MATH3714 Linear Regression and Robustness

Jochen Voss

University of Leeds, Semester 1, 2022/23

Contents

Preface

From previous modules we know how to fit a regression line through points $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^2$. The underlying model here is described by the equation

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

for all $i \in \{1, 2, \ldots, n\}$, and the aim is to find values for the intercept α and the slope β such that the residuals ε_i are as small as possible. This procedure, called simple linear regression, is illustrated in figure 1.

The variables x_i are called **input**, "features", or sometimes also "explanatory variables" or "independent variables". The variables y_i are called **output** or "response", or sometimes the "dependent variables". The values ε_i are called the **residuals** or errors.

Extending the situation of simple linear regression, here we will consider multiple linear regression, where the response y is allowed to depend on several input variables. The corresponding model is now

$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$

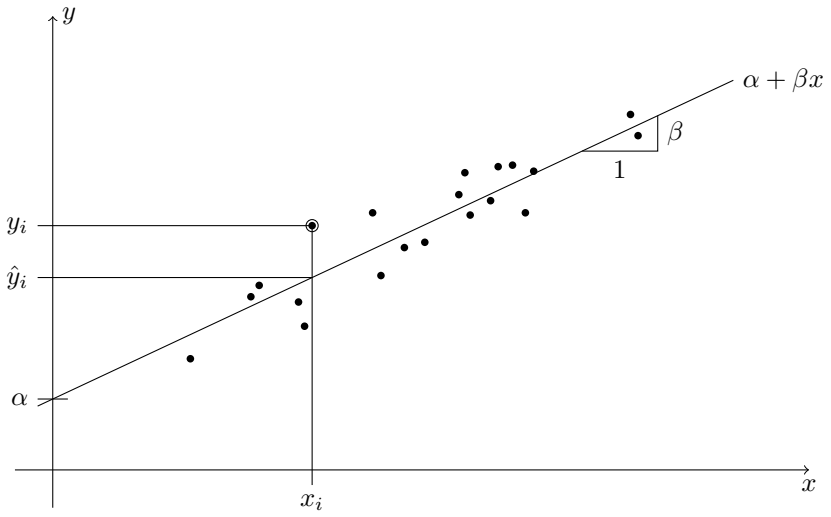


Figure 1: An illustration of linear regression. Each of the black circles in the plot stands for one paired sample (x_i, y_i). The regression line $x \mapsto \alpha + \beta x$, with intercept α and slope β, aims to predict the value of y using the observed value x. For the marked sample (x_i, y_i), the predicted y-value is $\widehat{y}_i$.

for all $i \in \{1, 2, \ldots, n\}$, where n is still the number of observations, and p is now the number of inputs we observe for each sample. Note that for multiple linear regression, we still consider a single response for each sample, only the number of inputs has been increased.

We will discuss multiple linear regression in much detail; our discussion will answer questions like the following:

1. Prediction: given a new input x, how can we predict the corresponding output y? To address this question we need to learn how to estimate the coefficients $\alpha, \beta_1, \ldots, \beta_p$.
2. By studying a fitted regression model, sometimes better understanding of the data can be achieved. For example, one could ask whether all of the p input variables carry information about the response y. We will use statistical hypothesis tests to answer such questions.
3. How to assess model fit?
4. How to deal with outliers in the data?

About MATH3714

This module is **MATH3714 Linear Regression and Robustness**. The module manager and lecturer is Dr Jochen Voss, with email J.Voss@leeds.ac.uk .

Notes and videos

The main way I expect you to learn the material for this course is by reading these notes and by watching the accompanying videos. We will cover two sections of notes each week.

Reading mathematics is a slow process. Each section roughly corresponds to one traditional lecture of 50 minutes. If you find yourself regularly getting through sections in much less than an hour, you're probably not reading carefully enough through each sentence of explanation and each line of mathematics, including understanding the motivation as well as checking the accuracy.

It is possible (but not recommended) to learn the material by only reading the notes and not watching the videos. It is not possible to learn the material by only watching the videos and not reading the notes.

Since we will be relying heavily on these notes, I am even more keen than usual to hear about errors mathematical, typographical or otherwise. Please email me if think you may have found any.

Lectures

The timetable lists three hours of lectures per week for this module. Given these notes and the video lectures, I anticipate that we will not need the full allocated three hours every week. My plan is to start out with two hours per week (Thuesday, 3-4pm and Friday, 1-2pm). We can adjust this arrangement as needed.

I these lectures I will run through the highlights of the notes, and there will plenty of opportunity to give extra examples and to answer any questions you may have. I am very keen to hear about things you'd like to go through in the lectures; please email me with your suggestions.

