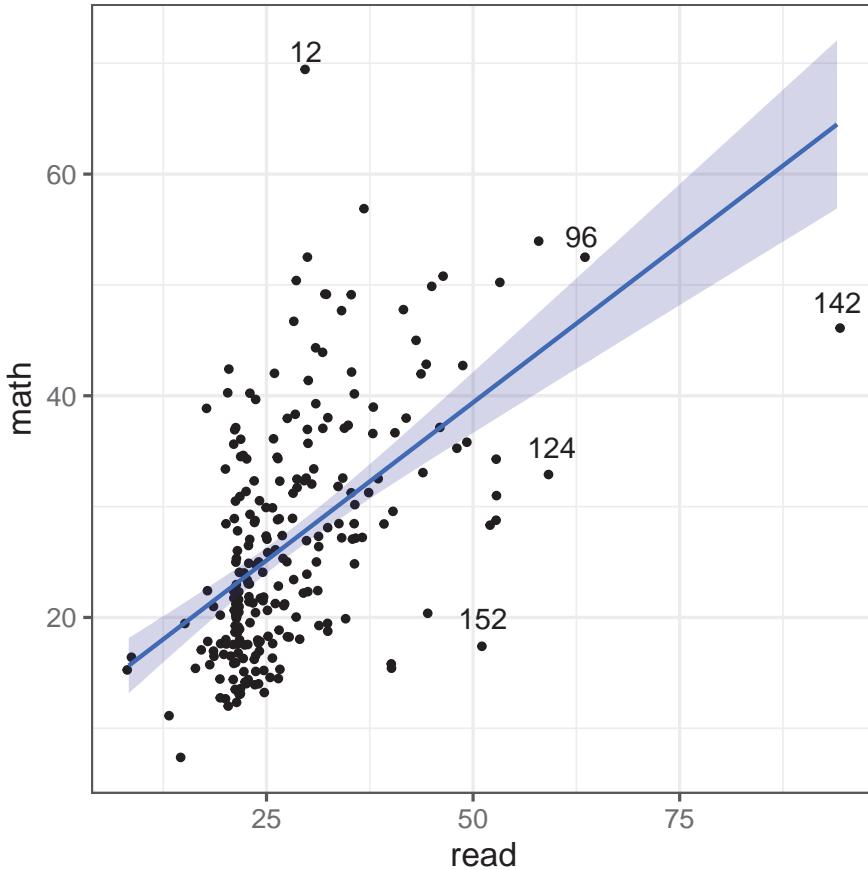


Figure 10.11: Scatterplot of mathematics score against reading score in the NLSY data, highlighting noteworthy points

```

coefs <- rownames(coef(NLSY.mod1))[-1]
linearHypothesis(NLSY.mod1, coefs, title = "income, educ = 0") |>
  print(SSP = FALSE)
#>
#> Multivariate Tests: income, educ = 0
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai        2   0.117    7.44      4     480 8.1e-06 ***
#> Wilks         2   0.884    7.59      4     478 6.2e-06 ***
#> Hotelling-Lawley 2   0.130    7.75      4     476 4.7e-06 ***
#> Roy          2   0.123   14.79      2     240 8.7e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

This joint multivariate test is more highly significant than either of those for the separate effects of the predictors, again because it pools strength.

Coefficient plots

As usual, you can display the coefficients using `coef()`. The `tidy` method for "`mle`" objects defined in `heplots` shows these in a tidy format with *t*-tests for each coefficient., arranged by the response variable.

```

coef(NLSY.mod1)
#>          read   math
#> (Intercept) 15.8848 8.7829
#> income      0.0137 0.0893
#> educ        0.9495 1.2755

tidy(NLSY.mod1)
#> # A tibble: 6 x 6
#>   response term      estimate std.error statistic p.value
#>   <chr>     <chr>      <dbl>     <dbl>      <dbl>     <dbl>
#> 1 read      (Intercept) 15.9       4.25      3.74  0.000233
#> 2 read      income      0.0137    0.0325     0.420 0.675
#> 3 read      educ        0.949     0.360      2.64  0.00894
#> 4 math      (Intercept) 8.78       4.33      2.03  0.0437
#> 5 math      income      0.0893    0.0331     2.70  0.00749
#> 6 math      educ        1.28       0.367     3.47  0.000607

```

However, a bivariate plot of these coefficients is more useful, because it provides visual tests of multivariate hypotheses. `heplots:::coefplot.mlm()` gives displays of the coefficients for a given *pair* of response variables. For interpretation, it adds the bivariate confidence ellipse for the coefficients, as well as univariate confidence intervals for each response. The univariate intervals are simply the horizontal and vertical shadows of the ellipses on the response variable axes.

A wrinkle here, as in Section 6.3, is that the coefficients are measured in different units and so coefficient plots for different predictors are more easily compared for standardized variables. To do this, I first re-fit the model using `scale(NLSY) ...`

```

NLSY_std <- scale(NLSY) |>
  as.data.frame()

NLSY_std.mod1 <- lm(cbind(read, math) ~ income + educ,
                      data = NLSY_std)

```

```

coefplot(NLSY_std.mod1, fill = TRUE,
         col = c("darkgreen", "brown"),
         lwd = 2,
         cex.lab = 1.5,
         ylim = c(-0.1, 0.5),
         xlab = "read coefficient (std)",
         ylab = "math coefficient (std)")

```

In Figure 10.12, the confidence ellipses for `income` and `educ` both exclude the origin, which represents the multivariate hypothesis $\mathcal{H}_0 : (\beta_{\text{read}}, \beta_{\text{math}}) = (0, 0)$, so this hypothesis is rejected. Note that if we only examined the univariate tests for each of the four parameters, we would conclude that for reading, `income` is not a significant predictor. The orientation of the confidence ellipses indicates the positive correlation between reading and mathematics scores.

10.5.1.1 Behavioral measures

Given that the parental background variables are highly predictive of student performance, we might want to know if the behavioral measures `antisoc` and `hyperact` add importantly to this. One way to do this is to add these predictors to the model and test for their additional contributions over and above the baseline model.

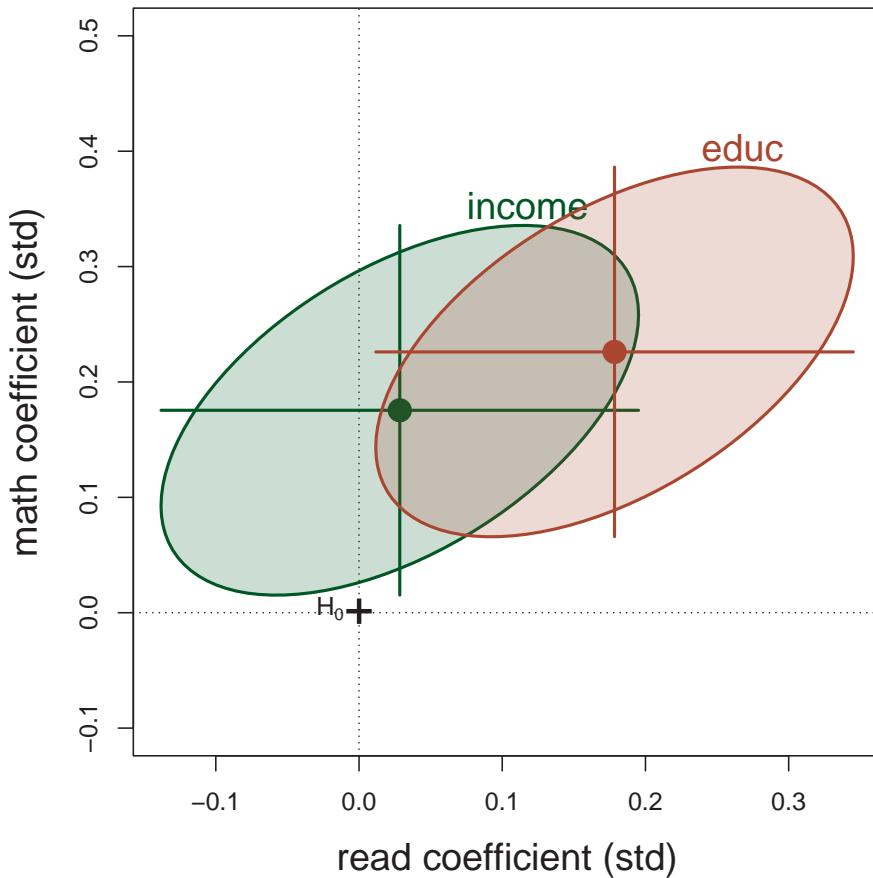


Figure 10.12: Bivariate coefficient plot for reading and math with 95% confidence ellipses. The variables have been standardized to make their units comparable.

You can do this using `update()`. In the model formula, “.” on the left hand side corresponds to the previous y variables; on the right-hand side it refers to the xs in the previous model, so I just add the new predictors to that.

```
NLSY.mod2 <- update(NLSY.mod1, . ~ . + antisoc + hyperact)
Anova(NLSY.mod2)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> income     1   0.0383    4.72      2    237 0.0098 ***
#> educ       1   0.0532    6.65      2    237 0.0015 ***
#> antisoc    1   0.0193    2.34      2    237 0.0988 .
#> hyperact   1   0.0144    1.74      2    237 0.1784
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Each of these new predictors are individually non-significant according to the Type II tests. Using `linearHypothesis()` you can test them jointly:

```

coefs <- rownames(coef(NLSY.mod2))[-1] |> print()
#> [1] "income"    "educ"       "antisoc"    "hyperact"

linearHypothesis(NLSY.mod2, coefs[3:4],
                  title = "NLSY.mod2 | NLSY.mod1") |>
  print(SSP = FALSE)
#>
#> Multivariate Tests: NLSY.mod2 | NLSY.mod1
#>          Df test stat approx F num Df den Df Pr(>F)
#> Pillai      2   0.024    1.45      4    476  0.218
#> Wilks       2   0.976    1.44      4    474  0.218
#> Hotelling-Lawley 2   0.024    1.44      4    472  0.218
#> Roy         2   0.022    2.64      2    238  0.073 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

10.5.2 Example: School data

Charnes et al. (1981) describe a large scale “social experiment” in public school education. Seventy school sites across the U.S. participated and a number of variables related to attributes of parents and teachers were used to predict aspects of students’ success in academic indicators (reading, mathematics), but also in their self-esteem. It was conceived in the late 1960’s in relation to a federally sponsored program charged with providing remedial assistance to educationally disadvantaged early primary school students.

The study was focused on the management styles used to guide educational planning across schools. In particular, it was primarily designed to compare schools using Program Follow Through (PFT) management methods of taking actions to achieve goals with those of Non Follow Through (NFT).

Here, I simply focus on the relations between outcome scores on tests of reading, mathematics and self-esteem in relation to five explanatory variables related to parents and teachers:

- **education** level of mother as measured by the percentage of high school graduates among female parents.
- **occupation**, highest occupation of a family member on a rating scale.
- **visit**, an index of the number of parental visits to the school site.
- **counseling**, a measure calculated from data on time spent with child on school-related topics such as reading together, etc.
- **teacher**, number of teachers at the given site.

The dataset, given in `schooldata` contains observations for 70 schools.⁵

Exploratory plots

There are eight variables in this example, so a scatterplot matrix or even a corrgram might not be sufficiently revealing. As usual, I tried a number of different methods and found a couple that were interesting and useful.

Multivariate normality is **not** required for all the variables in \mathbf{Y} and \mathbf{X} — it is only required for the residuals, $\boldsymbol{\varepsilon} = \mathbf{Y} - \widehat{\mathbf{Y}}$. Yet, for MMRA problems, sometimes an initial χ^2 QQ plot provides a handy way to flag possibly unusual values to pay attention to as the analysis proceeds. In Figure 10.13 we see five cases outside the 95% confidence envelope.

⁵In this, schools 1–49 were PFT sites and the remaining sites 50–70 were NFT. A separate dataset `schoolsites` provides other information on the schools, such as the general education style, region of the U.S., size of the city and that of the student population.

```

data(schooldata, package = "heplots")
res <- cqplot(schooldata, id.n = 5) |> print()
#>      DSQ quantile      p
#> 59 44.6      21.0 0.00714
#> 44 38.8      18.0 0.02143
#> 33 27.9      16.5 0.03571
#> 66 24.0      15.5 0.05000
#> 35 21.7      14.7 0.06429

# save the case ID numbers
outliers <- rownames(res) |> as.numeric() |> print()
#> [1] 59 44 33 66 35

```

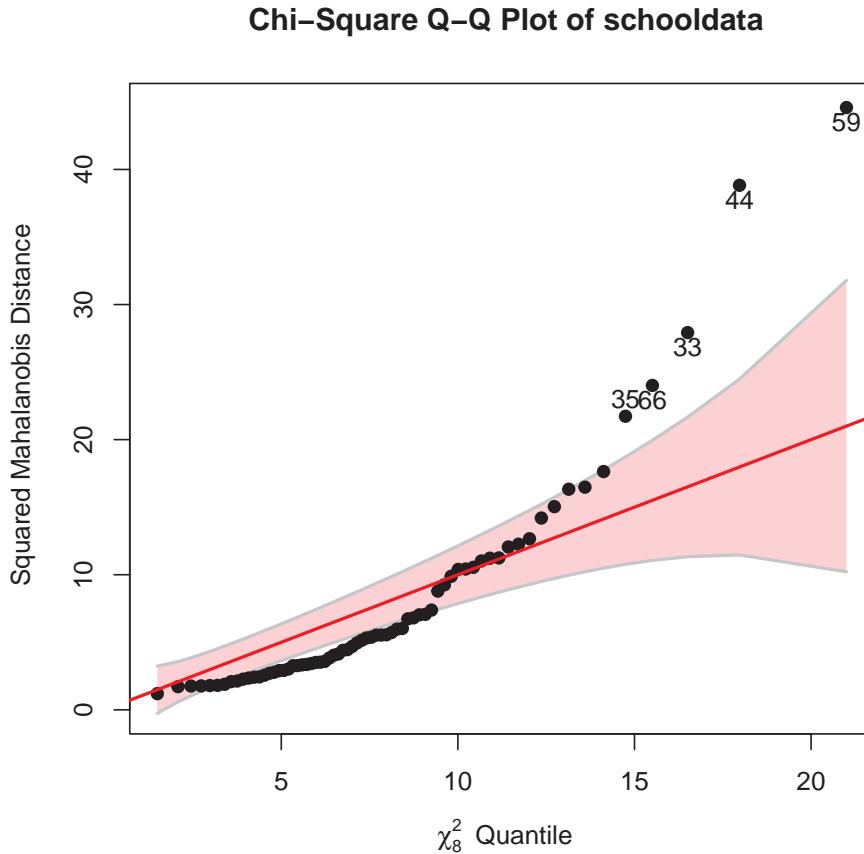


Figure 10.13: χ^2 QQ plot of the `schooldata` variables.

Rather than a complete 8×8 scatterplot matrix, it is useful here to examine the scatterplots *only* for each y variable against each of the predictors in \mathbf{X} .⁶ I'll take steps to flag some of these possibly unusual cases to see where they appear in these pairwise relations.

To prepare for this with `ggplot2`, it is necessary to reshape the data to long format *twice*—once for the ($q = 5$) x variables and again for the ($p = 3$) y responses to get all of their $q \times p$ combinations. That way, we get a data set with variables `x` and `y` whose variable names are by `xvar` and `yvar`.

⁶[GGally](#) has a function `ggduo()` that does something similar, plotting each of one set of variables against another. Like `ggpairs()` it allows for generalizations of a scatterplot where combinations of discrete factors and continuous variables can be displayed with appropriate visualizations for each.

```
# plot predictors vs each response
xvars <- names(schooldata)[1:5]
yvars <- names(schooldata)[6:8]

school_long <- schooldata |>
  tibble::rownames_to_column(var = "site") |>
  pivot_longer(cols = all_of(xvars),
                names_to = "xvar", values_to = "x") |>
  pivot_longer(cols = all_of(yvars),
                names_to = "yvar", values_to = "y") |>
  mutate(xvar = factor(xvar, xvars),
         yvar = factor(yvar, yvars))

car::some(school_long, n=8)
#> # A tibble: 8 x 5
#>   site   xvar      x  yvar      y
#>   <chr> <fct>    <dbl> <fct>    <dbl>
#> 1 1     teacher    9  reading   54.5
#> 2 18    education  28 mathematics 38.2
#> 3 26    teacher    7  selfesteem 31.2
#> 4 49    occupation 5.29 reading   12.2
#> 5 53    counseling 26.3 mathematics 22.0
#> 6 61    occupation 2.59 mathematics 7.1
#> 7 62    visit      9.89 reading   9.35
#> 8 64    occupation 8.91 mathematics 24.5
```

With this data structure, each scatterplot is a plot of a y against and x , and we can facet this using `facet_grid(yvar ~ xvar)`, giving Figure 10.14.

```
p1 <- ggplot(school_long, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, formula = y ~ x) +
  stat_ellipse(geom = "polygon",
               level = 0.95, fill = "blue", alpha = 0.2) +
  facet_grid(yvar ~ xvar, scales = "free") +
  labs(x = "predictor", y = "response") +
  theme_bw(base_size = 16)

# label the 3 most unusual points in each panel
p1 + geom_text_repel(data = school_long |>
                        filter(site %in% outliers[1:3]),
                        aes(label = site))
```

All of the predictors except for number of teachers show very strong linear relations with the outcome scores. Among the identified points, cases 44 and 59 stand out in all the plots, case 59 being particularly high on all the measures. As well, there is a small cluster of unusual points in the plots for number of teachers.

Fiting models

Let's proceed to fit the multivariate regression model. Here “.” on the right-hand side of the model formula means all the other variables in the dataset.

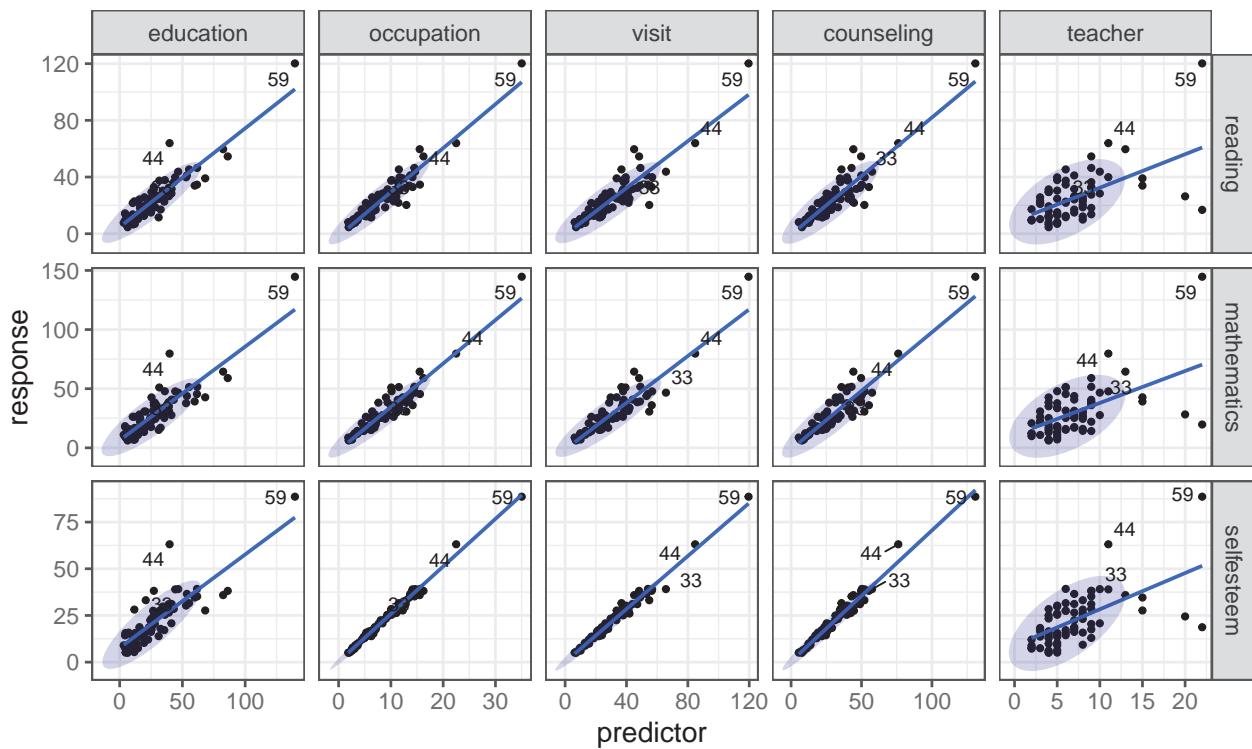


Figure 10.14: Scatterplots of each of the three response variables against each of the five predictors in the schooldata dataset. Three of the points identified as possible multivariate outliers are labeled.

```

school.mod <- lm(cbind(reading, mathematics, selfesteem) ~ .,
                  data=schooldata)
car::Anova(school.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> education   1    0.376     12.43      3     62 1.8e-06 ***
#> occupation  1    0.567     27.02      3     62 2.7e-11 ***
#> visit        1    0.260      7.27      3     62 0.00029 ***
#> counseling   1    0.065      1.43      3     62 0.24297
#> teacher      1    0.049      1.07      3     62 0.37003
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

These multivariate tests have a seemingly simple interpretation: parent's education and occupation and their visits to the schools are highly predictive of student's outcomes; their counseling efforts and the number of teachers in the schools, not so much.

You can get an assessment of the *strength* of multivariate association from the R^2 for each of the responses using `glance()` for the MLM. All of these are very high.

```

glance(school.mod)
#> # A tibble: 3 x 8
#>   response    r.squared sigma fstatistic numdf dendf  p.value  nobs
#>   <chr>        <dbl>    <dbl>       <dbl>    <dbl>    <dbl>    <dbl>    <int>

```

```
#> 1 reading      0.929  4.83      167.     5    64 2.34e-35    70
#> 2 mathematics 0.917  6.16      141.     5    64 3.37e-33    70
#> 3 selfesteem   0.993  1.17     1852.     5    64 8.47e-68    70
```

Similarly `etasq()` for an MLM gives an R^2 -like measure called η^2 of the *partial association* accounted for each of the *predictor terms* in the model. These are analogous to the Type II tests from `Anova()`, which test the additional contribution of each term in the model beyond all the others.

In spite of the overwhelming significance of the first three predictors, their variance accounted for is more modest. It is highest for parent's occupation, followed by education. Parent counseling and teachers contribute very little.

```
etasq(school.mod)
#>                 eta^2
#> education    0.3756
#> occupation   0.5666
#> visit        0.2603
#> counseling   0.0647
#> teacher       0.0491
```

10.6 Model diagnostics for MLMs

Model building, visualization and interpretation is often an iterative process. You fit a model and calculate some goodness of fit measures (R^2 for responses, η^2 for predictors). If these are reasonably strong, you feel happy and proceed to graphical displays to help you understand what you've found and explain it to others.

But **wait**: did you check the assumptions of the MLM? As in univariate models, diagnostic plots can help you spot problems in the data (unusual cases) or in the model (nonlinear relations, omitted predictors or interactions). You sometimes need to go circle back and fit a revised model, starting the process again.

For multivariate regression models, I consider the assumed multivariate normality of residuals and multivariate influence here. For MANOVA models question of homogeneity of covariance matrices is deferred until Chapter 12.

10.6.1 Multivariate normality of residuals

One easy thing to do is to check for multivariate normality of the residuals. Given that we found a few noteworthy points in Figure 10.13, a χ^2 QQ plot of the residuals in the model will tell us if any of these are really problematic. The pattern of points relative to the confidence band gives a rough indication of overall multivariate normality.

```
cqplot(school.mod, id.n = 5)
```

So, you can see that among the cases that stood out in the `cqplot()` of the observed variables (Figure 10.13), only case 35 attracts attention here, and it is well within the confidence band. Case 59, which was the largest in all the pairwise scatterplots (Figure 10.14) seems not unusual in the fitted model. It is a high-leverage point, but appeared to be well-fitted in all the simple regressions, except in those for `teacher`.

It is useful to contrast this with what we get from formal tests that the residuals are *strictly* multivariate normal. The **MVN** package (Korkmaz et al., 2025) provides `mvn()` for this, which performs a wide variety of

Chi-Square Q-Q Plot of school.mod

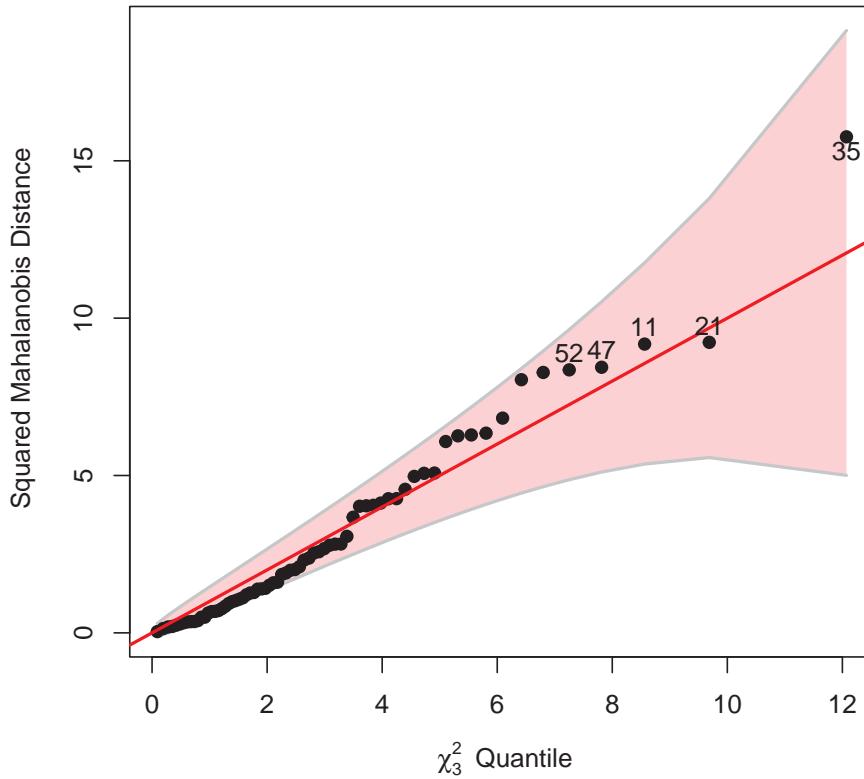


Figure 10.15: χ^2 QQ plot of the residuals in the schooldat multivariate regression model.

normality tests. The most widely used of these is due to Mardia (1974), which gives multivariate tests of skewness (lack of symmetry) and kurtosis (length of the tails).

Applying this to the **residuals** from the schools multivariate regression shows that multivariate normality is rejected here. Based on other evidence, this doesn't seem particularly troubling.

```

school.mvn <- mvn(residuals, mvn_test = "mardia")
# print multivariate and univariate tests
summary(school.mvn, select = "mvn")
#>           Test Statistic p.value     Method      MVN
#> 1 Mardia Skewness     1.92  0.750 asymptotic Normal
#> 2 Mardia Kurtosis    -1.16  0.244 asymptotic Normal
summary(school.mvn, select = "univariate")
#>           Test Variable Statistic p.value Normality
#> 1 Anderson-Darling start     0.307  0.524   Normal
#> 2 Anderson-Darling amount    0.625  0.085   Normal

```

`mvn()` also provides a variety of tests for univariate normality for each of the response variables. These are all OK.

10.6.2 Distance plot

Another useful screening plot (suggested by Peter J. Rousseeuw et al. (2004)) is a plot of Mahalanobis distances of the predictors against the Mahalanobis distances of the corresponding residuals for a fitted model. This diagnostic plot combines the information on leverage points and regression outliers in an interesting way.

It is much more useful than plotting them individually (against the theoretical values) as in a χ^2 QQ plot. Moreover, plotting them against each other is visually informative, because it places the leverage points (unusual in \mathbf{X}) and the outliers (unusual in \mathbf{Y}) in *different regions* of the plot. To judge notably “large” values, they suggest using cutoffs of $\sqrt{\chi_p^2(0.975)}$ for the predictor distances and $\sqrt{\chi_q^2(0.975)}$ for the residuals.

Such plots are produced by `heplots::distPlot()`. Running this on the `school.mod` model gives Figure 10.16. Points beyond the horizontal and vertical cutoff values are labeled with their case numbers.

```
distancePlot(school.mod, cex = 1.5, cex.lab = 1.2)
#> 0.975 X, Y distance cutoffs: 3.58 3.06
```

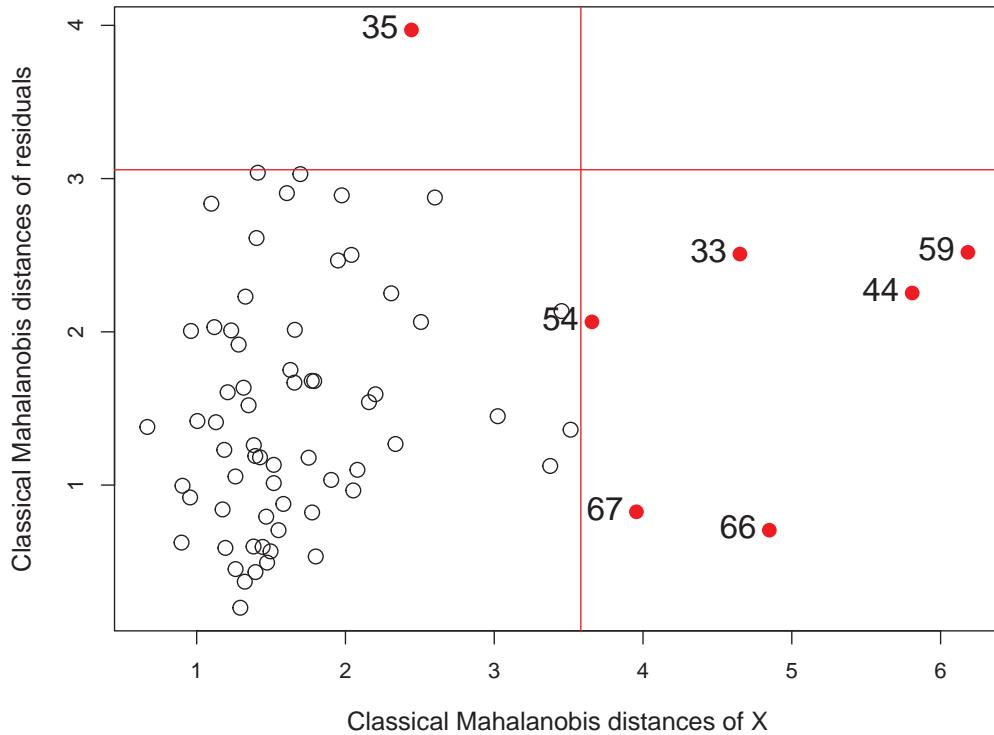


Figure 10.16: Plot of Mahalanobis distances of the least squares residuals vs. Mahalanobis distances of the predictors in the model

Most of the points identified in the χ^2 QQ plot Figure 10.13 are labeled here. Cases 44 and 59 are high leverage points with large \mathbf{X} distances. Case 35 is the only one beyond the cutoff for residuals. Interestingly, case 66 appeared near 35 in Figure 10.13, but is unusual for a different reason as we can see in Figure 10.16.

Peter J. Rousseeuw et al. (2004) also suggested methods of **robust multivariate regression** using robust estimates of location and scatter rather than the classical $\bar{\mathbf{y}}$ and \mathbf{S} . The *minimum covariance determinant* (*MCD*) estimator is a robust estimator with high breakdown value and bounded influence. It looks for the subset of size h , whose covariance matrix has the smallest determinant, where $h : \lceil n/2 \rceil < h < n$ controls the robustness. The `distPlot()` function implements this, as well as a *minimum variance ellipse* (*MVE*) method. Some robust methods are illustrated in Section 13.5.

10.6.3 Multivariate influence

TODO Sort out coverage here vs. Chapter 13

Again, what we see in simple scatterplots can be misleading because they ignore all the other variables in a model. But looking back after fitting a model and examining diagnostic plots can often be illuminating. Among the ideas we inherit from univariate models, the influence of particular observations on the results of analysis should be high on your list.

The multivariate extension of the diagnostic measures of leverage and influence, and influence plots (Section 6.6) is provided by the `mvinfluence` package (Friendly, 2025a). The theory behind this is due to Barrett & Ling (1992) and better illustrated in Barrett (2003). Mathematical details of this generalization are given in `help("mvinfluence-package")`.

Figure 10.17 shows one form of an influence plot for the `school.mod` model. Because multiple response variables are involved, this plots a measure of the *squared* studentized residuals for each observation against a generalized version of hat values, so potentially “bad” observations appear in the upper left corner. The size of the bubble symbol is proportional to a generalization of Cook’s distance, the measure of influence based on the change in all coefficients if each case was deleted from the analysis.

```
influencePlot(school.mod, id.n=4,
              type="stres",
              cex.lab = 1.5)
#>      H      Q CookD      L      R
#> 33 0.328 0.2017 0.7054 0.488 0.3001
#> 35 0.101 0.2825 0.3038 0.112 0.3142
#> 44 0.503 0.2984 1.6022 1.014 0.6009
#> 59 0.568 0.4937 2.9938 1.317 1.1441
#> 66 0.355 0.0174 0.0657 0.551 0.0269
```

That does not look good! You can see that cases 44 and 59 are actually quite troublesome here, but it turns out for different reasons. Take another look at Figure 10.14. You can see that case 59 is the most extreme on all the predictors, giving it very high leverage and therefore pulling the regression lines toward it in most of the plots, except those for number of teachers, where it also has large residuals. Case 44, on the other hand, stands out as high-leverage on only a few of the predictor-response combinations, but enough to give it a large multivariate hat value. It is also a point that is furthest from the regression lines.

What should be done? An appropriate action would be to re-fit the model, reducing the impact of these cases, in what I call a **sensitivity test**:

- Do the main conclusions change with those cases removed? That is, do any model terms change in significance tests or do coefficients change sign?
- Do the relative sizes of effects for predictors change enough to affect interpretation? That is, how much do the coefficients change?

The easiest solution is to just omit these troubling cases. You can do this using `update()`, specifying the `data` argument to be the subset of rows without the bad boys.

```
bad <- c(44, 59)
OK <- (1:nrow(schooldata)) |> setdiff(bad)
school.mod2 <- update(school.mod, data = schooldata[OK,])
Anova(school.mod2)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>           Df test stat approx F num Df den Df Pr(>F)
#> education    1     0.211      5.35       3     60   0.0025 **
```

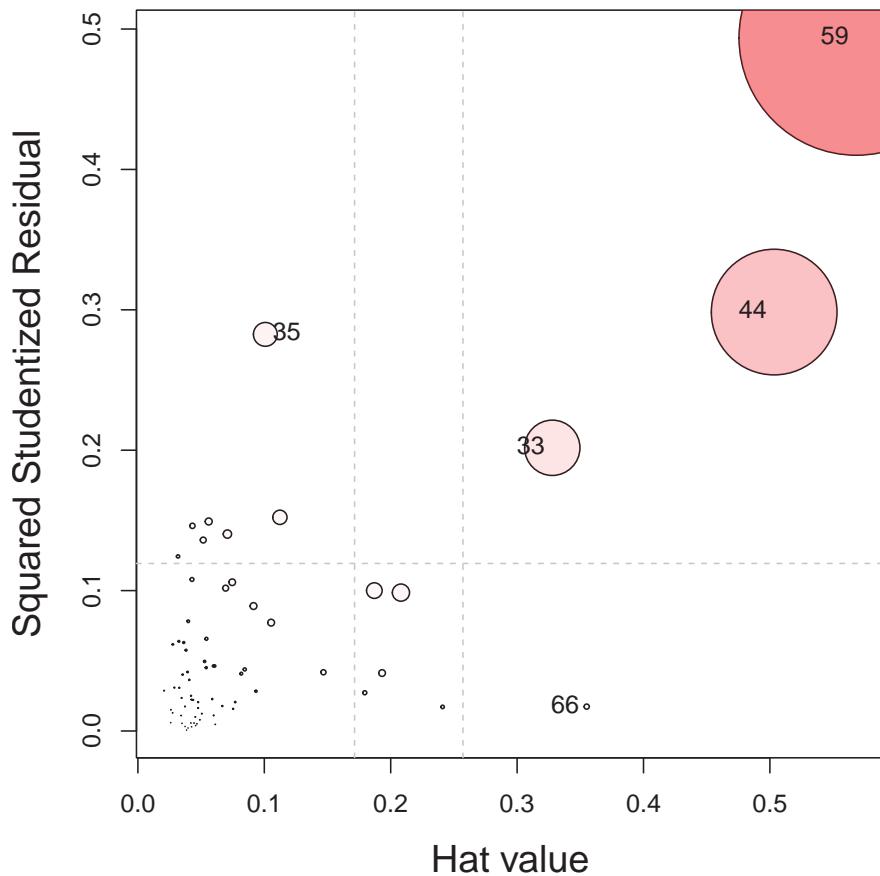


Figure 10.17: Influence plot for the `schooldat` multivariate regression model. Five cases are labeled as “noteworthy” on either axis.

```
#> occupation   1    0.422    14.62     3    60  2.9e-07 ***
#> visit        1    0.191     4.73     3    60   0.0050 **
#> counseling   1    0.053     1.13     3    60   0.3459
#> teacher      1    0.064     1.37     3    60   0.2618
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results of `Anova()` on this model tell us that the three significant predictors—occupation, education and visit—are still so, but slightly less so for the last two of these. To compare the coefficients in the new model compared to the old you can calculate the *relative difference*, $|\mathbf{B}_1 - \mathbf{B}_2|/\mathbf{B}_1$, which are

```
reldiff <- function(x, y, pct=TRUE) {
  res <- abs(x - y) / x
  if (pct) res <- 100 * res
  res
}

reldiff(coef(school.mod)[-1,], coef(school.mod2)[-1,]) |>
  round(1)
#>           reading mathematics selfesteem
```

```
#> education      10.1      19.8     -75.7
#> occupation    11.0      10.2      19.4
#> visit        -103.6     -76.8      2.8
#> counseling   -332.2     305.3     -115.4
#> teacher       -7.5      -5.6      7.0
```

As you can see, the effects on the coefficients for `visit` and `counseling` are dramatic.

10.7 ANCOVA → MANCOVA

TODO: Consider moving this to Chapter 11 and use much of the heplots MMRA vignette.

In univariate linear models, analysis of covariance (ANCOVA) is most often used in a situation where we want to compare the mean response, \bar{y}_j for different groups (defined by one or more factors), but where there are one or more quantitative predictors \mathbf{x}_1, \dots that should be taken into account for our comparisons to make sense. The simplest case is when \mathbf{x} is a pre-test score on the same measure, or when it is a background measure like age or level of education that we want to control for, to adjust for differences among the groups.

More generally, ANCOVA and its' multivariate MANCOVA brother are used for situations where the model matrix \mathbf{X} contains a mixture of factor variables and quantitative predictors, called "covariates". In this wider context, there are two flavors of analysis with different emphasis on the factors or the covariates:

- **true ANCOVA/MANOVA:** Attention is centered on the differences between the group means, but controlling for any difference in the covariate(s). This requires assuming that the slopes for the groups are all the same.
- **homogeneity of regression:** Here the focus is on the regression relations between the `ys` and the predictor `xs`, but we might also want to determine if the regression slopes are the same for all groups defined by the factors.

In the ANCOVA flavor, the model fits additive effects of the group factor(s) and the covariate(s), while the homogeneity of regression flavor adds interaction terms between groups and the `xs`. The test for homogeneity of regression is the added effect of the interaction terms:

```
mod1 <- lm(y ~ Group + x)          # ANCOVA model
mod2 <- lm(y ~ Group + x + Group:x) # allow separate slopes
mod2 <- lm(y ~ Group * x)           # same as above

anova(mod1, mod2)                  # test homogeneity of regression
```

Figure 10.18 illustrates these cases for a hypothetical two-group design studying the effect of an exercise program treatment on weight, recorded pre- (x) and post- (y) compared to a control group given no treatment. In panel (a) the slopes for the two groups are approximately equal, so the effect of treatment can be estimated by the difference in the fitted values of \hat{y}_i at the average value of x . In panel (b), the slope for the treated group is considerably greater than that for the control group, so the difference between the groups varies with x .

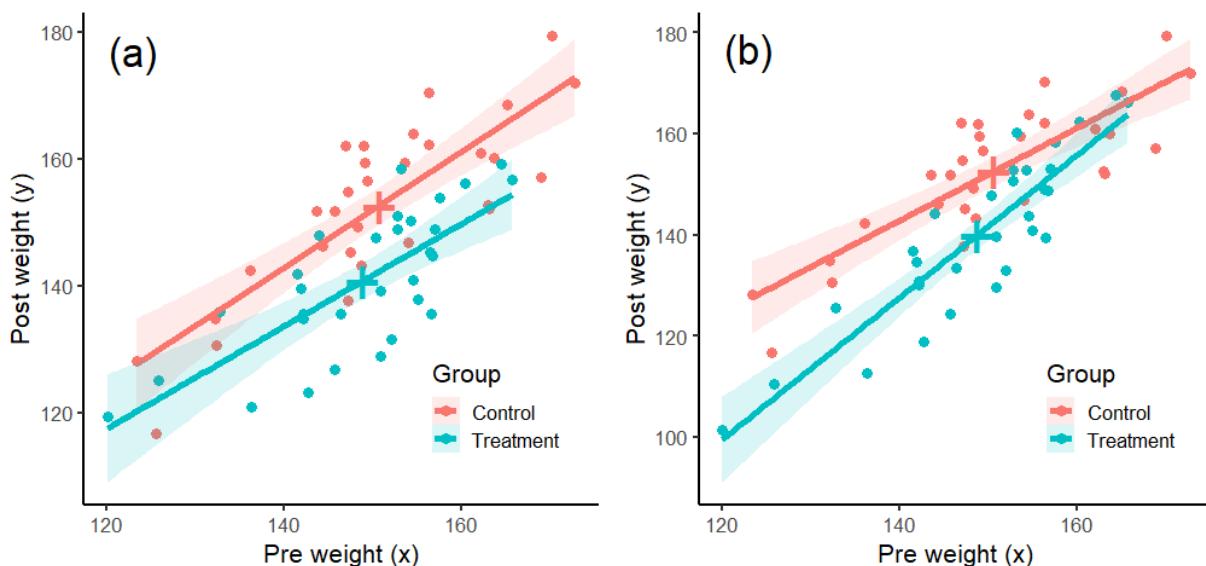


Figure 10.18: Two possible outcome patterns for a two-group design assessing the effect of a treatment on weight, measured pre- and post-treatment. (a) Additive effects of Group and x ; (b) Different slopes for the two groups. Plus signs show the means (\bar{x}_i, \bar{y}_i) for the two groups.

10.7.1 Example: Paired-associate tasks and academic performance

To what extent can simple tests of paired-associate learning⁷ predict measures of aptitude and achievement in kindergarten children? This was the question behind an experiment by William Rohwer at the University of California, Berkeley.

There were three outcome measures, one verbal and two visually-based:

- A student achievement test (**SAT**),
- Peabody Picture Vocabulary test (**PPVT**),
- Raven Progressive Matrices test (**Raven**).

Four paired-associate tasks were used, which differed in the syntactic and semantic relationship between the stimulus and response terms in each pair. These are called *named* (**n**), *still* (**s**), *named still* (**ns**), *named action* (**na**), and *sentence still* (**ss**).

Rohwer's data, taken from Timm (1975), is given in **Rohwer**. But there's a MANCOVA wrinkle: Performance on the academic tasks is well-known to vary with socioeconomic status of the parents or the school they attend. A simple design was to collect data from children in two schools, one in a low SES neighborhood ($n = 37$) and the other an upper-class high SES one ($n = 32$). The data look like this:

```
data(Rohwer, package = "heplots")
set.seed(42)
Rohwer |> dplyr::sample_n(6)
#>   group SES SAT PPVT Raven n  s ns na ss
#> 49     2  Hi  88  105     21 2 11 10 26 22
#> 65     2  Hi  50   96     13 5  8 20 28 26
#> 25     1  Lo   6   57     10 0  1 16 15 17
```

⁷Paired-associate learning are among the simplest tests of memory and learning. The subject is given a list of pairs of words or nonsense syllables, like “banana - house” or “YYZ - Toronto” to learn. On subsequent trials she is given the stimulus term of each pair (“banana”, “YYZ”) and asked to reply with the correct response (“house”, “Toronto”).

```
#> 18      1 Lo  45  54    10 0  6  6 14 16
#> 69      2 Hi  50  78    19 5 10 18 27 26
#> 64      2 Hi  24 102   16 4 17 21 27 31
```

Following the scheme for reshaping the data used in Figure 10.14, a set of scatterplots of each predictor against each response will give a useful initial look at the data. There's a lot to see here, so the plot in Figure 10.19 focuses attention on the regression lines for the two groups and their data ellipses.

```
yvars <- c("SAT", "PPVT", "Raven")      # outcome variables
xvars <- c("n", "s", "ns", "na", "ss")    # predictors

Rohwer_long <- Rohwer %>%
  dplyr::select(-group) |>
  tidyr::pivot_longer(cols = all_of(xvars),
                       names_to = "xvar", values_to = "x") |>
  tidyr::pivot_longer(cols = all_of(yvars),
                       names_to = "yvar", values_to = "y") |>
  dplyr::mutate(xvar = factor(xvar, levels = xvars),
                 yvar = factor(yvar, levels = yvars))

ggplot(Rohwer_long, aes(x, y, color = SES, shape = SES, fill = SES)) +
  geom_jitter(size=0.8) +
  geom_smooth(method = "lm",
              se = FALSE,
              formula = y ~ x,
              linewidth = 1.5) +
  stat_ellipse(geom = "polygon", alpha = 0.1) +
  labs(x = "Predictor (PA task)",
       y = "Response (Academic)") +
  facet_grid(yvar ~ xvar,           # plot matrix of Y by X
             scales = "free") +
  theme_bw(base_size = 16) +
  theme(legend.position = "bottom")
```

You can see here that the high-SES group generally performs better than the low group. The regression lines have similar slopes in some of the panels, but not all. The low SES group also appears to have larger variance on most of the PA tasks.

MANCOVA model

Nevertheless, I fit the MANCOVA model that allows a test of different means for the two SES groups on the responses, but constrains the slopes for the PA covariates to be equal. Only two of the PA tasks (na and ns) show individually significant effects in the multivariate tests.

```
# Make SES == 'Lo' the reference category
Rohwer$SES <- relevel(Rohwer$SES, ref = "Lo")

Rohwer.mod1 <- lm(cbind(SAT, PPVT, Raven) ~ SES + n + s + ns + na + ss,
                   data=Rohwer)
Anova(Rohwer.mod1)
#>
#> Type II MANOVA Tests: Pillai test statistic
```

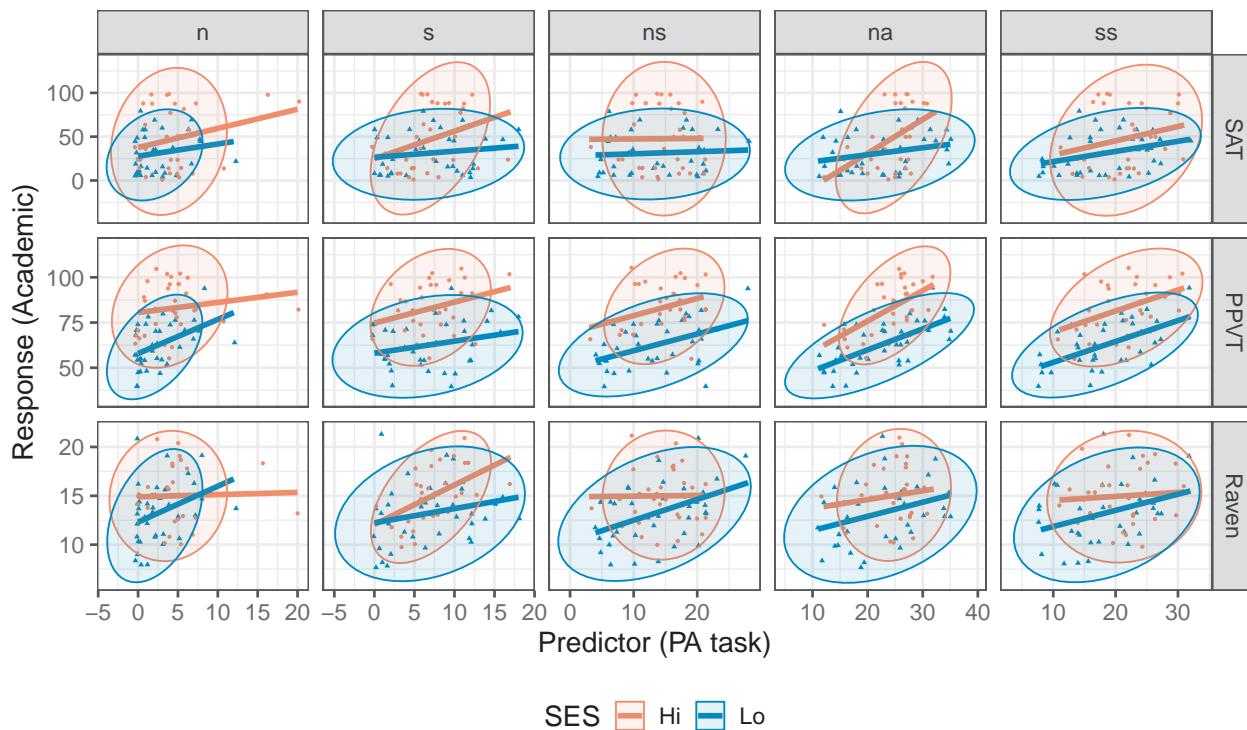


Figure 10.19: Scatterplots of each of the three response variables against each of the five predictors in the Rohwer dataset.