Workshops and Problem Sheets

There will be 5 problem sheets, corresponding to workshops in weeks 2, 4, 6, 8 and 10. The main goal of the workshops will be to go over your answers to the problems sheets.

My recommended approach to problem sheets and workshops is the following:

- Work through the problem sheet before the workshop, spending plenty of time on it, and making multiple efforts at questions you get stuck on. I recommend spending *at least three hours* on each problem sheet, in more than one block. Collaboration is encouraged when working through the problems, but I recommend writing up your work on your own.
- Take advantage of the workshops to ask for help or clarification on questions you weren't able to complete.
- After the workshop, attempt again the questions you were previously stuck on.
- If you're still unable to complete a question after this second round of attempts, *then* consult the solutions.

Software

For the module we will use the statistical computing package R. This program is free software, and you can find the program and documentation at the R project homepage. In particular, R will be used in the (assessed) practial.

My recommendation would be to install the RStudio environment, which includes R, on your own computer and use this for your work. (Choose the open source version, "RStudio Desktop", on the download page.) Alternatively you can use RStudio or plain R on the university computers.

Assessments

Your final mark for the module will be based on a computer practical (20%) and a final exam (80%). For the practical (I believe it will take place in week 9) you will need to solve some problem using R and the methods you learned in the course and to present your results in a short report.

1 Simple Linear Regression

As a reminder, we consider simple linear regression in this section. My hope is, that all of you have seen this material before at some stage, *e.g.* in school or in some first or second year modules.

In preparation for notation introduced in the next section, we rename the parameters from α and β to the new names β_0 for the intercept and β_1 for the slope.

1.1 Residual Sum of Squares

In simple linear regression, the aim is to find a regression line $y = \beta_0 + \beta_1 x$, such that the line is "close" to given data points $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^2$ for $i \in \{1, 2, \dots, n\}$. The ususal way to find β_0 and β_1, and thus the regression line, is by minimising the **residual sum of squares**:

$$r(\beta_0, \beta_1) = \sum_{i=1}^{n} \big(y_i - (\beta_0 + \beta_1 x_i)\big)^2. \tag{1}$$

For given β_0 and β_1, the value $r(\beta_0, \beta_1)$ measures how close (in vertical direction) the given data points (x_i, y_i) are to the regression line $\beta_0 + \beta_1 x$. By minimising $r(\beta_0, \beta_1)$ we find the regression line which is "closest" to the data. The solution of this minimisation problem is usually expressed in terms of the sample variance s_x and the sample covariance s_{xy}.

Definition 1.1. The **sample covariance** of $x_1, \dots, x_n \in \mathbb{R}$ and $y_1, \dots, y_n \in \mathbb{R}$ is given by

$$\mathrm{s}_{xy} := \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}),$$

where $\bar{x}$ and $\bar{y}$ are the sample means.

The **sample variance** of $x_1, \dots, x_n \in \mathbb{R}$ is given by

$$\mathrm{s}_x^2 := \mathrm{s}_{xx} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2,$$

where, again, $\bar{x}$ is the sample mean of the x_i.

Lemma 1.1. *Assume that $\mathrm{s}_x^2 > 0$. Then the function $r(\beta_0, \beta_1)$ from (1) takes its minimum at the point (β_0, β_1) given by*

$$\hat{\beta}_1 = \frac{\mathrm{s}_{xy}}{\mathrm{s}_x^2}, \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

where $\bar{x}, \bar{y}$ are the sample means, s_{xy} is the sample covariance and s_x^2 is the sample variance.

Proof. We could find the minimum of r by differentiating and setting the derivatives to zero. Here we follow a different approach which uses a "trick" to simplify the algebra: Let $\tilde{x}_i = x_i - \bar{x}$ and $\tilde{y}_i = y_i - \bar{y}$ for all $i \in \{1, \dots, n\}$. Then we have

$$\sum_{i=1}^{n} \tilde{x}_i = \sum_{i=1}^{n} x_i - n\bar{x} = 0$$

and, similarly, $\sum_{i=1}^n \tilde{y}_i = 0$. Using the new coordinates $\tilde{x}_i$ and $\tilde{y}_i$ we find

$$\begin{aligned} r(\beta_0, \beta_1) &= \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 \\ &= \sum_{i=1}^n (\tilde{y}_i + \bar{y} - \beta_0 - \beta_1 \tilde{x}_i - \beta_1 \bar{x})^2 \\ &= \sum_{i=1}^n \Big((\tilde{y}_i - \beta_1 \tilde{x}_i) + (\bar{y} - \beta_0 - \beta_1 \bar{x}) \Big)^2 \\ &= \sum_{i=1}^n (\tilde{y}_i - \beta_1 \tilde{x}_i)^2 + 2(\bar{y} - \beta_0 - \beta_1 \bar{x}) \sum_{i=1}^n (\tilde{y}_i - \beta_1 \tilde{x}_i) + n(\bar{y} - \beta_0 - \beta_1 \bar{x})^2 \end{aligned}$$