```
#>      Df test stat approx F num Df den Df Pr(>F)
#> SES   1    0.379    12.18     3    60 2.5e-06 ***
#> n     1    0.040     0.84     3    60  0.4773
#> s     1    0.093     2.04     3    60  0.1173
#> ns    1    0.193     4.78     3    60  0.0047 **
#> na    1    0.231     6.02     3    60  0.0012 **
#> ss    1    0.050     1.05     3    60  0.3770
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You can also examine the tests of the univariate ANCOVA models for each of the responses using `glance()` or `heplots::uniStats()`. All are significantly related, but the PPVT measure has the largest R^2 by far.

```
uniStats(Rohwer.mod1)
#> Univariate tests for responses in the multivariate linear model Rohwer.mod1
#>
#>          R^2      F df1 df2 Pr(>F)
#> SAT    0.295  4.33    6   62  0.001 **
#> PPVT   0.628 17.47    6   62  1e-11 ***
#> Raven  0.211  2.76    6   62  0.019 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To help interpret these effects, bivariate coefficient plots for the paired associate tasks are shown in Figure 10.20. (The coefficients for the group variable SES are on a different scale and so are omitted here.) From this you

can see that the named still and named action tasks have opposite signs: contrary to expectations, **ns** is negatively associated with the measures of aptitude and achievement (when the other predictors are adjusted for).

```
coefplot(Rohwer.mod1, parm = 2:6,  
         fill = TRUE,  
         level = 0.68,  
         cex.lab = 1.5)  
coefplot(Rohwer.mod1, parm = 2:6, variables = c(1,3),  
         fill = TRUE,  
         level = 0.68,  
         cex.lab = 1.5)
```

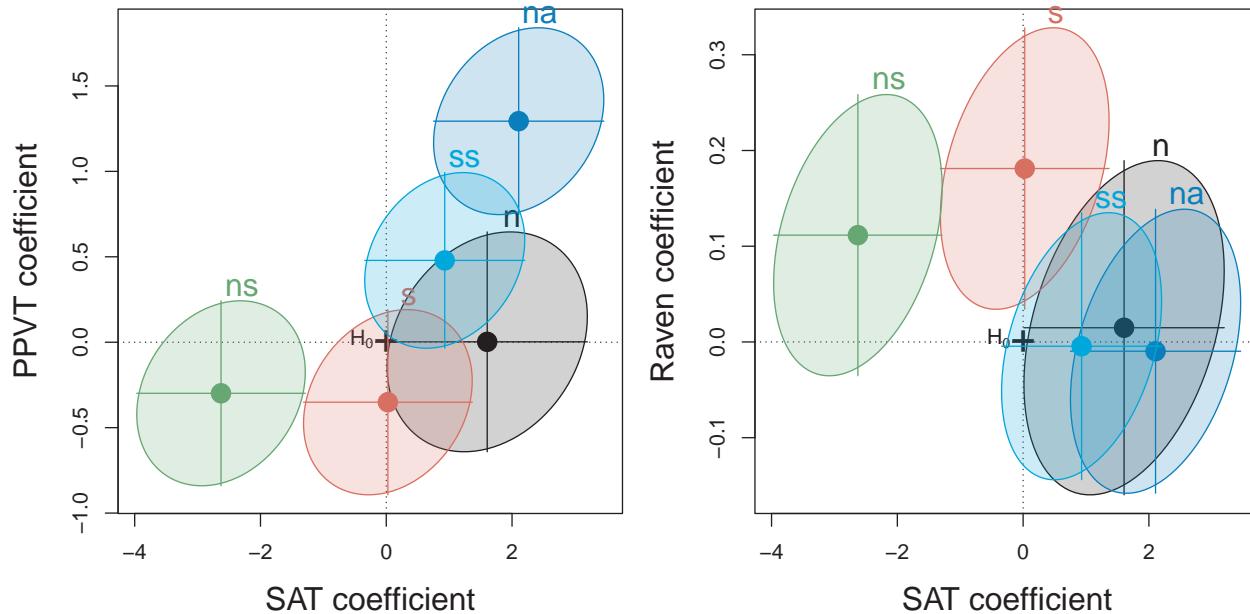


Figure 10.20: Bivariate coefficient plots for the MANCOVA model with confidence ellipses of 68% coverage.

TODO: For interpretation, it would be nice to know how many items were used for each of the PA tasks. The range of na goes up to 35, but others are less.

Adjusted means

From the analysis of covariance perspective, interest is often centered on estimating the differences between the group means, but adjusting or controlling for differences on the covariates. From the table of means below, you can see that the high SES group performs better on all three response variables, but this group also has higher scores on the paired associate tasks.

```
#> 1 Lo      1 31.3 62.6 13.2 2.54 6.92 13.5 22.4 18.4
#> 2 Hi      2 47.7 83.1 15     4.59 7.25 14.5 24.3 21.5
```

The adjusted mean differences are simply the values estimated by the coefficients for **SES** in the model. These are smaller than the differences between the observed means.

```
means[2, 3:5] - means[1, 3:5]
#>   SAT PPVT Raven
#> 1 16.4 20.4 1.76

# adjusted means
coef(Rohwer.mod1)[2,]
#>   SAT  PPVT Raven
#> 8.80 16.88 1.59
```

TODO: do this with a CI for the effects

Homogeneity of regression

The MANCOVA model, **Rohwer.mod1**, has relatively simple interpretations (a large effect of **SES**, with **ns** and **na** as the major predictors) but the test of the **SES** effect relies on the assumption of homogeneity of slopes for the predictors. We can test this assumption as follows, by adding interactions of **SES** with each of the covariates:

```
Rohwer.mod2 <- lm(cbind(SAT, PPVT, Raven) ~ SES * (n + s + ns + na + ss),
                     data=Rohwer)
Anova(Rohwer.mod2)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> SES      1    0.391    11.78      3     55 4.5e-06 ***
#> n        1    0.079     1.57      3     55 0.20638
#> s        1    0.125     2.62      3     55 0.05952 .
#> ns       1    0.254     6.25      3     55 0.00100 ***
#> na       1    0.307     8.11      3     55 0.00015 ***
#> ss       1    0.060     1.17      3     55 0.32813
#> SES:n    1    0.072     1.43      3     55 0.24417
#> SES:s    1    0.099     2.02      3     55 0.12117
#> SES:ns   1    0.118     2.44      3     55 0.07383 .
#> SES:na   1    0.148     3.18      3     55 0.03081 *
#> SES:ss   1    0.057     1.12      3     55 0.35094
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It appears from the above that there is only weak evidence of unequal slopes from the separate **SES**: terms; only that for **SES:na** is individually significant. The evidence for heterogeneity is stronger, however, when these terms are tested *collectively* using **linearHypothesis()**. I use a small **grep()** trick here to find the interaction terms, which have a ":" in their names.

```
# test interaction terms jointly
coefs <- rownames(coef(Rohwer.mod2))
interactions <- coefs[grep(":", coefs)]
```

```

print(linearHypothesis(Rohwer.mod2, interactions), SSP=FALSE)
#>
#> Multivariate Tests:
#>          Df test stat approx F num Df den Df Pr(>F)
#> Pillai      5   0.418    1.85     15    171 0.0321 *
#> Wilks       5   0.624    1.89     15    152 0.0277 *
#> Hotelling-Lawley 5   0.539    1.93     15    161 0.0240 *
#> Roy         5   0.385    4.38      5    57 0.0019 **
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Separate models

Model `Rohwer.mod2` with all interaction terms essentially fits a separate slope for each of the low and high SES groups for all responses with each of the predictor PA tasks. This is similar in spirit to what we would get if we fit a separate multivariate regression model for each of the groups, but parameterized differently: The heterogeneous regression model gives, for the interaction terms estimates of the difference in slopes between groups, while the separate-regressions approach gives separate slope estimates for each of the groups. These are equivalent, in the sense that the estimates for each approach can be derived from the other.

They are not equivalent in testing however, because the full model uses a combined pooled within-group error covariance, allows hypotheses about equality of slopes and intercepts to be tested directly and has greater power because it uses the total sample size. Here, I simply illustrate the mechanics of fitting separate models using the `subset` argument to `lm()`.

```

Rohwer.sesHi <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Hi")
Anova(Rohwer.sesHi)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> n     1   0.202    2.02      3    24 0.1376
#> s     1   0.310    3.59      3    24 0.0284 *
#> ns    1   0.358    4.46      3    24 0.0126 *
#> na    1   0.465    6.96      3    24 0.0016 ***
#> ss    1   0.089    0.78      3    24 0.5173
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Rohwer.sesLo <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Lo")
Anova(Rohwer.sesLo)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> n     1   0.0384    0.39      3    29 0.764
#> s     1   0.1118    1.22      3    29 0.321
#> ns    1   0.2252    2.81      3    29 0.057 .
#> na    1   0.2675    3.53      3    29 0.027 *
#> ss    1   0.1390    1.56      3    29 0.220
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The strength of evidence for the predictors `na` and `ns` is weaker here than when tested in the full heterogeneous model.

10.8 What we've learned

- The multivariate linear model is an extension of the (univariate) general linear model. Three major forms of the multivariate linear model are the MANOVA, MMRA and MANCOVA, all of which have their equivalents in ANOVA, MRA and ANCOVA.
- What the multivariate linear model offers over the univariate linear model is not just more powerful statistical tests, but also examinations of how the associations amongst the response variables varies with the explanatory variables.
- Just as with the general linear model, model diagnostics transfer to the multivariate linear model and are important for identifying potential issues in the validity of the estimated statistical model.
- Given that the multivariate linear model can do everything that the general linear model does, there is no reason to not consider the multivariate linear model. Confining discussions of multivariate response models to structural equation modeling may pedagogically create a cognitive blind spot that the association amongst response variables even *can* vary with the explanatory variables. Rather than something separate, for the purposes of causal inference, it would be better to use structural equation modeling as a more confirmatory statistical model than the multivariate linear model, which is perfectly well suited for exploratory purposes.
- Overall, we also see a glimpse of the importance of data visualization in the analysis of data using the multivariate linear model. The following chapter will explore this issue in more depth.

Packages used here:

13 packages used here: `broom`, `car`, `carData`, `dplyr`, `ggplot2`, `ggrepel`, `heplots`, `knitr`, `matlib`, `mvinfluence`, `MVN`, `patchwork`, `tidyR`

11

Visualizing Multivariate Models

The methods discussed in Chapter 10 provide the basis for a rather complete multivariate analysis of traditional univariate methods for the same designs. You can carry out multiple regression, ANOVA, or indeed, any classical linear model with the standard collection of analysis tools you use for a single outcome variable, but naturally extended in most cases to having several outcomes to analyse together. The key points are:

- Everything you know about the usual univariate models—regression coefficients, main effects and contrasts for factors, interactions of model terms, . . . applies here.
- By a rather clever design, called “matrix algebra” the separate univariate models can be combined, by turning vectors of responses, $\mathbf{y}_1, \mathbf{y}_2, \dots$ into a matrix \mathbf{Y} . Bingo! We get a multivariate extension.
- You can treat this as a collection of separate models, one for each response, because the coefficients are the same. But, the benefit of a multivariate approach is that you also get an overall multivariate test for each term in the model.

As nice as these mathematical and statistical ideas might be, the fact that the analysis is conducted for the response variables *collectively*, means that it may be harder to interpret and explain what this means about the separate responses. Here’s where multivariate model visualization comes to the rescue!

- **HE plots:** The tests of multivariate models, including multivariate analysis of variance (MANOVA) for group differences and multivariate multiple regression (MMRA) can be easily visualized by plots of a hypothesis (“H”) data ellipse for the fitted values, relative to the corresponding plot of the error ellipse (“E”) of the residuals, which I call the HE plot framework.
- **contrasts:** For factors in MANOVA, contrasts and linear hypotheses (Section 10.3.1) provide a way to decompose an overall multivariate test into portions directed at meaningful *specific* research questions regarding **how** the groups differ.
- **CDA:** For more than a few response variables, these result can be projected onto a lower-dimensional “canonical” space providing an even simpler description, accounting for most of the “juice”. Vectors for the response variables in this space show how these relate to the canonical dimension, facilitating interpretation.

Huang (2019) and others have criticized these methods as (a) difficult to understand because they are framed in terms of linear combinations of the the responses; (b) more complicated and limited in interpreting MANOVA effects and (c) unwieldy post hoc strategies often employed for interpretation.

These difficulties in understanding can, I believe, be cured by accessible graphical methods for visualizing hypothesis tests and for visualizing what these linear combinations reflect in terms of the observed variables. The HE plot framework described below provides powerful graphic methods available in easy used software.

This chapter describes this framework and illustrates some concrete examples, first for MANOVA designs which are conceptually and visually simpler, and then for MMRA designs with quantitative predictors and finally for MANCOVA models. Many more worked examples are available in vignettes for the `heplots`.¹

Packages

In this chapter I use the following packages. Load them now.

¹See the heplots vignettes [HE plot MANOVA Examples](#) and [HE plot MMRA Examples](#).

```
library(car)
library(heplots)
library(candisc)
library(ggplot2)
library(dplyr)
library(tidyr)
library(ggpubr)
library(patchwork)
```

11.1 HE plot framework

Chapter 9 illustrated the basic ideas of the framework for visualizing multivariate linear models in the context of a simple two group design, using Hotelling's T^2 . The main ideas were illustrated in Figure 9.9.

Having described the statistical ideas behind the MLM in Chapter 10, we can proceed to extend this framework to larger designs. Figure 11.1 illustrates these ideas using the simple one-way MANOVA design of the dogfood data from Section 10.2.1.

- In (a) **data space**, each group is summarized in (b) by its **data ellipse**, representing the means and covariances.
- Variation against the hypothesis of equal means can be seen by the **H** ellipse in the (c) **HE plot**, representing the data ellipse of the fitted values. Error variance is shown in the **E** ellipse, representing the pooled within-group covariance matrix, \mathbf{S}_p and the data ellipse of the residuals from the model. For the dogfood data, the group means have a negative relation: longer time to start eating is associated with a smaller amount eaten.
- The MANOVA (or Hotelling's T^2) is formally equivalent to a **discriminant analysis**, predicting group membership from the response variables which can be seen in data space. (The main difference is emphasis and goals: MANOVA seeks to test differences among group means, while discriminant analysis aims at classification of the observations into groups.)
- This effectively projects the p -dimensional space of the predictors into the smaller (d) **canonical space** that shows the greatest differences among the groups. As in a biplot, vectors show the relations of the response variables with the canonical dimensions.

For more complex models such as MANOVA with multiple factors or multivariate multivariate regression with several predictors, there is one sum of squares and products matrix (SSP), and therefore one **H** ellipse for *each term* in the model. For example, in a two-way MANOVA design with the model formula $(y_1, y_2) \sim A + B + A*B$ and equal sample sizes in the groups, the total sum of squares accounted for by the model is the sum of their separate effects,

$$\begin{aligned} \text{SSP}_{\text{Model}} &= \text{SSP}_A + \text{SSP}_B + \text{SSP}_{AB} \\ &= \mathbf{H}_A + \mathbf{H}_B + \mathbf{H}_{AB} . \end{aligned}$$

{#eq-HE-model}

All of these hypotheses can be overlaid in a single HE plot showing their effects together in a comprehensive view.

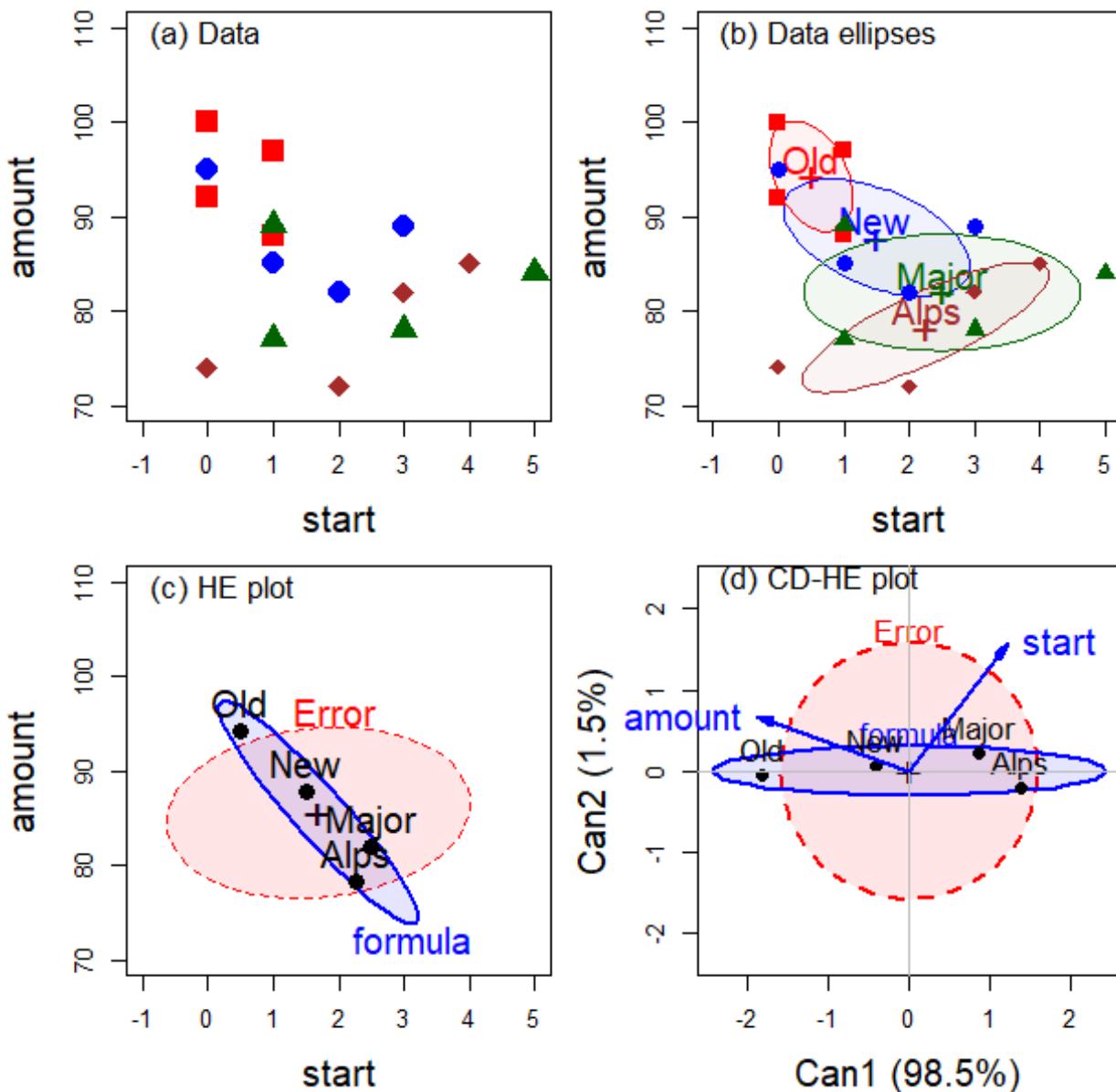


Figure 11.1: Dogfood quartet: Illustration of the conceptual ideas of the HE plot framework for the dogfood data. (a) Scatterplot of the data; (b) Summary using data ellipses; (c) HE plot shows the variation in the means in relation to pooled within group variance; (d) Transformation from data space to canonical space

11.2 HE plot construction

The HE plot is constructed to allow a direct visualization of the “size” of hypothesized terms in a multivariate linear model in relation to unexplained error variation. These can be displayed in 2D or 3D plots, so I use the term “ellipsoid” below to cover all cases.

Error variation is represented by a standard 68% data ellipsoid of the \mathbf{E} matrix of the residuals in $\boldsymbol{\varepsilon}$. This

is divided by the residual degrees of freedom, so the size of \mathbf{E}/df_e is analogous to a mean square error in univariate tests. The choice of 68% coverage allows you to “read” the residual standard deviation as the half-length of the shadow of the \mathbf{E} ellipsoid on any axis (see Figure 3.11).

The \mathbf{E} ellipsoid is then translated to the overall (grand) means $\bar{\mathbf{y}}$ of the variables plotted, which allows us to show the means for factor levels on the same scale, facilitating interpretation. In the notation of Equation 3.2, the error ellipsoid \mathcal{E}_c of size c is given by

$$\mathcal{E}_c(\bar{\mathbf{y}}, \mathbf{E}) = \bar{\mathbf{y}} \oplus c\mathbf{E}^{1/2}, \quad (11.1)$$

where $c = \chi^2_2(0.68)$ for 2D plots and $c = \chi^2_3(0.68)$ for 3D plots of standard 68% coverage.² Ellipses of various coverage were shown in Figure 3.9.

An ellipsoid representing variation in the means of a factor (or any other term reflected in a general linear hypothesis test, Equation 10.9) uses the corresponding \mathbf{H} matrix is simply the data ellipse of the fitted values for that term. But there is a question of the relative scaling of the \mathbf{H} and \mathbf{E} ellipsoids for interpretation.

Dividing the hypothesis matrix by the error degrees of freedom, giving \mathbf{H}/df_e , puts this on the same scale as the \mathbf{E} ellipse. I refer to this as *effect size scaling*, because it is similar to an effect size index used in univariate models, e.g., $ES = (\bar{y}_1 - \bar{y}_2)/s_e$ in a two-group, univariate design. An alternative, *significance scaling* (Section 11.4) provides a visual test of significance of a model \mathbf{H} term.

To illustrate this concretely, consider the HE plot for the `dogfood` shown in Figure 11.1 (c), reproduced here as Figure 11.2.

```
data(dogfood, package="heplots")
dogfood.mod <- lm(cbind(start, amount) ~ formula, data=dogfood)

heplot(dogfood.mod,
       fill = TRUE, fill.alpha = 0.1,
       cex.lab = 1.5, cex = 1.5,
       xlim = c(-1, 4.5),
       ylim = c(70, 100))
```

From the analysis in Section 10.2.2, we found the \mathbf{H} matrix for the `formula` effect in the `dogfood.mod` model to be as shown below. The negative covariance, -70.94, reflects a correlation of -0.94 between the means of start time and amount eaten.

```
dogfood.aov <- Anova(dogfood.mod)
SSP_H <- dogfood.aov$SSP[[1]] |> print()
#>           start amount
#> start     9.69   -70.9
#> amount   -70.94  585.7
```

Similarly, the \mathbf{E} matrix, shown below, reflects a slight positive correlation, 0.12, for dogs fed the same formula.

```
SSP_E <- dogfood.aov$SSPE |> print()
#>           start amount
#> start     25.8    11.8
#> amount    11.8   390.3
```

Example 11.1. Iris data

Perhaps the most famous (or infamous) dataset in the history of multivariate data analysis is that of

²In smallish samples ($n < 30$) we use the better approximations, $c = \sqrt{2F_{2,n-2}^{0.68}}$ for 2D plots and $c = \sqrt{3F_{3,n-3}^{0.68}}$ for 3D.

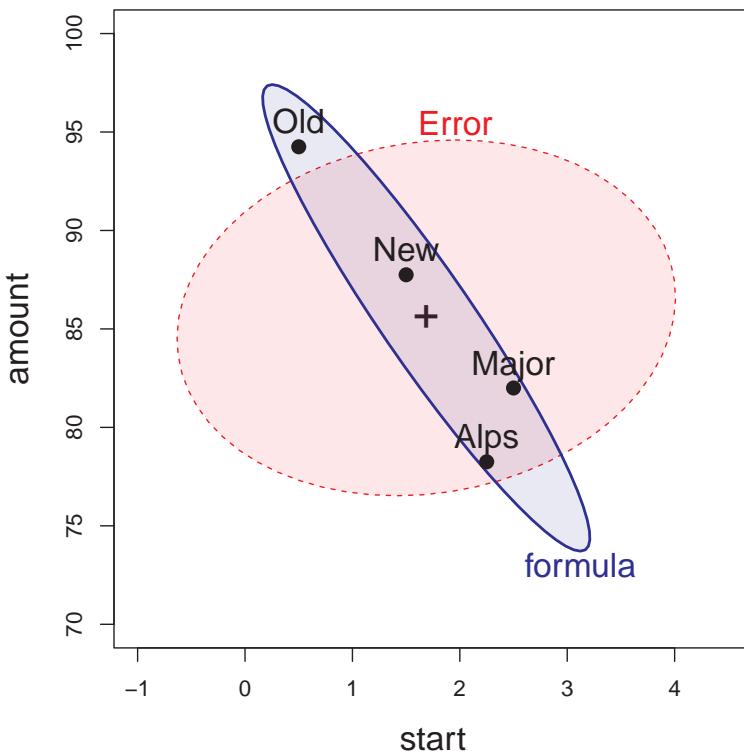


Figure 11.2: HE plot for the dogfood data, showing the means of the four groups, which generates the **H** ellipse for the effect of `formula`. The **E** ellipse labeled ‘Error’ shows the within-group variances and covariance.

measurements on three species of Iris flowers collected by Edgar Anderson (1935) in the Gaspé Peninsula of Québec, Canada. Anderson wanted to quantify the outward appearance (“morphology”: shape, structure, color, pattern, size) of species as a method to study variation within and between such groups. Although Anderson published in the obscure *Bulletin of the American Iris Society*, R. A. Fisher (1936) saw this as a challenge and opportunity to introduce the method now called discriminant analysis—how to find a weighted composite of variables to best discriminate among existing groups.

i History corner

I said “infamous” above because Fisher published in the *Annals of Eugenics*. He was an ardent eugenicist himself, and the work of eugenacists was often pervaded by prejudice against racial, ethnic and disabled groups. Through guilt by association, the Iris data, having mistakenly been called “Fisher’s Iris Data”, has become deprecated, even called “racist data”.³ The voices of the *Setosa*, *Versicolor* and *Virginica* of Gaspé protest: we don’t have a racist bone in our body and nor prejudice against any other species, to no avail.

Bodmer et al. (2021) present a careful account of Fisher’s views on eugenics within the context of his time and his contributions to modern statistical theory and practice. Fisher’s views on race were largely formed by Darwin and Galton, but “nearly all of Fisher’s statements were about populations, groups of populations, or the human species as a whole”. Regardless, the `iris` data were Anderson’s and should not be blamed. After all, if Anderson had gave his car to Fisher, would the car be tainted by Fisher’s eugenicist leanings?

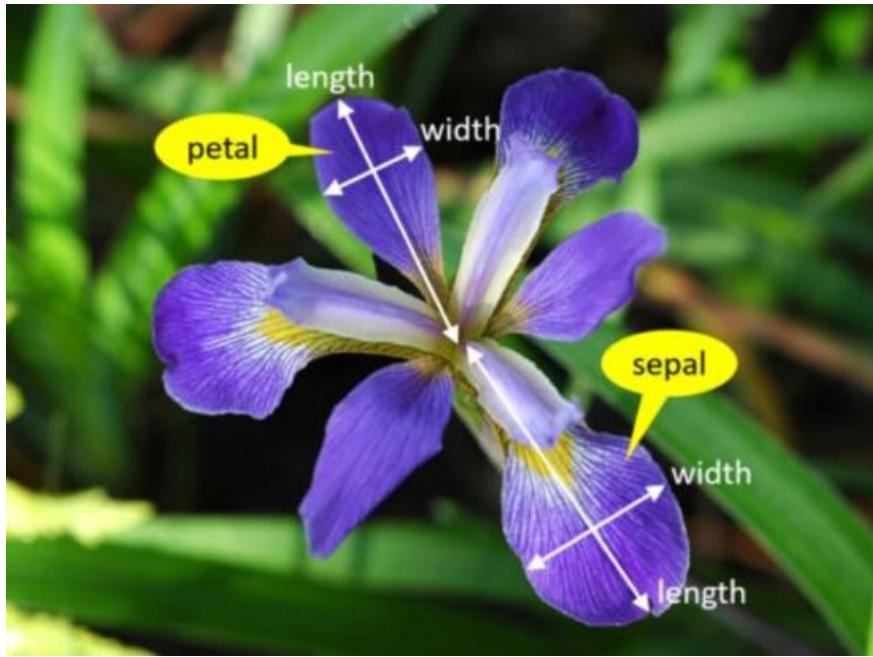


Figure 11.3: Diagram of an iris flower showing the measurements of petal and sepal size. Each flower has three sepals and three alternating petals. The sepals have brightly colored central sections. *Source:* Gayan De Silva (2020)

So that we understand what the measurements represent, Figure 11.3 superposes labels on a typical iris flower, having three sepals which alternate with three petals. Sepals are like ostentatious petals, with attractive decorations in the central section. Length is the distance from the center to the tip and width is the transverse dimension.

As always, it is useful to start with overview displays to see the data. A scatterplot matrix (Figure 11.4) shows that *versicolor* and *virginica* are more similar to each other than either is to *setosa*, both in their pairwise means (*setosa* are smaller) and in the slopes of regression lines. Further, the ellipses suggest that the assumption of constant within-group covariance matrices is problematic: While the shapes and sizes of the concentration ellipses for *versicolor* and *virginica* are reasonably similar, the shapes and sizes of the ellipses for *setosa* are different from the other two.

```
iris_colors <- c("blue", "darkgreen", "brown4")
scatterplotMatrix(~ Sepal.Length + Sepal.Width +
                  Petal.Length + Petal.Width | Species,
  data = iris,
  col = iris_colors,
  pch = 15:17,
  smooth=FALSE,
  regLine = TRUE,
  ellipse=list(levels=0.68, fill.alpha=0.1),
  diagonal = FALSE,
  legend = list(coords = "bottomleft",
                cex = 1.3, pt.cex = 1.2))
```

³For example, Megan Stodel in a blog post [Stop using iris](#) says, “*It is clear to me that knowingly using work that was itself used in pursuit of racist ideals is totally unacceptable.*” A Reddit discussion on this topic, [Is it socially acceptable to use the Iris dataset?](#) has some interesting replies.

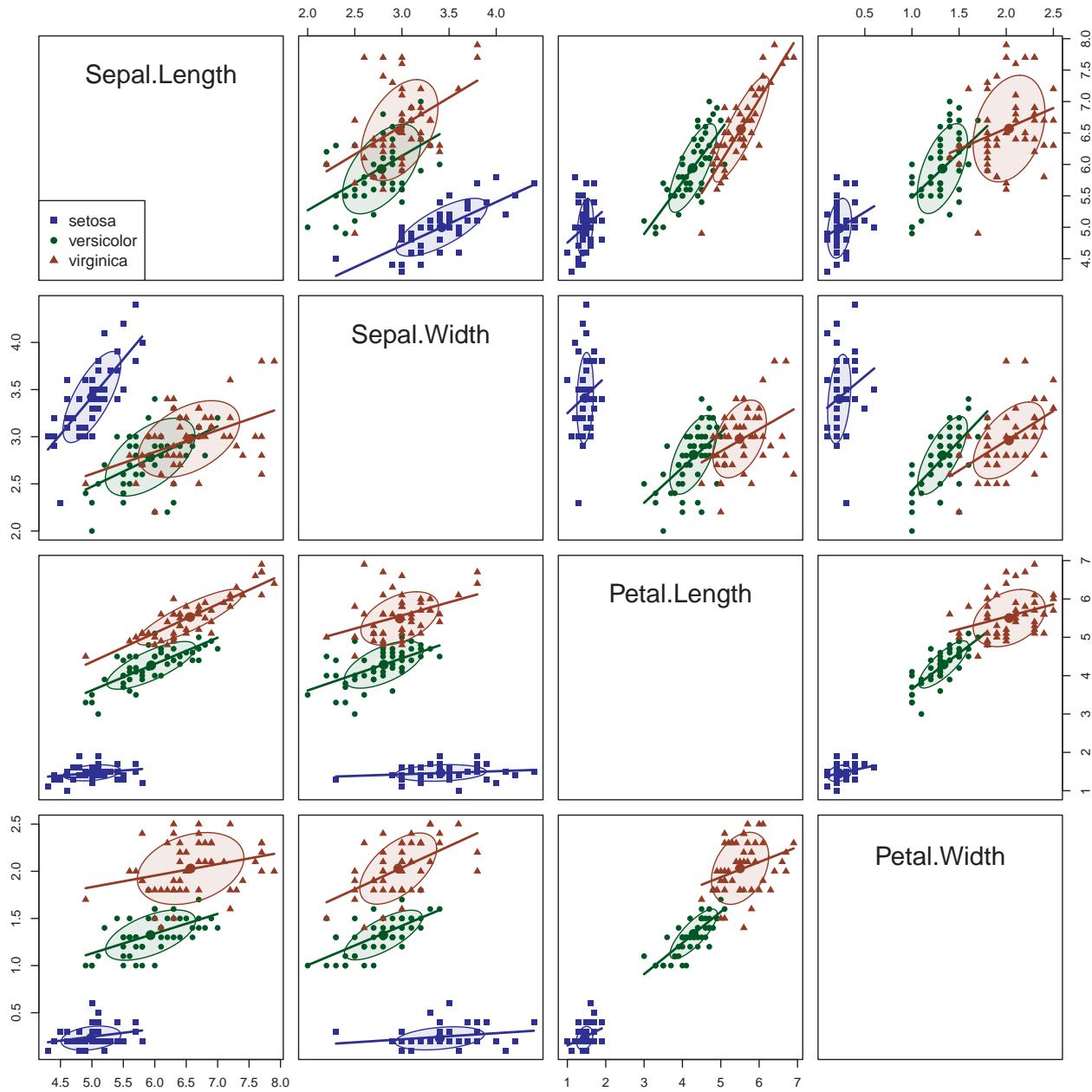


Figure 11.4: Scatterplot matrix of the `iris` dataset. The species are summarized by 68% data ellipses and linear regression lines in each pairwise plot.

**TODO*: Should this be a numbered section?

11.2.1 MANOVA model

We proceed nevertheless to fit a multivariate one-way ANOVA model to the iris data. The MANOVA model for these data addresses the question: “Do the means of the Species differ significantly for the sepal and petal variables taken together?”

$$\mathcal{H}_0 : \mu_{\text{setosa}} = \mu_{\text{versicolor}} = \mu_{\text{virginica}}$$

Because there are three species, the test involves $s = \min(p, g - 1) = 2$ degrees of freedom, and we are entitled

to represent this by two 1-df contrasts, or sub-questions. From the separation among the groups shown in Figure 11.4 (or more botanical knowledge), it makes sense to compare:

- Setosa vs. others: $\mathbf{c}_1 = (1, -\frac{1}{2}, -\frac{1}{2})$
- Versicolor vs. Virginica: : $\mathbf{c}_1 = (0, 1, -1)$

You can do this by putting these vectors as columns in a matrix and assigning this to the `contrasts()` of `Species`. It is important to do this *before* fitting with `lm()`, because the contrasts in effect determine how the `X` matrix is setup, and hence the names of the coefficients representing `Species`.

```
C <- matrix(c(1,-1/2,-1/2,
             0,   1,  -1), nrow=3, ncol=2)
contrasts(iris$Species) <- C
contrasts(iris$Species)
#>           [,1] [,2]
#> setosa      1.0   0
#> versicolor -0.5   1
#> virginica  -0.5  -1
```

Now let's fit the model. As you would expect from Figure 11.4, the differences among groups are highly significant.

```
iris.mod <- lm(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~
                 Species, data=iris)
Anova(iris.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> Species  2     1.19      53.5     8    290 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As a quick follow-up, it is useful to examine the univariate tests for each of the iris variables, using `heplots::glance()` or `heplots::uniStats()`. It is of interest that the univariate R^2 values are much larger for the petal variables than the sepal length and width.⁴ For comparison, `heplots::etasq()` gives the overall η^2 proportion of variance accounted for in all responses.

```
glance(iris.mod)
#> # A tibble: 4 x 8
#>   response    r.squared sigma fstatistic numdf dendf p.value   nobs
#>   <chr>        <dbl> <dbl>       <dbl> <dbl> <dbl> <dbl> <int>
#> 1 Sepal.Length  0.619  0.515      119.    2     147 1.67e-31   150
#> 2 Sepal.Width   0.401  0.340       49.2    2     147 4.49e-17   150
#> 3 Petal.Length  0.941  0.430      1180.   2     147 2.86e-91   150
#> 4 Petal.Width   0.929  0.205      960.    2     147 4.17e-85   150

etasq(iris.mod)
#>      eta^2
#> Species 0.596
```

But these statistics don't help to understand *how* the species differ. For this, we turn to HE plots.

⁴Recall that R^2 for a linear model is the the proportion of variation in the response that is explained by the model, calculated as $R^2 = SS_H/SS_T = SS_H/(SS_H + SSE)$. For a multivariate model, these are obtained from the diagonal elements of \mathbf{H} and \mathbf{E} .

11.3 HE plots

The `heplot()` function takes a "mle" object and produces an HE plot for one pair of variables specified by the `variables` argument. By default, it plots the first two. Figure 11.5 shows the HE plots for the two sepal and the two petal variables.

```
heplot(iris.mod, size = "effect",
       cex = 1.5, cex.lab = 1.5,
       fill = TRUE, fill.alpha = c(0.3, 0.1))
heplot(iris.mod, size = "effect", variables = 3:4,
       cex = 1.5, cex.lab = 1.5,
       fill = TRUE, fill.alpha = c(0.3, 0.1))
```

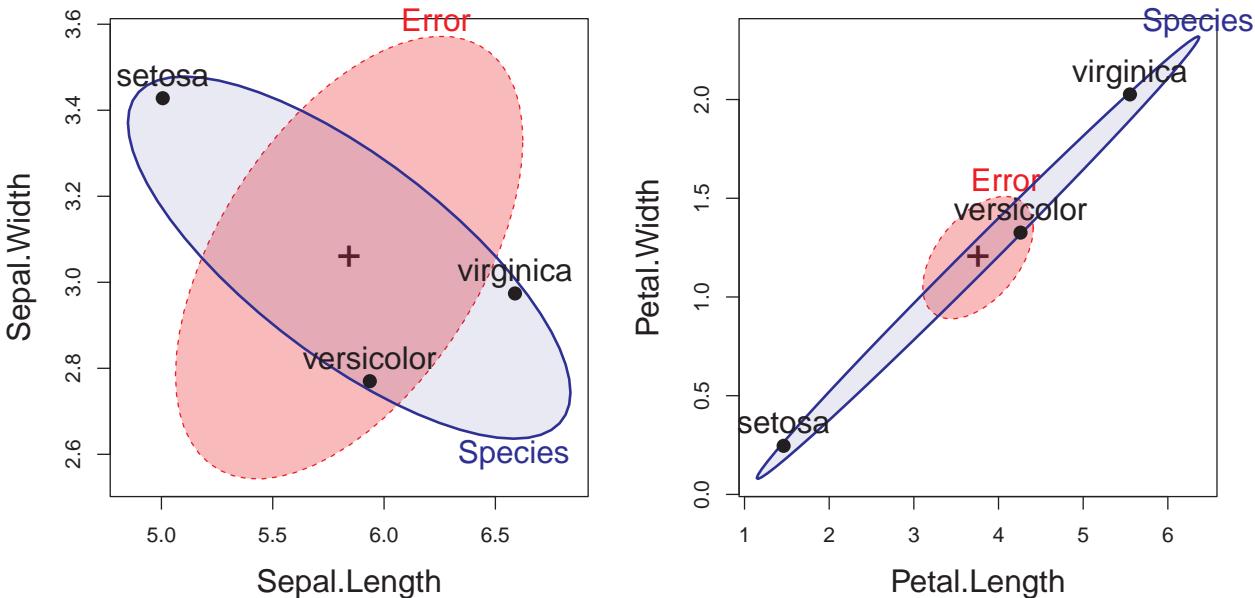


Figure 11.5: HE plots for the multivariate model `iris.mod`. The left panel shows the plot for the Sepal variables; the right panel plots the Petal variables.

The interpretation of the plots in Figure 11.5 is as follows:

- For the Sepal variables, length and width are positively correlated *within* species (the **E** = “Error” ellipsoid). The means of the groups (the **H** = “Species” ellipsoid), however, are negatively correlated. This plot is the HE plot representation of the data shown in row 2, column 1 of Figure 11.4. It reflects the relative **shape** of the iris sepals: shorter and wider for *setosa* than the other two species.
- For the Petal variables length and width are again positively correlated *within* species, but now the means of the groups are positively correlated: longer petals go with wider ones across species. This reflects the relative **size** of the iris petals. The analogous data plot appears in row 4, column 3 of Figure 11.4.

11.4 Significance scaling

The geometry of ellipsoids and multivariate tests allow us to go further with another re-scaling of the \mathbf{H} ellipsoid that gives a *visual test of significance* for any term in a MLM. This is done simply by dividing \mathbf{H}/df_e further by the α -critical value of the corresponding test statistic to show the strength of evidence against the null hypothesis.

Among the various multivariate test statistics, Roy's maximum root test, based on the largest eigenvalue λ_1 of \mathbf{HE}^{-1} , gives $\mathbf{H}/(\lambda_\alpha df_e)$ which has the visual property that the scaled \mathbf{H} ellipsoid will protrude *somewhere* outside the standard \mathbf{E} ellipsoid if and only if Roy's test is significant at significance level α . The critical value λ_α for Roy's test is

$$\lambda_\alpha = \left(\frac{df_1}{df_2} \right) F_{df_1, df_2}^{1-\alpha},$$

where $df_1 = \max(p, df_h)$ and $df_2 = df_h + df_e - df_1$.

For these data, the HE plot using significance scaling is shown in the right panel of Figure 11.6. The left panel is the same as that shown for sepal width and length in Figure 11.5, but with axis limits to make the two plots directly comparable.

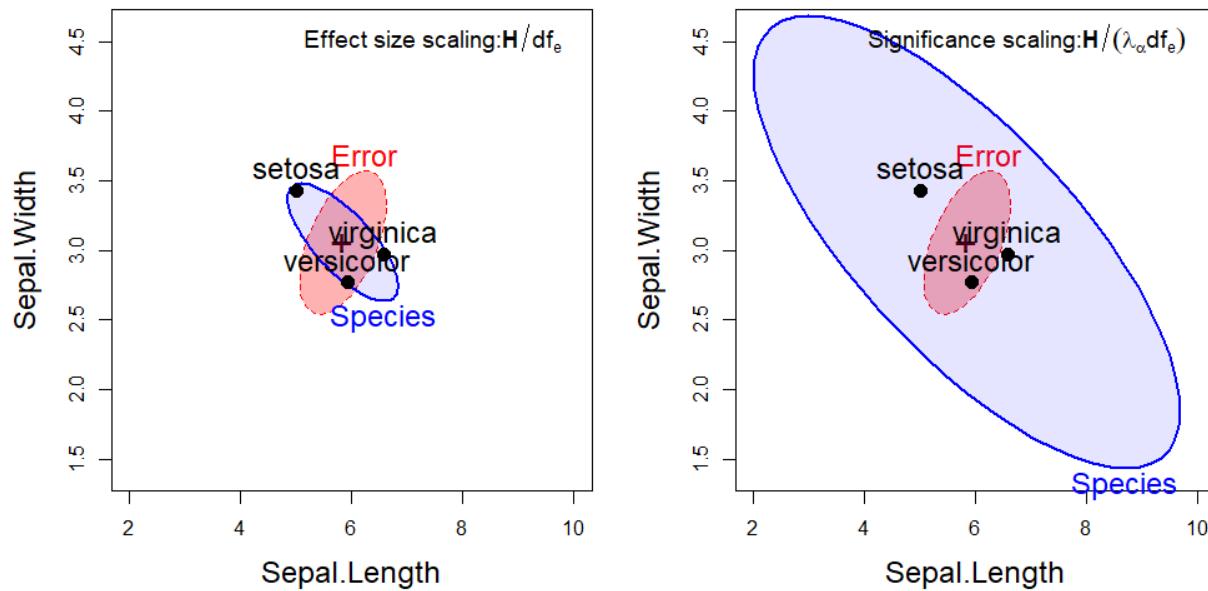


Figure 11.6: HE plots for sepal width and sepal length in the iris dataset. Left: *effect* scaling of the \mathbf{H} matrix; right: *significance* scaling, where protrusion of \mathbf{H} outside \mathbf{E} indicates a significant effect by Roy's test.

You can interpret the plot using effect scaling to indicate that the overall “size” of variation of the group means is roughly the same as that of within-group variation for the two sepal variables. Significance scaling weights the evidence against the null hypothesis that a given effect is zero. Clearly, the species vary significantly on the sepal variables, and the direction of the \mathbf{H} ellipse suggests that those whose sepals are longer are also less wide.

11.5 Visualizing contrasts and linear hypotheses

As described in Section 5.1.3, tests of linear hypotheses and contrasts represented by the general linear test $\mathcal{H}_0 : \mathbf{C} \mathbf{B} = \mathbf{0}$ provide a powerful way to probe the *specific* effects represented within the global null hypothesis, $\mathcal{H}_0 : \mathbf{B} = \mathbf{0}$, that all effects are zero.

In this example the contrasts \mathbf{c}_1 (`Species1`) and \mathbf{c}_2 (`Species2`) among the iris species are orthogonal, i.e., $\mathbf{c}_1^\top \mathbf{c}_2 = 0$. Therefore, their tests are statistically independent, and their \mathbf{H} matrices are additive. They fully decompose the general question of differences among the groups into two independent questions regarding the contrasts.

$$\mathbf{H}_{\text{Species}} = \mathbf{H}_{\text{Species1}} + \mathbf{H}_{\text{Species2}} \quad (11.2)$$

`car::linearHypothesis()` is the means for testing these statistically, and `heplot()` provides the way to show these tests visually. Using the contrasts set up in Section 11.2.1, \mathbf{c}_1 , representing the difference between `setosa` and the other species is labeled `Species1` and the comparison of `versicolor` with `virginica` is `Species2`. The coefficients for these in \mathbf{B} give the differences in the means. The line for `(Intercept)` gives grand means of the variables.

```
coef(iris.mod)
#>           Sepal.Length Sepal.Width Petal.Length Petal.Width
#> (Intercept)      5.843       3.057      3.758       1.199
#> Species1        -0.837       0.371     -2.296      -0.953
#> Species2        -0.326      -0.102      -0.646      -0.350
```

Numerical tests of hypotheses using `linearHypothesis()` can be specified in a very general way: A matrix (or vector) \mathbf{C} giving linear combinations of coefficients by rows, or a character vector giving the hypothesis in symbolic form. A character variable or vector tests whether the named coefficients are different from zero for all responses.

```
linearHypothesis(iris.mod, "Species1") |> print(SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.97    1064     4   144 <2e-16 ***
#> Wilks       1   0.03    1064     4   144 <2e-16 ***
#> Hotelling-Lawley 1   29.55   1064     4   144 <2e-16 ***
#> Roy         1   29.55   1064     4   144 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

linearHypothesis(iris.mod, "Species2") |> print(SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.745    105     4   144 <2e-16 ***
#> Wilks       1   0.255    105     4   144 <2e-16 ***
#> Hotelling-Lawley 1   2.925   105     4   144 <2e-16 ***
#> Roy         1   2.925   105     4   144 <2e-16 ***
```

The various test statistics are all equivalent here—they give the same F statistics—because they have 1 degree of freedom.

In passing, from Equation 11.2, note that the *joint* test of these contrasts is exactly equivalent to the overall test of **Species** (results not shown).

```
linearHypothesis(iris.mod, c("Species1", "Species2"))
```

We can show these contrasts in an HE plot by supplying a named list for the **hypotheses** argument. The names are used as labels in the plot. In the case of a 1-df multivariate test, the **H** ellipses plot as a degenerate line.