Since $\sum_{i=1}^n \tilde{x}_i = \sum_{i=1}^n \tilde{y}_i = 0$, the second term on the right-hand side vanishes and we get

$$r(\beta_0, \beta_1) = \sum_{i=1}^n (\tilde{y}_i - \beta_1 \tilde{x}_i)^2 + n(\bar{y} - \beta_0 - \beta_1 \bar{x})^2. \tag{2}$$

Both of these terms are positive and we can minimise the second term (without changing the first term) by setting $\beta_0 = \bar{y} - \beta_1 \bar{x}$.

To find the value of β_1 which minimises the first term on the right-hand side of (2) we now set the (one-dimensional) derivative w.r.t. β_1 equal to 0. We get the condition

$$\begin{aligned} 0 &\stackrel{!}{=} \frac{d}{d\beta_1} \sum_{i=1}^n (\tilde{y}_i - \beta_1 \tilde{x}_i)^2 \\ &= \sum_{i=1}^n 2(\tilde{y}_i - \beta_1 \tilde{x}_i) \frac{d}{d\beta_1} (\tilde{y}_i - \beta_1 \tilde{x}_i) \\ &= -2 \sum_{i=1}^n (\tilde{y}_i - \beta_1 \tilde{x}_i) \tilde{x}_i \\ &= -2 \sum_{i=1}^n \tilde{x}_i \tilde{y}_i + 2\beta_1 \sum_{i=1}^n \tilde{x}_i^2. \end{aligned}$$

The only solution to this equation is

$$\begin{aligned} \beta_1 &= \frac{\sum_{i=1}^n \tilde{x}_i \tilde{y}_i}{\sum_{i=1}^n \tilde{x}_i^2} \\ &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \\ &= \frac{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} \\ &= \frac{\mathrm{s}_{xy}}{\mathrm{s}_x^2}. \end{aligned}$$

Since the second derivative is $2 \sum_{i=1}^n \tilde{x}_i^2 \geq 0$, this is indeed a minimum and the proof is complete. □

1.2 Linear Regression as a Parameter Estimation Problem

In statistics, any analysis starts by making a statistical model of the data. This is done by writing random variables which have the same structure as the data, and which are chosen so that the data "looks like" a random sample from these random variables.

To construct a model for the data used in a simple linear regression problem, we use random variables $Y_1, \dots, Y_n$ such that

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{3}$$

for all $i \in \{1, 2, \dots, n\}$, where $\varepsilon_1, \dots, \varepsilon_n$ are i.i.d. random variables with $\mathbb{E}(\varepsilon_i) = 0$ and $\mathrm{Var}(\varepsilon_i) = \sigma^2$.

- Here we assume that the x-values are fixed and known. The only random quantities in the model are ε_i and Y_i. (There are more complicated models which also allow for randomness of x, but we won't consider such models here.)
- The random variables ε_i are called **residuals** or **errors**. In a scatter plot, the residuals correspond to the vertical distance between the samples and the regression line. Often one assumes that $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ for all $i \in \{1, 2, \dots, n\}$.
- The values β_0, β_1 and σ^2 are parameters of the model. To fit the model to data, we need to estimate these parameters.

This model is more complex than the models considered in some introductory courses to statistics:

- The data consists now of pairs of numbers, instead of just single numbers.
- We have

  $$\mathbb{E}(Y_i) = \mathbb{E}(\beta_0 + \beta_1 x_i + \varepsilon_i) = \beta_0 + \beta_1 x_i + \mathbb{E}(\varepsilon_i) = \beta_0 + \beta_1 x_i.$$

  Thus, the expectation of Y_i depends on x_i and, at least for $\beta_1 \neq 0$, the random variables Y_i are not identically distributed.

In this setup, we can consider the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ from the previous subsection as statistical parameter estimates for the model parameters β_0 and β_1.