```
hyp <- list("S:Vv" = "Species1", "V:v" = "Species2")
heplot(iris.mod, hypotheses=hyp,
       cex = 1.5, cex.lab = 1.5,
       fill = TRUE, fill.alpha = c(0.3, 0.1),
       col = c("red", "blue", "darkgreen", "darkgreen"),
       lty = c(0,0,1,1), label.pos = c(3, 1, 2, 1),
       xlim = c(2, 10), ylim = c(1.4, 4.6))
```

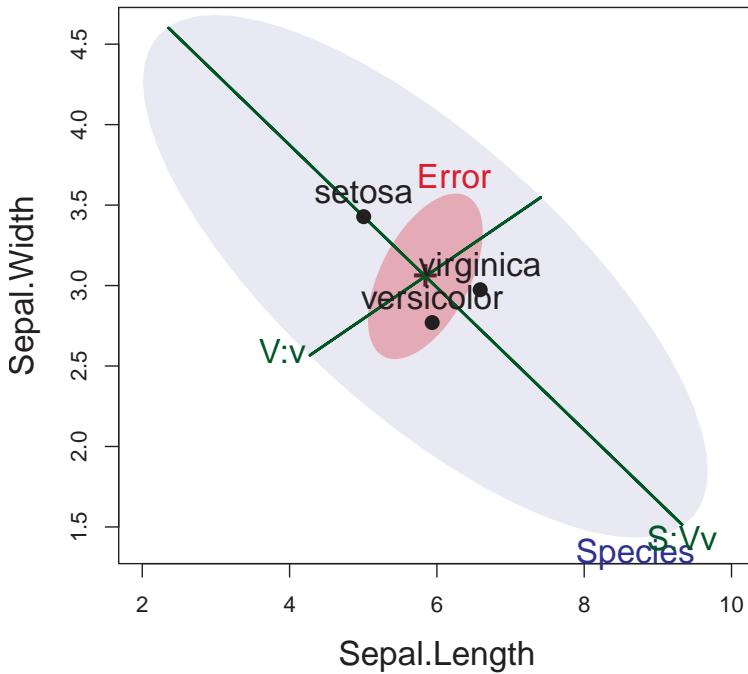


Figure 11.7: HE plot for sepal length and width in the `iris` data showing the tests of the two contrasts, using significance scaling.

This HE plot shows that, for the two sepal variables, the greatest between-species variation is accounted for by the contrast (**S:Vv**) between *setosa* and the others, for which the effect is very large in relation to error variation. The second contrast (**V:v**), between the *versicolor* and *virginica* species is relatively smaller, but still explains significant variation of the sepal variables among the species.

The directions of these hypotheses in a given plot show *how* the group means differ in terms of a given

contrast.⁵ For example, the contrast $\mathbf{S}:\mathbf{Vv}$ is the line that separates *setosa* from the others and indicates that *setosa* flowers have shorter but wider sepals.

11.6 HE plot matrices

In base R graphics, 2D scatterplots are extended to all pairwise views of multivariate data with a `pairs()` method. For multivariate linear models, the `heplots` defines a `pairs.mlm()` method to display HE plots for all pairs of the response variables.

```
pairs(iris.mod,
      fill=TRUE, fill.alpha=c(0.3, 0.1))
```

Figure 11.8 provides a fairly complete visualization of the results of the multivariate tests and answers the question: **how** do the species differ? Sepal length and the two petal variables have the group means nearly perfectly correlated, in the order *setosa* < *versicolor* < *virginica*. For Sepal width, however, *setosa* has the largest mean, and so the **H** ellipses show a negative correlation in the second row and column.

11.7 Low-D views: Canonical analysis

The HE plot framework so far provides views of all the effects in a MLM in *variable* space. We can view this in 2D for selected pairs of response variables, or for all pairwise views in scatterplot matrix format, as in Figure 11.8.

There is also an `heplot3d()` function giving plots for three response variables together. The 3D plots are interactive, in that they can be rotated and zoomed by mouse control, and dynamic, in that they can be made to spin and saved as movies. To save space, these plots are not shown here.

However in a one-way MANOVA design with more than response three variables, it is difficult to visualize how the groups vary on *all* responses together, and how the different variables contribute to discrimination among groups. In this situation, **canonical discriminant analysis** (CDA) is often used, to provide a low-D visualization of between-group variation.

When the predictors are also continuous, the analogous term is **canonical correlation analysis** (CCA). The advantage in both cases is that we can also show the relations of the response variables to these dimensions, similar to a biplot (Section 4.3) for a PCA of purely quantitative variables.

The key to this is the eigenvalue decomposition, $\mathbf{H}\mathbf{E}^{-1}\lambda_i = \lambda_i\mathbf{v}_i$ (Equation 10.7) of **H** relative to **E**. The eigenvalues, λ_i , give the “size” of each s orthogonal dimensions on which the multivariate tests are based (Section 10.2.3). But the corresponding eigenvectors, \mathbf{v}_i , give the *weights* for the response variables in s linear combinations that maximally discriminate among the groups or equivalently maximize the (canonical) R^2 of a linear combination of the predictor **X**s with a linear combination of the response **Y**s.

Thus, CDA amounts to a transformation of the p responses $\mathbf{Y}_{n \times p}$ into scores \mathbf{Z} in the canonical space, $\mathbf{Z}_{n \times s} = \mathbf{Y} \mathbf{E}^{-1/2} \mathbf{V}$,

where **V** contains the eigenvectors of $\mathbf{H}\mathbf{E}^{-1}$ and $s = \min(p, df_h)$ dimensions, the degrees of freedom for the hypothesis. It is well-known (e.g., Gittins (1985)) that *canonical discriminant plots* of the first two (or three,

⁵That the **H** ellipses for the contrasts subtend that for the overall test of **Species** is no accident. In fact, this is true in p -dimensional space for *any* linear hypothesis, and orthogonal contrasts have the additional geometric property that they form *conjugate axes* for the overall **H** ellipsoid relative to the **E** ellipsoid (Friendly et al., 2013).

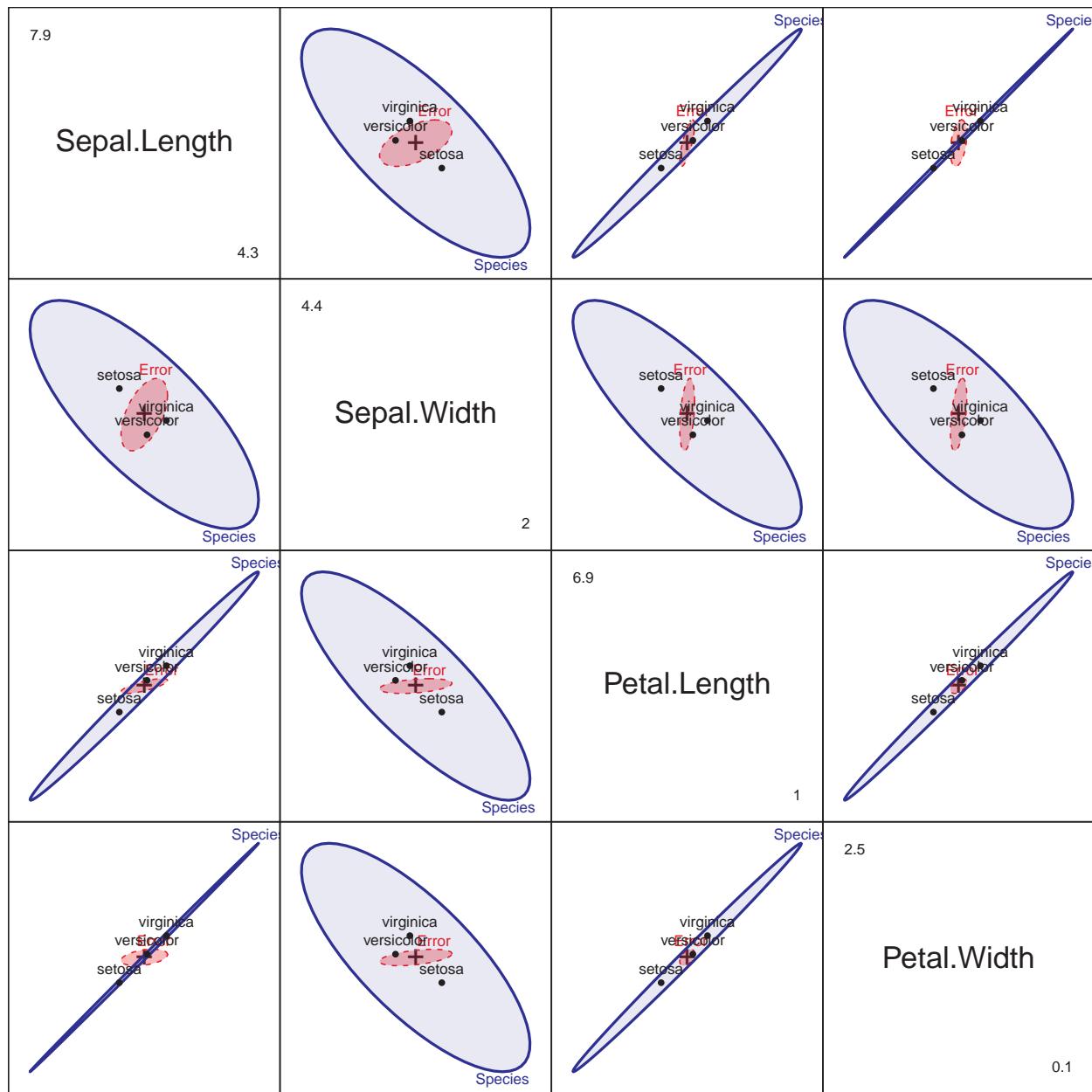


Figure 11.8: All pairwise HE plots for the iris data.

in 3D) columns of \mathbf{Z} corresponding to the largest canonical correlations provide an optimal low-D display of the variation between groups relative to variation within groups.

Canonical discriminant analysis is typically carried out in conjunction with a one-way MANOVA design. The `candisc` package (Friendly & Fox, 2025) generalizes this to multi-factor designs in the `candisc()` function. For any given term in a "mlm", the *generalized canonical discriminant analysis* amounts to a standard discriminant analysis based on the \mathbf{H} matrix for *that* term in relation to the full-model \mathbf{E} matrix.⁶

Tests based on the eigenvalues λ_i , initially stated by Bartlett (1938), use Wilks' Λ likelihood ratio tests of

⁶The related `candiscList()` function performs a generalized canonical discriminant analysis for *all* terms in a multivariate linear model, returning a list of the results for each factor.

these. This allow you to determine the number of significant canonical dimensions, or the number of different aspects to consider for the relations between the responses and predictors. This is a big win over univariate analyses for each dependent variable separately as follow-ups for a significant MANOVA result.

These take the form of *sequential* global tests of the hypothesis that the canonical correlation in the current row and all that follow are zero. Thus, if you find The canonical R^2 , `CanRsq`, gives the R-squared value of fitting the i th response canonical variate to the corresponding i th canonical variate for the predictors.

For the iris data, we get the following printed summary:

```
iris.can <- candisc(iris.mod) |> print()
#>
#> Canonical Discriminant Analysis for Species:
#>
#>   CanRsq Eigenvalue Difference Percent Cumulative
#> 1  0.970      32.192       31.9  99.121      99.1
#> 2  0.222       0.285       31.9   0.879     100.0
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>   LR test stat approx F numDF denDF Pr(> F)
#> 1        0.023     199.1      8    288 < 2e-16 ***
#> 2        0.778     13.8      3    145 5.8e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This analysis shows a very simple result: The differences among the iris species can be nearly entirely accounted for by the first canonical dimension (99.1%). Interestingly, the second dimension is also highly significant, even though it accounts for only 0.88%.

11.7.1 Coeficients

The `coef()` method for “`candisc`” objects returns a matrix of weights for the response variables in the canonical dimensions. By default, these are given for the response variables *standardized* to $\bar{y} = 0$ and $s_y^2 = 1$.

The `type` argument also allows for *raw score* weights (`type = "raw"`) used to compute scores for the observations on the canonical variables `Can1`, `Can2`, Using `type = "structure"` gives the canonical structure coefficients, which are the correlations between each response and the canonical scores.

```
coef(iris.can, type = "std")
#>           Can1   Can2
#> Sepal.Length  0.427  0.0124
#> Sepal.Width   0.521  0.7353
#> Petal.Length -0.947 -0.4010
#> Petal.Width   -0.575  0.5810
coef(iris.can, type = "structure")
#>           Can1   Can2
#> Sepal.Length -0.792  0.218
#> Sepal.Width   0.531  0.758
#> Petal.Length -0.985  0.046
#> Petal.Width   -0.973  0.223
```

The standardized (or raw score) weights are interpreted in terms of their signs and magnitudes, just as in

coefficient weights in a multiple regression. From the numbers, `Can1` seems to be a contrast between the sepal and petal variables. For `Can2`, sepal length doesn't matter and the result contrasts the two width variables against petal length.

I find it easier to interpret the correlations between the observed and canonical variables, given as the canonical structure coefficients. These are easily visualized as vectors in canonical space (similar to biplots for a PCA), as shown below (Figure 11.9).

11.7.2 Canonical scores plot

The `plot()` method for "candisc" objects gives a plot of these observation scores. `ellipse=TRUE` overlays this with their standard data ellipses for each species, as shown in Figure 11.9. The response variables are shown as vectors, using the structure coefficients, as in a biplot. Thus, the relative size of the projection of these vectors on the canonical axes reflects the correlation of the observed response on the canonical dimension. For ease of interpretation I flipped the sign of the first canonical dimension (`rev.axes`), so that the positive `Can1` direction corresponds to larger flowers.

```
vars <- names(iris)[1:4] |>
  stringr::str_replace("\\\\.", "\\n")
plot(iris.can,
  var.labels = vars,
  var.col = "black",
  var.lwd = 2,
  ellipse=TRUE,
  scale = 9,
  col = iris_colors,
  pch = 15:17,
  cex = 0.7, var.cex = 1.25,
  rev.axes = c(TRUE, FALSE),
  xlim = c(-10, 10),
  cex.lab = 1.5)
```

The interpretation of this plot is simple: in canonical space, variation of the means for the iris species is essentially one-dimensional (99.1% of the effect of `Species`), and this dimension corresponds to overall size of the iris flowers: The Setosas have smaller flowers.

All variables except for `Sepal.Width` are positively aligned with this axis, but the two petal variables show the greatest discrimination. The negative direction for `Sepal.Width` reflects the pattern seen in Figure 11.8, where *setosa* have wider sepals.

For the second dimension, look at the projections of the variable vectors on the `Can2` axis. All are positive, but this is dominated by `Sepal.Width`. We could call this a flower shape dimension.

11.7.3 Canonical HE plot

For a one-way design, the *canonical HE plot* is simply the HE plot of the canonical scores in the analogous MLM model that substitutes **Z** for **Y**. In effect, it is a more compact visual summary of the plot shown in Figure 11.9.

This is shown in Figure 11.10 for the `iris` data. In canonical space, the residuals are always uncorrelated, so the **E** ellipse plots as a circle. The **H** ellipse for `Species` here reflects a data ellipse for the fitted values—group means—shown as labeled points in the plot. The differences among `Species` are so large that this plot uses `size = "effect"` scaling, making the axes comparable to those in Figure 11.9.⁷

⁷If significance scaling was used the interpretation of the canonical HE plot would be the same as before: if the hypothesis ellipse extends beyond the error ellipse, then that dimension is significant.

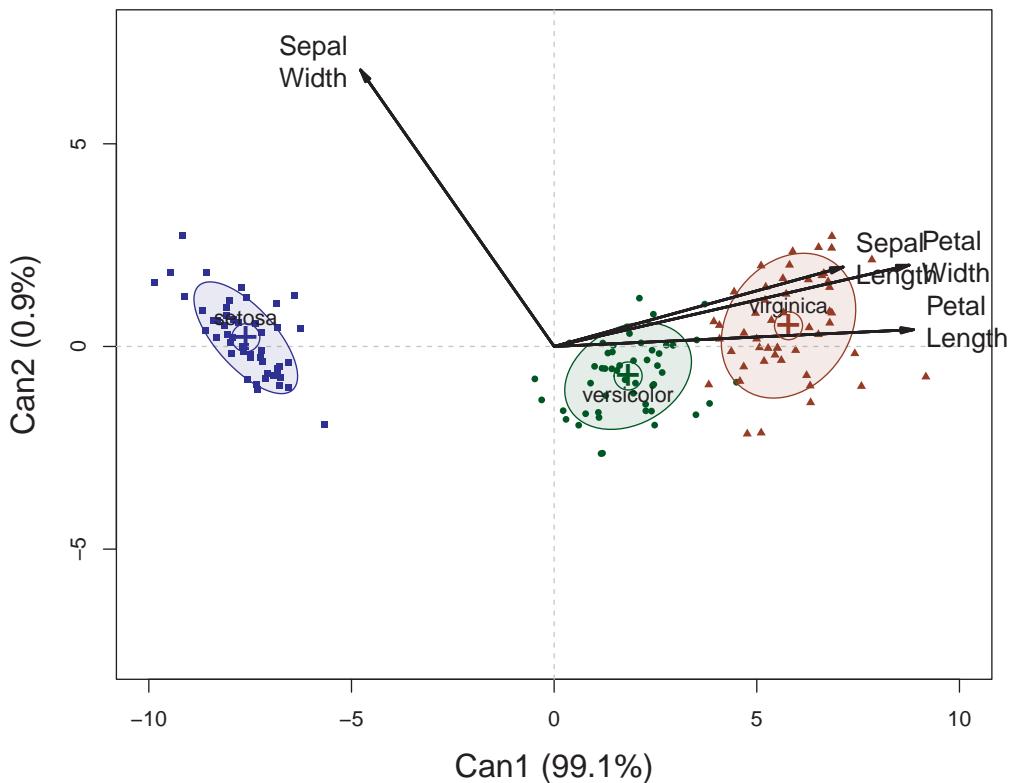


Figure 11.9: Plot of canonical scores for the iris data. Ellipses give 68% coverage data ellipses for the canonical scores. Variable vectors make angles with the Can1 and Can2 axes indicating their correlations.

The vectors for each predictor are the same structure coefficients as in the ordinary canonical plot. They can again be reflected for easier interpretation and scaled in length to fill the plot window.

```
heplot(iris.can,
      size = "effect",
      scale = 8,
      var.labels = vars,
      var.col = "black",
      var.lwd = 2,
      var.cex = 1.25,
      fill = TRUE, fill.alpha = 0.2,
      rev.axes = c(TRUE, FALSE),
      xlim = c(-10, 10),
      cex.lab = 1.5)
```

The collection of plots shown for the iris data here can be seen as progressive visual summaries of the data and increased visual understanding of

- The scatterplot matrix in Figure 11.4 shows the iris flowers in the data space of the sepal and petal variables.
- Canonical analysis substitutes for these the two linear combinations reflected in Can1 and Can2 . The plot in Figure 11.9 portrays *exactly* the same relations among the species, but in the reduced canonical space of only two dimensions.
- The HE plot version, shown in Figure 11.10 summarizes the separate data ellipses for the species with pooled, within-group variance of the \mathbf{E} matrix for the canonical variables, which are always uncorrelated. The variation among the group means is reflected in the size and shape of the ellipse for the \mathbf{E} matrix.

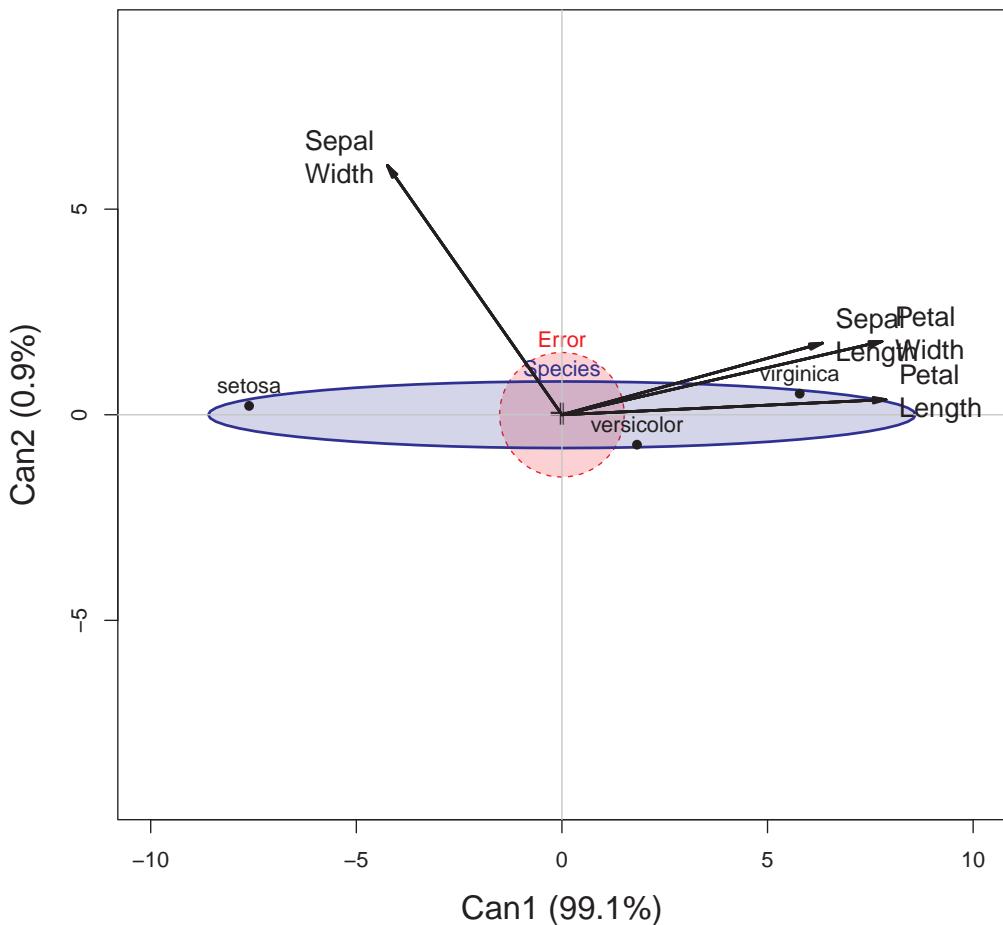


Figure 11.10: Canonical HE plot for the iris data. Compared with Figure 11.9, it substitutes canonical \mathbf{H} and \mathbf{E} ellipses for the canonical scores shown there.

11.8 Factorial MANOVA

When there are two or more factors, the overall model is comprised of main effects and possible interactions as shown in [?@eq-HE-model](#). A significant advantage of HE plots is that the main effects and interactions can be overlaid in the same plot showing how each term contributes to assessment of differences among the groups.

I illustrate this below for a simple 2×2 factorial design with three response variables.

Example 11.2. Plastic film data

An industrial experiment was conducted to determine the optimal conditions for extruding plastic film. Of interest were three responses: resistance to `tear`, film `gloss` and the `opacity` of the film. Two factors were manipulated, both at two levels, labeled High and Low: change in `rate` of extrusion (-10%, +10%) and amount of some `additive` (1%, 1.5%), with $n = 5$ runs at each combination of the factor levels. The dataset `heplots::Plastic` comes from Johnson & Wichern (1998), Example 6.12.

```
data(Plastic, package="heplots")
str(Plastic)
#> 'data.frame': 20 obs. of 5 variables:
#> $ tear    : num 6.5 6.2 5.8 6.5 6.5 6.9 7.2 6.9 6.1 6.3 ...
#> $ gloss   : num 9.5 9.9 9.6 9.6 9.2 9.1 10 9.9 9.5 9.4 ...
#> $ opacity : num 4.4 6.4 3 4.1 0.8 5.7 2 3.9 1.9 5.7 ...
#> $ rate    : Factor w/ 2 levels "Low","High": 1 1 1 1 1 1 1 1 1 ...
#> $ additive: Factor w/ 2 levels "Low","High": 1 1 1 1 1 2 2 2 2 2 ...
```

Multivariate tests

The MANOVA model `plastic.mod` fits the main effects of `rate` and `additive` and the interaction `rate:additive`. The `Anova()` summary shows a strong effect for `rate` of extrusions for the three responses jointly, a lesser, but still significant effect of `additive` and a non-significant interaction.

```
plastic.mod <- lm(cbind(tear, gloss, opacity) ~ rate*additive, data=Plastic)
Anova(plastic.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> rate       1    0.618    7.55      3     14   0.003 **
#> additive    1    0.477    4.26      3     14   0.025 *
#> rate:additive 1    0.223    1.34      3     14   0.302
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As a reminder, if you want to see the results of univariate tests for the responses separately, `heplots::uniStats()` (or `heplots::glance.mlm()`) gives some answers, including R^2 and univariate F tests.

```
uniStats(plastic.mod)
#> Univariate tests for responses in the multivariate linear model plastic.mod
#>
#>          R^2    F df1 df2 Pr(>F)
#> tear    0.586 7.56   3  16 0.0023 **
#> gloss   0.483 4.99   3  16 0.0125 *
#> opacity 0.125 0.76   3  16 0.5315
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

HE plots

We can visualize *all* these effects for pairs of variables in an HE plot, showing the “size” and orientation of hypothesis variation (**H**) for each model term, in relation to error variation (**E**), as ellipsoids. When, as here, the model terms have 1 degree of freedom, the **H** ellipsoids for `rate`, `additive` and `rate:additive` each degenerate to a line.

Figure 11.11 shows the HE plot for the responses `tear` and `gloss`, the strongest by univariate tests. This plot takes advantage of another feature of `heplot()`: You can overlay plots using `add = TRUE`, as is done here to show both significance and effect size scaling in a single plot.

```
colors = c("red", "darkblue", "darkgreen", "brown4")
heplot(plastic.mod, size="significance",
```

```

  col=colors, cex=1.5, cex.lab = 1.5,
  fill=TRUE, fill.alpha=0.1)
heplot(plastic.mod, size="effect",
  col=colors, lwd=6,
  add=TRUE, term.labels=FALSE)

```

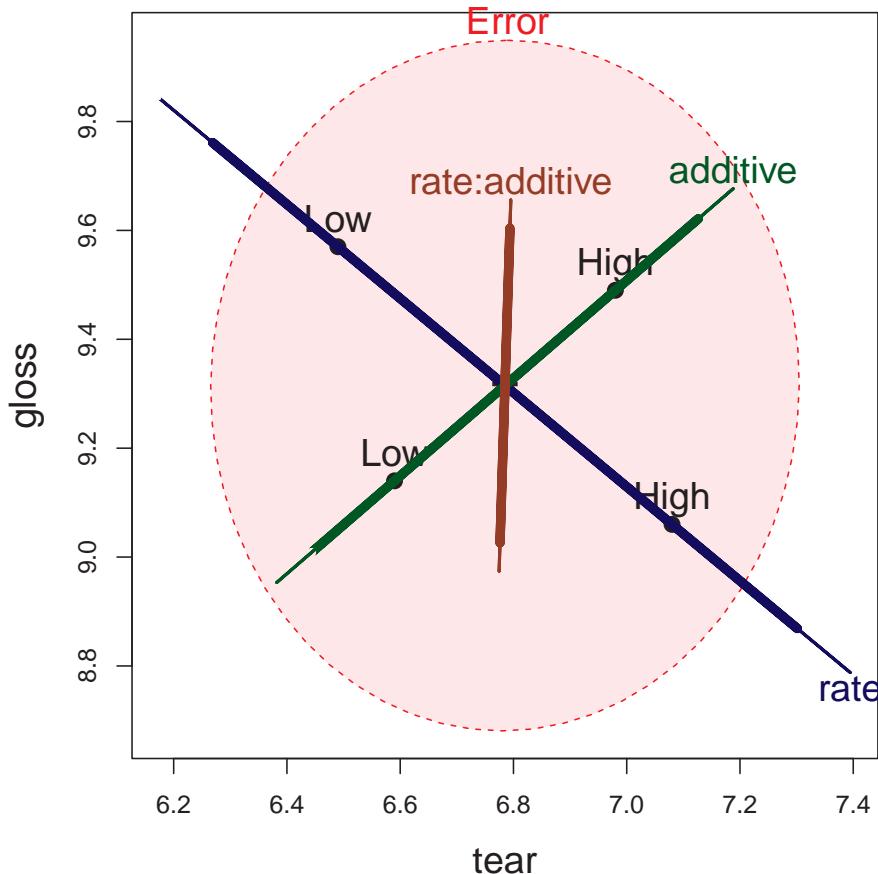


Figure 11.11: HE plot for effects on `tear` and `gloss` according to the factors `rate`, `additive` and their interaction, `rate:additive`. The thicker lines show effect size scaling; thinner lines show significance scaling.

In this view, the effect of extrusion `rate` is highly significant, with the high level giving larger tear resistance and lower gloss on average. High level of `additive` produces greater tear resistance and higher gloss. The interaction effect, `rate:additive`, while non-significant, points nearly entirely in the direction of `gloss`.

Understanding interactions

To understand an interaction effect, the simplest thing is to plot the cell means and error bars in a traditional line plot with one factor on the horizontal axis and the other as separate lines within the panel. Here (Figure 11.12) I use `ggpubr::gglne()`, which simplifies such plots, though at the expense of a bit of control for `ggplot2`.⁸

```

p1 <- gglne(Plastic,
  x = "rate", y = "tear",

```

⁸Line plots like this are nearly always more understandable with labels directly on the lines, rather than in a legend. Wrappers for `ggplot2` simplify some things but can't accommodate this kind of customization.

```

color = "additive", shape = "additive",
add = c("mean_se"),
position = "dodge",
point.size = 5, linewidth = 1.5,
ggtheme = theme_pubr(base_size = 16)
)

p2 <- ggline(Plastic,
  x = "rate", y = "gloss",
  color = "additive", shape = "additive",
  add = c("mean_se"),
  position = "dodge",
  point.size = 5, linewidth = 1.5,
  ggtheme = theme_pubr(base_size = 16)
)

p1 + p2

```

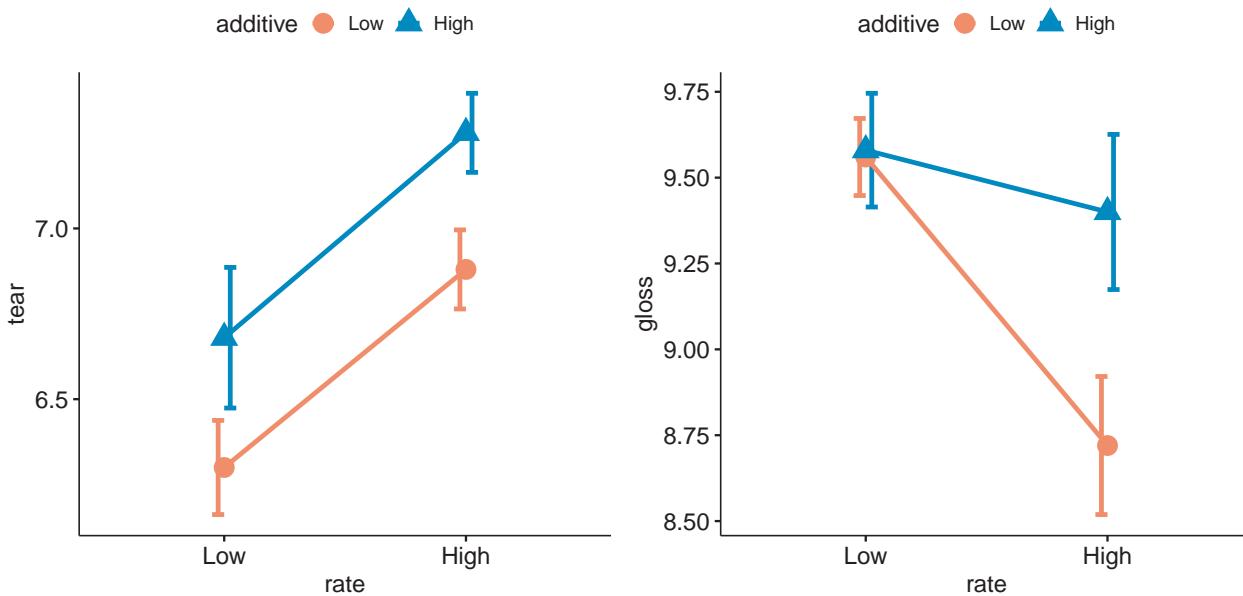


Figure 11.12: Line plots of means and their standard errors for the response `tear` (left) and `gloss` (right)"

For resistance to `tear`, the lines are parallel, so there is no interaction, which is why the `rate:additive` effect had no horizontal component in Figure 11.11. In the panel for `gloss`, the means for `additive` don't differ at high `rate`, but do substantially at the low extrusion rate.

But what if you really wanted to show the means for the combinations of rate and additive in an HE plot? By design, means for the levels of interaction terms are not shown in the HE plot, because doing so in general can lead to messy displays.

We can add them here for the term `rate:additive` as shown in Figure 11.13. This uses `heplots::termMeans()` to find the cell means for the combinations of the two factors and then `lines()` to connect the pairs of points for the low and high :levels of `additive`.

```

par(mar = c(4,4,1,1)+.1)
heplot(plastic.mod, size="evidence",
       col=colors, cex=1.5, cex.lab = 1.5,
       lwd = c(1, 5),
       fill=TRUE, fill.alpha=0.05)

## add interaction means
intMeans <- termMeans(plastic.mod, 'rate:additive',
                       abbrev.levels=3)
points(intMeans[,1], intMeans[,2], pch=18, cex=1.2, col="brown4")
text(intMeans[,1], intMeans[,2], rownames(intMeans),
     adj=c(0.5, 1), col="brown4")
lines(intMeans[c(1,3),1], intMeans[c(1,3),2],
      col="brown4", lwd = 3)
lines(intMeans[c(2,4),1], intMeans[c(2,4),2],
      col="brown4", lwd = 3)

```

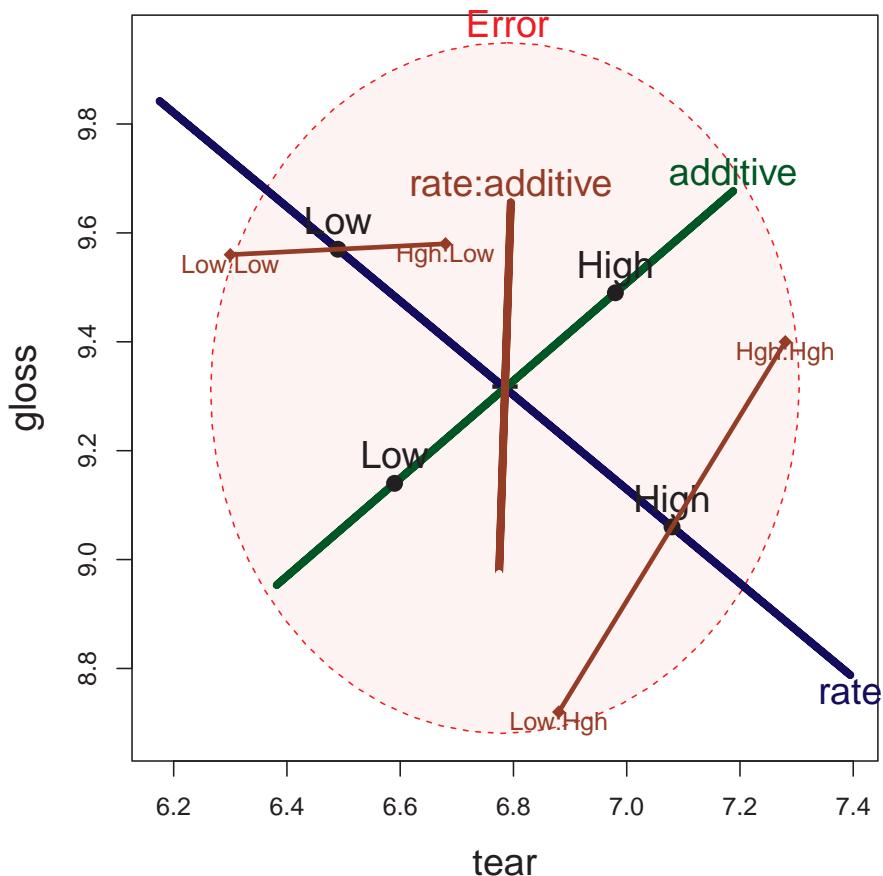


Figure 11.13: HE plot for effects on `tear` and `gloss` using significance scaling. To this is added points showing the means for the combinations of `rate` and `additive`.

11.9 Quantitative predictors: MMRA

The ideas behind HE plots extend naturally to multivariate multiple regression (MMRA). A purely visual feature of HE plots in these cases is that the **H** ellipse for a quantitative predictor with 1 df appears as a degenerate line. But consequently, the angles between these for different predictors has a simple interpretation as as the correlation between their predicted effects. Moreover, it is easy to show visual overall tests of *joint* linear hypotheses for two or more predictors together.

TODO: Use these examples below ? * `heplots::Hernior`: Recovery from Elective Herniorrhaphy -> HE_mmra vignette – Use for exercise

Example 11.3. NLSY data

Here I'll continue the analysis of the NLSY data from Section 10.5.1. In the model `NLSY.mod1`, I used only father's income and education to predict scores in reading and math, and both of these demographic variables were highly significant. Figure 11.14 shows what this looks like in an HE plot.

```
data(NLSY, package = "heplots")
NLSY.mod1 <- lm(cbind(read, math) ~ income + educ,
                 data = NLSY)

heplot(NLSY.mod1,
       fill=TRUE, fill.alpha = 0.2,
       cex = 1.5, cex.lab = 1.5,
       lwd=c(2, 3, 3),
       label.pos = c("bottom", "top", "top")
     )
```

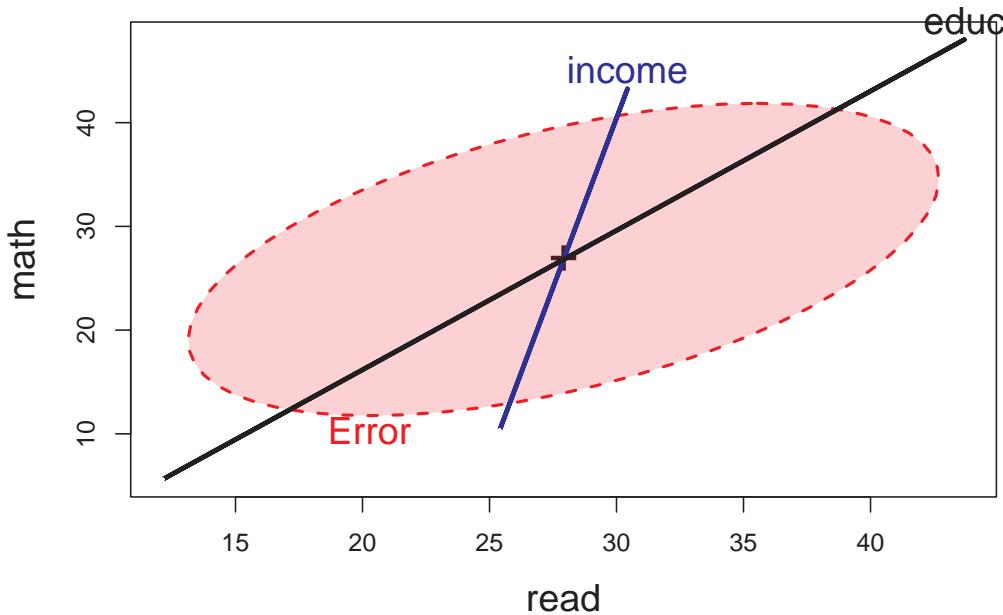


Figure 11.14: HE plot for the simple model for the NLSY data fitting reading and math scores from income and education.

Fathers income and education are positively correlated in their effects on the outcome scores. From the

angles in the plot, income is most related to the math score, while education is related to both, but slightly more to the reading score.

The overall joint test for both predictors can then be visualized as the test of the linear hypothesis $\mathcal{H}_0 : \mathbf{B} = [\beta_{\text{income}}, \beta_{\text{educ}}] = \mathbf{0}$. For `heplot()`, we specify the names of the coefficients to be tested with the `hypotheses` argument.

```
coefs <- rownames(coef(NLSY.mod1))[-1] |> print()
#> [1] "income" "educ"

heplot(NLSY.mod1,
       hypotheses = list("Overall" = coefs),
       fill=TRUE, fill.alpha = 0.2,
       cex = 1.5, cex.lab = 1.5,
       lwd=c(2, 3, 3, 2),
       label.pos = c("bottom", "top"))
```

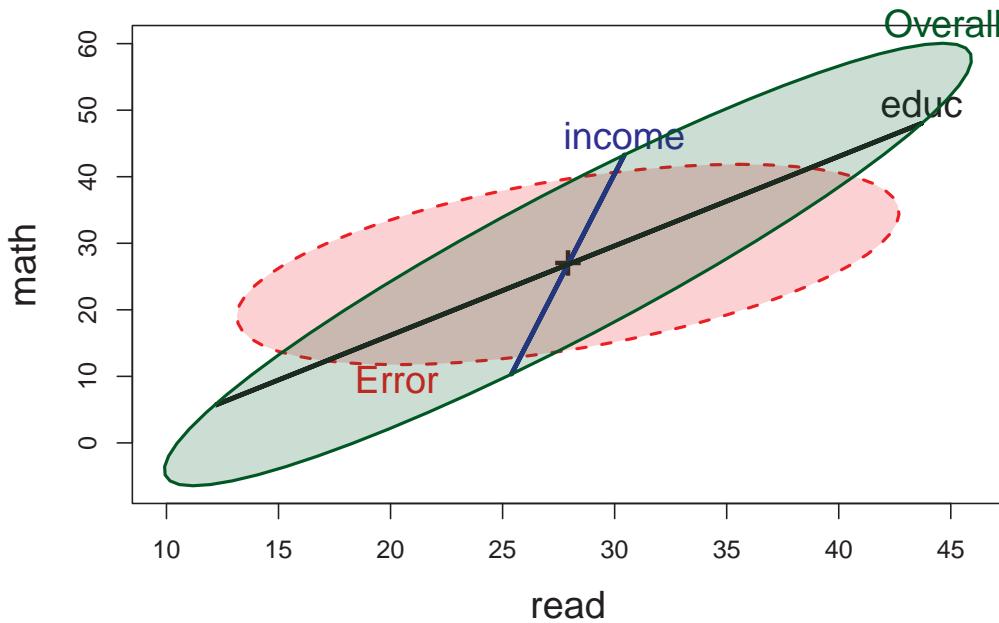


Figure 11.15: HE plot adding the **H** ellipse for the overall test that both predictors have no effect on the outcome scores.

The geometric relations of the **H** ellipses for the overall test and the individual predictors in Figure 11.15 is of worth noting here. Those for the separate coefficients always lie within the overall ellipse. The contribution for income makes the overall ellipse larger in the direction of the `math` score, while the contribution of education makes it larger in both directions.

Example 11.4. School data: HE plots

The `schooldata` dataset analyzed in Section 10.5.2 can also be illuminated by the methods of this chapter. There I fit the multivariate regression model predicting students scores on reading, mathematics and a measure of self-esteem using as predictors measures of parents' education, occupation, school visits, counseling help with school assignments and number of teachers per school.

But I also found two highly influential observations (cases 44, 59; see Figure 10.17) whose effect on the coefficients is rather large; so, I remove them from the analysis here.⁹

```
data(schooldata, package = "heplots")

bad <- c(44, 59)
OK <- (1:nrow(schooldata)) |> setdiff(bad)
school.mod2 <- lm(cbind(reading, mathematics, selfesteem) ~ .,
                   data=schooldata[OK, ])
```

In this model, parent's education and occupation and their visits to the schools were highly predictive of student's outcomes but their counseling efforts and the number of teachers in the schools did not contribute much. However, the nature of these relationships was largely uninterpreted in that analysis.

Here is where HE plots can help. You can think of this as a way to visualize what is entailed in the coefficients for this model by showing the *magnitude* of the predictor effects by their size and their relations to the outcome variable by their *direction*. The table of raw score coefficients isn't very helpful in this regard.

```
coef(school.mod2)
#>           reading mathematics selfesteem
#> (Intercept) 2.7096      3.561     0.39751
#> education    0.2233      0.132    -0.01088
#> occupation   3.3336      4.284     1.79574
#> visit        0.0101     -0.123     0.20005
#> counseling   -0.3953     -0.293     0.00868
#> teacher       -0.1945     -0.360     0.01129
```

Figure 11.16 shows the HE plot for reading and mathematics scores in this model, using the default significance scaling.

```
heplot(school.mod2,
       fill=TRUE, fill.alpha=0.1,
       cex = 1.5,
       cex.lab = 1.5,
       label.pos = c(rep("top", 4), "bottom", "bottom"))
```

Parent's occupation and education are both significant in this view, but what is more important is their orientation. Both are positively associated with reading and math scores, but education is somewhat more related to reading than to mathematics. Number of teachers and degree of parental counseling have a similar orientation, with teachers having a greater relation to mathematics scores. Visits to school and number of teachers are not significant in this plot, but both are positively correlated with reading and math and are coincident in the plot. The parent time counseling measure, while also insignificant, tilts in the opposite direction, having different signs for reading and math.

In the `pairs()` plot for all three responses (Figure 11.17), we see something different in the relations for self-esteem. While occupation has a large positive relation in all the plots in the third row and column, education, counseling and teachers have negative relations in these plots, particularly with mathematics scores.

⁹An alternative to fitting the model removing specific cases deemed troublesome is to use a `robust` method, such as `heplots::roblm()`. This uses re-weighted least squares to down-weight observations with large residuals or other problems. These methods are illustrated in Section 13.5.

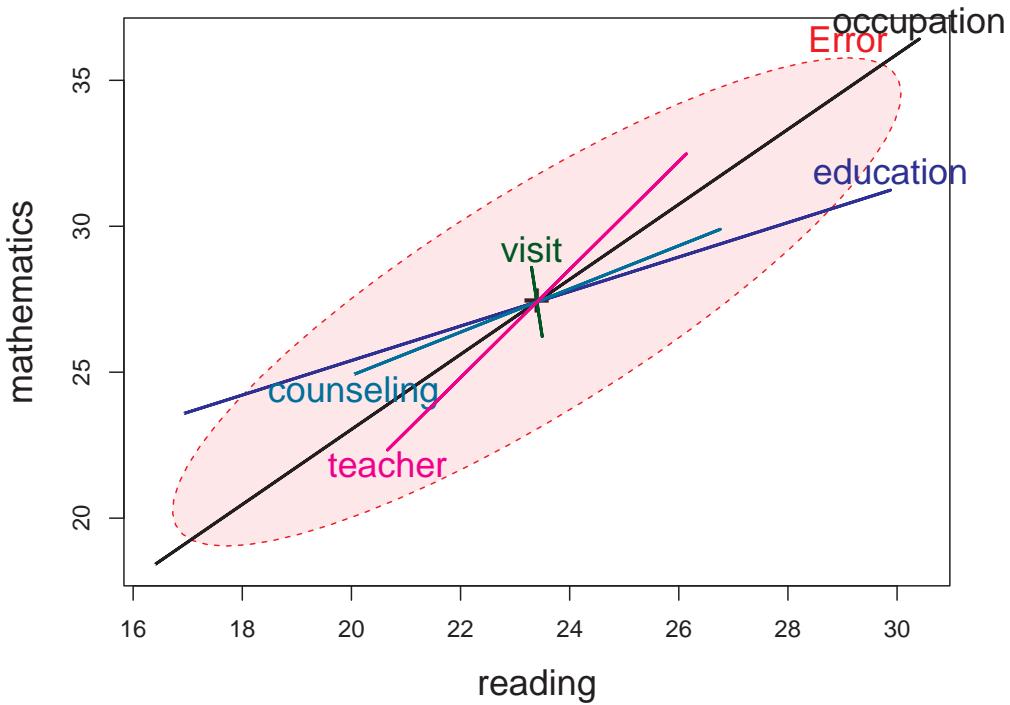


Figure 11.16: HE plot for reading and mathematics scores in the multivariate regression model...

```
pairs(school.mod2,
      fill=TRUE, fill.alpha=0.1,
      var.cex = 2.5,
      cex = 1.3)
```

The analysis of this data is continued below in Example 11.5.

11.10 Canonical correlation analysis

Just as we saw for MANOVA designs, a canonical analysis for multivariate regression involves finding a low-D view of the relations between predictors and outcomes that maximally explains their relations in terms of linear combinations of each. That is, the goal is to find weights for one set of variables, say \mathbf{X} *not* to predict each of the other set $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots]$ *individually*, but rather to also find weights for the \mathbf{y} s which is most highly correlated with the linear combination of the \mathbf{x} s.

In this sense, canonical correlation analysis (CCA) is *symmetric* in the x and y variables: the y set is not considered responses. Rather the goal is simply to explain the correlations between the two sets. For a thorough treatment of this topic, see Gittins (1985).

Geometrically, these linear combinations are vectors representing projections in the observation space of the x and y variables, and CCA can also be thought of as minimizing the angle between these vectors or maximizing the cosine of this angle. This is illustrated in Figure 11.18.