In order to fit a linear model we also need to estimate the residual variance σ^2. This can be done using the estimator

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \frac{1}{n-2} \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2. \tag{4}$$

To understand the form of this estimator, we have to remember that σ^2 is the variance of the ε_i. Thus, using the standard estimator for the variance, we could estimate σ^2 as

$$\sigma^2 \approx \frac{1}{n-1} \sum_{i=1}^{n} \big(\varepsilon_i - \bar{\varepsilon}\big)^2 \approx \frac{1}{n-1} \sum_{i=1}^{n} \big(\hat{\varepsilon}_i - \bar{\hat{\varepsilon}}\big)^2, \tag{5}$$

where $\bar{\varepsilon}$ and $\bar{\hat{\varepsilon}}$ are the averages of the ε_i and the $\hat{\varepsilon}_i$, respectively. One can show that $\bar{\hat{\varepsilon}} = 0$. The estimates of β_0 and β_1 are sensitive to fluctuations in the data, with the effect that the estimated regression line is, on average, slightly closer to the data points than the true regression line would be. This causes the sample variance of the $\hat{\varepsilon}_i$, on average, to be slightly smaller than the true residual variance σ^2 and the thus the estimator (5) is slightly biased. A more detailed analysis reveals that an unbiased estimator can be obtained if one replaces the pre-factor $1/(n-1)$ in equation (5) with $1/(n-2)$. This leads to the estimator (4).

The main advantage gained by considering a statistical model is, that we now can consider how close the estimators $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma}^2$ are to the true values. Results one can obtain include the following:

- The estimators $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma}^2$ are unbiased: This means that when we plug in random data (x_i, Y_i) from the model (3), on average we get the correc answer: $\mathbb{E}(\hat{\beta}_0) = \beta_0$, $\mathbb{E}(\hat{\beta}_1) = \beta_1$, $\mathbb{E}(\hat{\sigma}^2) = \sigma^2$.

- One can ask about the average distance between the estimated parameters $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma}^2$ and the (unknown) true values β_0, β_1 and σ^2. One measure for these distances is the root mean squared error of the estimators.
- One can consider confidence intervals for the parameters β_0, β_1 and σ^2.
- One can consider statistical hypothesis tests to answer yes/no questions about the parameters. For example, one might ask whether the data could have come from the model with $\beta_0 = 0$.
- One can consider whether the data is compatible with the model at all, irrespective of parameter values. If there is a non-linear relationship between x and y, the model (3) will no longer be appropriate.

We will consider most of these questions over the course of the module.

1.3 Matrix Notation

To conclude this section, we will rewrite the results of this section in a form which we will extensively use for multiple linear regression in the rest of this module. The idea here is to arrange all quantities in the problem as matrices and vectors in order to simplify notation. We write

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \in \mathbb{R}^{n\times 2}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n, \qquad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \in \mathbb{R}^n, \qquad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \in \mathbb{R}^2$$

Using this notation, we can rewrite the n equations $y_i = \beta_0 + x_i\beta_1 + \varepsilon_i$ for $i \in \{1, \dots, n\}$ as one vector-valued equation in $\mathbb{R}^n$: we get

$$y = X\beta + \varepsilon,$$

and we want to "solve" this vector-valued equation for β. The sum of squares can now be written as

$$r(\beta) = \sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon^\top \varepsilon = (y - X\beta)^\top (y - X\beta) = y^\top y - 2\beta^\top X^\top y + \beta^\top X^\top X\beta.$$

In the next section we will see that the minimum of r is attained for

$$\hat{\beta} = (XX^\top)^{-1} X^\top y$$

and one can check that the components of this vector $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1)$ coincide with the estimates we obtained above.

Summary

- simple linear regression is the case where there is only one input
- a regression line is fitted by minimising the residual sum of squares
- linear regression is a statistical parameter estimation problem
- the problem can be conveniently written in matrix/vector notation

2 Least Squares Estimates

2.1 Data and Models

For multiple linear regression we assume that there are p inputs and one output. If we have a sample of n obervations, we have np inputs and n outputs in total. Here we denote the ith observation of the jth input by x_{ij} and the corresponding output by y_j.

As an example, we consider the `mtcars` dataset built into R. This is a small dataset, which contains information about 32 automobiles (1973–74 models). The table lists fuel consumption `mpg`, gross horsepower `hp`, and 9 other aspects of these cars. Here we consider `mpg` to be the output, and the other listed aspects to be inputs. Type `help(mtcars)` in R to learn more about this dataset:

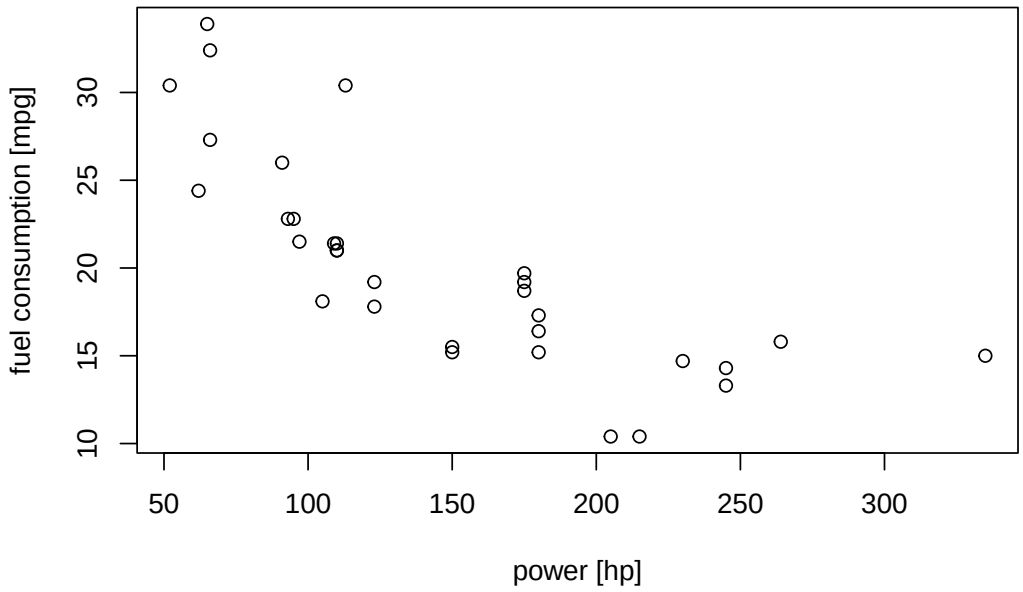
```
mtcars
```

```
#                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
# Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
# Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
# Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
# Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
# Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
# Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
# Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
# Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
# Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
# Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
# Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
# Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
# Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
# Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
# Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
# Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
# Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
# Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
# Honda Civic       30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
# Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
# Toyota Corona     21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
# Dodge Challenger  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
# AMC Javelin       15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
# Camaro Z28        13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
# Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
# Fiat X1-9         27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
# Porsche 914-2     26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
# Lotus Europa      30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
# Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
# Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
# Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
# Volvo 142E        21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

For this dataset we have $n = 32$ (number of cars), and $p = 10$ (number of attributes, excluding `mpg`). The values $y_1, \dots, y_{32}$ are listed in the first column of the table, the values $x_{i,1}$ for $i \in \{1, \dots, 32\}$ are shown in the second column, and the values $x_{i,10}$ are shown in the last column.

It is easy to make scatter plots which show how a single input affects the output. For example, we can show how the engine power affects fuel consumption:

```r
plot(mtcars$hp, mtcars$mpg,
     xlab = "power [hp]", ylab = "fuel consumption [mpg]")
```



We can see that cars with stronger engines tend to use more fuel (*i.e.* a gallon of fuel lasts for fewer miles; the curve goes down), but leaving out the other inputs omits a lot of information. It is not easy to make a plot which takes all inputs into account. Is is also not immediately obvious which of the variables are most important.

In linear regression, we assume that the output depends on the inputs in a linear (or more precisely, *affine*) way. We write this as

$$y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \varepsilon_i \tag{6}$$

where the residuals ε_i are assumed to be "small". Here, i ranges over all n samples, so (6) represents n simultaneous equations.

The parameters β_j can be interpreted as the expected change in the response y per unit change in x_j when all other regressor variables are held fixed. For this reason the parameters β_j (for $j = 1, \ldots, p$) are sometimes called *partial* regression coefficients.

This model describes a hyperplane in the $(p+1)$-dimensional space of the inputs x_j and the output y. The hyperplane is easily visualized when $p = 1$ (as a line in $\mathbb{R}^2$), and visualisation can be attempted for $p = 2$ (as a plane in $\mathbb{R}^3$) but is hard for $p > 2$.

We defer making a proper statistical model for multiple linear regression until the next section.

2.2 The Normal Equations

Similar to what we did in Section 1.3, we rewrite the model using matrix notation. We define the vectors

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n, \qquad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \in \mathbb{R}^n, \qquad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \in \mathbb{R}^{1+p}$$

as well as the matrix

$$X = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,p} \\ 1 & x_{2,1} & \cdots & x_{2,p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,p} \end{pmatrix} \in \mathbb{R}^{n\times(1+p)}.$$

The parameters β_i are called the **regression coefficients** and the matrix X is called the **design matrix**.

Using this notation, the model (6) can be written as

$$y = X\beta + \varepsilon, \tag{7}$$

where again $X\beta$ is a matrix-vector multiplication which "hides" the sums in equation (6), and (7) is an equation of vectors of size n, which combines the n individual equations from (6).

To simplify notation, we index the columns of X by $0, 1, \dots, p$ (instead of the more conventional $1, \dots, p+1$), so that we can for example write

$$(X\beta)_i = \sum_{j=0}^{p} x_{i,j}\beta_j = \beta_0 + \sum_{j=1}^{p} x_{i,j}\beta_j.$$

As before, we find the regression coefficients by minimising the residual sum of squares:

$$\begin{aligned} r(\beta) &= \sum_{i=1}^{n} \varepsilon_i^2 \\ &= \sum_{i=1}^{n} \big(y_i - (\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p})\big)^2. \end{aligned}$$

In practice, this notation turns out to be cumbersome, and we will use matrix notation in the following proof.

Lemma 2.1. *Assume that the matrix $X^\top X \in \mathbb{R}^{(1+p)\times(1+p)}$ is invertible. Then the function $r(\beta)$ takes its minimum at the vector $\widehat{\beta} \in \mathbb{R}^{p+1}$ given by*

$$\widehat{\beta} = (X^\top X)^{-1} X^\top y.$$

Proof. Using the vector equation $\varepsilon = y - X\beta$, we can also write the residual sum of squares as

$$\begin{aligned} r(\beta) &= \sum_{i=1}^{n} \varepsilon_i^2 \\ &= \varepsilon^\top \varepsilon \\ &= (y - X\beta)^\top (y - X\beta) \\ &= y^\top y - y^\top X\beta - (X\beta)^\top y + (X\beta)^\top (X\beta). \end{aligned}$$

Using the linear algebra rules from Appendix A.2 we find that $y^\top X\beta = (X\beta)^\top y = \beta^\top X^\top y$ and $(X\beta)^\top (X\beta) = \beta^\top X^\top X\beta$. Thus we get

$$r(\beta) = y^\top y - 2\beta^\top X^\top y + \beta^\top X^\top X\beta.$$

Note that in this equation X is a matrix, y and β are vectors, and $r(\beta)$ is a number.

To find the minimum of this function, we set all partial derivatives $\frac{\partial}{\partial \beta_i} r(\beta)$ equal to 0. Going through the terms in the formula for $r(\beta)$ we find: (1) $y^\top y$ does not depend on β, so we have $\frac{\partial}{\partial \beta_i} y^\top y = 0$ for all i, (2) we have

$$\frac{\partial}{\partial \beta_i} \beta^\top X^\top y = \frac{\partial}{\partial \beta_i} \sum_{j=1}^{p+1} \beta_j (X^\top y)_j = (X^\top y)_i$$

and (3) finally

$$\frac{\partial}{\partial\beta_i}\beta^\top X^\top X\beta = \frac{\partial}{\partial\beta_i}\sum_{j,k=1}^{p+1}\beta_j(X^\top X)_{j,k}\beta_k = 2\sum_{k=1}^{p+1}(X^\top X)_{i,k}\beta_k = 2\big((X^\top X)\beta\big)_i.$$

(Some care is needed, when checking that the middle equality sign in the previous equation is correct.) Combining these derivatives, we find

$$\frac{\partial}{\partial\beta_i}r(\beta) = 0 - 2(X^\top y)_i + 2\big(X^\top X\beta\big)_i \tag{8}$$

for all $i \in \{0, 1, \dots, p\}$. At a local minimum of r, all of these partial derivatives must be zero and using a vector equation we find that a necessary condition for a minimum is

$$X^\top X\beta = X^\top y. \tag{9}$$

Since we assumed that $X^\top X$ is invertible, there is exactly one vector β which solves (9). This vector is given by

$$\hat{\beta} := (X^\top X)^{-1}X^\top y.$$

As for one-dimensional minimisation, there is a condition on the second derivatives which must be checked to see which local extrema are local minima. Here we are only going to sketch this argument: A sufficient condition for $\hat{\beta}$ to be a minimum is for the second derivative matrix (the Hessian matrix) to be positive definite (see appendix A.2.8). Using equation (8) we find

$$\frac{\partial}{\partial\beta_i\partial\beta_j}r(\beta) = 2(X^\top X)_{i,j}$$

And thus the Hessian matrix is $H = 2X^\top X$. Using results from linear algebra, one can show that this matrix is indeed positive definite and thus $\hat{\beta}$ is the unique minimum of r. □

Equation (9) gives a system of $p + 1$ linear equations with $p + 1$ unknowns. This system of linear equations, $X^\top X\beta = X^\top y$ is called the **normal equations**. If $X^\top X$ is invertible, as assumed in the lemma, this system of equations has $\hat{\beta}$ as its unique solution. Otherwise, there may be more than one β which leads to the same value $r(\beta)$ and the minimum will no longer be unique. This happens for example, if two of the inputs are identical to each other (or, more generally, one input is linearly dependent on one or more other inputs).

The condition that $X^\top X$ must be invertible in lemma 2.1 corresponds to the condition $s_x^2 > 0$ in lemma 1.1.

The value $\hat{\beta}$ found in the lemma is called the **least squares estimator** for β.

2.3 Fitted Values

Let us again consider our model

$$y = X\beta + \varepsilon,$$

using the matrix notation introduced above. Here we can think of $X\beta$ as the **true values**, while ε are the errors. The design matrix X (containing the inputs) and the response y are known to us, while the true coefficients β and the errors ε are unknown. Solving for ε we find that the errors satisfy

$$\varepsilon = y - X\beta.$$

Using the least squares estimate $\hat{\beta}$ we can estimate the true values as

$$\hat{y} = X\hat{\beta}. \tag{10}$$

These estimates are called the **fitted values**. Using the definition of $\hat{\beta}$ we get

$$\hat{y} = X(X^\top X)^{-1}X^\top y =: Hy.$$

The matrix $H = X(X^\top X)^{-1}X^\top$ is commonly called the **hat matrix** (because it "puts the hat on y").

Finally, we can estimate the errors using the residuals

$$\hat{\varepsilon} = y - X\hat{\beta} = y - \hat{y} = y - Hy = (I - H)y, \tag{11}$$

where I is the $n \times n$ identity matrix.

2.4 Example

To conclude this section, we demonstrate how these methods can be used in R. For this we consider the `mtcars` example from the beginning of the section again. I will first show how to do the analysis "by hand", and later show how the same result can be obtained using R's built-in functions.

We first split `mtcars` into the respons column `y` (the first column) and the design matrix `X` (a column of ones, followed by columns 2 to 11 of `mtcars`):