Specifically, we want to find one set of weights \mathbf{a}_1 for the x variables and another for the y variables to give the linear combinations \mathbf{u}_1 and \mathbf{v}_1 ,

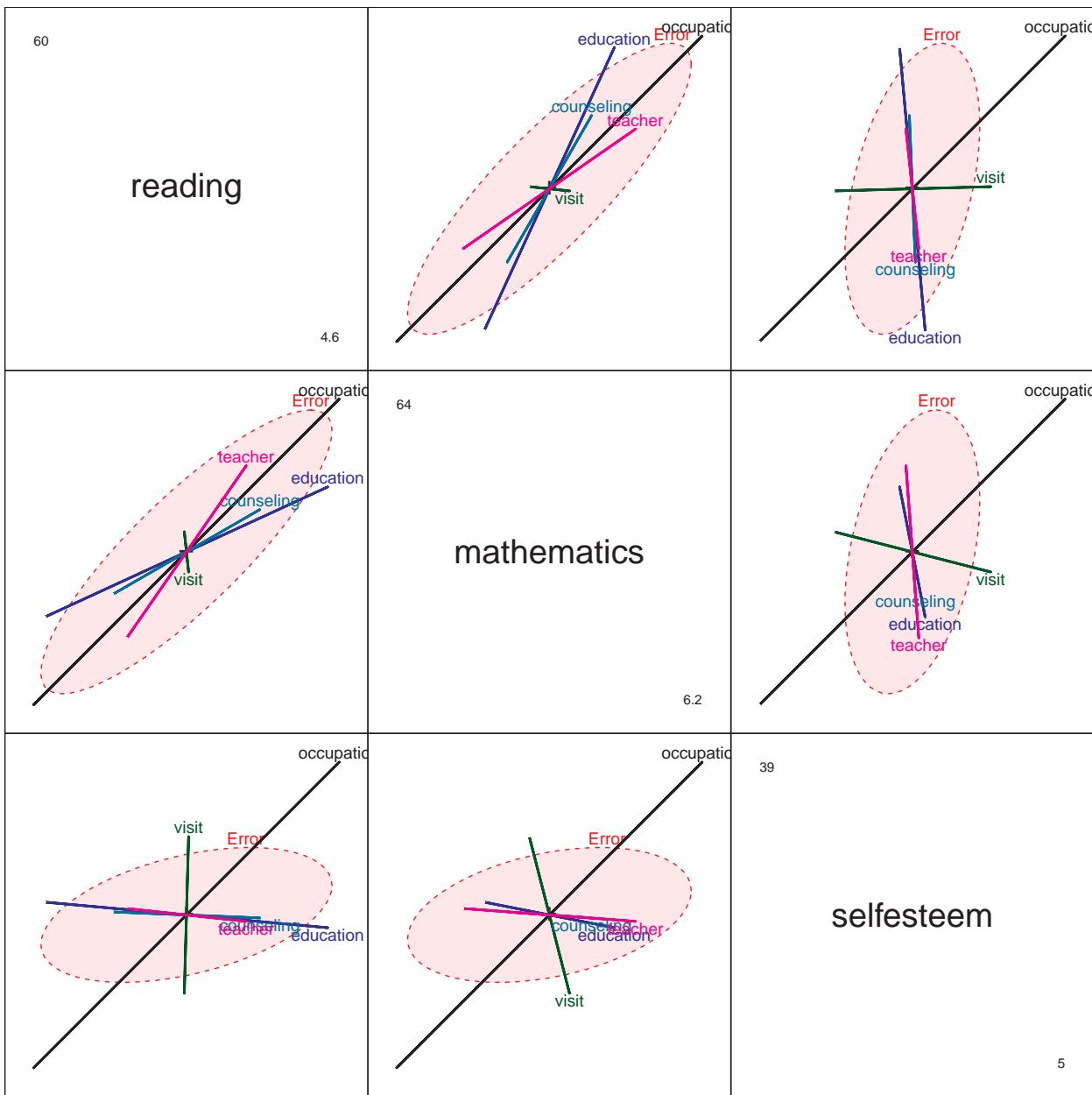


Figure 11.17: Pairwise HE plots for the three outcome variables in the multivariate regression model ...

$$\begin{aligned}\mathbf{u}_1 &= \mathbf{X} \mathbf{a}_1 = a_{11}\mathbf{x}_1 + a_{12}\mathbf{x}_2 + \cdots + a_{1q}\mathbf{x}_q \\ \mathbf{v}_1 &= \mathbf{Y} \mathbf{b}_1 = b_{11}\mathbf{y}_1 + b_{12}\mathbf{y}_2 + \cdots + b_{1p},\end{aligned}$$

such that the correlation $\rho_1 = \text{corr}(\mathbf{u}_1, \mathbf{v}_1)$ is maximized, or equivalently, minimizing the angle between them.

Using \mathbf{S}_{xx} , \mathbf{S}_{yy} to represent the covariance matrices of the x and y variables, and \mathbf{S}_{xy} for the cross-covariances between the two sets, the correlation between the linear combinations of each can be expressed as

$$\begin{aligned}\rho_1 &= \text{corr}(\mathbf{u}_1, \mathbf{v}_1) = \text{corr}(\mathbf{X} \mathbf{a}_1, \mathbf{Y} \mathbf{b}_1) \\ &= \frac{\mathbf{a}_1^\top \mathbf{S}_{xy} \mathbf{b}_1}{\sqrt{\mathbf{a}_1^\top \mathbf{S}_{xx} \mathbf{a}_1} \sqrt{\mathbf{b}_1^\top \mathbf{S}_{yy} \mathbf{b}_1}}\end{aligned}$$

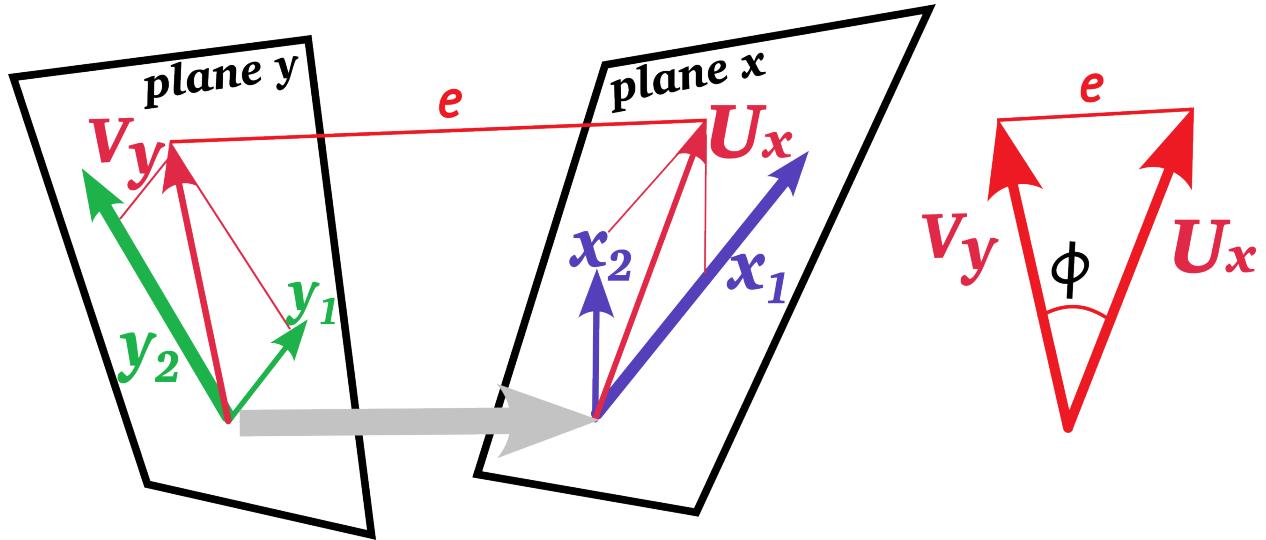


Figure 11.18: Diagram illustrating canonical correlation. For two y variables, all linear combinations are vectors in their plane, and similarly for the x variables. Maximizing the correlation between linear combinations of each is equivalent to making the angle ϕ between them as small as possible, or maximizing $\cos(\theta)$, shown in the diagram at the right. The thick grey arrow indicates that the two planes should be overlaid at a common origin. *Source:* Re-drawn by Udi Alter following a Cross-Validated discussion by user ‘ttnphns’, <https://bit.ly/4dgq2cp>

But, the y variables lie in a p -dimensional (observation) space, and the x in q dimensions, so what they have in common is a space of $s = \min(p, q)$ dimensions. Therefore, we can find additional pairs of canonical variables,

$$\begin{aligned} \mathbf{u}_2 &= \mathbf{X} \mathbf{a}_2 & \mathbf{v}_2 &= \mathbf{Y} \mathbf{b}_2 \\ &\vdots && \\ \mathbf{u}_s &= \mathbf{X} \mathbf{a}_s & \mathbf{v}_s &= \mathbf{Y} \mathbf{b}_s \end{aligned}$$

such that each pair $(\mathbf{u}_i, \mathbf{v}_i)$ has the maximum possible correlation and all distinct pairs are uncorrelated:

$$\begin{aligned} \rho_i &= \max_{\mathbf{a}_i, \mathbf{b}_i} \{ \mathbf{u}_i^\top \mathbf{v}_i \} = \\ \|\mathbf{u}_i\| &= 1, \quad \|\mathbf{v}_i\| = 1, \\ \mathbf{u}_i^\top \mathbf{u}_j &= 0, \quad \mathbf{v}_i^\top \mathbf{v}_j = 0 \quad \forall j \neq i : i, j \in \{1, 2, \dots, s\}. \end{aligned}$$

In words, the correlations among canonical variables are zero except when they are associated with the same canonical correlation or the weights $(\mathbf{a}_i, \mathbf{b}_i)$ for the same pair. Alternatively, all $p \times q$ correlations the variables in \mathbf{Y} and \mathbf{X} are fully summarized in the $s = \min(p, q)$ canonical correlations ρ_i for $i = 1, 2, \dots, s$.

The solution, developed by Hotelling (1936), is a form of a generalized eigenvalue problem, that can be stated in two equivalent ways,

$$\begin{aligned} (\mathbf{S}_{yx} \mathbf{S}_{xx}^{-1} \mathbf{S}_{xy} - \rho^2 \mathbf{S}_{yy}) \mathbf{b} &= \mathbf{0} \\ (\mathbf{S}_{xy} \mathbf{S}_{yy}^{-1} \mathbf{S}_{yx} - \rho^2 \mathbf{S}_{xx}) \mathbf{a} &= \mathbf{0}. \end{aligned}$$

Both equations have the same form and have the same eigenvalues. And, given the eigenvectors for one of these equations, we can find the eigenvectors for the other.

TODO: Fill in details of canonical correlations

Example 11.5. School data: Canonical analysis

With $p = 3$ responses and $q = 5$ predictors there are three possible sets of canonical variables which together account for 100% of the total linear relations between them. `heplots::cancor()` gives the percentage associated with each of the eigenvalues and the canonical correlations.

For this dataset, the first canonical variates, with $\text{Can } R = 0.995$, accounts for 98.6%, so you might think that that is sufficient. Yet the likelihood ratio tests show that the second set, with $\text{Can } R = 0.774$, is also significant, even though it only accounts for 1.3%.

```
# bad <- c(44, 59)
# OK <- (1:nrow(schooldata)) |> setdiff(bad)
school.can2 <- cancor(cbind(reading, mathematics, selfesteem) ~
                       education + occupation + visit + counseling + teacher,
                       data=schooldata[OK, ])
school.can2
#>
#> Canonical correlation analysis of:
#>   5   X  variables: education, occupation, visit, counseling, teacher
#>   with    3   Y  variables: reading, mathematics, selfesteem
#>
#>     CanR  CanRSQ   Eigen percent   cum
#> 1 0.9946 0.9892 91.41999 98.57540 98.58
#> 2 0.7444 0.5541  1.24267  1.33994 99.92
#> 3 0.2698 0.0728  0.07852  0.08466 100.00
#>                               scree
#> 1 ****
#> 2
#> 3
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>   CanR LR test stat approx F numDF denDF Pr(> F)
#> 1 0.995      0.004      67.5     15    166 < 2e-16 ***
#> 2 0.744      0.413      8.5      8    122 4.1e-09 ***
#> 3 0.270      0.927      1.6      3     62    0.19
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The virtue of CCA is that *all* correlations between the X and Y variables are completely captured in the correlations between the *pairs* of canonical scores: The $p \times q$ correlations between the sets are entirely represented by the $s = \min(p, q)$ canonical ones. Whether the second dimension is useful here depends on whether it adds some interpretable increment to what is going on in these relations. One could be justifiably happy with an explanation based on the first dimension that accounts for nearly all the total association between the sets.

The class "cancor" object returned by `cancor()` contains the canonical coefficients, for which there is a `coef()` method as in `candisc()`, and also a `scores()` method to return the scores on the canonical variables, called `Xcan1`, `Xcan2`, ... and `Ycan1`, `Ycan2`.

```
names(school.can2)
#> [1] "cancor"      "names"       "ndim"        "dim"         "coef"
```

```
#> [6] "scores"      "X"           "Y"           "weights"     "structure"
#> [11] "call"        "terms"
```

You can use the `plot()` method or `heplot()` method to visualize and help interpret the results. The `plot()` method plots the canonical `scores$X` against the `scores$Y` for a given dimension (selected by the `which` argument). The `id.n` argument gives a way to flag noteworthy observations.

```
plot(school.can2,
      pch=16, id.n = 3,
      cex.lab = 1.5, id.cex = 1.5,
      ellipse.args = list(fill = TRUE, fill.alpha = 0.1))
text(-2, 1.5, paste("Can R =", round(school.can2$cancor[1], 3)),
      cex = 1.4, pos = 4)

plot(school.can2, which = 2,
      pch=16, id.n = 3,
      cex.lab = 1.5, id.cex = 1.5,
      ellipse.args = list(fill = TRUE, fill.alpha = 0.1))
text(-3, 3, paste("Can R =", round(school.can2$cancor[2], 3)),
      cex = 1.4, pos = 4)
par(op)
```

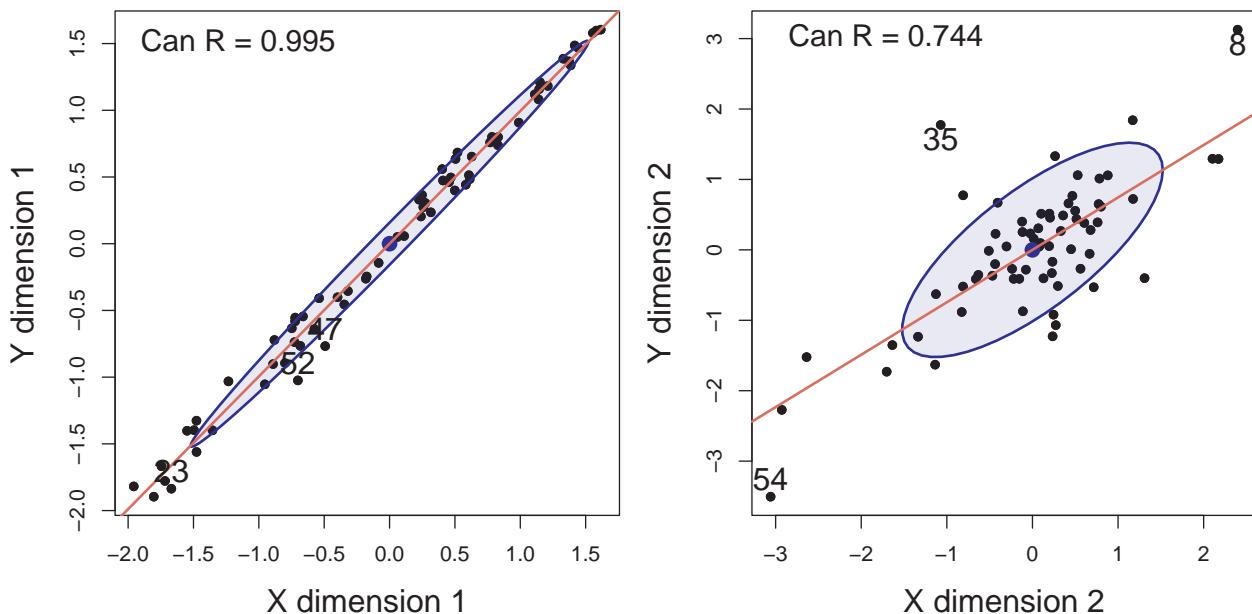


Figure 11.19: Plots of canonical scores for the first two canonical dimensions of the `schooldata` dataset, omitting the two highly influential cases.

It is worthwhile to look at an analogous plot of canonical scores for the original dataset including the two highly influential cases. As you can see in Figure 11.20, cases 44 and 59 are way outside the range of the rest of the data. Their influence increases the canonical correlation to a near perfect $\rho = 0.997$.

```
school.can <- cancor(cbind(reading, mathematics, selfesteem) ~
                      education + occupation + visit + counseling + teacher,
                      data=schooldata)
```

```
plot(school.can,
      pch=16, id.n = 3,
      cex.lab = 1.5, id.cex = 1.5,
      ellipse.args = list(fill = TRUE, fill.alpha = 0.1))
text(-5, 1, paste("Can R =", round(school.can$cancor[1], 3)),
      cex = 1.4, pos = 4)
```

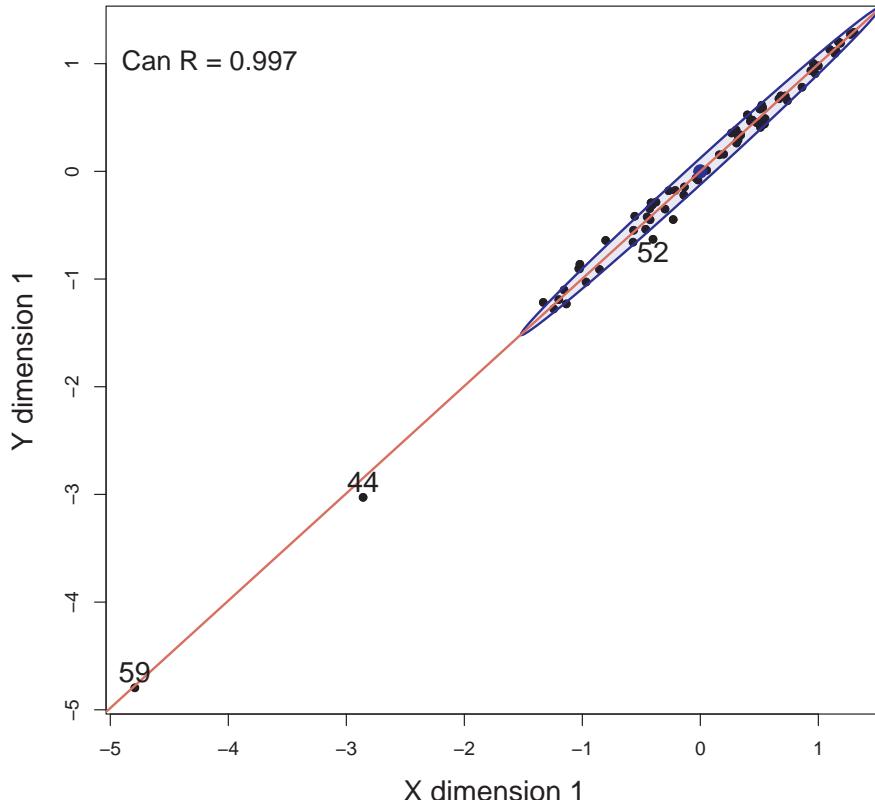


Figure 11.20: Plots of canonical scores on the first canonical dimension for the `schooldata`, including the influential cases, which stand out as so far from the rest of the observations.

Plots of canonical scores tell us of the strength of the canonical dimensions, but do not help interpreting the analysis in relation to the original variables. The HE plot version for canonical correlation analysis re-fits a multivariate regression model for the Y variables against the Xs, but substitutes the canonical scores for each, essentially projecting the data into canonical space.

TODO: Check out signs of structure coeffs from `cancor()`. Would be better to reflect the vectors for `Ycan1`.

```
heplot(school.can2,
       fill = TRUE, fill.alpha = 0.2,
       var.col = "red",
       asp = NA, scale = 0.25,
       cex.lab = 1.5, cex = 1.25,
       prefix="Y canonical dimension ")
```

The `red` variable vectors shown in these plots are intended only to show the correlations of Y variables with the canonical dimensions. The fact that they are so closely aligned reflects the fact that the first dimension

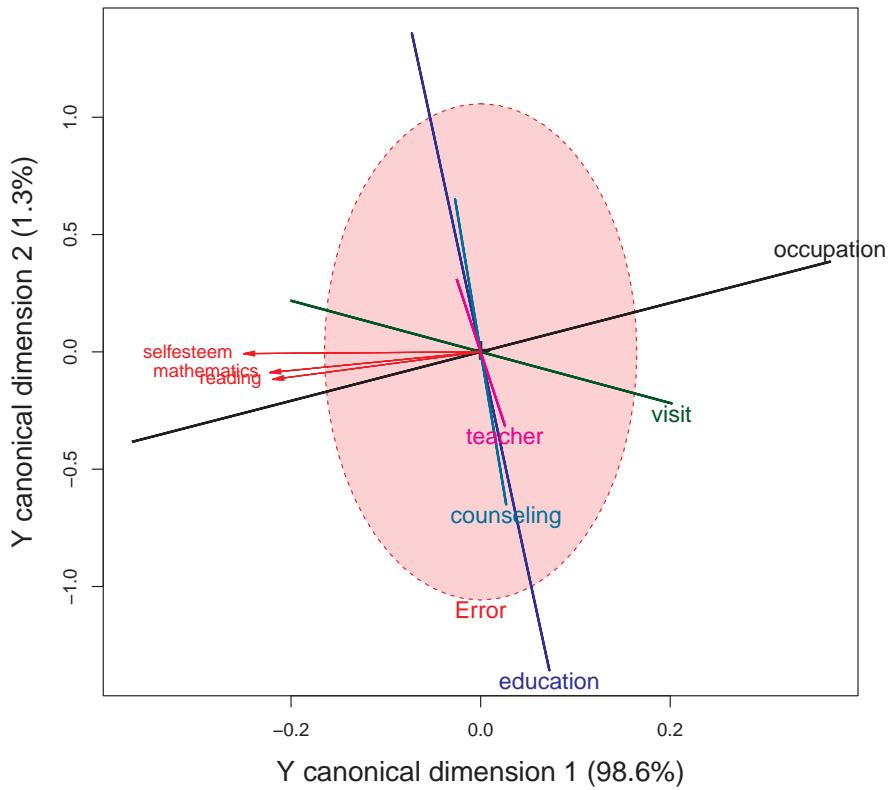


Figure 11.21: HE plot for the canonical correlation analysis of the schooldata. Vectors for the variables indicate their correlations with the canonical dimensions.

accounts for nearly all of their associations with the predictors. The orientation of the **H** ellipses/lines reflects the projection of those from Figure 11.17 into canonical space

Only their *relative lengths* and angles with respect to the Y canonical dimensions have meaning in these plots. Relative lengths correspond to proportions of variance accounted for in the Y canonical dimensions plotted; angles between the variable vectors and the canonical axes correspond to the structure correlations. The absolute lengths of these vectors are arbitrary and are typically manipulated by the `scale` argument to provide better visual resolution and labeling for the variables.

11.11 MANCOVA models

HE plots for designs containing a collection of quantitative predictors and one or more factors are quite simple in MANCOVA models where the effects are additive, i.e., don't involve interactions. They are a bit more challenging when you allow *separate* slopes for groups on all quantitative variables, because there get to be too many terms to usefully display. But these models are more complicated!

If the evidence for heterogeneity of regressions is not very strong, it is still useful to fit the MANCOVA model and display it in an HE plot.

An alternative is to fit separate models for the groups and display these as HE plots. As noted earlier (Section 10.7.1), this is not ideal for testing hypotheses, but provides a useful and informative display of the relations between the predictors and responses and the groups effect. I illustrate these approaches for the Rohwer data, encountered in Section 10.7.1, below.

Example 11.6. Rohwer data

In Section 10.7.1 I fit several models for Rohwer's data on the relations between paired-associate tasks and scholastic performance. The first model was the MANCOVA model testing the difference between the high and low SES groups, controlling for, or taking into account differences on the paired-associate task.

```
Rohwer.mod1 <- lm(cbind(SAT, PPVT, Raven) ~ SES + n + s + ns + na + ss,
                    data=Rohwer)
```

HE plots for this model for the pairs (SAT, PPVT) and (SAT, Raven) is shown in Figure 11.22. The result of an overall test for *all* predictors, $\mathcal{H}_0 : \mathbf{B} = \mathbf{0}$, is added to the basic plot using the `hypotheses` argument.

```
colors <- c("red", "blue", rep("black",5), "#969696")
covariates <- rownames(coef(Rohwer.mod1))[-(1:2)]
pairs(Rohwer.mod1,
      col=colors,
      hypotheses=list("Regr" = covariates),
      fill = TRUE, fill.alpha = 0.1,
      cex=1.5, cex.lab = 1.5, var.cex = 3,
      lwd=c(2, rep(3,5), 4))
```

The positive effect of SES on the outcome measures is seen in all pairwise plots: the high SES group is better on all responses. The positive orientation of the `Regr` ellipses for the covariates shows that the predicted values for all three responses are positively correlated (more so for SAT and PPVT): higher performance on the paired associate tasks, in general, is associated with higher academic performance. The two significant predictors, `na` and `ns` are the only ones that extend outside the error ellipses, but their orientations differ.

Homogeneity of regression

A second model tested the assumption of homogeneity of regression by adding interactions of SES with the PA tasks, allowing separate slopes for the two groups on each of the other predictors.

```
Rohwer.mod2 <- lm(cbind(SAT, PPVT, Raven) ~ SES * (n + s + ns + na + ss),
                    data=Rohwer)
```

This model has 11 terms, excluding the intercept: `SES`, plus 5 main effects (`xs`) for the predictors and 5 interactions (slope differences), too many for an understandable display. To visualize this in an HE plot (Figure 11.23), I simplify, by showing the interaction terms *collectively* by a single ellipse, representing their joint effect, and specified as a linear hypothesis called `slopes` that picks out the interaction effects.

The argument `terms` limits the `H` ellipses for the right-hand-side of the model which are shown to just those terms specified. The combined effect of the interaction terms is specified as an hypothesis (`slopes`) testing the interaction terms (which have a ":" in their name). Because `SES` is "treatment-coded" in this model, the interaction terms reflect the difference in slopes for the high SES group compared to the low.

```
(coefs <- rownames(coef(Rohwer.mod2)))
#> [1] "(Intercept)" "SESLo"        "n"          "s"
#> [5] "ns"          "na"          "ss"         "SESLo:n"
#> [9] "SESLo:s"     "SESLo:ns"    "SESLo:na"   "SESLo:ss"

colors <- c("red", "blue", rep("black",5), "#969696")
heplot(Rohwer.mod2, col=c(colors, "darkgreen"),
       terms=c("SES", "n", "s", "ns", "na", "ss"),
```

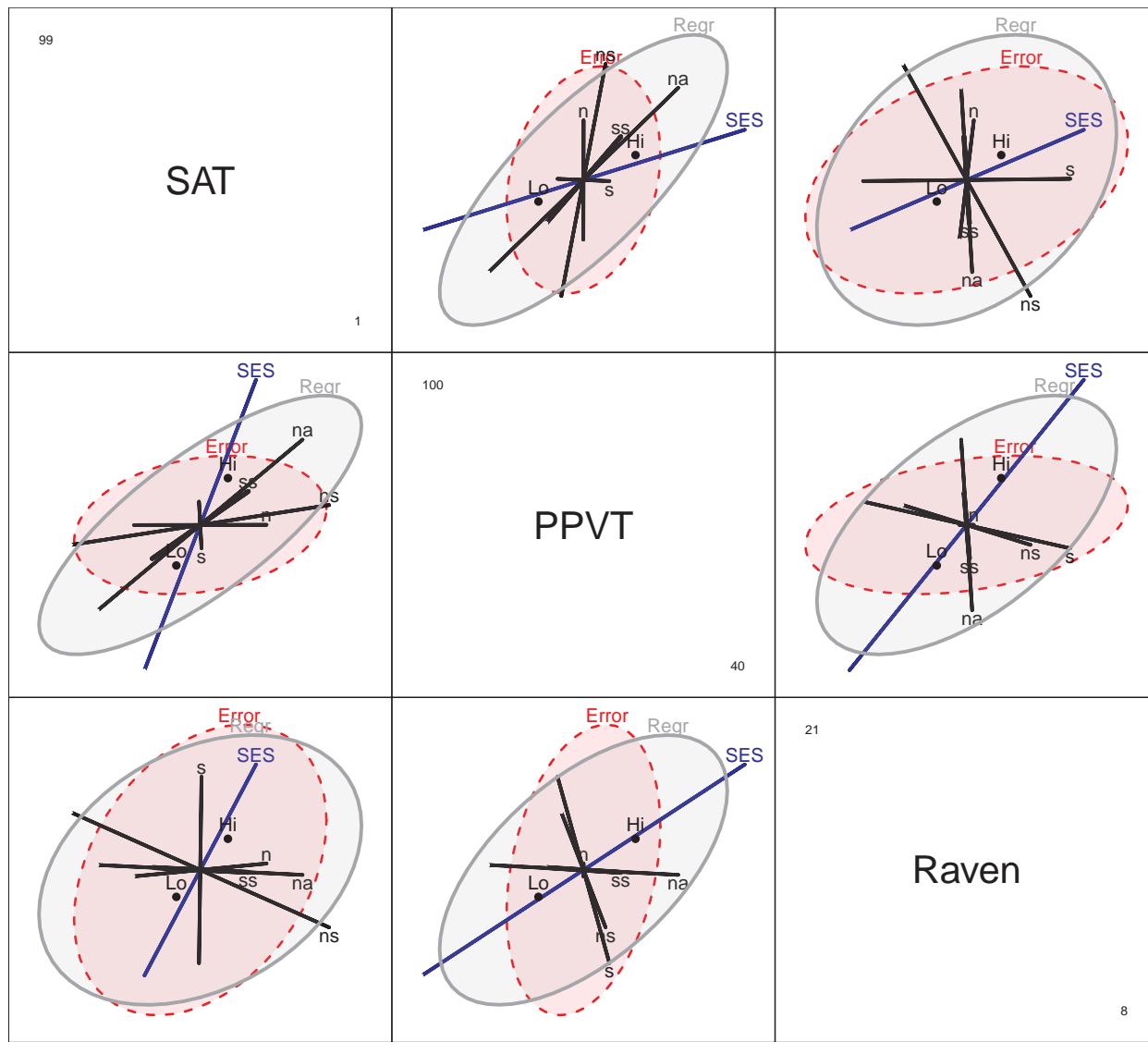


Figure 11.22: All-pairs HE plot for SAT, PPVT and Raven using the MANCOVA model. The ellipses labeled ‘Regr’ show the test of the overall effect of the quantitative predictors.

```

hypotheses=list("Regr" = c("n", "s", "ns", "na", "ss"),
                 "Slopes" = coefs[grep(":", coefs)]),
fill = TRUE, fill.alpha = 0.2, cex.lab = 1.5)

```

Separate models

When there is heterogeneity of regressions, using submodels for each of the groups has the advantage that you can easily visualize the slopes for the predictors in each of the groups, particularly if you overlay the individual HE plots. In this example, I'm using the models `Rohwer.sesLo` and `Rohwer.sesHi` fit to each of the groups.

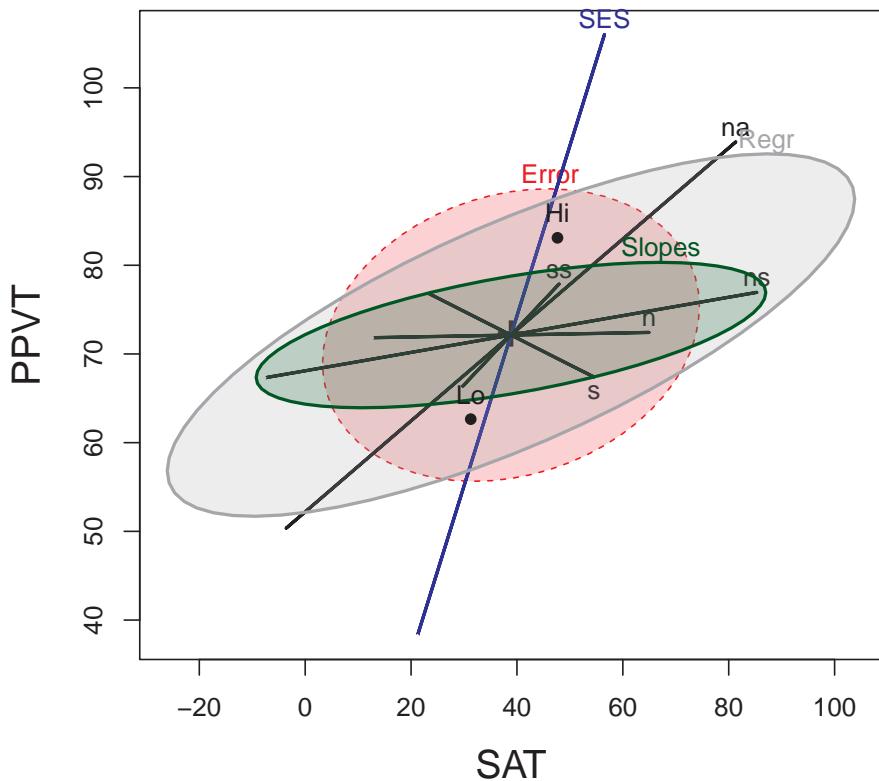


Figure 11.23: HE plot for SAT and PPVT using the heterogeneous regression model. The ellipse labeled ‘Regr’ shows the test of the covariates combined, and the ellipse labeled ‘slopes’ shows the combined difference in slopes between the two groups.

```
Rohwer.sesLo <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Lo")
Rohwer.sesHi <- lm(cbind(SAT, PPVT, Raven) ~ n + s + ns + na + ss,
                     data=Rohwer, subset = SES=="Hi")
```

Here I make use of the fact that several HE plots can be overlaid using the option `add=TRUE` as shown in Figure 11.24. The axis limits may need adjustment in the first plot so that the second one will fit.

```
heplot(Rohwer.sesLo,
       xlim = c(0,100),                      # adjust axis limits
       ylim = c(40,110),
       col=c("red", "black"),
       fill = TRUE, fill.alpha = 0.1,
       lwd=2, cex=1.2, cex.lab = 1.5)
heplot(Rohwer.sesHi,
       add=TRUE,
       col=c("brown", "black"),
       grand.mean=TRUE,
       error.ellipse=TRUE,                   # not shown by default when add=TRUE
       fill = TRUE, fill.alpha = 0.1,
       lwd=2, cex=1.2)
```

We can readily see the difference in means for the two SES groups (Hi has greater scores on both variables)

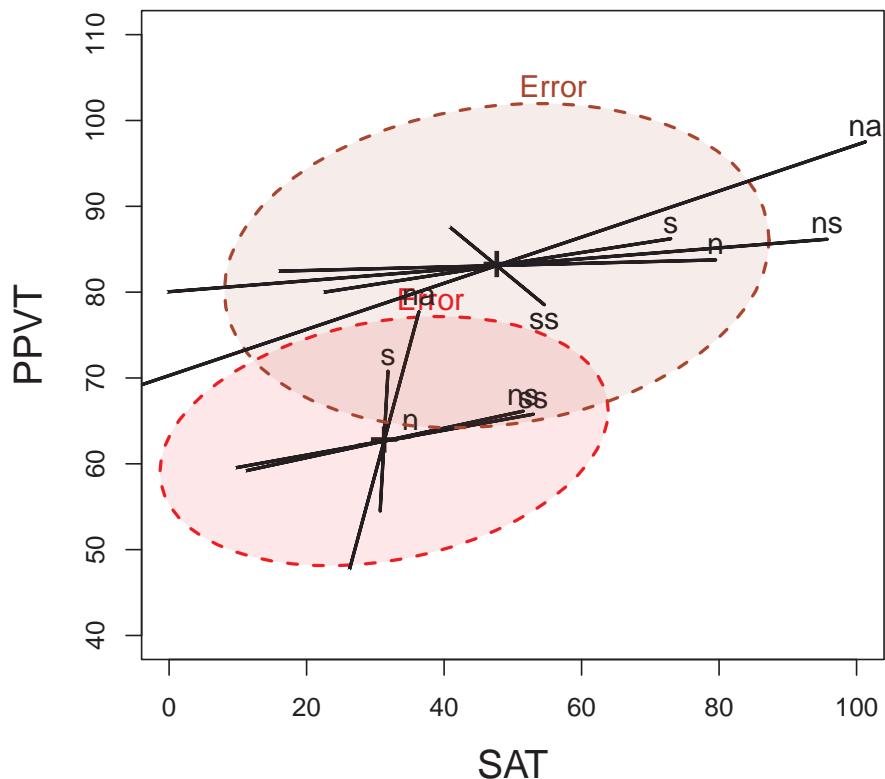


Figure 11.24: Overlaid HE plots for SAT and PPVT, for the low and high SES groups, when each group is fit separately.

and it also appears that the slopes of the **s** and **n** predictor ellipses are shallower for the High than the Low group, indicating greater relation with the SAT score. As well, the error ellipses show that on these measures, error variation is somewhat smaller in the low SES group.

11.12 What have we learned?

- **HE plots clarify complex multivariate models into enlightening visualizations** - The HE plot framework brilliantly addresses the interpretability problem of multivariate models by visualizing hypothesis (H) ellipses against error (E) ellipses. Rather than navigate a confusing maze of tables of coefficients and test statistics, with HE plots you can see which effects matter, how they relate to each other, and whether they're statistically significant. All of these benefits are given in a single, intuitive plot that reveals the geometric structure underlying your multivariate analysis.
- **Canonical space is your secret weapon for high-dimensional visualization** - When you have many response variables, canonical discriminant analysis and canonical correlation analysis project the complex multivariate relationships into a lower-dimensional space that captures the essential patterns. This isn't just dimension reduction—it's insight amplification, revealing the fundamental directions of variation that matter most while showing how your original variables contribute to these meaningful dimensions.
- **Visual hypothesis testing beats p-value hunting every time** - HE plots make hypothesis testing immediate and intuitive: if a hypothesis ellipse extends outside the error ellipse (under significance scaling), the effect is significant. No more scanning tables of p-values or wrestling with multiple comparisons—the

geometry tells the story directly. You can even decompose overall effects into meaningful contrasts and see their individual contributions as separate ellipses.

- **Ellipse orientations reveal the hidden correlational structure of your effects** - The angles between hypothesis ellipses in HE plots directly show how different predictors relate to your response variables and to each other. When effect ellipses point in similar directions, those predictors have similar multivariate signatures; when they're orthogonal, they capture independent aspects of variation. This geometric insight goes far beyond what correlation matrices can reveal.
- **Multivariate models become tractable through progressive visual summaries** - The chapter demonstrates a powerful visualization strategy: start with scatterplot matrices to see the raw data structure, move to HE plots to understand model effects, then project to canonical space for the clearest possible view of multivariate relationships. Each step preserves the essential information while making it more interpretable, turning the complexity of multivariate analysis into a comprehensible visual narrative.

12

Visualizing Equality of Covariance Matrices

To make the preliminary test on variances is rather like putting to sea in a rowing boat to find out whether conditions are sufficiently calm for an ocean liner to leave port. — G. E. P. Box (1953)

This chapter concerns the extension of tests of homogeneity of variance from the classical univariate ANOVA setting to the analogous multivariate (MANOVA) setting. Such tests are a routine but important aspect of data analysis, as particular violations can drastically impact model estimates and appropriate conclusions that can be drawn (Lix & Keselman, 1996).

Beyond issues of model assumptions, the question of equality of covariance matrices is often of general interest itself. For instance, variability is often an important issue in studies of strict equivalence in laboratories comparing across multiple patient measurements and in other applied contexts (see Gastwirth et al., 2009 for other exemplars).

Moreover the outcome of such tests often have important consequences for the details of a main method of analysis. Just as the Welsh t -test (Welch, 1947) is now commonly used and reported for a two-group test of differences in means under *unequal* variances, a preliminary test of equality of covariance matrices is often used in discriminant analysis to decide whether linear (LDA) or quadratic discriminant analysis (QDA) should be applied in a given problem. In such cases, the data at hand should inform the choice of statistical analysis to utilize.

I provide some answers to the following questions:

- **Visualization:** How can we visualize differences among group variances and covariance matrices, perhaps in a way that is analogous to what is done to visualize differences among group means? As will be illustrated (Section 12.6), differences among covariance matrices can be comprised of spread in overall size (“scatter”) and shape (“orientation”). These can be seen in data space with data ellipses, particularly if the data is centered by shifting all groups to the grand mean.
- **Low-D views:** When there are more than a few response variables, what low-dimensional views can show the most interesting properties related to the equality of covariance matrices? Projecting the data into the space of the principal components serves well again here (Section 12.7). Surprisingly, we will see that the *small dimensions* contain useful information about differences among the group covariance matrices, similar to what we find for outlier detection.
- **Other statistics:** Box’s M -test is most widely used. Are there other worthwhile test statistics? You will see that graphical methods suggest alternatives in Section 12.8. Similarly, the univariate Levene’s test has a simple multivariate generalization which can be visualized using HE plots (Section 12.9).

The following sections provide a capsule summary of the issues in this topic. Most of the discussion is couched in terms of a one-way design for simplicity, but the same ideas can apply to two-way (and higher) designs, where a “group” factor is defined as the product combination (interaction) of two or more factor variables.

When there are also numeric covariates, this topic can also be extended to the multivariate analysis of covariance (MANCOVA) setting. This is accomplished simply by applying these techniques to the *residuals* from predictions by the covariates alone.

Packages

In this chapter I use the following packages. Load them now

```
library(car)
library(helplots)
library(candisc)
library(ggplot2)
library(dplyr)
library(tidyr)
```

The main methods described here are implemented in the `helplots` package.

12.1 Homogeneity of Variance in Univariate ANOVA

In classical (Gaussian) univariate ANOVA models, the main interest is typically on tests of mean differences in a response y according to one or more factors. The validity of the typical F test, however, relies on the assumption of *homogeneity of variance*: all groups have the same (or similar) variance,

$$\sigma_1^2 = \sigma_2^2 = \dots = \sigma_g^2 .$$

It turns out that the F test for differences in means is relatively robust to violation of this assumption (Harwell et al., 1992), as long as the group sample sizes are roughly equal.¹ This applies to Type I error α rates, which are not much affected. However, unequal variance makes the ANOVA tests less efficient: you lose power to detect significant differences.

A variety of classical test statistics for homogeneity of variance are available, including Hartley's F_{max} (Hartley, 1950), Cochran's C (Cochran, 1941), and Bartlett's test (Bartlett, 1937), but these have been found to have terrible statistical properties (Rogan & Keselman, 1977), which prompted Box's famous quote.

Levene (1960) introduced a different form of test, based on the simple idea that when variances are equal across groups, the average *absolute values* of differences between the observations and group means will also be equal, i.e., substituting an L_1 norm for the L_2 norm of variance. In a one-way design, this is equivalent to a test of group differences in the means of the auxilliary variable $z_{ij} = |y_{ij} - \bar{y}_i|$.

More robust versions of this test were proposed by M. B. Brown & Forsythe (1974). These tests substitute the group mean by either the group **median** or a **trimmed mean** in the ANOVA of the absolute deviations. Some suggest these should be almost always preferred to Levene's version using the mean deviation. See Conover et al. (1981) for an early review and Gastwirth et al. (2009) for a general discussion of these tests. In what follows, we refer to this class of tests as "Levene-type" tests and suggest a multivariate extension described below (Section 12.2).

These deviations from a group central can be calculated using `helplots::colDevs()` and the central value can be a function, like `mean`, `median` or an anonymous one like `function(x) mean(x, trim = 0.1)`) that trims 10% off each side of the distribution. With a response `Y` Levene-type tests then be performed "by hand" as follows:

```
# Levine
Z.mean <- abs( colDevs(Y, group) )
lm(Z.mean ~ group)

# Brown-Forsythe
```

¹If group sizes are greatly unequal **and** homogeneity of variance is violated, then the F statistic is too liberal (actual rejection rate greater than the nominal α) when large sample variances are associated with small group sizes. Conversely, the F statistic is too conservative (actual $< \alpha$) if large variances are associated with large group sizes.

```
Z.med <- abs( colDevs(Y, group, median) )
lm(Z.med ~ group)
```

The function `car::leveneTest()` does this, so we could examine whether the variances are equal in the Penguin variables, one at a time, like so:

```
data(peng, package = "heplots")
leveneTest(bill_length ~ species, data=peng)
#> Levene's Test for Homogeneity of Variance (center = median)
#>          Df F value Pr(>F)
#> group     2   2.29   0.1
#>           330

leveneTest(bill_depth ~ species, data=peng)
#> Levene's Test for Homogeneity of Variance (center = median)
#>          Df F value Pr(>F)
#> group     2   1.91   0.15
#>           330

# ...

leveneTest(body_mass ~ species, data=peng)
#> Levene's Test for Homogeneity of Variance (center = median)
#>          Df F value Pr(>F)
#> group     2   5.13 0.0064 **
#>           330
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

More conveniently, `heplots:leveneTests()` with an “s”, does this for each of a set of response variables, specified in a data frame, a model formula or a "`mlm`" object. It also formats the results in a more pleasing way:

```
peng.mod <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~ species,
                 data = peng)
leveneTests(peng.mod)
#> Levene's Tests for Homogeneity of Variance (center = median)
#>
#>          df1 df2 F value Pr(>F)
#> bill_length     2 330   2.29 0.1033
#> bill_depth      2 330   1.91 0.1494
#> flipper_length  2 330   0.44 0.6426
#> body_mass       2 330   5.13 0.0064 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So, this tells us that the groups do not differ in variances on first three variables, but they do for `body_mass`.

12.2 Visualizing Levene's test

To gain some insight into the problem of homogeneity of variance it is helpful to see how the situation looks in terms of data. For the Penguin data, it might be simplest just to look at boxplots of the variables and try to see whether the **widths** of the central 50% boxes seem to be the same, as in Figure 12.1. However, it is perceptually difficult to focus on differences with widths of the boxes within each panel when their centers also differ from group to group.

```
source("R/penguin/penguin-colors.R")
col <- peng.colors("dark")
clr <- c(col, gray(.20))
peng_long <- peng |>
  pivot_longer(bill_length:body_mass,
               names_to = "variable",
               values_to = "value")

peng_long |>
  group_by(species) |>
  ggplot(aes(value, species, fill = species)) +
  geom_boxplot() +
  facet_wrap(~ variable, scales = 'free_x') +
  theme_penguins() +
  theme_bw(base_size = 14) +
  theme(legend.position = 'none')
```

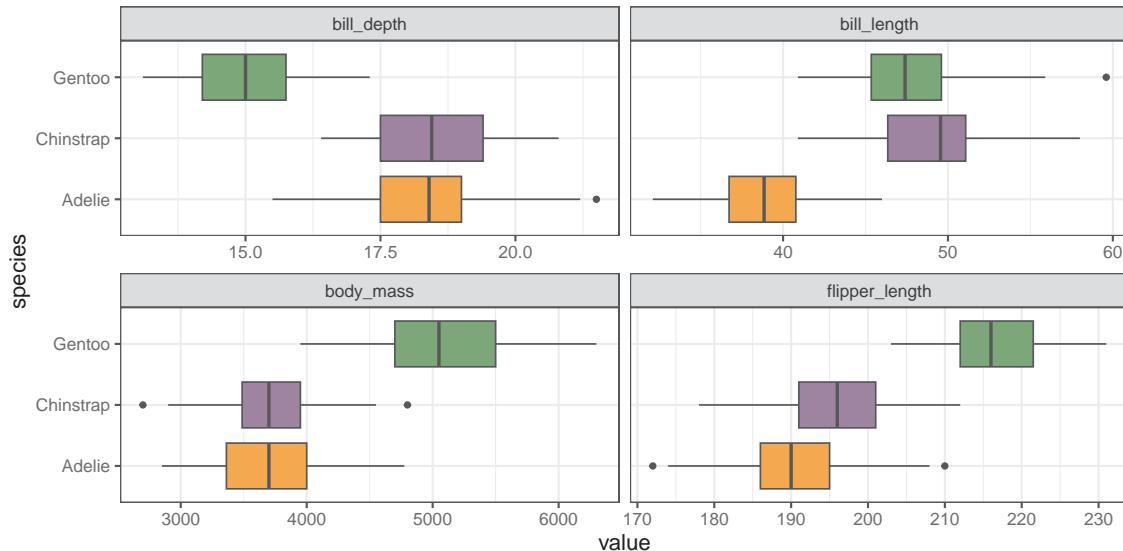


Figure 12.1: Boxplots for the Penguin variables. For assessing homogeneity of variance, we should be looking for differences in **width** of the central 50% boxes in each panel, rather than difference in central tendency.

Instead, you can see more *directly* what is tested by the Levene test by graphing the *absolute deviations* from the group means or medians. This is another example of the graphic idea that you can make visual comparisons easier by plotting quantities of direct interest. You can calculate the median deviation values as follows:

```
vars <- c("bill_length", "bill_depth", "flipper_length", "body_mass")
pengDevs <- colDevs(peng[, vars], peng$species, median) |>
  abs()
```

From a boxplot of the absolute deviations in Figure 12.2 your eye can now focus on the central value, shown by the **median** ‘|’ line, because Levene’s method is testing whether these differ across groups.