```
y <- mtcars[, 1]
X <- cbind(1, data.matrix(mtcars[, 2:11]))
```

Next we compute $X^\top X$ and solve the normal equations. Often it is faster, easier, and has smaller numerical errors to solve the normal equations rather than inverting the matrix $X^\top X$.

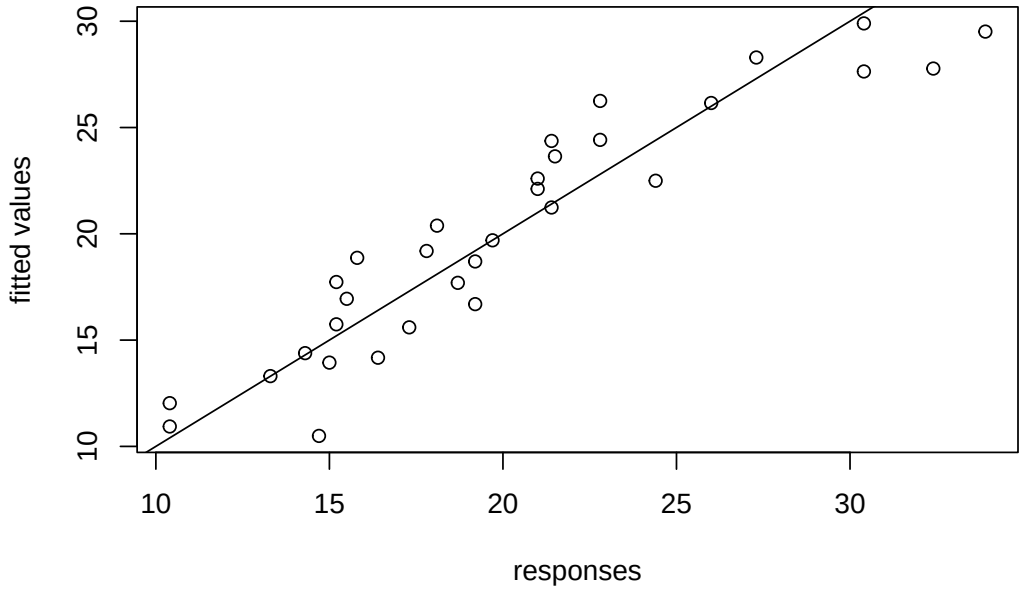
```
XtX <- t(X) %*% X
beta.hat <- solve(XtX, t(X) %*% y)
beta.hat
```

```
#              [,1]
#      12.30337416
# cyl  -0.11144048
# disp  0.01333524
# hp   -0.02148212
# drat  0.78711097
# wt   -3.71530393
# qsec  0.82104075
# vs    0.31776281
# am    2.52022689
# gear  0.65541302
# carb -0.19941925
```

Without further checks it is hard to know whether the result is correct, or whether we made a mistake somewhere along the lines. One good sign is that we argued earlier that higher `hp` should lead to lower `mpg`, and indeed the corresponding coefficient `-0.02148212` is negative.

Finally, compute the fitted values and generate a plot of fitted values against responses. If everything worked, we would expect the points in this plot to be close to the diagonal.