```
# calculate absolute differences from median
dev_long <- data.frame(species = peng$species, pengDevs) |>
  pivot_longer(bill_length:body_mass,
               names_to = "variable",
               values_to = "value")

dev_long |>
  group_by(species) |>
  ggplot(aes(value, species, fill = species)) +
  geom_boxplot() +
  facet_wrap(~ variable, scales = 'free_x') +
  xlab("absolute median deviation") +
  theme_penguins() +
  theme_bw(base_size = 14) +
  theme(legend.position = 'none')
```

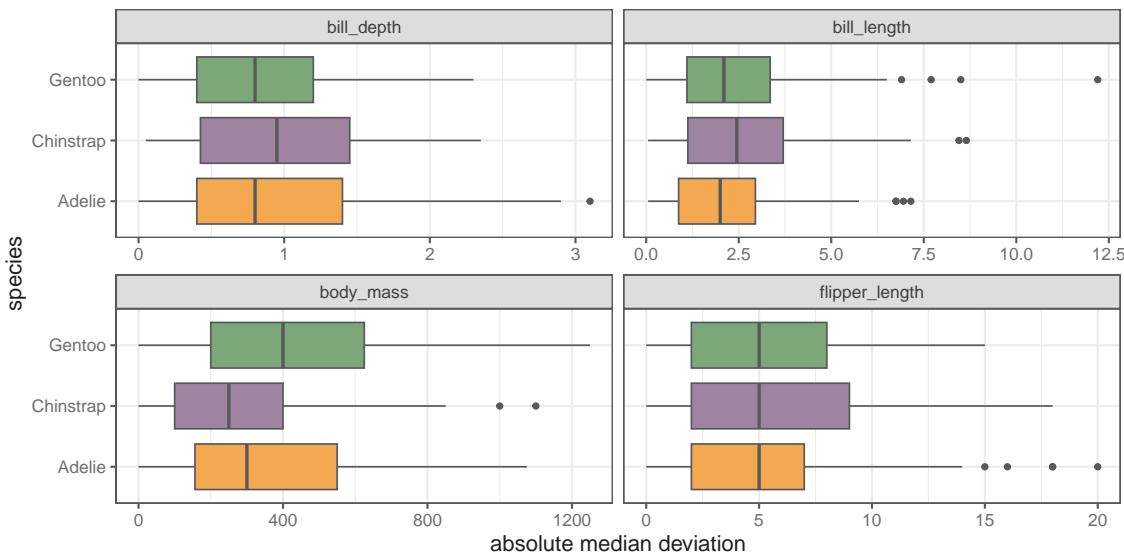


Figure 12.2: Boxplots for absolute differences from group medians for the Penguin data. The visual test of equality of variance is whether the median lines in the boxplots align.

It is now easy to see that the medians largely align for all the variables except for `body_mass`.

12.3 Homogeneity of variance in MANOVA

In the MANOVA context, the main emphasis, of course, is on differences among mean vectors, testing

$$\mathcal{H}_0 : \mu_1 = \mu_2 = \cdots = \mu_g .$$

However, the standard test statistics (Wilks' Lambda, Hotelling-Lawley trace, Pillai-Bartlett trace, Roy's maximum root) rely upon the analogous assumption that the within-group covariance **matrices** Σ_i are equal for all groups,

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_g .$$

This is much stronger than in the univariate case, because it also requires that all the correlations between pairs of variables are the same for all groups. For example, for two responses, there are three parameters $(\rho, \sigma_1^2, \sigma_2^2)$ assumed equal across all groups; for p responses, there are $p(p+1)/2$ assumed equal. The variances relate to *size* differences among data ellipses while the differences in the covariances appear as differences in *shape*.

Penguin data: Covariance ellipses

To preview a main example, Figure 12.3 shows data ellipses for the main size variables in the Penguins data (`peng`). `heplots::covEllipses()` is specialized for viewing the relations among the data ellipsoids representing the sample covariance matrices, $\mathbf{S}_1 = \mathbf{S}_2 = \cdots = \mathbf{S}_g$. It draws the data ellipse for each group, and also for the pooled within-group \mathbf{S}_p , as shown in Figure 12.3 for bill length and bill depth.

You can see that the sizes and shapes of the data ellipses are sort of similar in the left panel. The visual comparison becomes more precise when the data ellipses are all shifted to a common origin at the grand means (using `center = TRUE`). From this you can see that the Adelie group differs most from the others.

```
op <- par(mar = c(4, 4, 1, 1) + .5,
           mfrow = c(c(1,2)))
covEllipses(cbind(bill_length, bill_depth) ~ species, data=peng,
            fill = TRUE,
            fill.alpha = 0.1,
            lwd = 3,
            col = clr)

covEllipses(cbind(bill_length, bill_depth) ~ species, data=peng,
            center = TRUE,
            fill = c(rep(FALSE,3), TRUE),
            fill.alpha = .1,
            lwd = 3,
            col = clr,
            label.pos = c(1:3,0))
par(op)
```

All such pairwise plots in scatterplot matrix format are produced using the `variables` argument to `covEllipses()`, giving Figure 12.4.

```
clr <- c(peng.colors(), "black")
covEllipses(peng[,3:6], peng$species,
            variables=1:4,
            col = clr,
            fill=TRUE,
            fill.alpha=.1)
```

The covariance ellipses in Figure 12.4 look pretty similar in size, shape and orientation. But what does Box's \mathcal{M} test (described below) say? As you can see, it concludes strongly against the null hypothesis, because the test is highly sensitive to small differences among the covariance matrices.

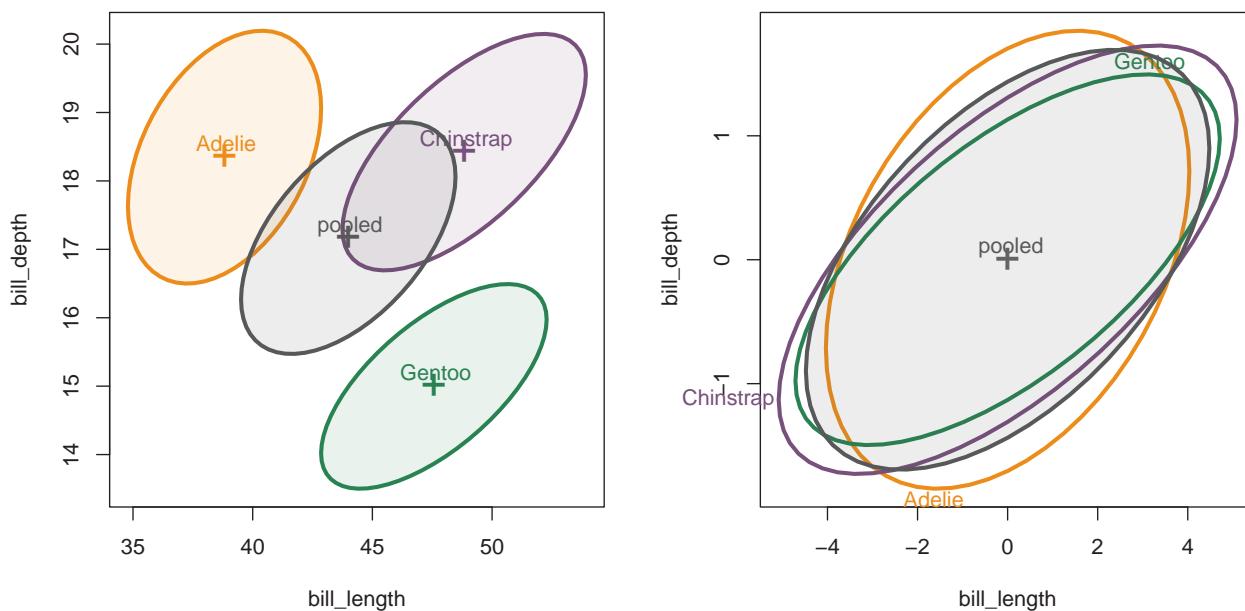


Figure 12.3: Data ellipses for bill length and bill depth in the penguins data, also showing the pooled covariance. Left: As is; right: these are centered at the grand means for easier comparison.

```

peng.boxm <- boxM(cbind(bill_length, bill_depth, flipper_length, body_mass)
  ~ species,
  data=peng) |>
  print()
#>
#> Box's M-test for Homogeneity of Covariance Matrices
#>
#> data: peng
#> Chi-Sq (approx.) = 75, df = 20, p-value = 3e-08

```

Iris data: Covariance ellipses

It will be useful to have another example as we proceed, so Figure 12.5 shows an analogous plot for the iris data we examined in Section 11.6.

Even when these are shown uncentered, the differences in size, shape and orientation are much more apparent. Iris *setosa* stands out as having smaller variance on some of the variables, while the ellipses for *virginica* tend to be larger. Their orientation (slopes) also differ quite a bit.

```

iris.colors <- c("red", "darkgreen", "blue")
covEllipses(iris[,1:4], iris$Species,
  variables=1:4,
  fill = TRUE,
  fill.alpha=.1,
  col = c(iris.colors, "black"),
  label.pos=c(1:3,0))

#>
#> Box's M-test for Homogeneity of Covariance Matrices
#>

```

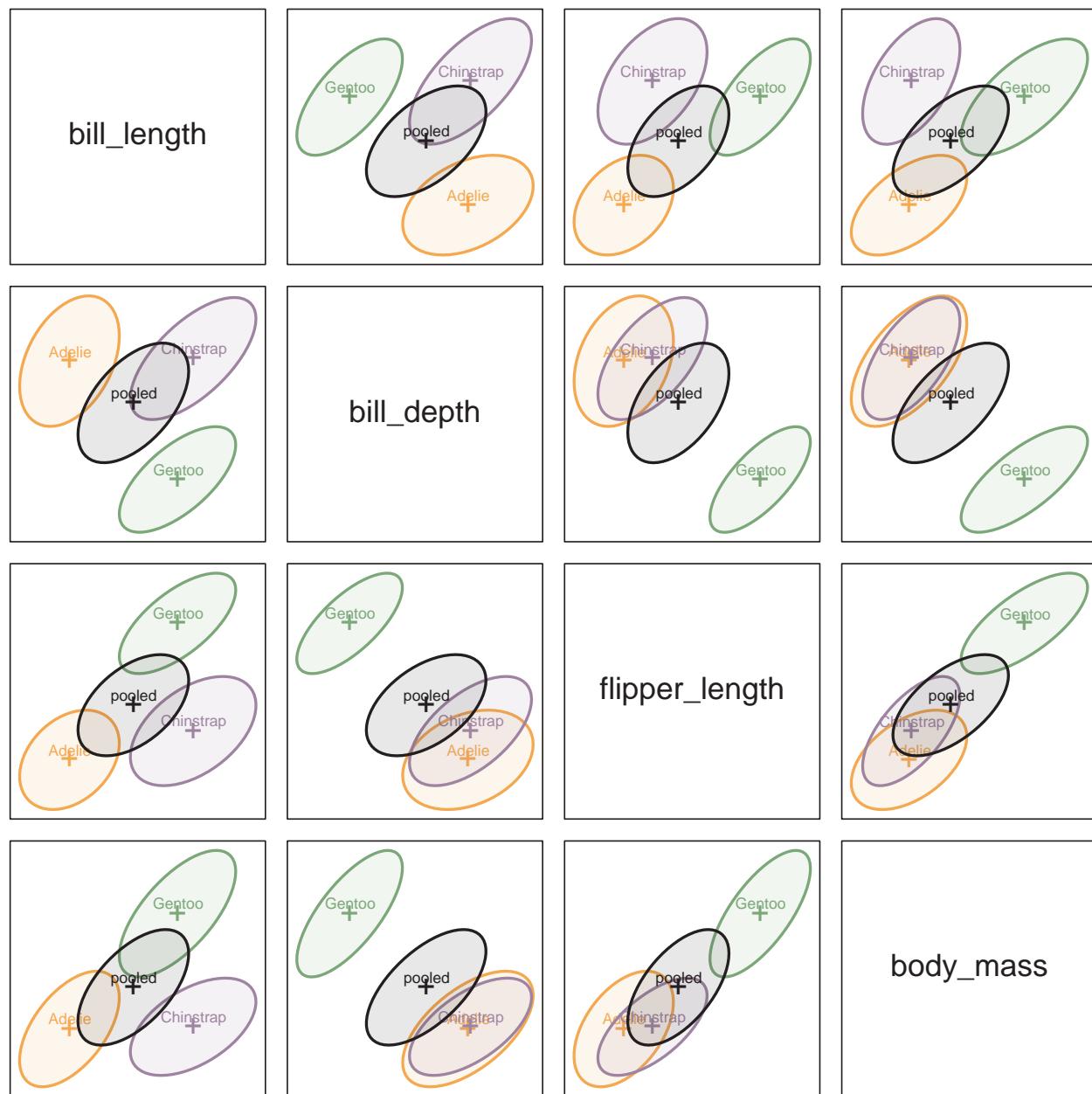


Figure 12.4: All pairwise covariance ellipses for the penguins data. The covariance matrices are homogeneous when the ellipses for the groups all have the same size and shape as that for the mean-centered pooled data (shown in black).

```
#> data: iris
#> Chi-Sq (approx.) = 141, df = 20, p-value <2e-16
```

Unsurprisingly, Box's \mathcal{M} tests gives $\chi^2_{20} = 140.94$ with a p -value $< 2.2\text{e-}16$, putting numbers to the visual conclusion from Figure 12.5.

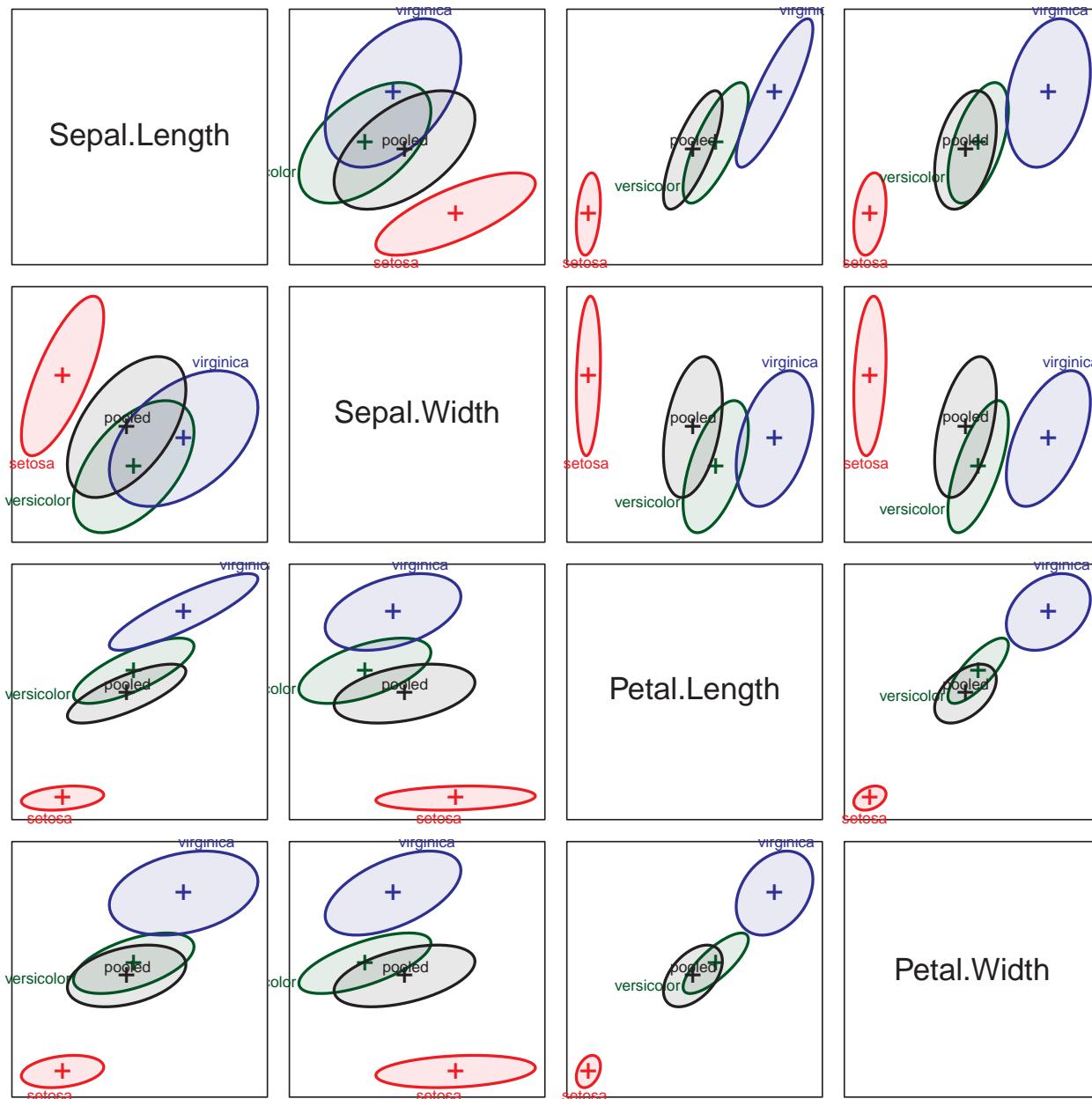


Figure 12.5: All pairwise covariance ellipses for the iris data. The large differences in size and shape indicate substantial heterogeneity in variances and covariances.

12.4 Box's \mathcal{M} test

Take a moment and think, “How could we generalize a test of equality of variances, $s_1^2 = s_2^2 = \dots = s_g^2$, to the multivariate case, where we have $(p \times p)$ matrices, $\mathbf{S}_1 = \mathbf{S}_2 = \dots = \mathbf{S}_g$ for each of g groups?”. Multivariate thinking suggests that that we calculate some measure of “size” of each \mathbf{S}_i , in a similar way to what is done in multivariate tests comparing \mathbf{H} and \mathbf{E} matrices.

Box (1949) proposed the following likelihood-ratio test (LRT) statistic \mathcal{M} for testing the hypothesis of equal covariance matrices, using the log of the determinant $|\mathbf{S}_i|$ as the measure of size.

$$\mathcal{M} = (N - g) \ln |\mathbf{S}_p| - \sum_{i=1}^g (n_i - 1) \ln |\mathbf{S}_i| , \quad (12.1)$$

where $N = \sum n_i$ is the total sample size and

$$\mathbf{S}_p = (N - g)^{-1} \sum_{i=1}^g (n_i - 1) \mathbf{S}_i$$

is the pooled covariance matrix.

\mathcal{M} can thus be thought of as a ratio of the determinant of the pooled \mathbf{S}_p to the geometric mean of the determinants of the separate \mathbf{S}_i .

In practice, there are various transformations of the value of M to yield a test statistic with an approximately known distribution (Timm, 1975). Roughly speaking, when each $n_i > 20$, a χ^2 approximation is often used; otherwise an F approximation is known to be more accurate.

Asymptotically, $-2 \ln(\mathcal{M})$ has a χ^2 distribution. The χ^2 approximation due to Box (1949, 1950) is that

$$X^2 = -2(1 - c_1) \ln(\mathcal{M}) \sim \chi_{df}^2$$

with $df = (g - 1)p(p + 1)/2$ degrees of freedom, and a bias correction constant:

$$c_1 = \left(\sum_i \frac{1}{n_i - 1} - \frac{1}{N - g} \right) \frac{2p^2 + 3p - 1}{6(p + 1)(g - 1)} .$$

In this form, Bartlett's test for equality of variances in the univariate case is the special case of Box's \mathcal{M} when there is only one response variable, so Bartlett's test is sometimes used as univariate follow-up to determine which response variables show heterogeneity of variance.

Yet, like its univariate counterparts, Box's test is well-known to be highly sensitive to violation of (multivariate) normality and the presence of outliers², as Box (1953) suggested in the opening chapter quote. Yet, it provides a nice framework for thinking about this problem more generally. . .

12.5 Visualizing heterogeneity

A larger goal of this chapter is to use this background as another illustration of multivariate thinking, here, for visualizing and testing the heterogeneity of covariance matrices in multivariate designs.

While researchers often rely on a single number to determine if their data have met a particular threshold, such compression will often obscure interesting information, particularly when a test concludes that differences exist, and one is left to wonder “why?”. It is within this context where, again, visualizations often reign supreme.

We have already seen one useful method in Section 12.3, which uses direct visualization of the information

²For example, Tiku & Balakrishnan (1984) concluded from simulation studies that the normal-theory LRT provides poor control of Type I error under even modest departures from normality. O'Brien (1992) proposed some robust alternatives, and showed that Box's normal theory approximation suffered both in controlling the null size of the test and in power. Zhang & Boos (1992) also carried out simulation studies with similar conclusions and used bootstrap methods to obtain corrected critical values.

in the \mathbf{S}_i and \mathbf{S}_p using *data ellipsoids* to show size and shape as minimal schematic summaries; In what follows, I propose three additional visualization-based approaches to questions of heterogeneity of covariance in MANOVA designs:

- (a) a simple dotplot of the components of Box's \mathcal{M} test: the log determinants of the \mathbf{S}_i together with that of the pooled \mathbf{S}_p . Extensions of these simple plots raise the question of whether measures of heterogeneity other than that captured in Box's test might also be useful; and,
- (b) PCA low-rank views to highlight features more easily seen there than in the full data space.
- (c) the connection between Levene-type tests and an ANOVA (of centered absolute differences) suggests a parallel with a multivariate extension of Levene-type tests and a MANOVA. We explore this with a version of Hypothesis-Error (HE) plots we have found useful for visualizing mean differences in MANOVA designs.

These methods are described and illustrated in Friendly & Sigal (2018).

12.6 Visualizing Box's \mathcal{M}

Box's test is based on a comparison of the log determinants of the \mathbf{S}_i relative to that of the pooled \mathbf{S}_p , so the simplest thing to do is just plot them!

`boxM()` produces a "boxm" object, for which there are `summary()` (details) and `plot()` methods. The `plot()` method gives a dot plot of the log determinants $\ln |\mathbf{S}_i|$ together with that for the pooled covariance $\ln |\mathbf{S}_p|$. Cai et al. (2015) provide the theory for the (asymptotic) confidence intervals shown.

```
plot(peng.boxm, gplabel="species", cex.lab = 1.5)
plot(iris.boxm, gplabel="Species", cex.lab = 1.5)
```

In these plots (Figure 12.6), the value for the pooled covariance appears within the range of the groups, because it is a weighted average. If you take a moment to look back at Figure 12.4, you'll see that the data ellipses for Gentoo are slightly smaller in most pairwise views, but it is much easier to see this in a plot that summarizes this, like Figure 12.6.

From Figure 12.5, it is clear that *setosa* shows the smallest within-group variability. The numeric scale values on the horizontal axis give a sense that the range across groups is considerably greater for the iris data than for the Penguins. Some other generalizations of Box's test using other measures are illustrated in Section 12.8.

12.7 Low-rank views

With $p > 3$ response variables, a simple alternative to the pairwise 2D plots in data space (shown in Figure 12.4 and Figure 12.5) is the projection into the principal component space accounting for the greatest amounts of total variance in the data (Friendly & Sigal (2018)). For the Iris data, a simple PCA of the covariance matrix shows that nearly 96% of total variance in the data is accounted for in the first two dimensions.

```
iris.pca <- prcomp(iris[,1:4], scale. = TRUE)
summary(iris.pca)
```

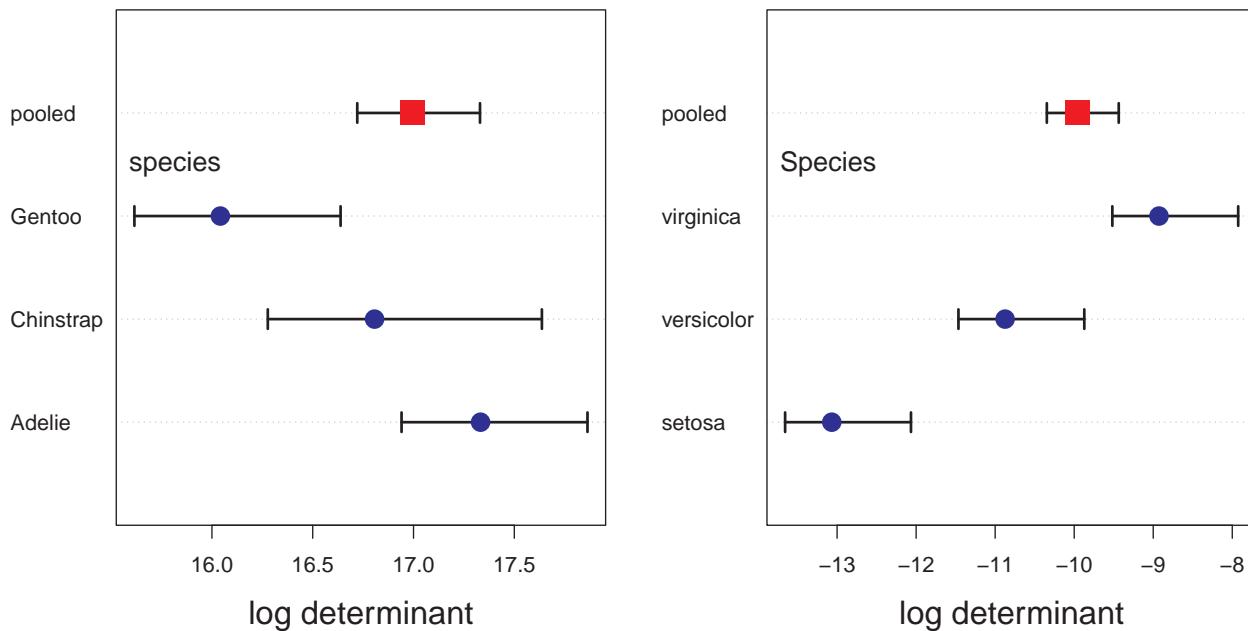


Figure 12.6: Plots of the contributions to Box's \mathcal{M} statistic for the Penguin and iris data.

```
#> Importance of components:
#>          PC1    PC2    PC3    PC4
#> Standard deviation   1.71 0.956 0.3831 0.14393
#> Proportion of Variance 0.73 0.229 0.0367 0.00518
#> Cumulative Proportion  0.73 0.958 0.9948 1.00000
```

Figure 12.7 shows the plots of the covariance ellipsoids for the first two principal component scores, uncentered (left panel) and centered (right panel). The dominant PC1 (73% of total variance) essentially orders the species by a measure of overall size of their sepals and petals. In the centered view, it can again be seen how *Setosa* differs in covariance from the other two species, and that while *Virginica* and *Versicolor* both have similar shapes to the pooled covariance matrix, *Versicolor* has somewhat greater variance on PC1.

```
source(here::here("R/util/text.usr.R"))

covEllipses(iris.pca$x, iris$Species,
  col = c(iris.colors, "black"),
  fill=TRUE, fill.alpha=.1,
  cex.lab = 1.5,
  label.pos = c(1, 3, 3, 0), asp=1)
text.usr(0.05, 0.95, "(a) Uncentered", pos = 4)

covEllipses(iris.pca$x, iris$Species,
  center=TRUE,
  col = c(iris.colors, "black"),
  fill=TRUE, fill.alpha=.1,
  cex.lab = 1.5,
  label.pos = c(1, 3, 3, 0), asp=1)
text.usr(0.95, 0.95, "(b) Centered", pos = 2)
```

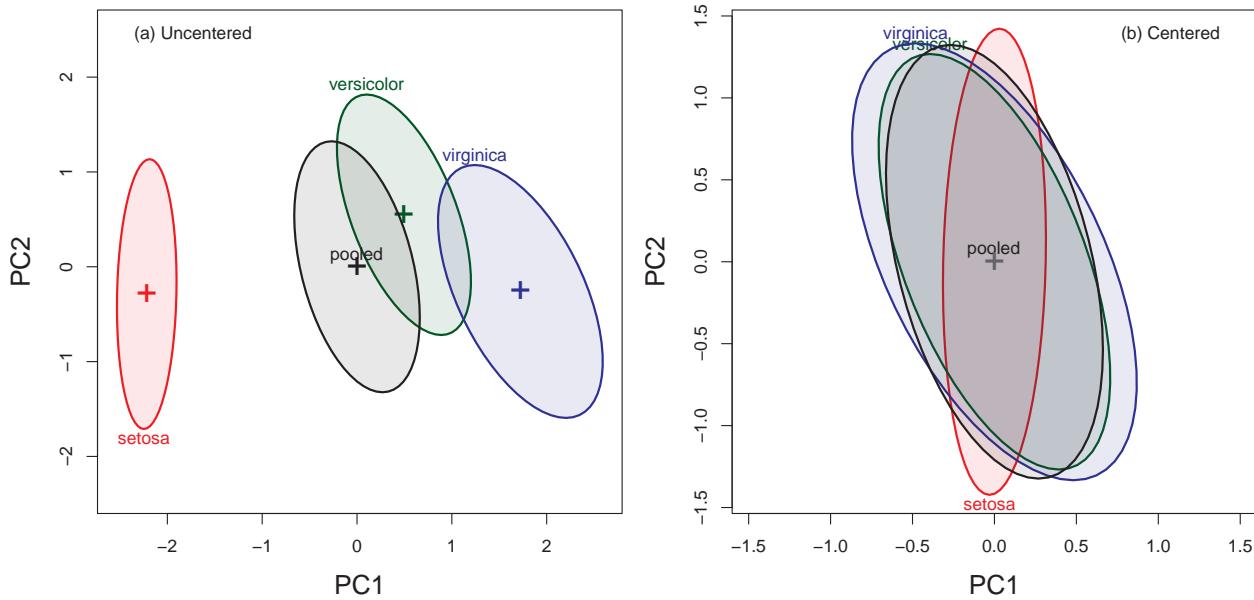


Figure 12.7: Covariance ellipsoids for the first two principal components of the iris data. Left (a): uncentered, showing group means on the principal components; right (b): centered at the origin.

12.7.1 Small dimensions can matter

For the Iris data, the first two principal components account for 96% of total variance, so we might think we are done here. Yet, as we've seen in other problems (outliers, collinearity), important information also exists in the space of the *smallest* principal component dimensions.

This is also true, as we will see for Box's \mathcal{M} test, because it is a (linear) function of all the eigenvalues of the between and within group covariance matrices, is therefore also subject to the influence of the smaller dimensions, where differences among \mathbf{S}_i and of \mathbf{S}_p can lurk.

```
covEllipses(iris.pca$x, iris$Species,
            variables = 3:4,
            center=TRUE,
            col = c(iris.colors, "black"),
            fill=TRUE, fill.alpha=.1,
            cex.lab = 1.5,
            label.pos = c(1, 3, 3, 4), asp=1)
```

Figure 12.8 shows the covariance ellipsoids in (PC3, PC4) space. Even though these dimensions contribute little to total variance, there are more pronounced differences in the within-group shapes (correlations) relative to the pooled covariance, and these contribute to a rejection of homogeneity by Box's \mathcal{M} test. Here we see that the correlation for *Virginica* is of opposite sign from the other two groups.

12.8 Other measures of heterogeneity

As we saw above Section 12.3, the question of equality of covariance matrices can be expressed in terms of the similarity in size and shape of the data ellipses for the individual group \mathbf{S}_i relative to that of \mathbf{S}_p . Box's \mathcal{M} test uses just one possible function to describe this size: the logs of their determinants.

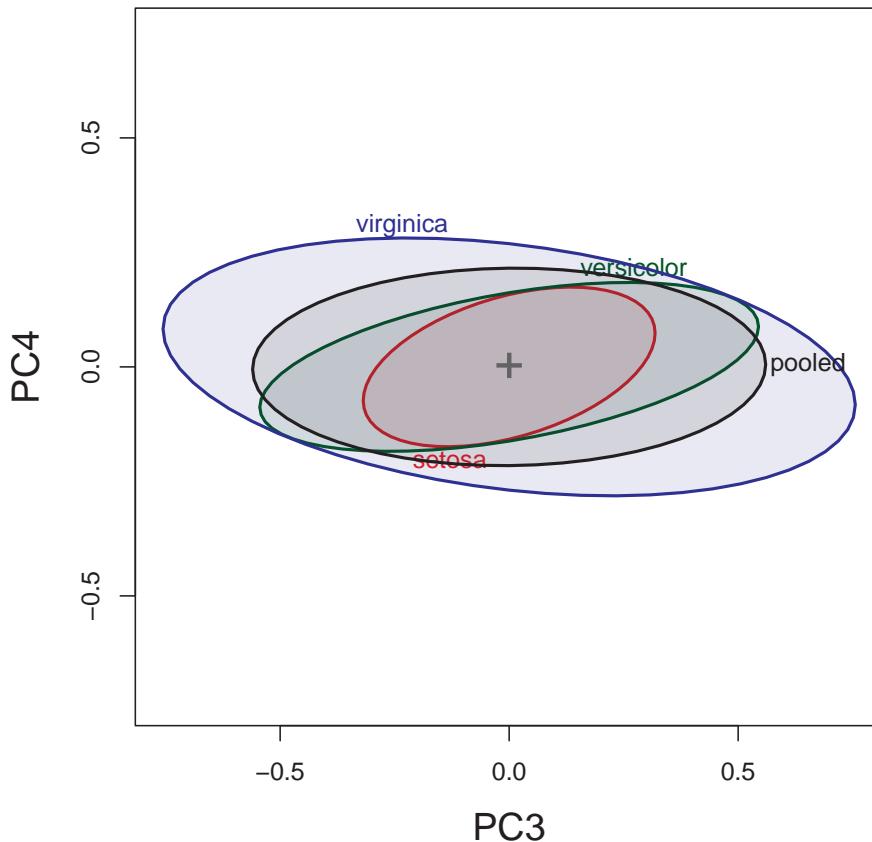


Figure 12.8: Covariance ellipses for the smallest principal components of the iris data.

When Σ is the covariance matrix of a multivariate vector \mathbf{y} with eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \lambda_p$, the properties shown in Table 12.1 represent methods of describing the size and shape of the ellipsoid in \mathbb{R}^p .³ Just as is the case for tests of the MLM itself where different functions of these give test statistics (Wilks' Λ , Pillai trace, etc.), one could construct other test statistics for homogeneity of covariance matrices.

Table 12.1: Statistical and geometrical properties of “size” of an ellipsoid

Size	Conceptual formula	Geometry	Function
(a) Generalized variance:	$\det \Sigma = \prod_i \lambda_i$	area, (hyper)volume	geometric mean
(b) Average variance:	$\text{tr}(\Sigma) = \sum_i \lambda_i$	linear sum	arithmetic mean
(c) Average precision:	$1/\text{tr}(\Sigma^{-1}) = 1/\sum_i (1/\lambda_i)$		harmonic mean
(d) Maximal variance:	λ_1	maximum dimension	supremum

Hence, for a sample covariance matrix \mathbf{S} , $|\mathbf{S}|$ is a measure of generalized variance and $\ln |\mathbf{S}|$ is a measure of average variance across the p dimensions.

The "boxM" plot methods in r"heplots" can compute and plot all of the functions of the eigenvalues in Table 12.1. The results are shown in Figure 12.9.

```
plot(peng.boxm, which="product", gplabel="species")
plot(peng.boxm, which="sum", gplabel="species")
```

³More general theory and statistical applications of the geometry of ellipsoids is given by Friendly et al. (2013).

```
plot(peng.boxm, which="precision", gplabel="species")
plot(peng.boxm, which="max", gplabel="species")
```

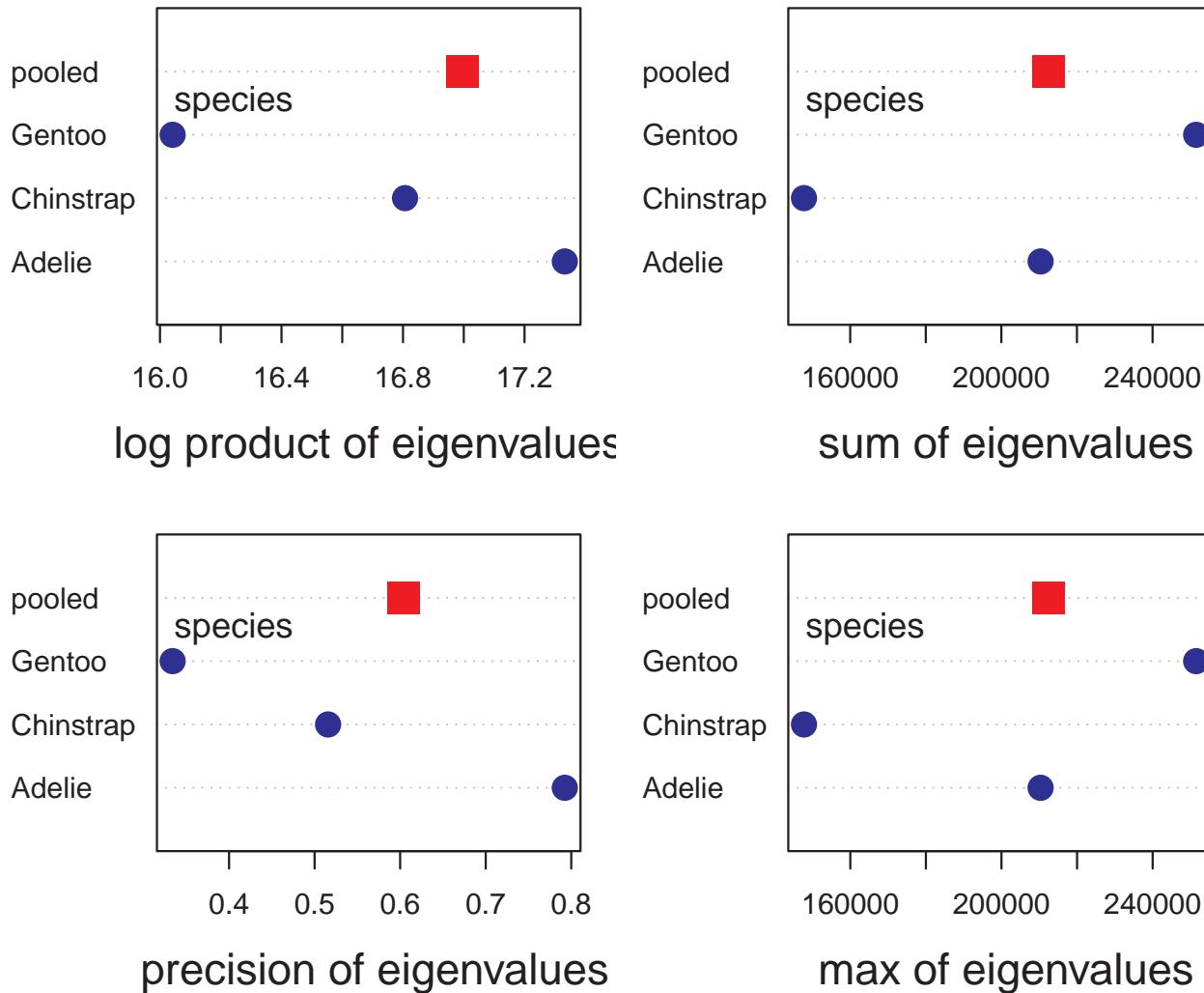


Figure 12.9: Plot of eigenvalue statistics of the covariance matrices for the Penguin data

Except for the absence of error bars, the plot for log product in the upper left panel of Figure 12.9 is the same as that in Figure 12.6, but with the sign reversed. In principle, it is possible to add such confidence intervals for all these measures through the use of bootstrapping, but this has not yet been implemented in the `heplots`

For this data set, the pattern of points in the plot for Box's \mathcal{M} is also more or less the same as that for the precision measure. The plots for the sum of and maximum eigenvalue are also similar to each other, but differ from those of the two measures in the left column of Figure 12.9. The main point is that these are not *all the same*, so different functions reflect different patterns of the eigenvalues, and could be used to define new statistical tests, perhaps with greater power or sensitivity to outliers.

12.9 Multivariate analog of Levine's test

The fact that Levine's test is just a simple ANOVA of the absolute deviations from the group means or medians suggests an easy multivariate generalization (which has not been well-studied) —Simply do a MANOVA of the absolute differences, $|\mathbf{Y} - \bar{\mathbf{Y}}_j|$, centering on the means or medians.⁴ For the `iris` data, this gives:

```
irisdev <- colDevs(iris[,1:4], iris$Species, median,
                     group.var = "Species") |>
  mutate(across(where(is.numeric), abs))

irisdev.mod <- lm(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~
                     Species, data=irisdev)
Anova(irisdev.mod)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>          Df test stat approx F num Df den Df Pr(>F)
#> Species   2     0.394      8.89     8    290 6.7e-11 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `irisdev.mod` model is just another MANOVA, so we can visualize it in an HE plot (Figure 12.10). But remember that this is showing differences among groups in multivariate dispersion rather than in means.

```
pairs(irisdev.mod, variables=1:4, fill=TRUE, fill.alpha=.1)
```

In this plot, the **H** ellipses for the groups show the variation in the overall *dispersion*—the size of absolute differences from the group medians. These are very highly correlated across groups in most of the subplots, particularly so for those involving sepal width.

12.9.1 Canonical discriminant analysis

But we can go further. Just as canonical discriminant analysis provides a low-dimensional view of differences in means, the same method can be used to visualize heterogeneity of dispersion expressed by the the Levene-type MANOVA in 1D or 2D.

For the model `irisdev.mod`, this shows that 98% of groups differences shown in the HE plots (Figure 12.10) can be found in a single canonical dimension. The 4D information can be reduced to 1D.

```
library(candisc)
irisdev.can <- candisc(irisdev.mod) |>
  print()
#>
#> Canonical Discriminant Analysis for Species:
#>
#>   CanRsq Eigenvalue Difference Percent Cumulative
#>   1  0.381      0.6154      0.602     97.9      97.9
```

⁴M. J. Anderson (2006) describes other procedures based on Euclidean distances $D_{ij} = [\sum(y_{ij} - \bar{y}_j)^2]^{1/2}$ of the observations from their group centroids (means or spatial medians). Rather than fit a parametric MANOVA model, he suggests a permutation test method (PERMDIST) which randomly permutes the group assignments and yields non-parametric tests.

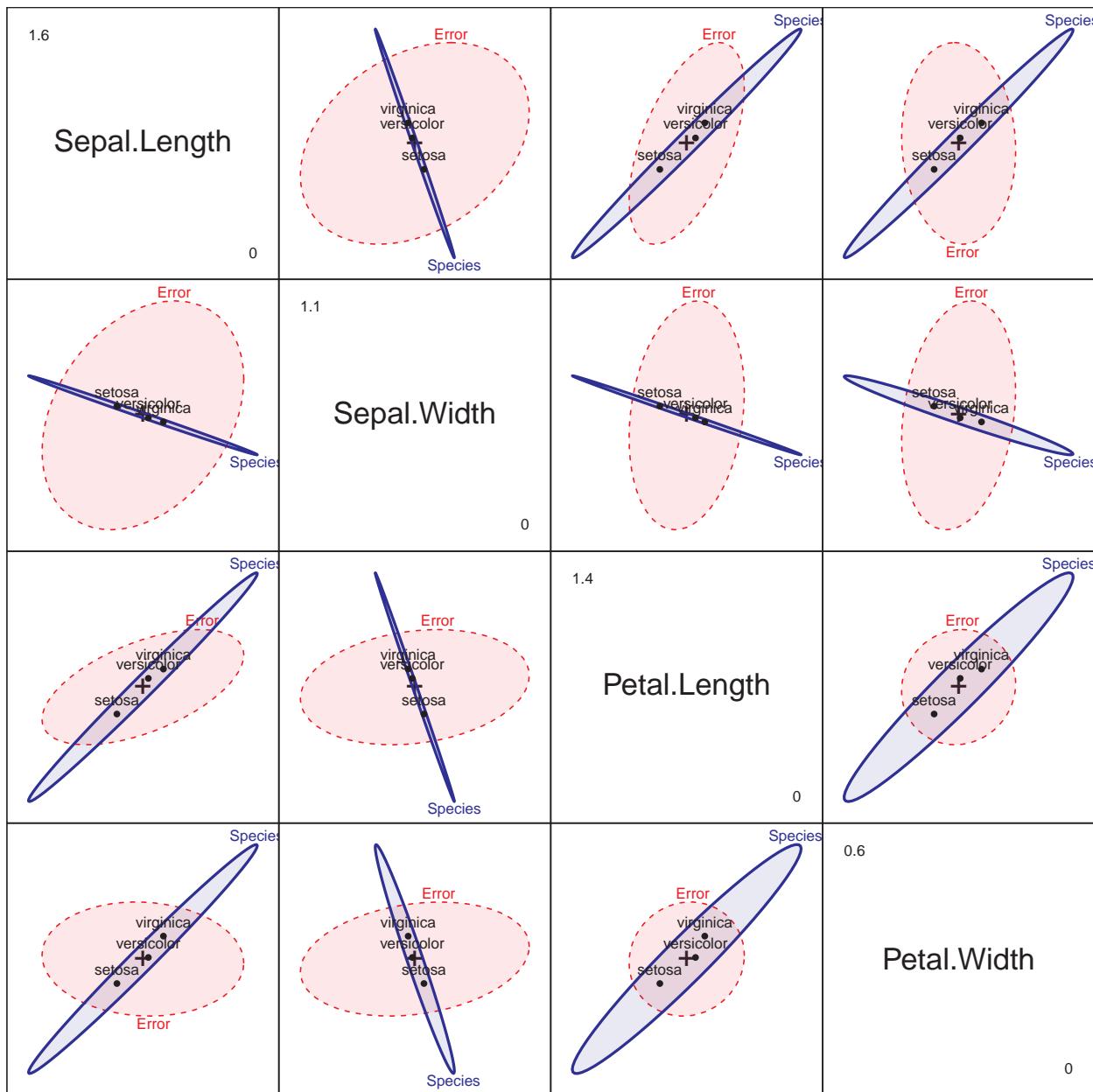


Figure 12.10: HE plots for the multivariate generalization of Levine's test based on MANOVA of absolute deviations from group medians.

```
#> 2 0.013      0.0132      0.602      2.1      100.0
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#> LR test stat approx F numDF denDF Pr(> F)
#> 1      0.611     10.06      8    288 2.3e-12 ***
#> 2      0.987     0.64      3    145   0.59
```

```
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Plotting the first canonical dimension gives boxplots for the scores and vectors for the weights of the iris variables.

```
plot(irisdev.can, which=1,
      fill.alpha = .3,
      col = iris.colors, lwd = 2,
      var.cex = 1.2,
      cex.lab = 1.5, cex.axis = 1.1)
```

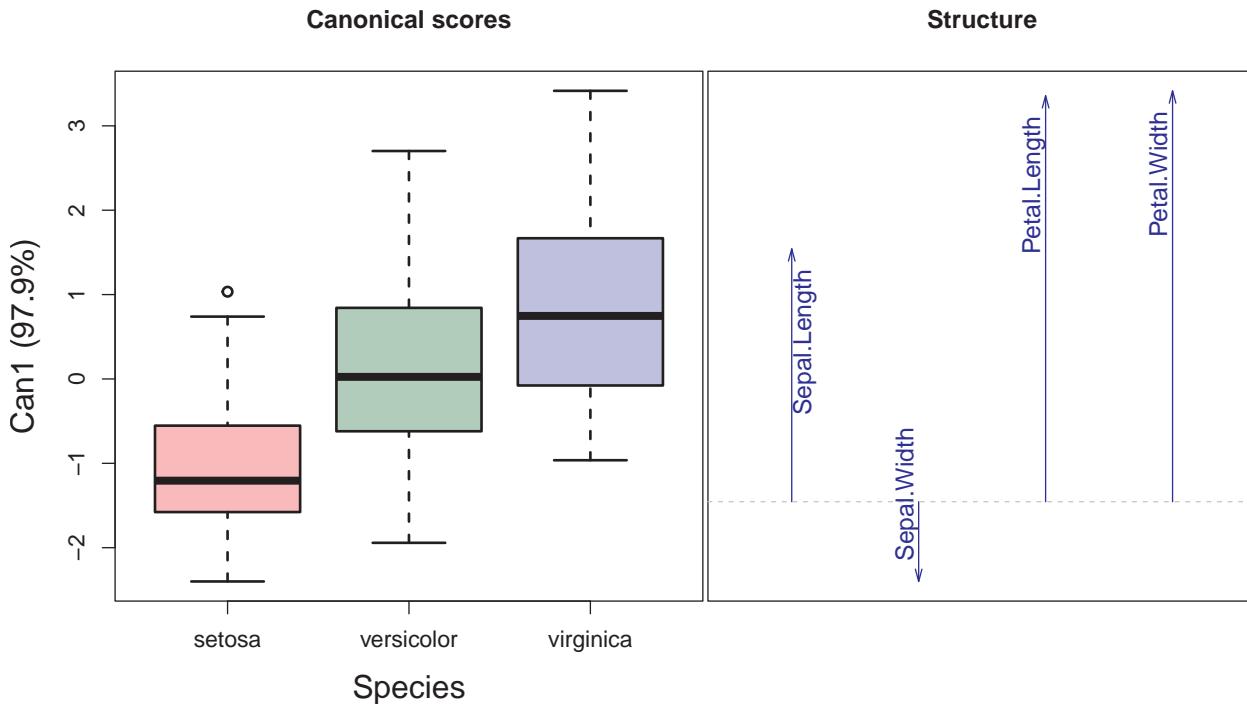


Figure 12.11: Canonical discriminant plot for the multivariate generalization of Levine's test based on MANOVA of absolute deviations from group medians.

Overall, the covariance matrices are smallest for setosa and largest for virginica. The structure coefficients for the variables reflect the pattern shown in the HE plots (Figure 12.10), where the direction of the \mathbf{H} ellipses is negative for sepal width but positive for the others.

12.10 What Have We Learned?

The quest to understand equality of covariance matrices in multivariate models has taken us on a journey from Box's famous row boat metaphor to some novel visualization techniques. Here are the essential insights that will transform how you think about and explore heterogeneity in your multivariate data:

- **Visualization beats test statistics alone** - While Box's \mathcal{M} -test gives you a single p -value, covariance

ellipses reveal the *why* behind heterogeneity. You can literally see differences in size (scatter) and shape (orientation) of group covariances, making the abstract concrete and interpretable.

- **Centering makes differences more apparent** - Shifting all group ellipses to a common center (the grand mean) is like removing visual noise to see the signal. This simple transformation makes differences in covariance structure apparent, turning subtle variations into obvious visual patterns.
- **Small dimensions often hold big secrets** - Don't ignore those "unimportant" principal components! The smallest PC dimensions frequently contain the most discriminating information about covariance differences. It's often where the action is hiding, away from the obvious patterns in major dimensions.
- **Multiple measures tell richer stories** - Box's \mathcal{M} -test uses log determinants, but why stop there? Exploring different eigenvalue functions (sum, precision, maximum) can reveal distinct patterns of heterogeneity, like having multiple lenses to examine the same phenomenon.
- **Levene meets MANOVA in beautiful harmony** - The multivariate extension of Levene's test (MANOVA on absolute deviations) creates a natural bridge between univariate and multivariate thinking, complete with interpretable HE plots that make variance differences as visual as mean differences.
- **Graphics simplify complex concepts** - These visualization tools transform an intimidating mathematical concept (equality of $p \times p$ matrices) into something any researcher can explore, understand, and communicate to others.

The chapter's strength lies in showing that covariance matrices aren't just mathematical abstractions—they're visual stories waiting to be told, patterns waiting to be discovered, and insights illustrating the power of multivariate thinking.

13

Multivariate Influence and Robust Estimation

Note

This chapter may be moved from the printed PDF copy to an online Appendix.

In the analysis of linear models, the identification and treatment of outliers and influential observations represents one of the most critical yet challenging aspects of statistical modeling. As you saw earlier (Section 6.6), even a single “bad” observation can completely alter the results of a linear model fit by ordinary least squares.

Univariate influence diagnostics have been well-established since the pioneering work of R. D. Cook (1977) and others (Belsley et al. (1980); R. D. Cook & Weisberg (1982)) and their wide implementation in R packages such as `stats` and `car` makes these readily accessible in statistical practice. If you seek statistical advice regarding a perplexing model, the consultant may well ask:

Did you make any influence or other diagnostic plots?

However, the extension to multivariate response models introduces additional complexity that goes far beyond simply applying univariate methods to each response variable separately. The multivariate case requires consideration of the *joint influence* structure across all responses simultaneously, accounting for the correlation patterns among dependent variables and the potential for observations to be influential in some linear combinations of responses while appearing benign when examined multivariate.

This multivariate perspective can reveal influence patterns that would otherwise remain hidden, as an observation might exert substantial leverage on the overall model fit through subtle but systematic effects across multiple responses. Detecting outliers and influential observations has now progressed to the point where the methods described below (Section 13.1) can usefully be applied to multivariate linear models.

But having found some troublesome cases, the question arises, what to do about them? We are generally reluctant to simply ignore them, unless there is evidence of a gross data error (as in the `Davis` data, Section 2.1.2). Instead, a large class of **robust** methods, which reduce the impact of outliers on the analysis, have been developed. These are described in Section 13.5 below.

Packages

In this chapter I use the following packages. Load them now.

```
library(dplyr)
library(tidyr)
library(car)
library(helplots)
library(candisc)
library(mvinfluence)
library(ggplot2)
library(patchwork)
```

13.1 Multivariate influence

An elegant extension of the ideas behind leverage, studentized residuals and measures of influence to the case of multivariate response data is due to Barrett & Ling (1992) (see also: Barrett (2003)). These methods have been implemented in the `mvinfluence` package (Friendly, 2025a) which makes available several forms of influence plots to visualize the results.

As in the univariate case, the measures of multivariate influence stem from case-deletion idea of comparing some statistic calculated from the full sample to that statistic calculated when case i is deleted. The Barrett-Ling approach generalized this to the case of deleting a set I of $m \geq 1$ cases. This can be useful because some cases can “mask” the influence of others in the sense that when one is deleted, others become much more influential. However, in most cases the default of deleting individual observations ($m = 1$) is sufficient.

13.1.1 Notation

It is useful to define some notation used to designate terms in the model calculated from the *complete* dataset versus those calculated with one or more observations *excluded*. As before, let \mathbf{X} be the model matrix in the multivariate linear model, $\mathbf{Y}_{n \times p} = \mathbf{X}_{n \times q} \mathbf{B}_{q \times p} + \mathbf{E}_{n \times p}$. As we know, the usual least squares estimate of \mathbf{B} is given by $\mathbf{B} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$.

Then let:

- \mathbf{X}_I be the *submatrix* of \mathbf{X} whose m rows are indexed by I ,
- $\mathbf{X}_{(-I)}$ is the *complement*, the submatrix of \mathbf{X} with the m rows in I deleted,

Matrices \mathbf{Y}_I , $\mathbf{Y}_{(-I)}$ are defined similarly, denoting the submatrix of m rows of \mathbf{Y} and the submatrix with those rows deleted, respectively.

The calculation of regression coefficients when the cases indexed by I have been removed has the form $\mathbf{B}_{(-I)} = (\mathbf{X}_{(-I)}^\top \mathbf{X}_{(-I)})^{-1} \mathbf{X}_{(-I)}^\top \mathbf{Y}_I$. The corresponding residuals are expressed as $\mathbf{E}_{(-I)} = \mathbf{Y}_{(-I)} - \mathbf{X}_{(-I)} \mathbf{B}_{(-I)}$.

13.1.2 Hat values and residuals

The influence measures defined by Barrett & Ling (1992) are functions of two matrices \mathbf{H}_I and \mathbf{Q}_I corresponding to hat values and residuals, defined as follows:

- For the full data set, the “hat matrix”, \mathbf{H} , is given by $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$,
- \mathbf{H}_I is the $m \times m$ the submatrix of \mathbf{H} corresponding to the index set I , $\mathbf{H}_I = \mathbf{X}(\mathbf{X}_I^\top \mathbf{X}_I)^{-1} \mathbf{X}_I^\top$,
- \mathbf{Q} is the analog of \mathbf{H} defined for the residual matrix \mathbf{E} , that is, $\mathbf{Q} = \mathbf{E}(\mathbf{E}^\top \mathbf{E})^{-1} \mathbf{E}^\top$, with corresponding submatrix $\mathbf{Q}_I = \mathbf{E}_I(\mathbf{E}_I^\top \mathbf{E}_I)^{-1} \mathbf{E}_I^\top$,

13.1.3 Cook's distance

Multivariate analogs of all the usual influence diagnostics (Cook's D, CovRatio, ...) can be defined in terms of \mathbf{H} and \mathbf{Q} . For instance, Cook's distance is defined for a univariate response by

$$D_I = (\mathbf{b} - \mathbf{b}_{(-I)})^\top (\mathbf{X}^\top \mathbf{X})(\mathbf{b} - \mathbf{b}_{(-I)}) / ps^2 ,$$

a measure of the squared distance between the coefficients \mathbf{b} for the full data set and those $\mathbf{b}_{(-I)}$ obtained when the cases in I are deleted.

In the multivariate case, Cook's distance is obtained by replacing the vector of coefficients \mathbf{b} by $\text{vec}(\mathbf{B})$, the result of stringing out the coefficients for all responses in a single $(n \times p)$ -length vector.

$$D_I = \frac{1}{p} [\text{vec}(\mathbf{B} - \mathbf{B}_{(-I)})]^T (S^{-1} \otimes \mathbf{X}^T \mathbf{X}) \text{vec}(\mathbf{B} - \mathbf{B}_{(-I)}) ,$$

where \otimes is the Kronecker (direct) product and $\mathbf{S} = \mathbf{E}^T \mathbf{E} / (n - p)$ is the covariance matrix of the residuals.

13.1.4 Leverage and residual components

We gain further insight by considering how far we can generalize from the case for a univariate response. When $m = 1$, Cook's distance can be re-written as a product of leverage and residual components as

$$D_i = \left(\frac{n-p}{p} \right) \frac{h_{ii} q_{ii}}{(1-h_{ii})^2} .$$

This suggests that we define a *leverage component* L_i and *residual component* R_i as

$$L_i = \frac{h_{ii}}{1-h_{ii}} \quad R_i = \frac{q_{ii}}{1-h_{ii}} .$$

R_i is the studentized residual here, and $D_i \propto L_i \times R_i$.

In the general, multivariate case there are analogous matrix expressions for \mathbf{L} and \mathbf{R} . When $m > 1$, the quantities \mathbf{H}_I , \mathbf{Q}_I , \mathbf{L}_I , and \mathbf{R}_I are $m \times m$ matrices. Where scalar quantities are needed, the `mvinfluence` functions apply a function, `FUN`, either `det()` or `tr()` to calculate a measure of "size". This is done as:

```
H <- sapply(x$H, FUN)
Q <- sapply(x$Q, FUN)
L <- sapply(x$L, FUN)
R <- sapply(x$R, FUN)
```

This is the same trick used in the calculation of the various multivariate test statistics like Wilks' Lambda and Pillai's trace. In this way, the full range of multivariate influence measures discussed by Barrett (2003) can be calculated.

13.2 The Mysterious Case 9

To illustrate these ideas, this example, from Barrett (2003), considers the simplest case, of one predictor (`x`) and two response variables, `y1` and `y2`.

```
Toy <- tibble(
  case = 1:9,
  x = c(1, 1, 2, 2, 3, 3, 4, 4, 10),
  y1 = c(0.10, 1.90, 1.00, 2.95, 2.10, 4.00, 2.95, 4.95, 10.00),
  y2 = c(0.10, 1.80, 1.00, 2.93, 2.00, 4.10, 3.05, 4.93, 10.00)
)
```

A quick peek (Figure 13.1) at the data indicates that `y1` and `y2` are nearly perfectly correlated with each other. Both of these are also strongly linear with `x` and there is one extreme point (case 9). The data is peculiar, but looking at these pairwise plots doesn't suggest that anything is terribly wrong. In the plots of `y1` and `y2` against `x`, case 9 simply looks like a good leverage point (Section 6.6).

```
scatterplotMatrix(~ y1 + y2 + x, data=Toy,
  cex=2,
```

```
col = "blue", pch = 16,
id = list(n=1, cex=2),
regLine = list(lwd = 2, col="red"),
smooth = FALSE)
```

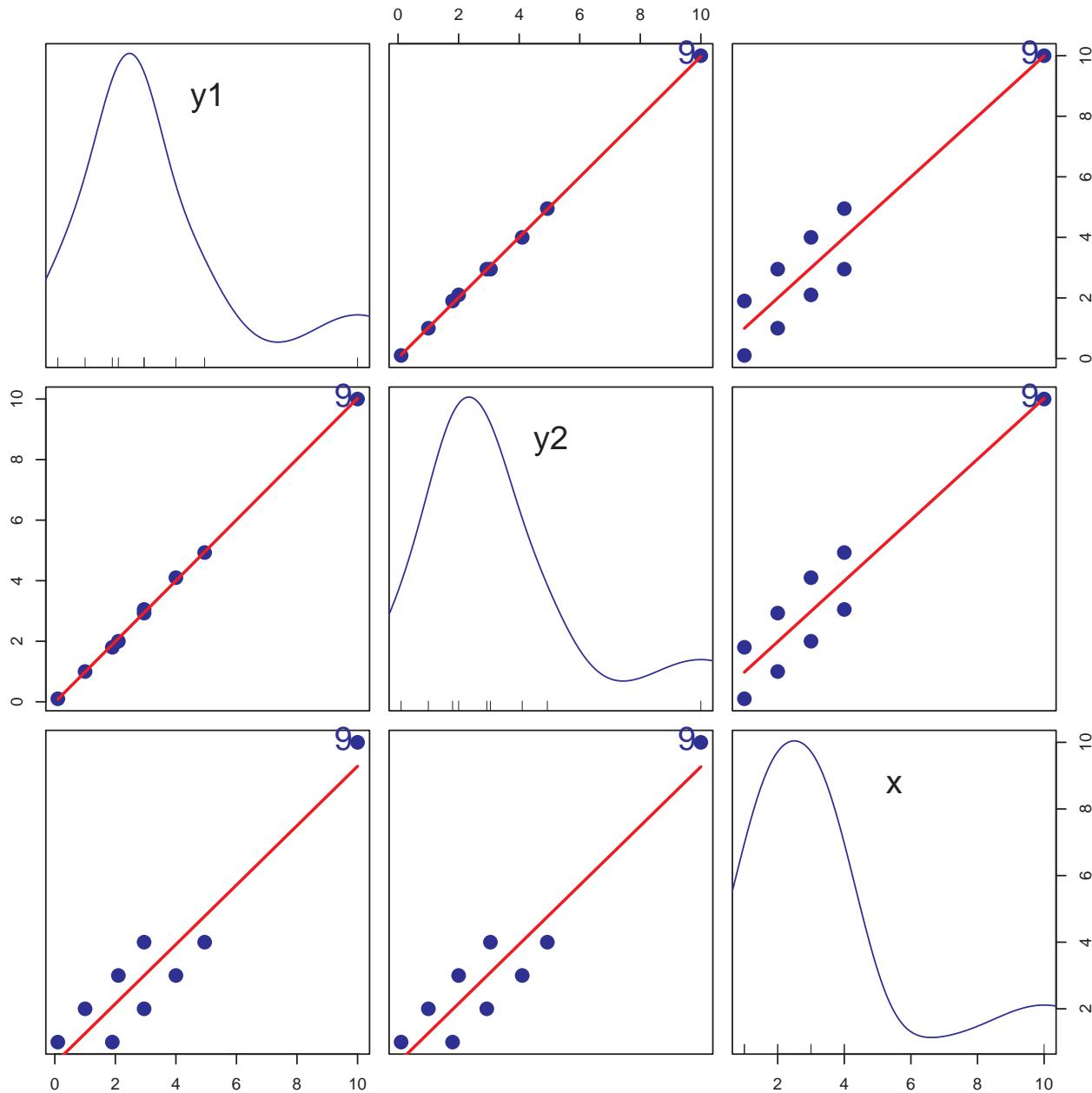


Figure 13.1: Scatterplot matrix for the toy example.

For this example, we fit the univariate models with y_1 and y_2 separately and then the multivariate model.

```
Toy.lm1 <- lm(y1 ~ x, data=Toy)
Toy.lm2 <- lm(y2 ~ x, data=Toy)
Toy.mlm <- lm(cbind(y1, y2) ~ x, data=Toy)
```

13.2.1 Cook's D

First, let's examine the Cook's D statistics for the models. Note that the function `cooks.distance()` invokes `stats::cooks.distance.lm()` for the univariate response models, but invokes `mvinfluence::cooks.distance.mlm()` for the multivariate model.

```
df <- Toy
df$D1 <- cooks.distance(Toy.lm1)
df$D2 <- cooks.distance(Toy.lm2)
df$D12 <- cooks.distance(Toy.mlm)

df
#> # A tibble: 9 x 7
#>   case     x     y1     y2      D1      D2     D12
#>   <int> <dbl> <dbl> <dbl>    <dbl>    <dbl>   <dbl>
#> 1     1     1   0.1   0.1   0.121   0.118   0.125
#> 2     2     1   1.9   1.8   0.124   0.103   0.298
#> 3     3     2     1     1   0.0901   0.0886   0.0906
#> 4     4     2   2.95   2.93   0.0829   0.0819   0.0831
#> 5     5     3   2.1     2   0.0548   0.0673   0.182
#> 6     6     3     4     4.1   0.0692   0.0852   0.232
#> 7     7     4   2.95   3.05   0.0793   0.0651   0.203
#> 8     8     4   4.95   4.93   0.0665   0.0643   0.0690
#> 9     9    10    10    10   0.00159   0.00830  3.22
```

The only thing remarkable here is for case 9: The univariate Cook's Ds, D1 and D2 are very small, yet the multivariate statistic, D12=3.22 is over 10 times the next largest value.

Let's see how case 9 stands out in the influence plots (Figure 13.2). It has an extreme hat value. But, because it's residual is very small, it does not have much influence on the fitted models for either y_1 or y_2 . Neither of these plots suggest that anything is terribly wrong with the univariate models—none of the points are in the “danger” zones of the upper- and lower-right corners.

```
ip1 <- car::influencePlot(Toy.lm1,
                           id = list(cex=1.5), cex.lab = 1.5)
ip2 <- car::influencePlot(Toy.lm2,
                           id = list(cex=1.5), cex.lab = 1.5)
```

TODO: Check how these are defined in `mvinfluence`

Contrast these results with what we get for the model for y_1 and y_2 jointly (Figure 13.3) In the multivariate version, `mvinfluence::influencePlot.mlm()` plots the squared studentized residual (denoted R in the output) against the hat value; this is referred to as a ‘type = “stres” plot.¹ Case 9 stands in Figure 13.3 out as wildly influential on the joint regression model. But there's more: The cases in Figure 13.2 with large Cook's D (bubble size) have only tiny influence in the multivariate model.

```
influencePlot(Toy.mlm, type = "stres",
              id.n=2, id.cex = 1.3,
              cex.lab = 1.5)
#>      H      Q CookD      L      R
#> 1 0.202 0.177 0.125 0.253 0.222
```

¹Similar to the univariate version, hat values greater than 2 or 3 times their average, $\bar{h} = p/n$ here, are considered large in the multivariate case. Values of the squared studentized residual R_i are calibrated by the Beta distribution, $\text{Beta}(\alpha = 0.95, q/2, (n - p - q)/2)$.

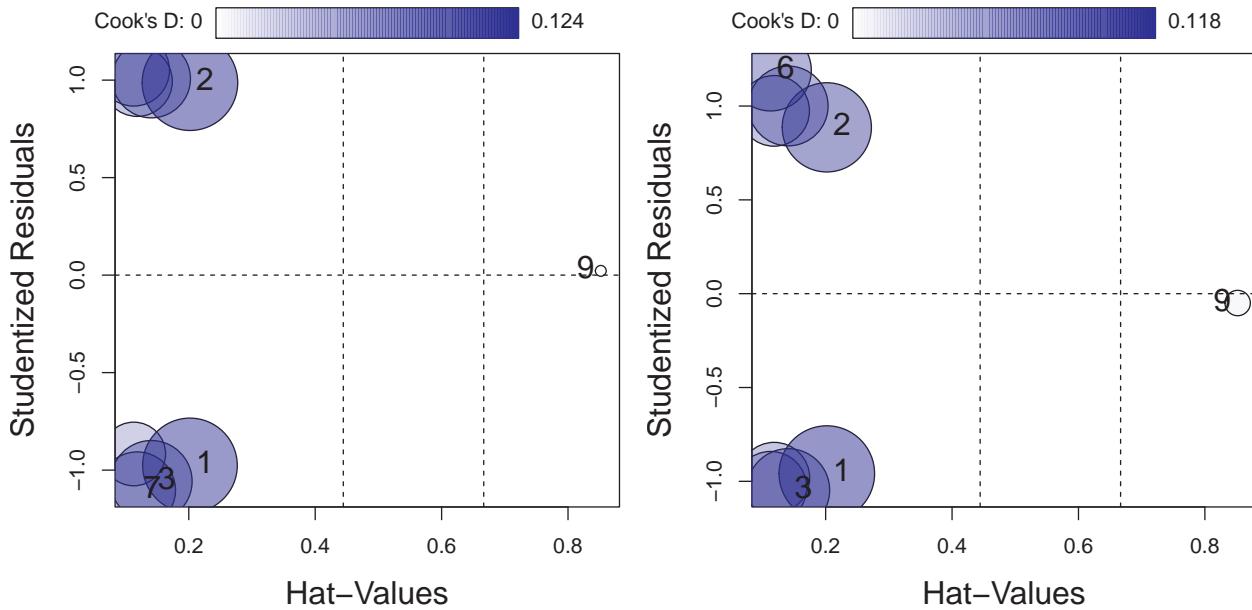


Figure 13.2: Influence plots for the univariate models for y_1 and y_2

```
#> 2 0.202 0.422 0.298 0.253 0.529
#> 6 0.113 0.587 0.232 0.127 0.662
#> 9 0.852 1.082 3.225 5.750 7.301
```

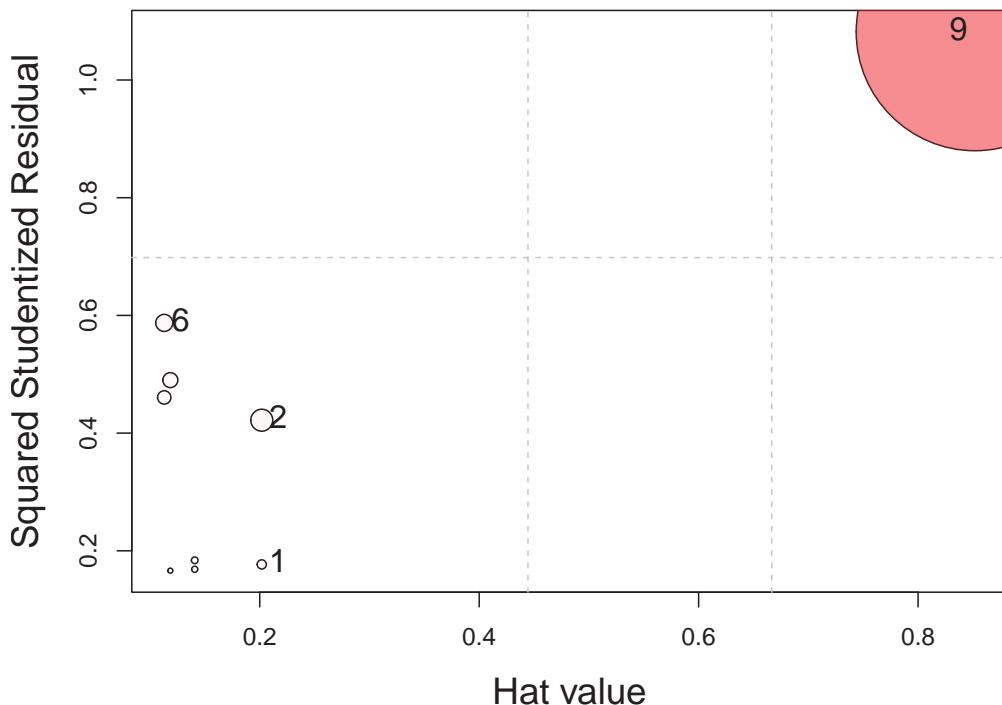


Figure 13.3: Studentized residual influence plot for the multivariate model $(y_1, y_2) \sim x$. Dotted vertical lines mark large hat values, $H > 2, 3p/n$. The dotted horizontal line marks large values of the squared studentized residual.

Theory into Practice

Chairman Mao said, “Theory into practice”, but Tukey (1959) said that, “The practical power of a statistical test is the product of its’ statistical power and the probability of use”. The story for multivariate influence here illustrates a nice feature of the connections between statistical theory, graphic development and implemented in software here.

A statistical development proposes a new way of thinking about a problem. People with a graphical bent look at this and think, “How can I visualize this?”. A software developer then solves the remaining problem of how to incorporate that into easy-to-use functions or applications. If only this was easy, but sometimes, all three roles appear within a given person.

The general formulation of Barrett (2003) suggests an alternative form of the multivariate influence plot (Figure 13.4) that uses the leverage (L) and residual (R) components (`type = "LR"`) directly.

Because influence is the product of leverage and residual, a plot of $\log(L)$ versus $\log(R)$ has the attractive property that contours of constant Cook’s distance fall on diagonal lines with slope = -1. Adjacent reference lines represent constant *multiples* of influence.

```
influencePlot(Toy.mlm, type="LR",
              id.n=2, id.cex = 1.3,
              cex.lab = 1.5) -> infl
```

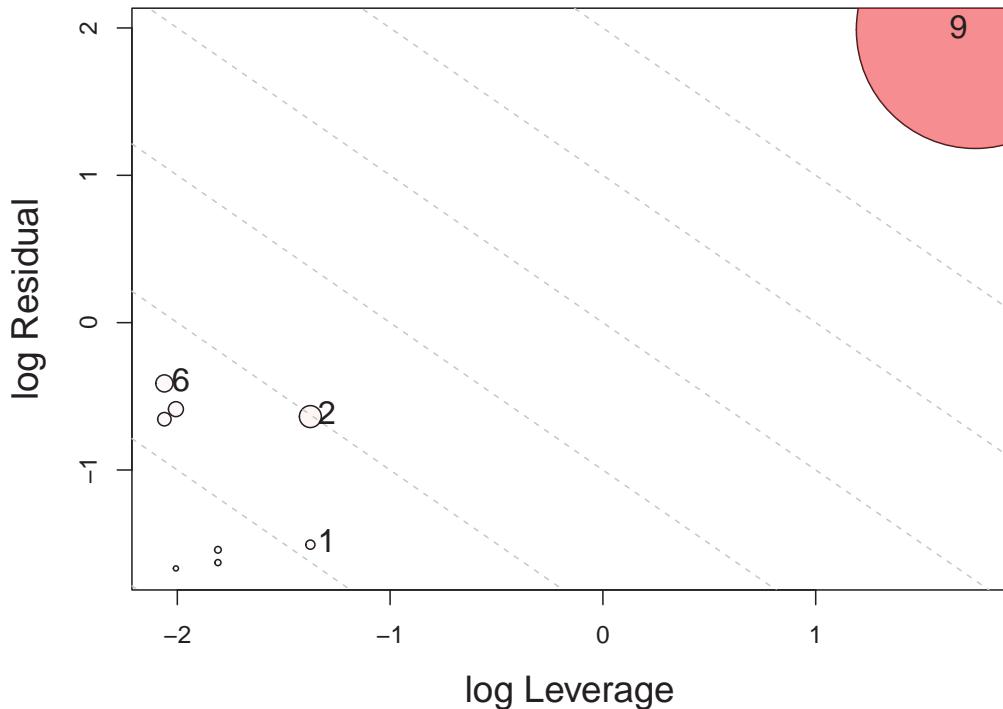


Figure 13.4: LR plot of $\log(L)$ versus $\log(R)$ for the multivariate model $(y_1, y_2) \sim x$. Dotted lines show contours of constant Cook’s distance.

13.2.2 DFBETAS

The DFBETAS statistics give the estimated change in the regression coefficients when each case is deleted in turn. We can gain some insight as to why case 9 is unremarkable in the univariate regressions by plotting these, shown in Figure 13.5. The values come from `stats::dfbetas()` and return the standardized values.

```

db1 <- as.data.frame(dfbetas(Toy.lm1))
gg1 <- ggplot(data = db1, aes(x=`(Intercept)`, y=x, label=rownames(db1))) +
  geom_point(size=1.5) +
  geom_label(size=6, fill="pink") +
  xlab(expression(paste("Deletion Intercept ", b[0]))) +
  ylab(expression(paste("Deletion Slope ", b[1]))) +
  ggtitle("dfbetas for y1") +
  theme_bw(base_size = 16)

db2 <- as.data.frame(dfbetas(Toy.lm2))
gg2 <- ggplot(data = db2, aes(x=`(Intercept)`, y=x, label=rownames(db2))) +
  geom_point(size=1.5) +
  geom_label(size=6, fill="pink") +
  xlab(expression(paste("Deletion Intercept ", b[0]))) +
  ylab(expression(paste("Deletion Slope ", b[1]))) +
  ggtitle("dfbetas for y2") +
  theme_bw(base_size = 16)

gg1 + gg2

```

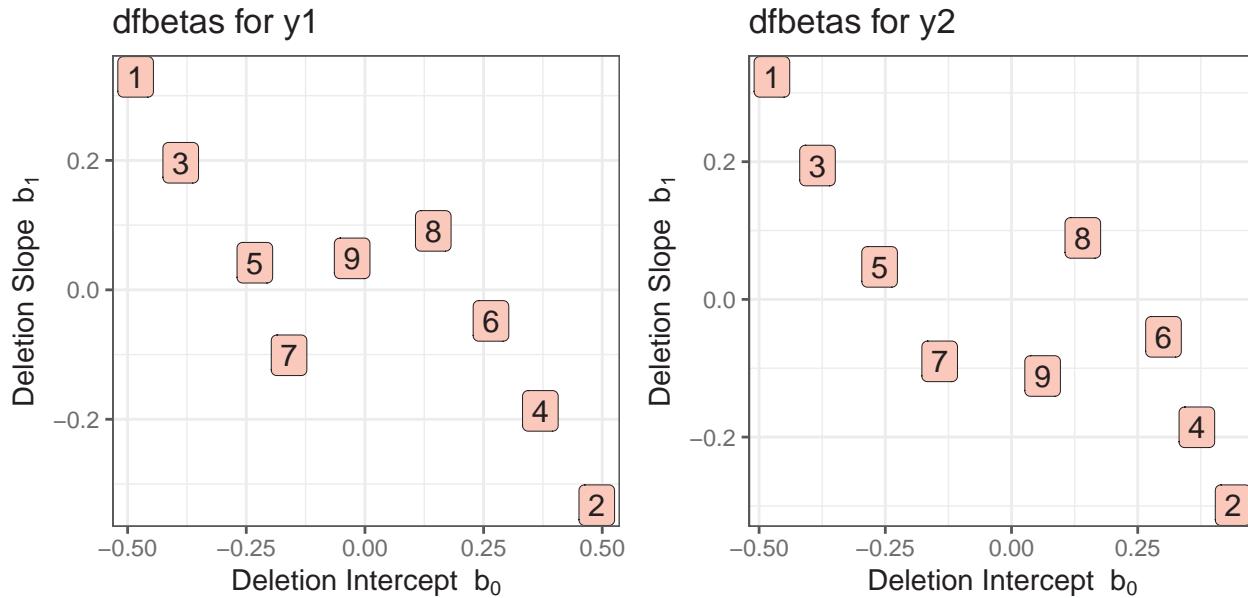


Figure 13.5

The values for case 9 are nearly (0, 0) in both plots, indicating that deleting this case has negligible effect in both *univariate* regressions. Yet, case 9 appeared very influential in the multivariate model. Why did this happen?

In this contrived example, the problem arose from the very high correlation between y_1 and y_2 , $r = 0.9997$ as can be seen in the (y_1, y_2) panel in Figure 13.1. Although each of the y_1 and y_2 values for the high-leverage cases are in-line with the univariate regressions (and thus have small univariate Cook's Ds), the ill-conditioning magnifies small discrepancies in their positions, making the multivariate Cook's D larger. And that's the solution to the Mysterious Case 9.

13.3 Example: NLSY data

The `heplots::NLSY` data introduced in Section 10.5.1 provides a more realistic example of how the *multivariate* influence measures and their plots contribute to understanding multivariate data. Some plots were suggested there (Figure 10.10) to identify “noteworthy” points. The model `NLSY.mod1` fits the child’s reading and math scores using parents’ education:

```
data(NLSY, package = "heplots")
NLSY.mod1 <- lm(cbind(read, math) ~ income + educ,
                 data = NLSY)
```

For influence plots, the `id.method = "noteworthy"` method for point labeling selects observations with large values for *any* of the standardized residual, hat value or Cook’s D, so 6 points are labeled in Figure 13.6.

```
influencePlot(NLSY.mod1,
              id.method = "noteworthy",
              id.n = 3, id.cex = 1.25,
              cex.lab = 1.5)
#>          H      Q   CookD      L      R
#> 12  0.01547 0.090101 0.11149 0.01571 0.091516
#> 19  0.14656 0.004801 0.05629 0.17173 0.005625
#> 54  0.16602 0.016554 0.21986 0.19907 0.019849
#> 142 0.00509 0.208258 0.08478 0.00511 0.209323
#> 152 0.00422 0.049821 0.01682 0.00424 0.050032
#> 221 0.05707 0.000873 0.00399 0.06053 0.000926
```

There is an interesting configuration of the points in Figure 13.6. One group of points (152, 12, 142) is apparent at the left side. These have relatively low leverage (hat value), and increasingly large residuals. Another group of points (221, 19, 54) have small residuals but increasingly large hat values. You should take a moment and compare these points with their positions in Figure 10.10. However, with `id.n = 2` only two points are flagged: 54 and 142.

The LR plot is also instructive here.

```
influencePlot(NLSY.mod1, type = "LR",
              id.method = "noteworthy",
              id.n = 3, id.cex = 1.25,
              cex.lab = 1.5)
#>          H      Q   CookD      L      R
#> 12  0.01547 0.090101 0.11149 0.01571 0.091516
#> 19  0.14656 0.004801 0.05629 0.17173 0.005625
#> 54  0.16602 0.016554 0.21986 0.19907 0.019849
#> 142 0.00509 0.208258 0.08478 0.00511 0.209323
#> 152 0.00422 0.049821 0.01682 0.00424 0.050032
#> 221 0.05707 0.000873 0.00399 0.06053 0.000926
```

In Figure 13.7 the noteworthy points are arrayed within a diagonal band corresponding to contours of equal Cook’s D, and this statistic increases multiplicatively with each step away from the origin. Cases 54 and 142 stand out somewhat here.

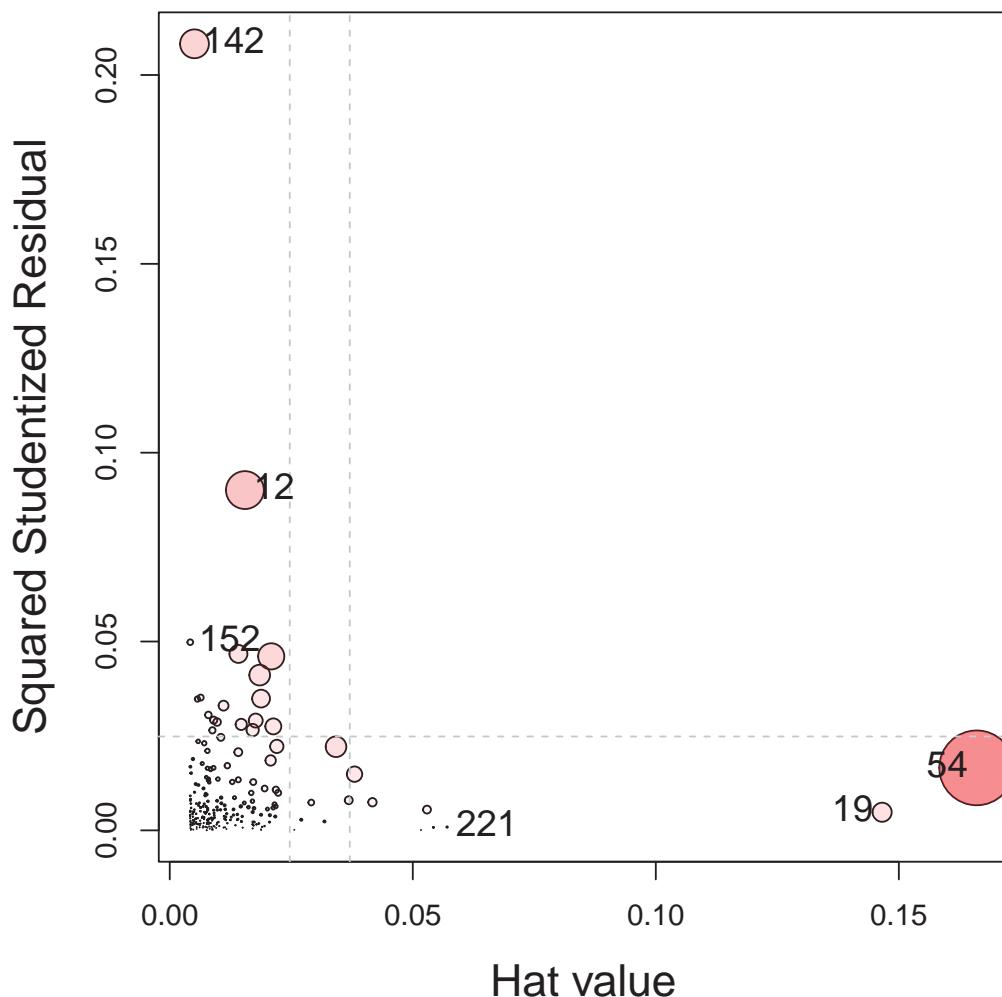


Figure 13.6: Influence plot for the NLSY data...

13.4 Example: Penguin data

Another example illustrates the appearance of influence plots with factors as predictors. Earlier chapters (Section 3.9.2, Section 4.7.1) presented a variety of plots (Figure 3.26, Figure 3.27, Figure 4.33) for the Penguin data in which a few cases were identified as unusual in exploratory analysis. But, are any of them influential in the context of a multivariate model?

Let's consider the simple model predicting `species` from the numeric variables

```
data(peng, package="heplots")
peng.mlm <- lm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~
  species, data=peng)
```

In the influence plot (Figure 13.8) the predictor `species` is of course discrete, so there are only three distinct hat-values and the values for each species appear as columns of points in this plot. For ease of interpretation, I use a little `dplyr` magic to label the species and their sample sizes.

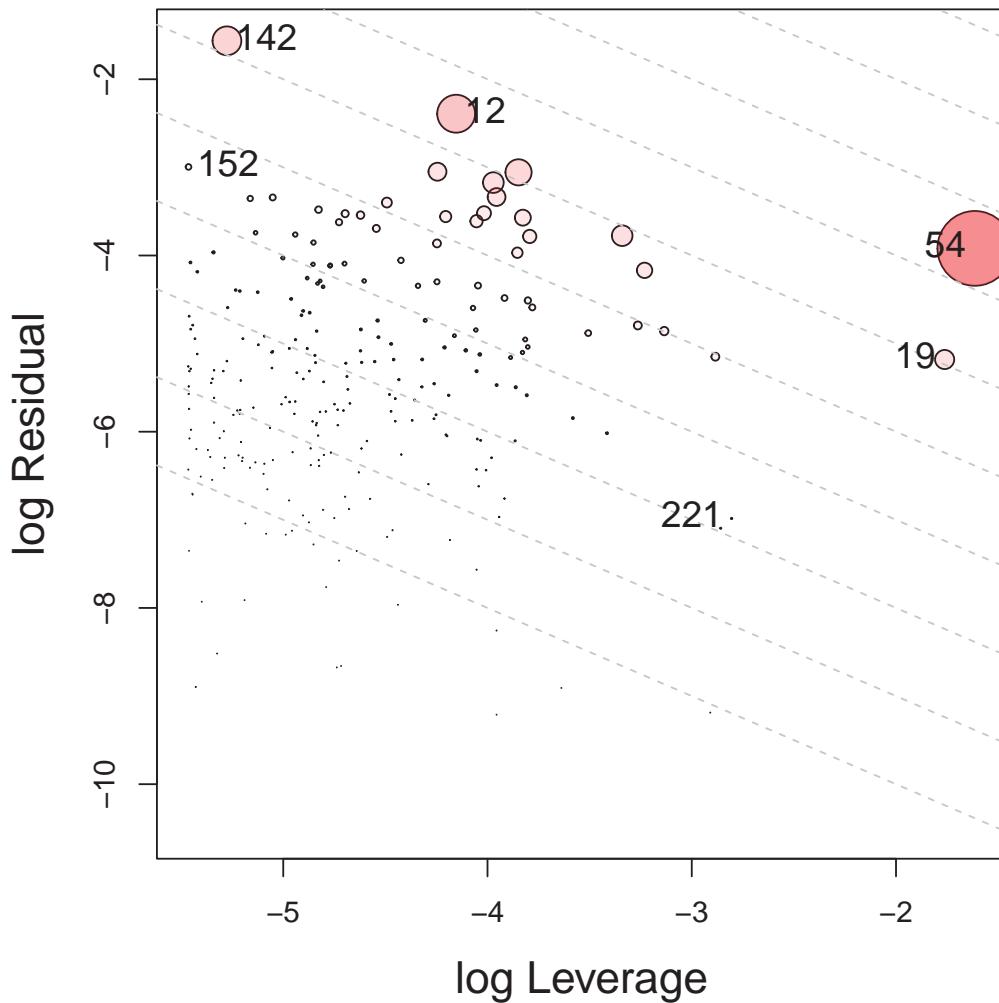


Figure 13.7: Influence plot for the NLSY data...

```

res <- influencePlot(peng.mlm, id.n=3, type="stres")
res |> arrange(desc(CookD)) |> print(digits = 2)
#>      H      Q   CookD      L      R
#> 283 0.0147 0.0919 0.1486 0.0149 0.0932
#> 286 0.0147 0.0322 0.0520 0.0149 0.0327
#> 179 0.0084 0.0512 0.0474 0.0085 0.0517
#> 10  0.0068 0.0562 0.0424 0.0069 0.0566
#> 268 0.0147 0.0079 0.0129 0.0149 0.0081
#> 267 0.0147 0.0037 0.0061 0.0149 0.0038
#> 266 0.0147 0.0021 0.0034 0.0149 0.0021

# labels for species, adding sample n
loc <- merge(peng |> add_count(species),
             res,
             by = "row.names") |>
  group_by(species) |>
  slice(1) |>

```

```
ungroup() |>
  select(species, H, n) |>
  mutate(label = glue::glue("{species}\n(n={n})"))
  text(loc$H, 0.10, loc$label, xpd=TRUE)
```

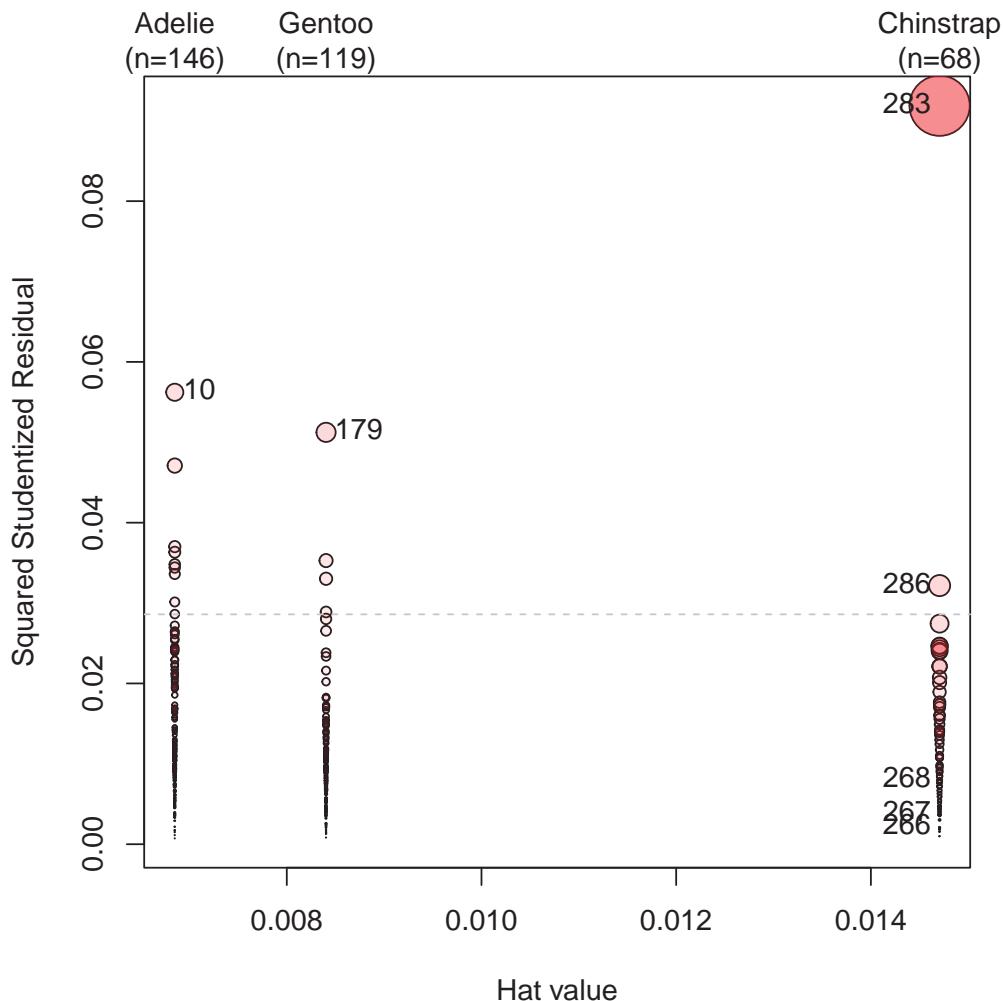


Figure 13.8: Influence plot for the Penguin data, showing the squared studentized residual vs. hat value. Unusual points on either variable or on the Cook's D statistic are identified with their case number.

We know that hat-values are proportional to how unusual the observations are from the means of the predictors, but what is this for a factor, like `species`? The answer is that $H_j \propto 1/n_j$, so Chinstrap with the smallest sample size ($n = 68$) is the most unusual.

The most influential case (283) here is our Chinstrap friend “Cyrano” (see Figure 3.27). Among the others, case 10 is the Adelie bird we labeled “Hook Nose”. To better understand *why* these cases are influential, we can make plots of the data in data space or in PCA space as we did earlier.

13.5 Robust Estimation

Robust methods for multivariate linear models aim to provide reliable parameter estimates and inference procedures that remain stable in the presence of outlying observations. As noted by Peter J. Rousseeuw et al. (2004), “*The main advantage of robust regression is that it provides reliable results even when some of the assumptions of classical regression are violated.*” But this advantage comes at the cost of increased computational complexity.

Robust regression is a compromise between excluding “bad” data points entirely from analysis vs. including all the data and treating them equally in classical OLS regression. The essential idea of robust regression is to weight the observations differently based on how well behaved these observations are. Roughly speaking, it is a form of *weighted* least squares regression. But because the weights are derived from the data, this must be done iteratively, requiring more computation.

Several general approaches have been developed for robust multivariate regression. These include M-estimators, S-estimators (Peter J. Rousseeuw & Yohai, 1984), and MM-estimators (Yohai, 1987). Each approach offers different trade-offs between robustness properties, computational efficiency, and statistical efficiency under ideal conditions. See the CRAN Task View: Robust Statistical Methods for an extensive list of robust methods in R and the vignette for the `rrcov` package for a general overview of multivariate robust methods.

The method implemented in the `heplots::robmlm()` function belongs to the class of **M-estimators**. This generalizes OLS estimation by replacing the least squares criterion with a more robust version. The key idea is to relax the least squares criterion of minimizing $Q(\mathbf{e}) = \sum e_i^2 = \sum (y_i - \hat{y}_i)^2$ by considering more general functions $Q(\mathbf{e}, \rho)$, where the function $\rho(e_i)$ can be chosen to reduce the impact of large outliers. In these terms,

- OLS uses $\rho(e_i) = e_i^2$
- L_1 estimation uses $\rho(e_i) = |e_i|$, the least absolute values
- A bit more complicated, the **biweight** function uses a squared measure of error up to some value c and then levels off thereafter,

$$\rho(e_i) = \begin{cases} \left[1 - \left(\frac{e_i}{c}\right)^2\right]^2 & |e_i| \leq c, \\ 1 & |e_i| > c. \end{cases}$$

These functions are more easily understood in a graph (Figure 13.9). The biweight function (ref?) has a property like Winsorizing—the squared error remains constant for residuals $e_i > c$, with the tuning constant $c = 4.685$ for `MASS::psi.bisquare()`.

But, there is a problem here, in that weighted least squares is designed for situations where the observation weights are *known in advance*, for instance, when data points have unequal variances or when some observations are more reliable than others.

The solution, called *iteratively reweighted least squares* (IRLS), is to substitute computation for theory, and iterate between estimating the coefficients (with given weights) and determining weights for the observations in the next iteration. This is implemented in `heplots::robmlm()`, which generalizes the methods used in `MASS::rlm()`. . . .

13.6 Example: Penguin data

In earlier analyses of the Penguin data, a few points appeared unusual for their species in various plots (Figure 3.21, Figure 3.27) and a few were deemed overly influential in Section 13.4 for the simple model

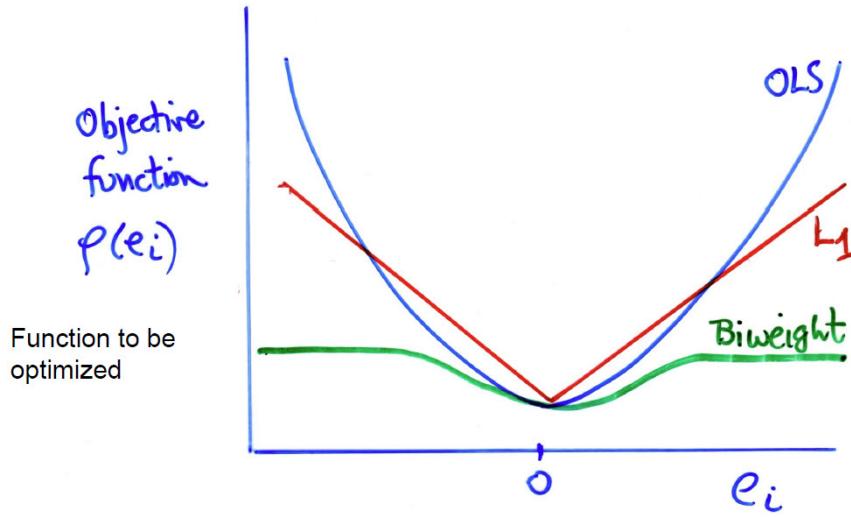


Figure 13.9: Diagram plotting the function $\rho(e_i)$ of the contributions of the residuals e_i to what is minimized in various fitting methods.

`peng.mlm` predicting `species`. What effect does fitting a robust MANOVA have on the results? We can just substitute `robmelm()` for `lm()`. to fit the robust model.

```
peng.rlm <- robmelm(cbind(bill_length, bill_depth, flipper_length, body_mass) ~
                      species, data=peng)
```

The `plot()` method for a "robmelm" object give an index plot of the observation weights in the final iteration. I take some care here to color the points by species, label the points with the lowest weights and add my names for a couple of birds.

```
source(here::here("R", "penguin", "penguin-colors.R"))
col = peng.colors("dark")[peng$species]
plot(peng.rlm,
      segments = TRUE,
      id.weight = 0.6,
      col = col,
      cex.lab = 1.3)
# label old friends
notables <- tibble(
  id = c(10, 283),
  name = c("HookNose", "Cyrano"),
  wts = peng.rlm$weights[id])
text(notables$id, notables$wts,
     label = notables$name, pos = 3,
     xpd = TRUE)
# label species in axis at the top
ctr <- split(seq(nrow(peng)), peng$species) |> lapply(mean)
axis(side = 3, at=ctr, labels = names(ctr), cex.axis=1.2)
```

The observations that are labeled here are among those which stood out as notable in other plots. In Figure 13.10, the lowest four points are down-weighted considerably. Our friend "Cyrano" (283), the greatest

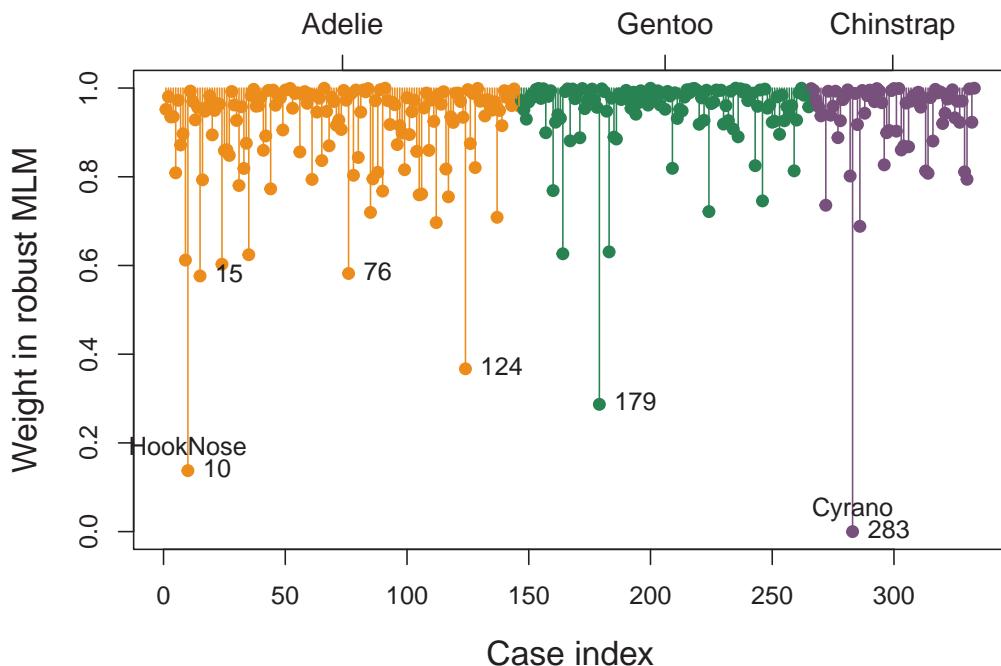


Figure 13.10: Index plot of the observation weights for the robust model, `peng.rlm`. Observations with weights < 0.5 are labeled with their case number.

outlier, gets a weight of exactly zero, so is effectively excluded from the model; “Hook Nose” (10) gets a weight of 0.13, and so he can’t do much damage.

```
cases <- c(10, 124, 179, 283)
lowwt <- peng.rlm$weights[cases]
names(lowwt) <- cases
lowwt
#>   10    124    179    283
#> 0.137 0.367 0.287 0.000
```

Another way to assess the effect of using a robust model is to examine its’ effects on the model coefficients. `heplots::rel_diff()` calculates the relative difference, $|x - y|/x$, expressed here as percent of change. As you can see below, the changes in most of the coefficients is rather small.

```
rel_diff(coef(peng.mlm), coef(peng.rlm)) |>
  print(digits=2)
#>                                bill_length bill_depth flipper_length body_mass
#> (Intercept)                 0.083       0.26       0.092      0.43
#> speciesChinstrap          0.572      78.10      8.053     76.20
#> speciesGentoo             0.857      -0.51      0.089      0.12
```

The largest differences are for the coefficients for `Chinstrap`, particularly for `bill_depth` and `body_mass`, which reflect the dramatic effect of effectively removing “Cyrano” from the analysis.

If you accept the premise of down-weighting cases with large residuals in robust methods, `robmle()` provides a principled way to handle them. Bye-bye Cyrano; it’s been fun, but we can get along nicely without you.

14

Case studies

This chapter presents some complete analyses of datasets that will be prominent in the book. Some of this material may later be moved to earlier chapters.

Packages

In this chapter I use the following packages. Load them now.

```
library(car)
library(helplots)
library(candisc)
library(ggplot2)
library(dplyr)
library(tidyr)
library(corrgram)
library(ggbiplot)

scatterplotMatrix(~ Speed + Attention + Memory + Verbal + Visual + ProbSolv | Dx,
  data=NeuroCog,
  plot.points = FALSE,
  smooth = FALSE,
  legend = FALSE,
  col = scales::hue_pal()(3),
  ellipse=list(levels=0.68))
```

In this figure, we can see that the regression lines have similar slopes and similar data ellipses for the groups, though with a few exceptions. You can also see that the **Normal** group has higher means on all the variables and that the **Schizophrenic** and **SchizoAffective** don't seem to differ very much.

Biplot

While still in exploratory mode, we can gain greater insight with our trusty multivariate juicer, the biplot (Section 4.3). A PCA of the cognitive variables shows that nearly 80% of the total variance is accounted for by two dimensions, of which the first dimension accounts for 69%.

```
neuro.pca <- NeuroCog |>
  select(Speed:Visual) |>
  prcomp(scale. = TRUE)
summary(neuro.pca)

#> Importance of components:
#>              PC1    PC2    PC3    PC4    PC5
#> Standard deviation   1.856  0.720  0.629  0.5761  0.5570
#> Proportion of Variance 0.689  0.104  0.079  0.0664  0.0621
#> Cumulative Proportion 0.689  0.793  0.872  0.9379  1.0000
```

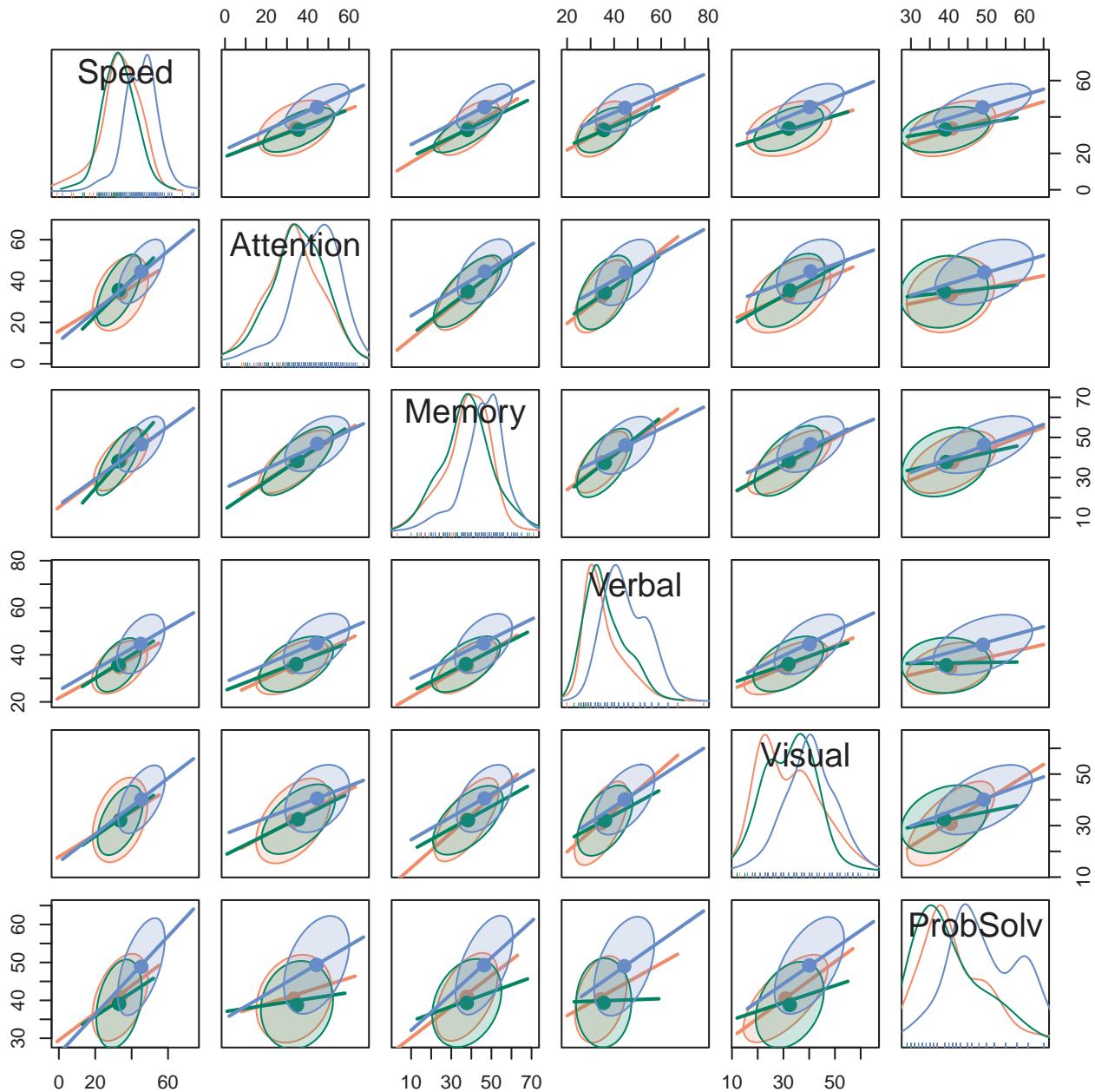


Figure 14.1: Scatterplot matrix of the NeuroCog data. Points are suppressed here, focusing on the data ellipses and regression lines. Colors for the groups: Schizophrenic (red), SchizoAffective (green), Control (blue).

The biplot for this analysis (Figure 14.2) provides a nice summary of what was seen in the scatterplot matrix (Figure 14.1): A large difference between the **Control** group and the others, which don't differ much from each other.

```
ggbiplot(neuro.pca,
  obs.scale = 1, var.scale = 1,
  groups = NeuroCog$Dx,
  varname.size = 5,
```

```

var.factor = 1.5,
ellipse = TRUE) +
scale_color_discrete(name = 'groups') +
theme_minimal() +
theme(legend.direction = 'horizontal',
legend.position = 'top')

```

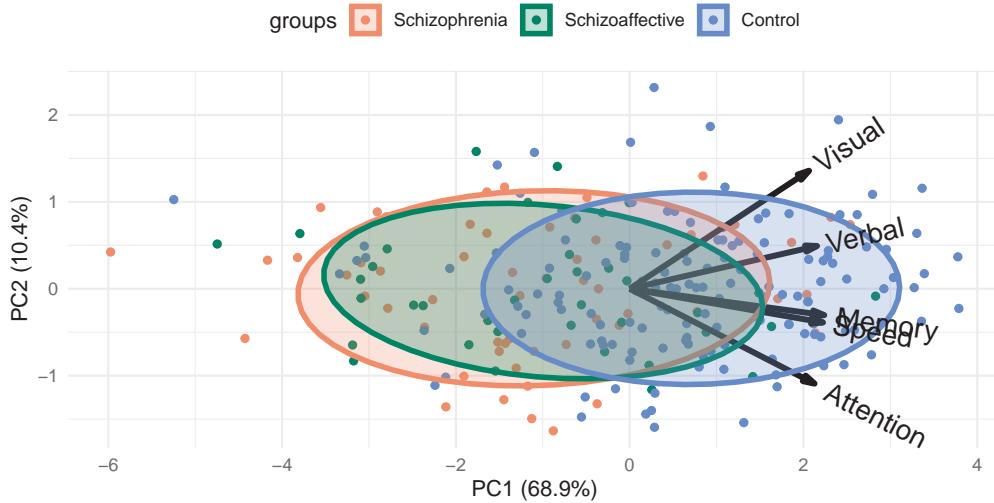


Figure 14.2: Biplot of the NeuroCog data. . .

The variable vectors in Figure 14.2 reflect the correlations of the response variables with each other and with the principal components, PC1 and PC2. All of the variables are positively correlated in this 2D view, **Memory** and **Speed** most highly so.

14.1 Fitting the MLM

We proceed to fit the one-way MANOVA model to answer the main research question of how well these variables distinguish among the groups.

```

NC.mlm <- lm(cbind(Speed, Attention, Memory, Verbal, Visual, ProbSolv) ~ Dx,
               data=NeuroCog)
Anova(NC.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>   Df test stat approx F num Df den Df Pr(>F)
#>   Dx  2      0.299      6.89      12     470 1.6e-11 ***
#>   ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The first research question is captured by the contrasts for the **Dx** factor shown above. We can test these with `car::linearHypothesis()`. The contrast **Dx1** for control vs. the diagnosed groups is highly significant,

```
# control vs. patients
print(linearHypothesis(NC.mlm, "Dx1"), SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.289    15.9     6   234 2.8e-15 ***
#> Wilks       1   0.711    15.9     6   234 2.8e-15 ***
#> Hotelling-Lawley 1   0.407    15.9     6   234 2.8e-15 ***
#> Roy         1   0.407    15.9     6   234 2.8e-15 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

but the second contrast, Dx2, comparing the schizophrenic and schizoaffective group, is not.

```
# Schizo vs SchizAff
print(linearHypothesis(NC.mlm, "Dx2"), SSP=FALSE)
#>
#> Multivariate Tests:
#>           Df test stat approx F num Df den Df Pr(>F)
#> Pillai      1   0.006    0.249     6   234 0.96
#> Wilks       1   0.994    0.249     6   234 0.96
#> Hotelling-Lawley 1   0.006    0.249     6   234 0.96
#> Roy         1   0.006    0.249     6   234 0.96
```

As a quick check on the model, a χ^2 QQ plot (Figure 14.3) reveals no problems with multivariate normality of residuals nor potentially harmful residuals.

```
cqplot(NC.mlm, id.n = 3)
```

14.1.1 HE plot

So the question becomes: how to understand these results.

`heplot()` shows the visualization of the multivariate model in the space of two response variables (the first two by default). The result (Figure 14.4) tells a very simple story: The control group performs higher on higher measures than the diagnosed groups, which do not differ between themselves.

(For technical reasons, to abbreviate the group labels in the plot, we need to `update()` the MLM model after the labels are reassigned.)

```
# abbreviate levels for plots
NeuroCog$Dx <- factor(NeuroCog$Dx,
                        labels = c("Schiz", "SchAff", "Contr"))
NC.mlm <- update(NC.mlm)
```

Then, feed the model to `heplot()` for a plot of the first two response variables, `Speed` and `Attention`.

```
heplot(NC.mlm,
       fill=TRUE, fill.alpha=0.1,
       cex.lab=1.3, cex=1.25)
par(op)
```

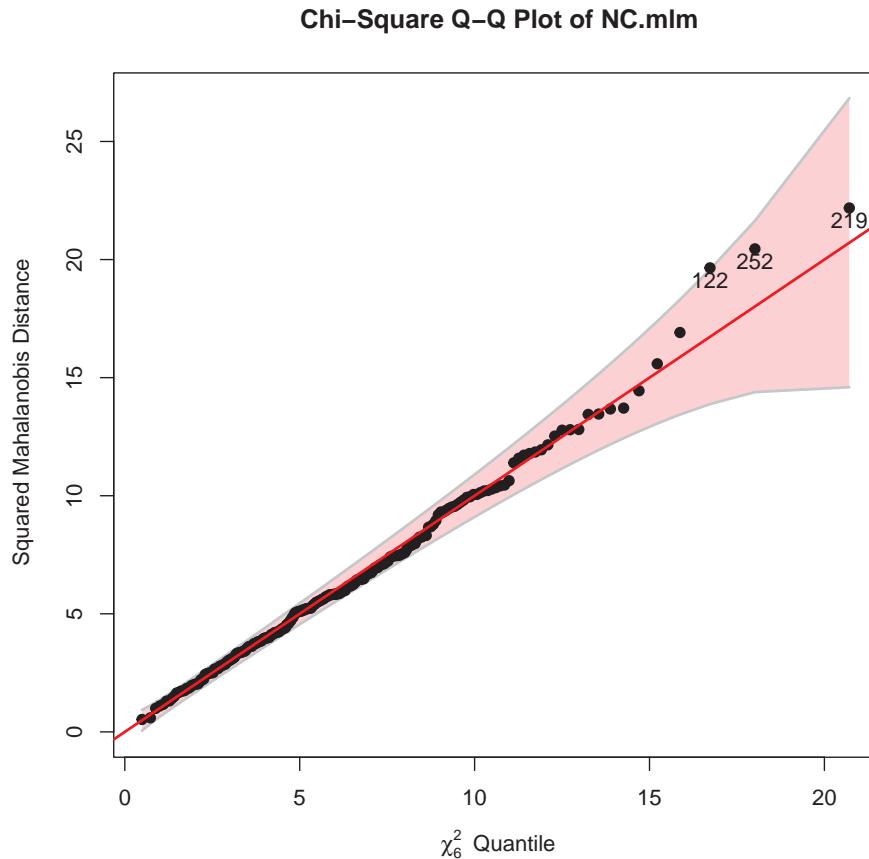


Figure 14.3: Chisquare QQ plot of the MANOVA model

This pattern, of the control group higher than the others, is consistent across all of the response variables, as we see from a plot of `pairs(NC.mlm)`:

```
pairs(NC.mlm,
      fill=TRUE, fill.alpha=0.1,
      var.cex=2)
```

It signals that we are likely to see a simpler representation of the data in canonical space.

14.1.2 Canonical space

We can gain further insight, and a simplified plot showing all the response variables by projecting the MANOVA into the canonical space, which is entirely 2-dimensional (because $df_h = 2$). However, the output from `candisc()` shows that 98.5% of the mean differences among groups can be accounted for in just one canonical dimension.

```
NC.can <- candisc(NC.mlm)
NC.can
#>
#> Canonical Discriminant Analysis for Dx:
#>
#>    CanRsq Eigenvalue Difference Percent Cumulative
```

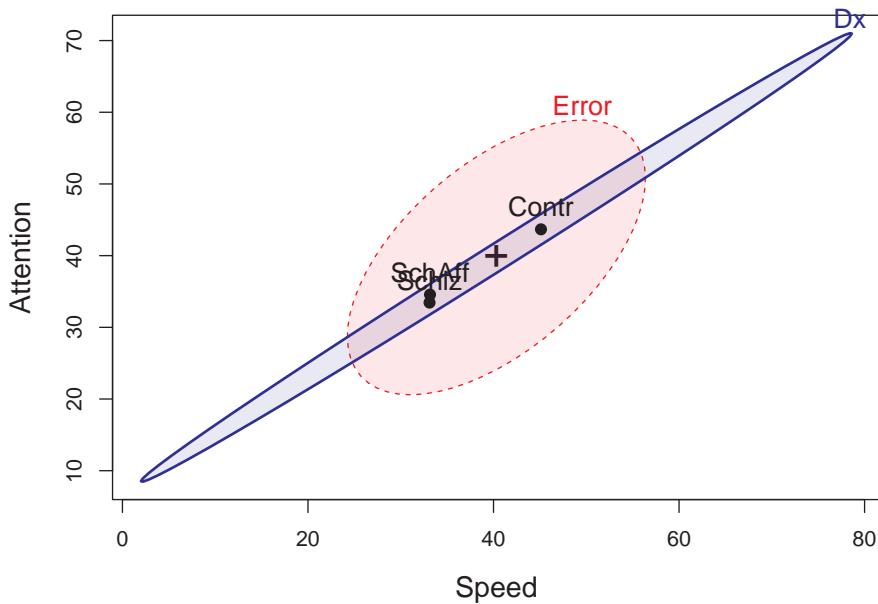


Figure 14.4: HE plot of Speed and Attention in the MLM for the NeuroCog data. The labeled points show the means of the groups on the two variables. The blue H ellipse for groups indicates the strong positive correlation of the group means.

```
#> 1 0.29295     0.41433      0.408     98.5      98.5
#> 2 0.00625     0.00629      0.408      1.5    100.0
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#> LR test stat approx F numDF denDF Pr(> F)
#> 1       0.703     7.53     12    468   9e-13 ***
#> 2       0.994     0.30      5    235     0.91
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 14.6 is the result of the `plot()` method for class "candisc" objects, that is, the result of calling `plot(NC.can, ...)`. It plots the two canonical scores, $Z_{n \times 2}$ for the subjects, together with data ellipses for each of the three groups.

```
pos <- c(4, 1, 4, 4, 1, 3)
col <- c("red", "darkgreen", "blue")
plot(NC.can,
      ellipse=TRUE,
      rev.axes=c(TRUE, FALSE),
      pch=c(7, 9, 10),
      var.cex=1.2, cex.lab=1.5, var.lwd=2, scale=4.5,
      col=col,
      var.col="black", var.pos=pos,
      prefix="Canonical dimension ")
```

The interpretation of Figure 14.6 is again fairly straightforward. As noted earlier (Section 11.7), the projections

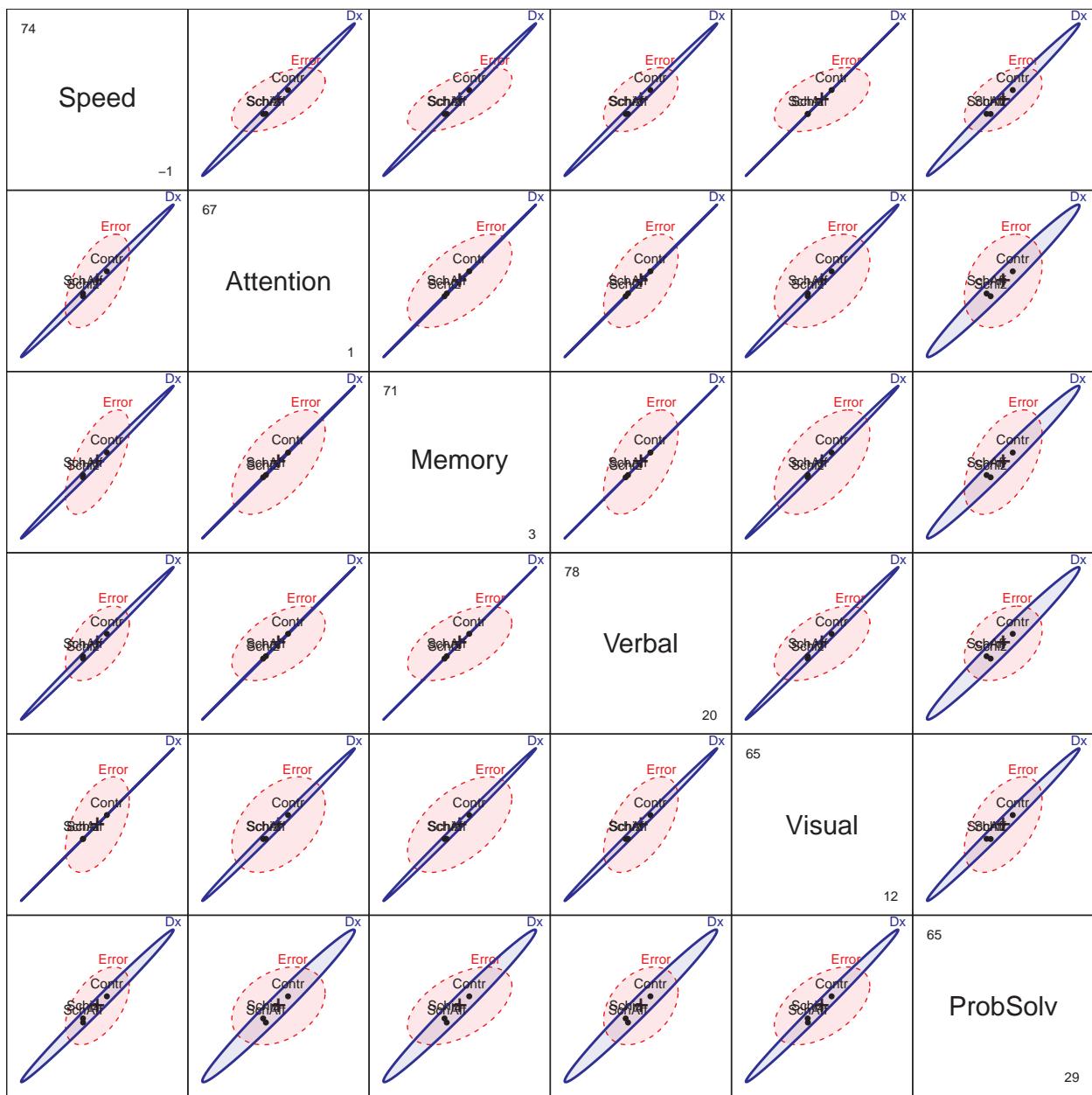


Figure 14.5: HE plot matrix of the MLM for NeuroCog data.

of the variable vectors in this plot on the coordinate axes are proportional to the correlations of the responses with the canonical scores. From this, we see that the normal group differs from the two patient groups, having higher scores on all the neurocognitive variables, most of which are highly correlated. The problem solving measure is slightly different, and this, compared to the cluster of memory, verbal and attention, is what distinguishes the schizophrenic group from the schizoaffectives.

The separation of the groups is essentially one-dimensional, with the control group higher on all measures. Moreover, the variables processing speed and visual memory are the purest measures of this dimension, but all variables contribute positively. The second canonical dimension accounts for only 1.5% of group mean differences and is non-significant (by a likelihood ratio test). Yet, if we were to interpret it, we would note

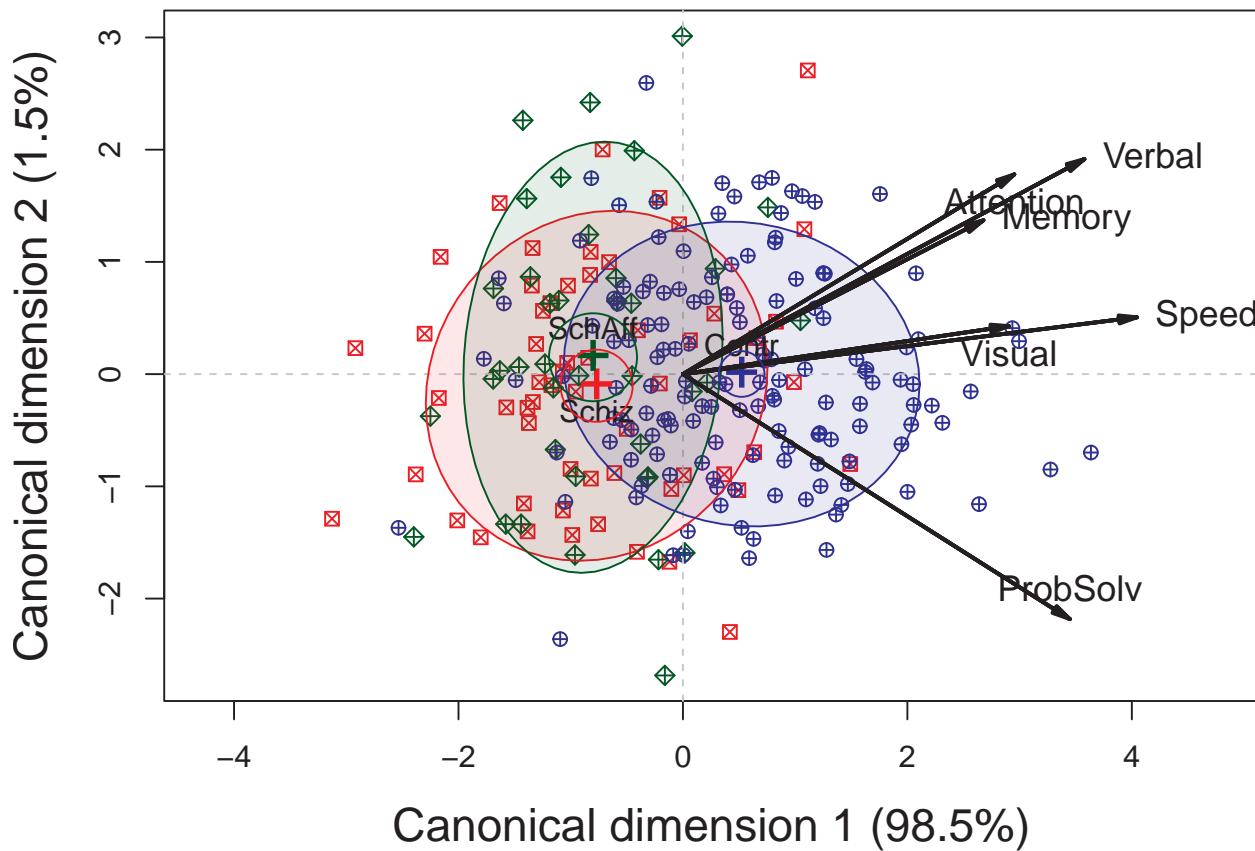


Figure 14.6: Canonical discriminant plot for the NeuroCog data MANOVA. Scores on the two canonical dimensions are plotted, together with 68% data ellipses for each group.

that the schizophrenia group is slightly higher on this dimension, scoring better in problem solving and slightly worse on working memory, attention, and verbal learning tasks.

Summary

This analysis gives a very simple description of the data, in relation to the research questions posed earlier:

- On the basis of these neurocognitive tests, the schizophrenic and schizoaffective groups do not differ significantly overall, but these groups differ greatly from the normal controls.
- All cognitive domains distinguish the groups in the same direction, with the greatest differences shown for the variables most closely aligned with the horizontal axis in Figure 14.6.

14.2 Social cognitive measures

The social cognitive measures were designed to tap various aspects of the perception and cognitive processing of emotions of others. Emotion perception was assessed using a Managing Emotions score from the MCCB. A “theory of mind” (ToM) score assessed ability to read the emotions of others from photographs of the eye region of male and female faces. Two other measures, externalizing bias (**ExtBias**) and personalizing bias (**PersBias**) were calculated from a scale measuring the degree to which individuals attribute internal, personal or situational causal attributions to positive and negative social events.

The analysis of the `SocialCog` data proceeds in a similar way: first we fit the MANOVA model, then test the overall differences among groups using `Anova()`. We find that the overall multivariate test is again significant,

```
data(SocialCog, package="heplots")
SC.mlm <- lm(cbind(MgeEmotions, ToM, ExtBias, PersBias) ~ Dx,
              data=SocialCog)
Anova(SC.mlm)
#>
#> Type II MANOVA Tests: Pillai test statistic
#>   Df test stat approx F num Df den Df Pr(>F)
#>   Dx  2      0.212     3.97      8    268 0.00018 ***
#>   ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Testing the same two contrasts using `linearHypothesis()` (results not shown), we find that the overall multivariate test is again significant, but now *both* contrasts are significant ($Dx1: F(4, 133) = 5.21, p < 0.001$; $Dx2: F(4, 133) = 2.49, p = 0.0461$), the test for $Dx2$ just barely.

```
# control vs. patients
print(linearHypothesis(SC.mlm, "Dx1"), SSP=FALSE)
# Schizo vs. SchizAff
print(linearHypothesis(SC.mlm, "Dx2"), SSP=FALSE)
```

These results are important, because, if they are reliable and make sense substantively, they imply that patients with schizophrenia and schizoaffective diagnoses *can* be distinguished by their performance on tasks assessing social perception and cognition. This was potentially a new finding in the literature on schizophrenia.

As we did above, it is useful to visualize the nature of these differences among groups with HE plots for the `SC.mlm` model. Each contrast has a corresponding **H** ellipse, which we can show in the plot using the `hypotheses` argument. With a single degree of freedom, these degenerate ellipses plot as lines.

```
heplot(SC.mlm,
       hypotheses=list("Dx1"="Dx1", "Dx2"="Dx2"),
       fill=TRUE, fill.alpha=.1,
       cex.lab=1.5, cex=1.2)
```

It can be seen that the three group means are approximately equally spaced on the `ToM` measure, whereas for `MgeEmotions`, the control and schizoaffective groups are quite similar, and both are higher than the schizophrenic group. This ordering of the three groups was somewhat similar for the other responses, as we could see in a `pairs(SC.mlm)` plot.

14.2.1 Model checking

Normally, we would continue this analysis, and consider other HE and canonical discriminant plots to further interpret the results, in particular the relations of the cognitive measures to group differences, or perhaps an analysis of the relationships between the neuro- and social-cognitive measures. We don't pursue this here for reasons of length, but this example actually has a more important lesson to demonstrate.

Before beginning the MANOVA analyses, extensive data screening was done by the client using SPSS, in which all the response *and* predictor variables were checked for univariate normality and multivariate normality

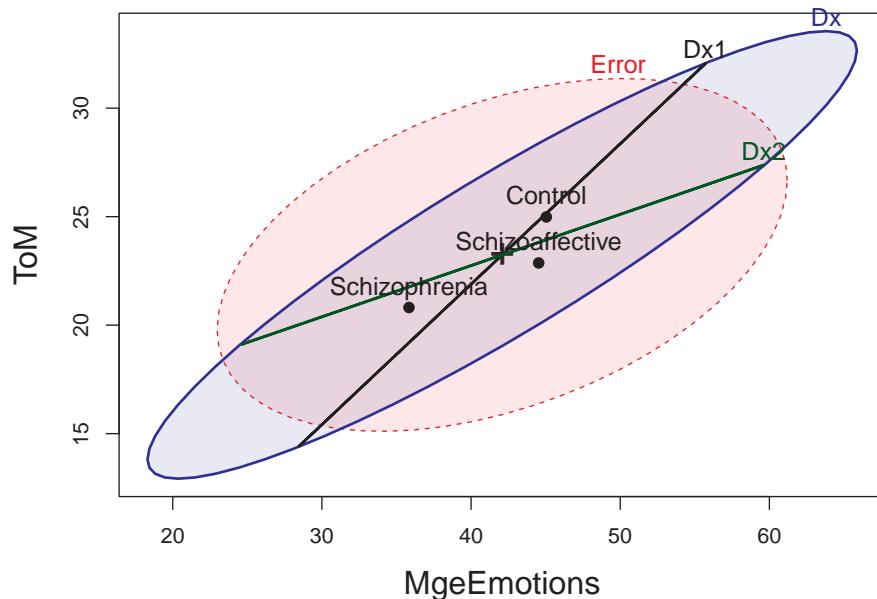


Figure 14.7: HE plot of Speed and Attention in the MLM for the `SocialCog` data. The labeled points show the means of the groups on the two variables. The lines for `Dx1` and `Dx2` show the tests of the contrasts among groups.

(MVN) for both sets. This traditional approach yielded a huge amount of tabular output and no graphs, and did not indicate any major violation of assumptions.¹

A simple visual test of MVN and the possible presence of multivariate outliers is related to the theory of the data ellipse: Under MVN, the squared Mahalanobis distances $D_M^2(\mathbf{y}) = (\mathbf{y} - \bar{\mathbf{y}})' \mathbf{S}^{-1} (\mathbf{y} - \bar{\mathbf{y}})$ should follow a χ_B^2 distribution. Thus, a quantile-quantile plot of the ordered D_M^2 values vs. corresponding quantiles of the χ_B^2 distribution should approximate a straight line (Cox, 1968; Healy, 1968). Note that this should be applied to the *residuals* from the model – `residuals(SC.mlm)` – and not to the response variables directly.

`heplots::cqplot()` implements this for "mle" objects Calling this function for the model `SC.mlm` produces Figure 14.8. It is immediately apparent that there is one extreme multivariate outlier; three other points are identified, but the remaining observations are nearly within the 95% confidence envelope (using a robust MVE estimate of \mathbf{S}).

```
cqplot(SC.mlm, method="mve",
       id.n=4,
       main="",
       cex.lab=1.25)
```

Further checking revealed that this was a data entry error where one case (15) in the schizophrenia group had a score of -33 recorded on the `ExtBias` measure, whose valid range was (-10, +10). In R, it is very easy to re-fit a model to a subset of observations (rather than modifying the dataset itself) using `update()`. The result of the overall Anova and the test of `Dx1` were unchanged; however, the multivariate test for the most interesting contrast `Dx2` comparing the schizophrenia and schizoaffective groups became non-significant at the $\alpha = 0.05$ level ($F(4, 133) = 2.18, p = 0.0742$).

¹Actually, multivariate normality of the predictors in \mathbf{X} is not required in the MLM. This assumption applies only to the conditional values $\mathbf{Y} | \mathbf{X}$, i.e., that the errors $\epsilon_i' \sim \mathcal{N}_p(\mathbf{0}, \Sigma)$ with constant covariance matrix. Moreover, the widely used MVN test statistics, such as Mardia's (1970) test based on multivariate skewness and kurtosis are known to be quite sensitive to mild departures in kurtosis (Mardia, 1974) which do not threaten the validity of the multivariate tests.

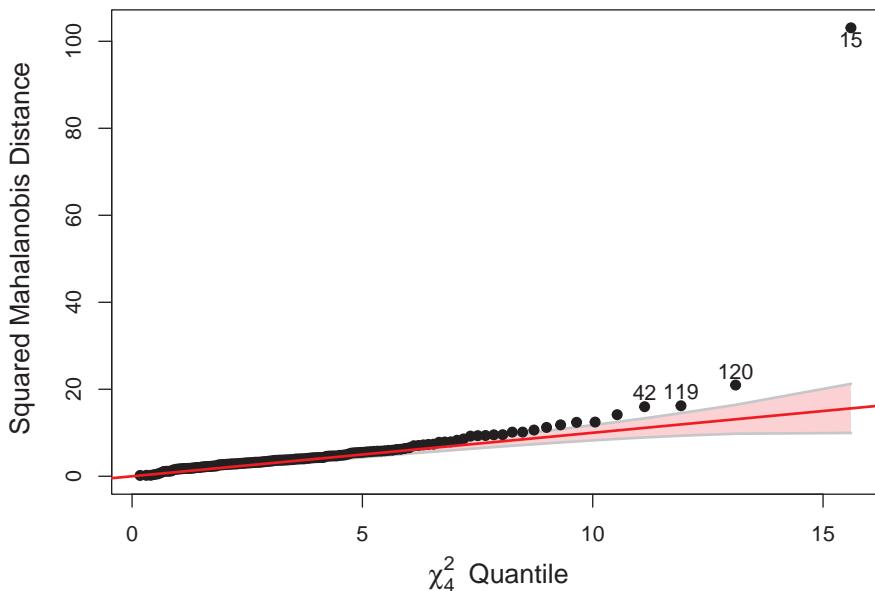


Figure 14.8: Chi-square quantile-quantile plot for residuals from the model `SC.mlm`. The confidence band gives a point-wise 95% envelope, providing information about uncertainty. One extreme multivariate outlier is highlighted.

```
SC.mlm1 <- update(SC.mlm,
                     subset=rownames(SocialCog) != "15")

Anova(SC.mlm1)
print(linearHypothesis(SC.mlm1, "Dx1"), SSP=FALSE)
print(linearHypothesis(SC.mlm1, "Dx2"), SSP=FALSE)
```

14.2.2 Canonical HE plot

This outcome creates a bit of a quandry for further analysis (do univariate follow-up tests? try a robust model?) and reporting (what to claim about the Dx2 contrast?) that we don't explore here. Rather, we proceed to attempt to interpret the MLM with the aid of canonical analysis and a canonical HE plot. The canonical analysis of the model `SC.mlm1` now shows that both canonical dimensions are significant, and account for 83.9% and 16.1% of between group mean differences respectively.

```
SC.can1 <- candisc(SC.mlm1)
SC.can1
#>
## Canonical Discriminant Analysis for Dx:
#>
##   CanRsq Eigenvalue Difference Percent Cumulative
## 1 0.1645      0.1969      0.159     83.9      83.9
## 2 0.0364      0.0378      0.159     16.1     100.0
#>
## Test of H0: The canonical correlations in the
## current row and all that follow are zero
#>
```

```
#>   LR test stat approx F numDF denDF Pr(> F)
#> 1      0.805     3.78     8   264 0.00032 ***
#> 2      0.964     1.68     3   133 0.17537
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
op <- par(mar=c(5,4,1,1)+.1)
heplot(SC.can1,
       fill=TRUE, fill.alpha=.1,
       hypotheses=list("Dx1"="Dx1", "Dx2"="Dx2"),
       lwd = c(1, 2, 3, 3),
       col=c("red", "blue", "darkgreen", "darkgreen"),
       var.lwd=2,
       var.col="black",
       label.pos=c(3,1),
       var.cex=1.2,
       cex=1.25, cex.lab=1.2,
       scale=2.8,
       prefix="Canonical dimension ")
par(op)
```

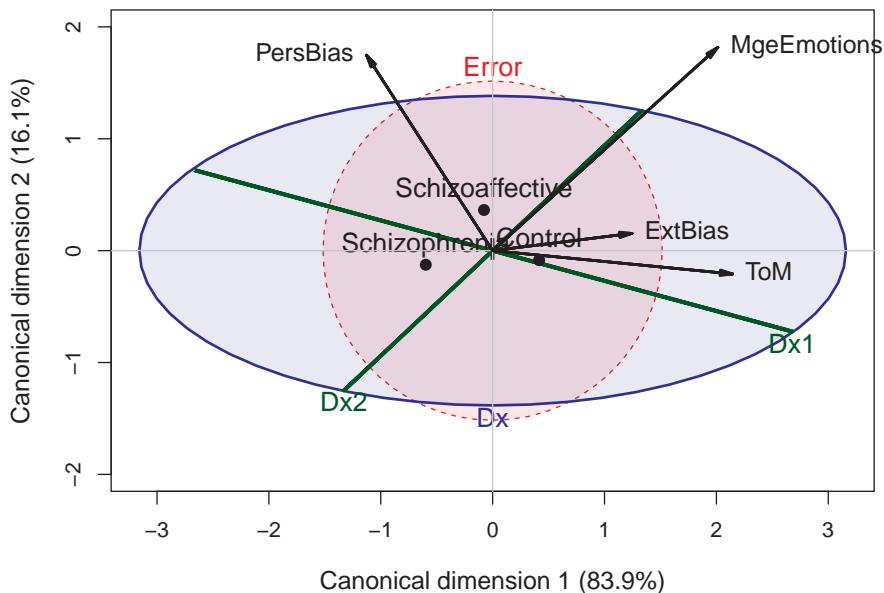


Figure 14.9: Canonical HE plot for the corrected SocialCog MANOVA. The variable vectors show the correlations of the responses with the canonical variables. The embedded green lines show the projections of the **H** ellipses for the contrasts **Dx1** and **Dx2** in canonical space.

The HE plot version of this canonical plot is shown in Figure 14.9. Because the `heplot()` method for a "`candisc`" object refits the original model to the **Z** canonical scores, it is easy to also project other linear hypotheses into this space. Note that in this view, both the **Dx1** and **Dx2** contrasts project outside **E** ellipse.².

This canonical HE plot has a very simple description:

²The direct application of significance tests to canonical scores probably requires some adjustment because these are computed to have the optimal between-group discrimination.

- Dimension 1 orders the groups from control to schizoaffective to schizophrenia, while dimension 2 separates the schizoaffective group from the others;
- Externalizing bias and theory of mind contributes most to the first dimension, while personal bias and managing emotions are more aligned with the second; and,
- The relations of the two contrasts to group differences and to the response variables can be easily read from this plot.

```
#cat("Packages used here:\n")
write_pkgs(file = .pkg_file)
#> 11 packages used here:
#> broom, candisc, car, carData, corrgram, dplyr, ggbiplot, ggplot2, heplots, knitr, tidyr
```


Part V

End matter

Colophon

This book was produced using [R version 4.5.1 \(2025-06-13 ucrt\)](#). Fundamental to this was the framework for reproducible documents provided by Yihui Xie's [knitr](#) package.

[Quarto](#) was used to compile and render the book in HTML and PDF formats. [** Don't really need all this**]

```
Quarto 1.7.29
[>] Checking environment information...
    Quarto cache location: C:\Users\friendly\AppData\Local\quarto
[>] Checking versions of quarto binary dependencies...
    Pandoc version 3.6.3: OK
    Dart Sass version 1.85.1: OK
    Deno version 1.46.3: OK
    Typst version 0.13.0: OK
[>] Checking versions of quarto dependencies.....OK
[>] Checking Quarto installation.....OK
    Version: 1.7.29
    CodePage: 1252
[>] Checking tools.....OK
    TinyTeX: (external install)
    Chromium: (not installed)
[>] Checking LaTeX.....OK
    Using: TinyTeX
    Version: 2025
[>] Checking Chrome Headless.....OK
    Using: Chrome found on system
    Source: Windows Registry
[>] Checking basic markdown render....OK
[>] Checking Python 3 installation....(None)
    Unable to locate an installed version of Python 3.
    Install Python 3 from https://www.python.org/downloads/
[>] Checking R installation.....OK
    Version: 4.5.1
    LibPaths:
        - C:/R/R-4.5.1/library
    knitr: 1.50
    rmarkdown: 2.29
[>] Checking Knitr engine render.....OK
```

Package versions

The principal R package versions used in examples and illustrations are listed below. These were captured via `sessioninfo:::package_info()` from all `library()` commands in the text, and scripts which also updated the references to packages.

At the time of writing, most of these were current on CRAN repositories but some development versions are indicated as “local” in the `source` column.

package	version	date	source
bayestestR	0.16.1	2025-07-01	CRAN
broom	1.0.9	2025-07-28	CRAN
candisc	0.9.2	2025-07-29	local
car	3.1-3	2024-09-27	CRAN
carData	3.0-5	2022-01-06	CRAN
corpcor	1.6.10	2021-09-16	CRAN
correlation	0.8.8	2025-07-08	CRAN
corrgram	1.14	2021-04-29	CRAN
corrplot	0.95	2024-10-14	CRAN
datawizard	1.2.0	2025-07-17	CRAN
dplyr	1.1.4	2023-11-17	CRAN
easystats	0.7.5	2025-07-11	CRAN
effects	4.2-4	2025-07-29	CRAN
effectsize	1.0.1	2025-05-27	CRAN
factoextra	1.0.7	2020-04-01	CRAN
FactoMineR	2.12	2025-07-23	CRAN
forcats	1.0.0	2023-01-29	CRAN
genridge	0.8.0	2024-12-02	CRAN
GGally	2.3.0	2025-07-18	CRAN
gganimate	1.0.10	2025-06-21	CRAN
ggbiplot	0.6.2	2024-01-08	CRAN
ggdensity	1.0.0	2023-02-09	CRAN
ggeffects	2.3.0	2025-06-13	CRAN
gggda	0.1.1	2025-08-05	Github (corybrunson/gggda@df6c7782e93f83c70c1cf43b11d06ffabc6c712a)
ggpcp	0.2.0	2022-11-28	CRAN
ggplot2	3.5.2	2025-04-09	CRAN
ggpubr	0.6.1	2025-06-27	CRAN
ggrepel	0.9.6	2024-09-07	CRAN
ggstats	0.10.0	2025-07-02	CRAN
heplots	1.7.8	2025-08-18	local
Hotelling	1.0-8	2021-09-09	CRAN
imager	1.0.5	2025-08-02	CRAN
insight	1.3.1	2025-06-30	CRAN
knitr	1.50	2025-03-16	CRAN
lubridate	1.9.4	2024-12-08	CRAN
magrittr	2.0.3	2022-03-30	CRAN
marginalEffects	0.28.0	2025-06-25	CRAN
MASS	7.3-65	2025-02-28	CRAN
matlib	1.0.1	2025-07-17	Github (friendly/matlib@d33440a7d98983c36f19f5cdc4d19d3f2ea8a50e)
modelbased	0.12.0	2025-07-10	CRAN
modelsummary	2.4.0	2025-06-08	CRAN
mvinfluence	0.9.3	2025-08-09	local
MVN	6.1	2025-06-10	CRAN
nestedLogit	0.3.2	2023-06-22	CRAN
parameters	0.27.0	2025-07-09	CRAN
patchwork	1.3.1	2025-06-21	CRAN
performance	0.15.0	2025-07-10	CRAN

package	version	date	source
purrr	1.1.0	2025-07-10	CRAN
qgraph	1.9.8	2023-11-03	CRAN
readr	2.1.5	2024-01-10	CRAN
report	0.6.1	2025-02-07	CRAN
Rtsne	0.17	2023-12-07	CRAN
see	0.11.0	2025-03-11	CRAN
stringr	1.5.1	2023-11-14	CRAN
tibble	3.3.0	2025-06-08	CRAN
tidyverse	1.3.1	2024-01-24	CRAN
tourr	2.0.0	2023-02-22	CRAN
vcd	1.4-13	2024-09-16	CRAN
VisCollin	0.1.2	2023-09-05	CRAN

References

- Abbott, E. A. (1884). *Flatland: A romance of many dimensions*. Buccaneer Books.
- Adler, D., & Murdoch, D. (2023). *Rgl: 3D visualization using OpenGL*. <https://CRAN.R-project.org/package=rgl>
- Aluja, T., Morineau, A., & Sanchez, G. (2018). *Principal component analysis for data science*. <https://pca4ds.github.io/>
- Anderson, E. (1935). The irises of the Gaspé peninsula. *Bulletin of the American Iris Society*, 35, 2–5.
- Anderson, M. J. (2006). Distance-based tests for homogeneity of multivariate dispersions. *Biometrics*, 62(1), 245–253. <https://doi.org/10.1111/j.1541-0420.2005.00440.x>
- Andrews, D. F. (1972). Plots of high dimensional data. *Biometrics*, 28, 123–136.
- Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician*, 27, 17–21.
- Arel-Bundock, V. (2025a). *Marginaleffects: Predictions, comparisons, slopes, marginal means, and hypothesis tests*. <https://marginaleffects.com/>
- Arel-Bundock, V. (2025b). *Modelsummary: Summary tables and plots for statistical models and data: Beautiful, customizable, and publication-ready*. <https://modelsummary.com>
- Asimov, D. (1985). Grand tour. *SIAM Journal of Scientific and Statistical Computing*, 6(1), 128–143.
- Barab'asi, A.-L. (2016). *Network science*. Cambridge University Press.
- Barrett, B. E. (2003). Understanding influence in multivariate regression. *Communications in Statistics - Theory and Methods*, 32(3), 667–680. <https://doi.org/10.1081/STA-120018557>
- Barrett, B. E., & Ling, R. F. (1992). General classes of influence measures for multivariate regression. *Journal of the American Statistical Association*, 87(417), 184–191. <https://www.jstor.org/stable/i314301>
- Bartlett, M. S. (1937). Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A*, 160(901), 268–282. <https://doi.org/10.2307/96803>
- Bartlett, M. S. (1938). Further aspects of the theory of multiple regression. *Mathematical Proceedings of the Cambridge Philosophical Society*, 34(1), 33–40. <https://doi.org/10.1017/s0305004100019897>
- Bashaw, W. L., & Findley, W. G. (Eds.). (1967). *Symposium on general linear model approach to the analysis of experimental data in educational research, final report*. <https://files.eric.ed.gov/fulltext/ED026737.pdf>
- Becker, R. A., Cleveland, W. S., & Shyu, M.-J. (1996). The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2), 123–155.
- Belsley, D. A. (1991). *Conditioning diagnostics: Collinearity and weak data in regression*. Wiley.
- Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley; Sons.
- Biecek, P., Baniecki, H., Krzyzinski, M., & Cook, D. (2023). *Performance is not enough: A story of the rashomon's quartet*. <https://arxiv.org/abs/2302.13356>
- Black, C., Southwell, C., Emmerson, L., Lunn, D., & Hart, T. (2018). Time-lapse imagery of adélie penguins reveals differential winter strategies and breeding site occupation. *PLOS ONE*, 13(3), e0193532. <https://doi.org/10.1371/journal.pone.0193532>
- Blishen, B., Carroll, W., & Moore, C. (1987). The 1981 socioeconomic index for occupations in canada. *Canadian Review of Sociology/Revue Canadienne de Sociologie*, 24(4), 465–488. <https://doi.org/10.1111/j.1755-618x.1987.tb00639.x>
- Bock, R. D. (1963). Programming univariate and multivariate analysis of variance. *Technometrics*, 5(1), 95–117. <https://doi.org/10.1080/00401706.1963.10490061>
- Bock, R. D. (1964). A computer program forunivariate and multivariate analysis of variance. *Proceedings of Scientific Symposium on Statistics*.
- Bodmer, W., Bailey, R. A., Charlesworth, B., Eyre-Walker, A., Farewell, V., Mead, A., & Senn, S. (2021).

- The outstanding scientist, r.a. Fisher: His views on eugenics and race. *Heredity*, 126(4), 565–576. <https://doi.org/10.1038/s41437-020-00394-6>
- Bondy, J. A., & Murty, U. S. R. (2008). *Graph theory*. Springer.
- Borg, I., & Groenen, P. J. F. (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.
- Borg, I., Groenen, P. J. F., & Mair, P. (2018). Applied multidimensional scaling and unfolding. In *SpringerBriefs in Statistics*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-73471-2>
- Box, G. E. P. (1949). A general distribution theory for a class of likelihood criteria. *Biometrika*, 36(3-4), 317–346. <https://doi.org/10.1093/biomet/36.3-4.317>
- Box, G. E. P. (1950). Problems in the analysis of growth and wear curves. *Biometrics*, 6, 362–389.
- Box, G. E. P. (1953). Non-normality and tests on variances. *Biometrika*, 40(3/4), 318–335. <https://doi.org/10.2307/2333350>
- Brown, M. B., & Forsythe, A. B. (1974). Robust tests for equality of variances. *Journal of the American Statistical Association*, 69(346), 364–367. <https://doi.org/10.1080/01621459.1974.10482955>
- Brown, P. J., & Zidek, J. V. (1980). Adaptive multivariate ridge regression. *The Annals of Statistics*, 8(1), 64–74. <http://www.jstor.org/stable/2240743>
- Brunson, J. C., & Gracey, J. (2025). *Gggda: A ggplot2 extension for geometric data analysis*. <https://github.com/corybrunson/gggda>
- Buja, A., Cook, D., Asimov, D., & Hurley, C. (2005). Computational methods for high-dimensional rotations in data visualization. In J. S. CR Rao EJ Wegman (Ed.), *Handbook of statistics* (pp. 391–413). Elsevier. [https://doi.org/10.1016/s0169-7161\(04\)24014-7](https://doi.org/10.1016/s0169-7161(04)24014-7)
- cagne, 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. <http://historical.library.cornell.edu/cgi-bin/cul.math/docviewer?did=00620001&seq=3>
- Cai, T. T., Liang, T., & Zhou, H. H. (2015). Law of log determinant of sample covariance matrix and optimal estimation of differential entropy for high-dimensional gaussian distributions. *Journal of Multivariate Analysis*, 137, 161–172. [https://doi.org/https://doi.org/10.1016/j.jmva.2015.02.003](https://doi.org/10.1016/j.jmva.2015.02.003)
- Caiani, F. (1926). Origins of fourth dimension concepts. *The American Mathematical Monthly*, 33(8), 397–406. <https://doi.org/10.1080/00029890.1926.11986607>
- Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2), 245–276. https://doi.org/10.1207/s15327906mbr0102_10
- Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. (1983). *Graphical methods for data analysis*. Wadsworth.
- Chambers, J. M., & Hastie, T. J. (1991). *Statistical models in s* (p. 624). Chapman & Hall/CRC.
- Charnes, A., Cooper, W. W., & Rhodes, E. (1981). Evaluating program and managerial efficiency: An application of data envelopment analysis to program follow through. *Management Science*, 27(6), 668–697. <http://www.jstor.org/stable/2631155>
- Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74, 829–836.
- Cleveland, W. S. (1985). *The elements of graphing data*. Wadsworth Advanced Books.
- Cleveland, W. S., & Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83, 596–610.
- Cleveland, W. S., & McGill, R. (1984). Graphical perception: Theory, experimentation and application to the development of graphical methods. *Journal of the American Statistical Association*, 79, 531–554.
- Cleveland, W. S., & McGill, R. (1985). Graphical perception and graphical methods for analyzing scientific data. *Science*, 229, 828–833.
- Clyde, D. J., Cramer, E. M., & Sherin, R. J. (1966). *Multivariate statistical programs*. Biometric Laboratory, University of Miami.
- Cochran, W. G. (1941). The distribution of the largest of a set of estimated variances as a fraction of their total. *Annals of Eugenics*, 11(1), 47–52. <https://doi.org/10.1111/j.1469-1809.1941.tb02271.x>
- Conover, W. J., Johnson, M. E., & Johnson, M. M. (1981). A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data. *Technometrics*, 23(4), 351–361. <https://doi.org/10.1080/00401706.1981.10487680>

- Cook, D., Buja, A., Cabrera, J., & Hurley, C. (1995). Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3), 155. <https://doi.org/10.2307/1390844>
- Cook, D., Buja, A., Lee, E.-K., & Wickham, H. (2008). Grand tours, projection pursuit guided tours, and manual controls. In *Handbook of data visualization* (pp. 295–314). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-33037-0_13
- Cook, D., & Laa, U. (2024). *Interactively exploring high-dimensional data and models in R*. Online. https://dicook.github.io/mulgar_book/
- Cook, D., & Swayne, D. F. (2007). *Interactive and dynamic graphics for data analysis : With R and GGobi*. Springer. <http://www.ggobi.org/book/>
- Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, 19(1), 15–18. <http://links.jstor.org/sici?&sici=0040-1706%28197702%2919%3A1%3C15%3ADOIOIL%3E2.0.CO%3B2-8>
- Cook, R. D. (1993). Exploring partial residual plots. *Technometrics*, 35(4), 351–362.
- Cook, R. D. (1996). Added-variable plots and curvature in linear regression. *Technometrics*, 38(3), 275–278. <https://doi.org/10.1080/00401706.1996.10484507>
- Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression*. Chapman; Hall.
- Cook, R. D., & Weisberg, S. (1994). ARES plots for generalized linear models. *Computational Statistics & Data Analysis*, 17(3), 303–315. [https://doi.org/10.1016/0167-9473\(92\)00075-3](https://doi.org/10.1016/0167-9473(92)00075-3)
- Costantini, G., Epskamp, S., Borsboom, D., Perugini, M., Möttus, R., Waldorp, L. J., & Cramer, A. O. J. (2015). State of the aRt personality research: A tutorial on network analysis of personality data in R. *Journal of Research in Personality*, 54, 13–29. <https://doi.org/10.1016/j.jrp.2014.07.003>
- Cotton, R. (2013). *Learning R*. O'Reilly Media.
- Cox, D. R. (1968). Notes on some aspects of regression analysis. *Journal of the Royal Statistical Society Series A*, 131, 265–279.
- Csárdi, G., Nepusz, T., Traag, V., Horvát, S., Zanini, F., Noom, D., & Müller, K. (2024). *igraph: Network analysis and visualization in r*. <https://doi.org/10.5281/zenodo.7682609>
- Curran, J., & Hersh, T. (2021). *Hotelling: Hotelling's t^2 test and variants*. <https://CRAN.R-project.org/package=Hotelling>
- Davies, R., Locke, S., & D'Agostino McGowan, L. (2022). *datasauRus: Datasets from the datasaurus dozen*. <https://CRAN.R-project.org/package=datasauRus>
- Davis, C. (1990). Body image and weight preoccupation: A comparison between exercising and non-exercising women. *Appetite*, 16(1), 84. [https://doi.org/10.1016/0195-6663\(91\)90115-9](https://doi.org/10.1016/0195-6663(91)90115-9)
- Dempster, A. P. (1969). *Elements of continuous multivariate analysis*. Addison-Wesley.
- Dempster, A. P. (1972). Covariance selection. *Biometrics*, 28(1), 157–175.
- Dixon, W. J. (1965). *BMD biomedical computer programs*. Health Sciences Computing Facility, School of Medicine, University of California; Health Sciences Computing Faculty.
- Dray, S., & Siberchicot, A. (2025). *Adegraphics: An S4 lattice-based package for the representation of multivariate data*. <http://pbil.univ-lyon1.fr/ADE-4/>
- Duncan, O. D. (1961). A socioeconomic index for all occupations. In Jr. A. J. Reiss, P. K. H. O. D. Duncan, & C. C. North (Eds.), *Occupations and social status*. The Free Press.
- Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32(2), 407–499.
- Emerson, J. W., Green, W. A., Schloerke, B., Crowley, J., Cook, D., Hofmann, H., & Wickham, H. (2013). The generalized pairs plot. *Journal of Computational and Graphical Statistics*, 22(1), 79–91. <http://www.tandfonline.com/doi/ref/10.1080/10618600.2012.694762>
- Euler, L. (1758). Elementa doctrinae solidorum. *Novi Commentarii Academiae Scientiarum Petropolitanae*, 4, 109–140. <https://scholarlycommons.pacific.edu/euler-works/230/>
- Farquhar, A. B., & Farquhar, H. (1891). *Economic and industrial delusions: A discourse of the case for protection*. Putnam.
- Fienberg, S. E. (1971). Randomization and social affairs: The 1970 draft lottery. *Science*, 171, 255–261.
- Finn, J. D. (1967). *MULTIVARIANCE: Fortran program for univariate and multivariate analysis of variance and covariance*. School of Education, State University of New York at Buffalo.
- Fisher, R. A. (1923). Studies in crop variation. II. The manurial response of different potato varieties. *The Journal of Agricultural Science*, 13(2), 311–320. <https://hdl.handle.net/2440/15179>

- Fisher, R. A. (1925b). *Statistical methods for research workers*. Oliver & Boyd.
- Fisher, R. A. (1925a). *Statistical methods for research workers* (6th ed.). Oliver & Boyd.
- Fisher, R. A. (1936). 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>
- Fishkeller, M. A., Friedman, J. H., & Tukey, J. W. (1974). PRIM-9, an interactive multidimensional data display and analysis system. *Proceedings of the Pacific ACM Regional Conference*.
- Flury, B., & Riedwyl, H. (1988). *Multivariate statistics: A practical approach*. Chapman & Hall.
- Fox, J. (1987). Effect displays for generalized linear models. In C. C. Clogg (Ed.), *Sociological methodology, 1987* (pp. 347–361). Jossey-Bass.
- Fox, J. (2003). Effect displays in R for generalized linear models. *Journal of Statistical Software*, 8(15), 1–27.
- Fox, J. (2016). *Applied regression analysis and generalized linear models* (Third edition.). SAGE.
- Fox, J. (2020). *Regression diagnostics* (2nd ed.). SAGE Publications, Inc. <https://doi.org/10.4135/9781071878651>
- Fox, J. (2021). *A mathematical primer for social statistics* (2nd ed.). SAGE Publications, Inc. <https://doi.org/10.4135/9781071878835>
- Fox, J., & Monette, G. (1992). Generalized collinearity diagnostics. *Journal of the American Statistical Association*, 87(417), 178–183.
- Fox, J., & Weisberg, S. (2018a). *An R companion to applied regression* (Third). SAGE Publications. <https://books.google.ca/books?id=uPNrDwAAQBAJ>
- Fox, J., & Weisberg, S. (2018b). Visualizing fit and lack of fit in complex regression models with predictor effect plots and partial residuals. *Journal of Statistical Software*, 87(9). <https://doi.org/10.18637/jss.v087.i09>
- Fox, J., Weisberg, S., & Price, B. (2023). *Car: Companion to applied regression*. <https://CRAN.R-project.org/package=car>
- Fox, J., Weisberg, S., Price, B., Friendly, M., & Hong, J. (2025). *Effects: Effect displays for linear, generalized linear, and other models*. <https://cran.r-project.org/package=effects>
- Friedman, J., Hastie, T., Tibshirani, R., Narasimhan, B., Tay, K., Simon, N., & Yang, J. (2025). *Glmnet: Lasso and elastic-net regularized generalized linear models*. <https://glmnet.stanford.edu>
- Friendly, M. (1991). *SAS System for statistical graphics* (1st ed.). SAS Institute. http://www.sas.com/service/doc/pubcat/uspubcat/ind_files/56143.html
- Friendly, M. (1994). Mosaic displays for multi-way contingency tables. *Journal of the American Statistical Association*, 89, 190–200. <http://www.jstor.org/stable/2291215>
- Friendly, M. (1999). Extending mosaic displays: Marginal, conditional, and partial views of categorical data. *Journal of Computational and Graphical Statistics*, 8(3), 373–395. <http://datavis.ca/papers/drew/drew.pdf>
- Friendly, M. (2002). Corrrgrams: Exploratory displays for correlation matrices. *The American Statistician*, 56(4), 316–324. <https://doi.org/10.1198/000313002533>
- Friendly, M. (2007). HE plots for multivariate general linear models. *Journal of Computational and Graphical Statistics*, 16(2), 421–444. <https://doi.org/10.1198/106186007X208407>
- Friendly, M. (2008). The Golden Age of statistical graphics. *Statistical Science*, 23(4), 502–535. <https://doi.org/10.1214/08-STS268>
- Friendly, M. (2011). *Generalized ridge trace plots: Visualizing bias and precision with the genridge R package*. SCS Seminar.
- Friendly, M. (2013). The generalized ridge trace plot: Visualizing bias and precision. *Journal of Computational and Graphical Statistics*, 22(1), 50–68. <https://doi.org/10.1080/10618600.2012.681237>
- Friendly, M. (2022). The life and works of André-Michel Guerry, revisited. *Sociological Spectrum*, 42(4-6), 233–259. <https://doi.org/10.1080/02732173.2022.2078450>
- Friendly, M. (2024). *Genridge: Generalized ridge trace plots for ridge regression*. <https://github.com/friendly/genridge>
- Friendly, M. (2025a). *Mvinfluence: Influence measures and diagnostic plots for multivariate linear models*. <https://github.com/friendly/mvinfluence>
- Friendly, M. (2025b). *vcdExtra: Vcd extensions and additions*. <https://friendly.github.io/vcdExtra/>
- Friendly, M., & Fox, J. (2025). *Candisc: Visualizing generalized canonical discriminant and canonical correlation analysis*. <https://github.com/friendly/candisc/>

- Friendly, M., Fox, J., & Chalmers, P. (2024). *Matlib: Matrix functions for teaching and learning linear algebra and multivariate statistics*. <https://github.com/friendly/matlib>
- Friendly, M., & Kwan, E. (2003). Effect ordering for data displays. *Computational Statistics and Data Analysis*, 43(4), 509–539. [https://doi.org/10.1016/S0167-9473\(02\)00290-6](https://doi.org/10.1016/S0167-9473(02)00290-6)
- Friendly, M., & Kwan, E. (2009). Where's Waldo: Visualizing collinearity diagnostics. *The American Statistician*, 63(1), 56–65. <https://doi.org/10.1198/tast.2009.0012>
- Friendly, M., & Meyer, D. (2016). *Discrete data analysis with R: Visualization and modeling techniques for categorical and count data*. Chapman & Hall/CRC.
- Friendly, M., Monette, G., & Fox, J. (2013). Elliptical insights: Understanding statistical methods through elliptical geometry. *Statistical Science*, 28(1), 1–39. <https://doi.org/10.1214/12-STS402>
- Friendly, M., & Sigal, M. (2018). Visualizing tests for equality of covariance matrices. *The American Statistician*, 72(4), 144–155. <https://doi.org/10.1080/00031305.2018.1497537>
- Friendly, M., & Wainer, H. (2021). *A history of data visualization and graphic communication*. Harvard University Press. <https://doi.org/10.4159/9780674259034>
- Fuller, W. (2006). *Measurement error models* (2nd ed.). John Wiley & Sons.
- Funkhouser, H. G. (1937). Historical development of the graphical representation of statistical data. *Osiris*, 3(1), 269–405. <http://tinyurl.com/32ema9>
- Gabriel, K. R. (1971). The biplot graphic display of matrices with application to principal components analysis. *Biometrics*, 58(3), 453–467. <https://doi.org/10.2307/2334381>
- Gabriel, K. R. (1981). Biplot display of multivariate matrices for inspection of data and diagnosis. In V. Barnett (Ed.), *Interpreting multivariate data* (pp. 147–173). John Wiley; Sons.
- Galton, F. (1863). *Meteorographica, or methods of mapping the weather*. Macmillan. <http://www.mugu.com/galton/books/meteorographica/index.htm>
- Galton, F. (1886). Regression towards mediocrity in hereditary stature. *Journal of the Anthropological Institute*, 15, 246–263. <http://www.jstor.org/cgi-bin/jstor/viewitem/09595295/dm995266/99p0374f/0>
- Galton, F. (1889). *Natural inheritance*. Macmillan. <http://galton.org/books/natural-inheritance/pdf/galton-nat-inh-1up-clean.pdf>
- Gannett, H. (1898). *Statistical atlas of the united states, eleventh (1890) census*. U.S. Government Printing Office.
- Gastwirth, J. L., Gel, Y. R., & Miao, W. (2009). The impact of Levene's test of equality of variances on statistical theory and practice. *Statistical Science*, 24(3), 343–360. <https://doi.org/10.1214/09-STS301>
- Gayan De Silva. (2020). *Exploring the world of artificial neural networks -a beginner's overview*. <https://doi.org/10.13140/RG.2.2.14790.14406>
- Gelman, A., Hullman, J., & Kennedy, L. (2023). *Causal quartets: Different ways to attain the same average treatment effect*. http://www.stat.columbia.edu/~gelman/research/unpublished/causal_quartets.pdf
- Gittins, R. (1985). *Canonical analysis: A review with applications in ecology*. Springer-Verlag.
- Goeman, J., Meijer, R., Chaturvedi, N., & Lueder, M. (2022). *Penalized: L1 (lasso and fused lasso) and L2 (ridge) penalized estimation in GLMs and in the cox model*. <https://doi.org/10.32614/CRAN.package.penalized>
- Gorman, K. B., Williams, T. D., & Fraser, W. R. (2014). Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PLoS ONE*, 9(3), e90081. <https://doi.org/10.1371/journal.pone.0090081>
- Gower, J. C., & Hand, D. J. (1996). *Biplots*. Chapman & Hall.
- Gower, J. C., Lubbe, S. G., & Roux, N. J. L. (2011). *Understanding biplots*. Wiley. <http://books.google.ca/books?id=66gQCi5JOKYC>
- Grandjean, M. (2016). A social network analysis of Twitter: Mapping the digital humanities community. *Cogent Arts & Humanities*, 3(1), 1171458. <https://doi.org/10.1080/23311983.2016.1171458>
- Graybill, F. A. (1961). *An introduction to linear statistical models*. McGraw-Hill.
- Greenacre, M. (1984). *Theory and applications of correspondence analysis*. Academic Press.
- Greenacre, M. (2010). *Biplots in practice*. Fundación BBVA. <https://books.google.ca/books?id=dv4LrFP7U/EC>
- Guerry, A.-M. (1833). *Essai sur la statistique morale de la France*. Crochard.

- Hahsler, M., Buchta, C., & Hornik, K. (2024). *Seriation: Infrastructure for ordering objects using seriation*. <https://github.com/mhahsler/seriation>
- Haitovsky, Y. (1987). On multivariate ridge regression. *Biometrika*, 74(3), 563–570. <https://doi.org/10.1093/biomet/74.3.563>
- Harrell, F. E. (2015). *Regression modeling strategies: With applications to linear models, logistic and ordinal regression, and survival analysis*. Springer International Publishing. <https://books.google.ca/books?id=sQ90rgEACAAJ>
- Harrison, P. (2023). Langevitour: Smooth interactive touring of high dimensions, demonstrated with scRNA-seq data. *The R Journal*, 15(2), 206–219. <https://doi.org/10.32614/RJ-2023-046>
- Harrison, P. (2025). *Langevitour: Langevin tour*. <https://logarithmic.net/langevitour/>
- Hart, C., & Wang, E. (2022). *Detourr: Portable and performant tour animations*. <https://CRAN.R-project.org/package=detourr>
- Hartigan, J. A. (1975a). *Clustering algorithms*. John Wiley; Sons.
- Hartigan, J. A. (1975b). Printer graphics for clustering. *Journal of Statistical Computing and Simulation*, 4, 187–213.
- Hartley, H. O. (1950). The use of range in analysis of variance. *Biometrika*, 37(3–4), 271–280. <https://doi.org/10.1093/biomet/37.3-4.271>
- Harwell, M. R., Rubinstein, E. N., Hayes, W. S., & Olds, C. C. (1992). Summarizing monte carlo results in methodological research: The one- and two-factor fixed effects ANOVA cases. *Journal of Educational and Behavioral Statistics*, 17(4), 315–339. <https://doi.org/10.3102/10769986017004315>
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference and prediction* (2nd ed.). Springer. <http://www-stat.stanford.edu/~tibs/ElemStatLearn/>
- Healy, M. J. R. (1968). Multivariate normal plotting. *Journal of the Royal Statistical Society Series C*, 17(2), 157–161.
- Herschel, J. F. W. (1833). On the investigation of the orbits of revolving double stars: Being a supplement to a paper entitled "micrometrical measures of 364 double stars". *Memoirs of the Royal Astronomical Society*, 5, 171–222.
- Hoaglin, D. C., & Welsch, R. E. (1978). The hat matrix in regression and ANOVA. *The American Statistician*, 32(1), 17–22. <https://doi.org/10.1080/00031305.1978.10479237>
- Hocking, R. R. (2013). *Methods and applications of linear models: Regression and the analysis of variance*. Wiley. <https://books.google.ca/books?id=iq2J-1iS6HcC>
- Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12, 55–67.
- Hoerl, A. E., Kennard, R. W., & Baldwin, K. F. (1975). Ridge regression: Some simulations. *Communications in Statistics*, 4(2), 105–123. <https://doi.org/10.1080/03610927508827232>
- Hofmann, H., VanderPlas, S., & Ge, Y. (2022). *Ggpcp: Parallel coordinate plots in the ggplot2 framework*. <https://github.com/heike/ggpcp>
- Hofstadter, D. R. (1979). *Gödel, escher, bach: An eternal golden braid*. Basic Books.
- Højsgaard, S., Edwards, D., & Lauritzen, S. (2012). *Graphical models with R*. Springer Science & Business Media.
- Horst, A., Hill, A., & Gorman, K. (2022). *Palmerpenguins: Palmer archipelago (antarctica) penguin data*. <https://allisonhorst.github.io/palmerpenguins/>
- Hotelling, H. (1931). The generalization of Student's ratio. *The Annals of Mathematical Statistics*, 2(3), 360–378. <https://doi.org/10.1214/aoms/1177732979>
- Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4), 321. <https://doi.org/10.2307/2333955>
- Huang, F. L. (2019). MANOVA: A procedure whose time has passed? *Gifted Child Quarterly*, 64(1), 56–60. <https://doi.org/10.1177/0016986219887200>
- Huberty, C. J., & Morris, J. D. (1989). Multivariate analysis versus multiple univariate analyses. *Psychological Bulletin*, 105(2), 302–308. <https://doi.org/10.1037/0033-2909.105.2.302>
- Husson, F., Josse, J., Le, S., & Mazet, J. (2025). *FactoMineR: Multivariate exploratory data analysis and data mining*. <http://factominer.free.fr>

- Husson, F., Le, S., & Pagès, J. (2017). *Exploratory multivariate analysis by example using r*. Chapman & Hall. <https://doi.org/10.1201/b21874>
- IBM. (1965). *Proceedings of the IBM scientific computing symposium on statistics: Oct 21-23, 1963* (L. Robinson, Ed.). IBM. <https://www.amazon.com/Proceedings-Scientific-Computing-Symposium-Statistics/dp/B000GL5RLU>
- Inselberg, A. (1985). The plane with parallel coordinates. *The Visual Computer*, 1, 69–91.
- Isvoranu, A.-M., Epskamp, S., Waldorp, L. J., & Borsboom, D. (2022). *Network psychometrics with r: A guide for behavioral and social scientists*. Routledge. <https://doi.org/10.4324/9781003111238>
- Johnson, R., & Wichern, D. (1998). *Applied multivariate statistical analysis* (4th ed.). Prentice Hall.
- Kassambara, A., & Mundt, F. (2020). *Factoextra: Extract and visualize the results of multivariate data analyses*. <http://www.sthda.com/english/rpkgs/factoextra>
- Kastellec, J. P., & Leoni, E. L. (2007). Using graphs instead of tables in political science. *Perspectives on Politics*, 5(04), 755–771. <https://doi.org/10.1017/S1537592707072209>
- Korkmaz, S., Goksuluk, D., & Zararsiz, G. (2025). *MVN: Multivariate normality tests*. <https://selcukkorkmaz.github.io/mvn-tutorial/>
- Krijthe, J. (2023). *Rtsne: T-distributed stochastic neighbor embedding using a barnes-hut implementation*. <https://github.com/jkrijthe/Rtsne>
- Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1), 1–27. <https://doi.org/10.1007/bf02289565>
- Kwan, E., Lu, I. R. R., & Friendly, M. (2009). Tableplot: A new tool for assessing precise predictions. *Zeitschrift für Psychologie / Journal of Psychology*, 217(1), 38–48. <https://doi.org/10.1027/0044-3409.217.1.38>
- Larmarange, J. (2025). *Ggstats: Extension to ggplot2 for plotting stats*. <https://larmarange.github.io/ggstats/>
- Larsen, W. A., & McCleary, S. J. (1972). The use of partial residual plots in regression analysis. *Technometrics*, 14, 781–790.
- Lauritzen, S. L. (1996). *Graphical models*. Oxford University Press.
- Lawless, J. F., & Wang, P. (1976). A simulation study of ridge and other regression estimators. *Communications in Statistics*, 5, 307–323.
- Lee, E.-K., & Cook, D. (2009). A projection pursuit index for large p small n data. *Statistics and Computing*, 20(3), 381–392. <https://doi.org/10.1007/s11222-009-9131-1>
- Lee, S. (2021). *Liminal: Multivariate data visualization with tours and embeddings*. <https://CRAN.R-project.org/package=liminal>
- Levene, H. (1960). Robust tests for equality of variances. In I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, & H. B. Mann (Eds.), *Contributions to probability and statistics: Essays in honor of Harold Hotelling* (pp. 278–292). Stanford University Press.
- Lix, J. M., L. M. Keselman, & Keselman, H. J. (1996). Consequences of assumption violations revisited: A quantitative review of alternatives to the one-way analysis of variance F test. *Review of Educational Research*, 66(4), 579–619. <https://doi.org/10.3102/00346543066004579>
- Longley, J. W. (1967). An appraisal of least squares programs for the electronic computer from the point of view of the user. *Journal of the American Statistical Association*, 62, 819–841. <https://doi.org/https://www.tandfonline.com/doi/abs/10.1080/01621459.1967.10500896>
- Lüdecke, D. (2025). *Ggeffects: Create tidy data frames of marginal effects for ggplot from model outputs*. <https://strengejackette.github.io/ggeffects/>
- Lüdecke, D., Ben-Shachar, M. S., Patil, I., Waggoner, P., & Makowski, D. (2021). performance: An R package for assessment, comparison and testing of statistical models. *Journal of Open Source Software*, 6(60), 3139. <https://doi.org/10.21105/joss.03139>
- Lüdecke, D., Ben-Shachar, M. S., Patil, I., Wiernik, B. M., & Makowski, D. (2022). Easystats: Framework for easy statistical modeling, visualization, and reporting. In CRAN. <https://easystats.github.io/easystats/>
- Maaten, L. van der, & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605. <http://www.jmlr.org/papers/v9/vandermaaten08a.html>
- Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3), 519–530. <https://doi.org/http://dx.doi.org/10.2307/2334770>
- Mardia, K. V. (1974). Applications of some measures of multivariate skewness and kurtosis in testing

- normality and robustness studies. *Sankhya: The Indian Journal of Statistics, Series B*, 36(2), 115–128. <http://www.jstor.org/stable/25051892>
- Marquardt, D. W. (1970). Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12, 591–612.
- Martí, R., & Laguna, M. (2003). Heuristics and meta-heuristics for 2-layer straight line crossing minimization. *Discrete Applied Mathematics*, 127(3), 665–678.
- Matejka, J., & Fitzmaurice, G. (2017, May). Same stats, different graphs. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. <https://doi.org/10.1145/3025453.3025912>
- Matloff, N. (2011). *The art of R programming: A tour of statistical software design*. No Starch Press.
- McDonald, G. C. (2009). Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1), 93–100. <https://doi.org/10.1002/wics.14>
- McGowan, L. D., Gerke, T., & Barrett, M. (2023). Causal inference is not just a statistics problem. *Journal of Statistics and Data Science Education*, 1–9. <https://doi.org/10.1080/26939169.2023.2276446>
- Meyer, D., Zeileis, A., Hornik, K., & Friendly, M. (2024). *Vcd: Visualizing categorical data*. <https://doi.org/10.32614/CRAN.package.vcd>
- Meyers, L. S., Gamst, G., & Guarino, A. J. (2006). *Applied multivariate research: Design and interpretation*. SAGE Publications.
- Monette, G. (1990). Geometry of multiple regression and interactive 3-D graphics. In J. Fox & S. Long (Eds.), *Modern methods of data analysis* (pp. 209–256). SAGE Publications.
- O'Brien, P. C. (1992). Robust procedures for testing equality of covariance matrices. *Biometrics*, 48(3), 819–827. <http://www.jstor.org/stable/2532347>
- Oksanen, J., Simpson, G. L., Blanchet, F. G., Kindt, R., Legendre, P., Minchin, P. R., O'Hara, R. B., Solymos, P., Stevens, M. H. H., Szoezs, E., Wagner, H., Barbour, M., Bedward, M., Bolker, B., Borcard, D., Borman, T., Carvalho, G., Chirico, M., De Caceres, M., ... Weedon, J. (2025). *Vegan: Community ecology package*. <https://vegandevs.github.io/vegan/>
- Otto, J., & Kahle, D. (2023). *Ggdensity: Interpretable bivariate density visualization with ggplot2*. <https://jamesotto852.github.io/ggdensity/>
- Pearson, K. (1896). Contributions to the mathematical theory of evolution—III, regression, heredity and panmixia. *Philosophical Transactions of the Royal Society of London*, 187, 253–318.
- Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 6(2), 559–572.
- Pearson, K. (1903). I. Mathematical contributions to the theory of evolution. —XI. On the influence of natural selection on the variability and correlation of organs. *Philosophical Transactions of the Royal Society of London*, 200(321–330), 1–66. <https://doi.org/10.1098/rsta.1903.0001>
- Pedersen, T. L., & Robinson, D. (2025). *Ganimate: A grammar of animated graphics*. <https://ganimate.com>
- Pineo, P. O., & Porter, J. (1967). Occupational prestige in canada*. *Canadian Review of Sociology*, 4(1), 24–40. <https://doi.org/10.1111/j.1755-618X.1967.tb00472.x>
- Pineo, P. O., & Porter, J. (2008). Occupational prestige in canada. *Canadian Review of Sociology*, 4(1), 24–40. <https://doi.org/10.1111/j.1755-618x.1967.tb00472.x>
- Playfair, W. (1786). *Commercial and political atlas: Representing, by copper-plate charts, the progress of the commerce, revenues, expenditure, and debts of england, during the whole of the eighteenth century*. Debrett; Robinson;; Sewell. <http://ucpj.uchicago.edu/Isis/journal/demo/v000n000/000000/000000.fg4.html>
- Playfair, W. (1801). *Statistical breviary; shewing, on a principle entirely new, the resources of every state and kingdom in Europe*. Wallis.
- Reaven, G. M., & Miller, R. G. (1968). Study of the relationship between glucose and insulin responses to an oral glucose load in man. *Diabetes*, 17(9), 560–569. <https://doi.org/10.2337/diab.17.9.560>
- Reaven, G. M., & Miller, R. G. (1979). An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia*, 16, 17–24.
- Robinaugh, D. J., Hoekstra, R. H. A., Toner, E. R., & Borsboom, D. (2019). The network approach to psychopathology: A review of the literature 2008–2018 and an agenda for future research. *Psychological Medicine*, 50(3), 353–366. <https://doi.org/10.1017/s0033291719003404>
- Rogan, J. C., & Keselman, H. J. (1977). Is the ANOVA f-test robust to variance heterogeneity when sample

- sizes are equal?: An investigation via a coefficient of variation. *American Educational Research Journal*, 14(4), 493–498. <https://doi.org/10.3102/00028312014004493>
- Rousseeuw, Peter J., Ruits, I., & Tukey, J. W. (1999). The bagplot: A bivariate boxplot. *The American Statistician*, 53(4), 382–387.
- Rousseeuw, Peter J., Van Aelst, S., Van Driessen, K., & Gulló, J. A. (2004). Robust multivariate regression. *Technometrics*, 46(3), 293–305. <https://doi.org/10.1198/004017004000000329>
- Rousseeuw, Peter J., & Yohai, V. J. (1984). Robust regression by means of S-estimators. In J. Franke, W. Härdle, & D. Martin (Eds.), *Robust and nonlinear time series analysis* (Vol. 26, pp. 256–272). Springer.
- Sarkar, D. (2025). *Lattice: Trellis graphics for r*. <https://lattice.r-forge.r-project.org/>
- Scheffé, H. A. (1960). *The analysis of variance*. Wiley.
- Schloerke, B., Cook, D., Larmarange, J., Briatte, F., Marbach, M., Thoen, E., Elberg, A., & Crowley, J. (2025). *GGally: Extension to ggplot2*. <https://ggobi.github.io/ggally/>
- Scott, D. W. (1992). *Multivariate density estimation: Theory, practice, and visualization*. Wiley.
- Searle, S. R., Speed, F. M., & Milliken, G. A. (1980). Population marginal means in the linear model: An alternative to least squares means. *The American Statistician*, 34(4), 216–221.
- Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3–4), 591–611. <https://doi.org/10.1093/biomet/52.3-4.591>
- Shepard, R. N. (1962a). The analysis of proximities: Multidimensional scaling with an unknown distance function. i. *Psychometrika*, 27(2), 125–140. <https://doi.org/10.1007/bf02289630>
- Shepard, R. N. (1962b). The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3), 219–246. <https://doi.org/10.1007/bf02289621>
- Shepard, R. N., Romney, A. K., Nerlove, S. B., & Board, M. S. S. (1972a). *Multidimensional scaling: theory and applications in the behavioral sciences: Vols. II. Applications*. Seminar Press. <https://books.google.ca/books?id=PpFAAQAAIAAJ>
- Shepard, R. N., Romney, A. K., Nerlove, S. B., & Board, M. S. S. (1972b). *Multidimensional scaling: Theory and applications in the behavioral sciences: Vols. I. Theory*. Seminar Press. <https://books.google.ca/books?id=pJRAAQAAIAAJ>
- Shoben, E. J. (1983). Applications of multidimensional scaling in cognitive psychology. *Applied Psychological Measurement*, 7(4), 473–490. <https://doi.org/10.1177/014662168300700406>
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman & Hall.
- Simpson, E. H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society, Series B*, 30, 238–241.
- 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., Lang, D. T., Buja, A., & Cook, D. (2003). GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4), 423–444. [https://doi.org/10.1016/s0167-9473\(02\)00286-4](https://doi.org/10.1016/s0167-9473(02)00286-4)
- Teator, P. (2011). *R cookbook*. O'Reilly Media.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B: Methodological*, 58, 267–288.
- Tiku, M. L., & Balakrishnan, N. (1984). Testing equality of population variances the robust way. *Communications in Statistics - Theory and Methods*, 13(17), 2143–2159. <https://doi.org/10.1080/03610928408828818>
- Timm, N. H. (1975). *Multivariate analysis with applications in education and psychology*. Wadsworth (Brooks/Cole).
- Torgerson, W. S. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4), 401–419. <https://doi.org/10.1007/bf02288916>
- Tufte, E. R. (1983). *The visual display of quantitative information*. Graphics Press.
- Tukey, J. W. (1959). A quick, compact, two sample test to Duckworth's specifications. *Technometrics*, 1, 31–48. <https://doi.org/10.2307/1266308>
- Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 71–86. <https://doi.org/10.1162/jocn.1991.3.1.71>
- VanderPlas, S., Ge, Y., Unwin, A., & Hofmann, H. (2023). Penguins go parallel: A grammar of graphics

- framework for generalized parallel coordinate plots. *Journal of Computational and Graphical Statistics*, 1–16. <https://doi.org/10.1080/10618600.2023.2195462>
- Velleman, P. F., & Welsh, R. E. (1981). Efficient computing of regression diagnostics. *The American Statistician*, 35(4), 234–242.
- Vinod, H. D. (1978). A survey of ridge regression and related techniques for improvements over ordinary least squares. *The Review of Economics and Statistics*, 60(1), 121–131. <http://www.jstor.org/stable/1924340>
- Vu, V. Q., & Friendly, M. (2024). *Ggbiplot: A grammar of graphics implementation of biplots*. <https://github.com/friendly/ggbiplot>
- Waddell, A., & Oldford, R. W. (2023). *Loon: Interactive statistical data visualization*. <https://CRAN.R-project.org/package=loon>
- Warne, F. T. (2014). A primer on multivariate analysis of variance(MANOVA) for behavioral scientists. *Practical Assessment, Research & Evaluation*, 19(1). <https://scholarworks.umass.edu/pare/vol19/iss1/17/>
- Wegman, E. J. (1990). Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411), 664–675.
- Wei, T., & Simko, V. (2024). *Corrplot: Visualization of a correlation matrix*. <https://github.com/taiyun/corrplot>
- Welch, B. L. (1947). The generalization of "student's" problem when several different population variances are involved. *Biometrika*, 34(1–2), 28–35. <https://doi.org/10.1093/biomet/34.1-2.28>
- West, D. B. (2001). *Introduction to graph theory*. Prentice hall.
- Whittaker, J. (1990). *Graphical models in applied multivariate statistics*. John Wiley; Sons.
- Wickham, H. (2014). *Advanced R*. Chapman and Hall/CRC.
- 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. (2025). *Tourrr: Tour methods for multivariate data visualisation*. <https://github.com/ggobi/tourrr>
- Wickham, H., Cook, D., Hofmann, H., & Buja, A. (2011). Tourrr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2). <https://doi.org/10.18637/jss.v040.i02>
- Wilkinson, G. N., & Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Applied Statistics*, 22(3), 392. <https://doi.org/10.2307/2346786>
- Winer, B. J. (1962). *Statistical principles in experimental design*. McGraw-Hill.
- Wood, S. N. (2006). *Generalized additive models: An introduction with r*. Chapman; Hall/CRC Press.
- Wright, K. (2021). *Corrogram: Plot a correlogram*. <https://kwstat.github.io/corrogram/>
- Xie, Y. (2021). *Animation: A gallery of animations in statistics and utilities to create animations*. <https://yihui.org/animation/>
- Xu, Z., & Oldford, R. W. (2021). *Loon.tour: Tour in 'loon'*. <https://cran.r-project.org/package=loon.tour>
- Yee, T. W. (2015). *Vector generalized linear and additive models: With an implementation in r*. Springer.
- Yee, T. W. (2025). *VGAM: Vector generalized linear and additive models*. <https://CRAN.R-project.org/package=VGAM>
- Yohai, V. J. (1987). High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics*, 15(2), 642–656.
- Zhang, J., & Boos, D. D. (1992). Bootstrap critical values for testing homogeneity of covariance matrices. *Journal of the American Statistical Association*, 87(418), 425–429. <http://www.jstor.org/stable/2290273>

Index

- 1970 draft lottery, 13–16
- AddHealth data, 309
- adegraphics package, 123
- animation package, xii
- anscombe data, 8
- Anscombe’s quartet, 8
- bagplot, 51–52
- banknote data, 281
- bar chart, 7
- bfi data, 94
- biplots
- collinearity, 245–248
- bivariate density plot, 53
- Canadian occupational prestige, 182–184
- candisc package, 352
- canonical correlation, 5
- car package, 30, 41, 177, 182, 397
- codingMatrices package, 169
- collinearity biplots, 245–248
- condition indices, 244
- corrgram, 69–73
- corrgram package, 69
- corrplot package, 69, 70
- data ellipse, 5, 9, 26, 34–50
- datasaurus dozen, 10
- datasauRus package, 10
- datasets, 382, 383
 - AddHealth, 309
 - anscombe, 8
 - banknote, 281
 - bfi, 94
 - Davis, 11
 - Diabetes, 132
 - dogfood, 292, 340
 - Draft1970, 14
 - longley, 255
 - mathscore, 273
 - NLSY, 315, 405
 - Parenting, 303
 - peng, 47
 - penguins, 47
 - pollen, xii
- Prestige, xv, 41, 182
- Rohwer, 331
- Salaries, 26, 101
- schooldata, 321
- schoolsites, 321
- workers, 109
- datasets package, 47
- Davis data, 11
- depth, 51
- Diabetes data, 132
- diagnostic plots, 181–182
- dogfood data, 292, 340
- Draft1970 data, 14
- Duncan occupational prestige, 178–181
- effects package, 203
- factoextra package, 123
- FactoMineR package, 113, 123
- genridge package, 235, 255
- GGally package, 76, 322
- ggridge package, 141
- ggbiplot package, 123
- ggdensity package, 53
- ggeffects package, 203
- gggda package, 51
- ggpcp package, 80
- ggplot2 package, xv, 27, 30
- ggstats package, 191
- glmnet package, 255
- gt package, 7
- HE plot, 5
- heplots package, 318, 339, 351, 378, 391
- high density region, 53
- igraph package, 95
- influence, 10, 12
- ipsatized scores, 237
- lack of fit, 10
- lattice package, xv, 30, 123, 203
- line graph, 7
- loess, 16
- longley data, 255
- loon package, 94

- M-estimators, 409
 Mahalanobis distance, 35
 marginaleffects package, 203
 MASS package, 255, 280
 mathscore data, 273
 matlib package, xv, 111
 metamer package, 10
 modelsummary package, 186
 mvinfluence package, 328, 398, 399
 MVN package, 291, 325
 NLSY data, 315, 405
 non-parametric
 bivariate density plot, 53
 ordering
 factor levels, 183
 variables, 84
 outliers, 10, 52, 248
 packages
 adegraphics, 123
 animation, xii
 candisc, 352
 car, 30, 41, 177, 182, 397
 codingMatrices, 169
 corrgram, 69
 corrplot, 69, 70
 datasauRus, 10
 datasets, 47
 effects, 203
 factoextra, 123
 FactoMineR, 113, 123
 genridge, 235, 255
 GGally, 76, 322
 ggridge, 141
 ggbiplot, 123
 ggdensity, 53
 ggeffects, 203
 gggda, 51
 ggpcp, 80
 ggplot2, xv, 27, 30
 ggstats, 191
 glmnet, 255
 gt, 7
 heplots, 318, 339, 351, 378, 391
 igraph, 95
 lattice, xv, 30, 123, 203
 loon, 94
 marginaleffects, 203
 MASS, 255, 280
 matlib, xv, 111
 metamer, 10
 modelsummary, 186
 mvinfluence, 328, 398, 399
 MVN, 291, 325
 palmerpenguins, 47
 penalized, 255
 rgl, 132
 rgl, xii
 Rtsne, 139
 seriation, 142
 stats, 397
 tinytable, 7
 tourr, 90, 93
 vcd, 74
 vcdExtra, 74
 vegan, 136
 VGAM, 289
 VisCollin, 235
 palmerpenguins package, 47
 parallel coordinate plots, 79–84
 Parenting data, 303
 penalized package, 255
 peng data, 47, 382–383
 penguins data, 47
 pie chart, 7
 pollen data, xii
 precision, 28
 Prestige data, xv, 41, 182
 principal components, 39
 projection, 5
 pure error, 9
 quartet
 Anscombe, 8, 25
 causal, 10
 Rashamon, 10
 regression, 181
 quartets
 dogfood, 340
 regression quartet, 181
 rgl package, xii
 rgl package, 132
 robust estimation, 409–411
 Rohwer data, 331
 Rtsne package, 139
 Salaries data, 26, 101
 scatterplot, 7
 scatterplot matrix, 25, 63–69
 generalized, 74–78
 schooldata data, 321
 schoolsites data, 321
 seriation package, 142
 Simpson’s paradox, 54–57, 197
 smoother, 26–30

loess, 16

non-parametric, 28

stats package, 397

stratifier, 26

Structural equation model, 4

tableplots, 245

tinytable package, 7

tourr package, 90, 93

Tukey, John, 25

uncertainty, 28

vcd package, 74

vcdExtra package, 74

vegan package, 136

VGAM package, 289

VisCollin package, 235

visual thinking, 7

visual thinning, 50, 67–69

workers data, 109