```
y.hat <- X %*% beta.hat
plot(y, y.hat, xlab = "responses", ylab = "fitted values")
abline(a = 0, b = 1) # plot the diagonal
```



For comparison we now re-do the analysis using built-in R commands. In the `lm()` command below, we use `data=mtcars` to tell R where the data is stored, and the formula `mpg ~ .` states that we want to model `mpg` as a function of all other variable (this is the meaning of `.`).

```
m <- lm(mpg ~ ., data = mtcars) # fit a linear model
coef(m) # get the estimated coefficients
```

```
# (Intercept)         cyl        disp          hp        drat          wt
# 12.30337416 -0.11144048  0.01333524 -0.02148212  0.78711097 -3.71530393
#        qsec          vs          am        gear        carb
#  0.82104075  0.31776281  2.52022689  0.65541302 -0.19941925
```

Comparing these coefficients to the vector `beta.hat` from above shows that we got the same result using both methods. The fitted values can be computed using `fitted.values(m)`. Here we just check that we get the same result as above:

```
max(abs(fitted.values(m) - y.hat))
```

```
# [1] 5.329071e-13
```

This result `5.329071e-13` stands for the number $5.329071 \cdot 10^{-13}$, which is extremely small. The difference between our results and R's result is caused by rounding errors.

Summary

- multiple linear regression allows for more than one input but still has only one output
- the least squared estimate for the coefficients is found by minimising the residual sum of squares
- the estimate can be computed as the solution to the normal equations
- the hat matrix transforms responses into fitted values

Interlude: Linear Regression in R

Fitting a Model

The function `lm()` is used to fit a linear model in R. There are different ways to specify the form of the model and the data to be used for fitting the model.

- The most basic way to call `lm()` is the case where the explanatory variables and the response variable are stored as separate vectors. Assuming, for example, that the explanatory variables are `x1`, `x2`, `x3` and that the response variable is `y` in R, we can tell R to fit the linear model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon$ by using the following command:

  ```
  lm(y ~ x1 + x2 + x3)
  ```

  Note that R automatically added the intercept term β_0 to this model. If we want to fit a model without an intercept, *i.e.* the model $y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon$, we have to add `0 +` in front of the explanatory variables:

  ```
  lm(y ~ 0 + x1 + x2 + x3)
  ```

  The general form of a model specification is the response variable, followed by `~`, followed by a plus-separated list of explanatory variables. For this form of calling `lm()`, the variables `y`, `x1`, `x2`, and `x3` in the examples above must be already defined before `lm()` is called. It may be a good idea to double-check that the variables have the correct values before trying to call `lm()`.

- Both for the response and for explanatory variables we can specify arbitrary R expressions to compute the numeric values to be used. For example, to fit the model $\log(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$ (assuming that all y_i are positive) we can use the following command:

  ```
  lm(log(y) ~ x1 + x2)
  ```

  Some care is needed, because `+`, `*` and `^` have a special meaning inside the first argument of `lm()`; any time we want to compute a variable for `lm()` using these operations, we need to surround the corresponding expression with `I()`, to tell R that `+`, `*` or `^` should have their usual, arithmetic meaning. For example, to fit a model of the form $y \sim \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$, we can use the following R command:

  ```
  lm(y ~ x + I(x^2))
  ```

  Here, the use of `I()` tells R that `x^2` is to be interpreted as the vector $(x_1^2, \ldots, x_n^2)$. Similarly, we can fit a model of the form $y = \beta_0 + \beta_1(x_1 + x_2) + \varepsilon$:

  ```
  lm(y ~ I(x1+x2))
  ```

  Here, the use of `I()` tells R that `x1+x2` indicates the vector $(x_{1,1} + x_{2,1}, \ldots, x_{1,n} + x_{2,n})$ instead of two separate explanatory variables.

  Details about how to specify models in calls to `lm()` can be found by using the command `help(formula)` in R.

- If the response and the explanatory variables are stored in the columns of a data frame, we can use the `data=...` argument to `lm()` to specify this data frame and then just use the column names to specify the regression model. For example, the `stackloss` dataset built into R consists of a data frame with columns `Air.Flow`, `Water.Temp`, `Acid.Conc.`, `stack.loss`. To predict `stackloss$stack.loss` from `stackloss$Air.Flow` we can write

  ```
  lm(stack.loss ~ Air.Flow, data=stackloss)
  ```

As a special case, a single dot "." can be used in place of the explanatory variables in the model to indicate that all columns except for the given response should be used. Thus, the following two commands are equivalent:

```
lm(stack.loss ~ ., data=stackloss)
lm(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., data=stackloss)
```

Understanding the Model

The output of the `lm()` function is an R object which can be used the extract information about the fitted model. A good way to work with this object is to store it in a variable and then use commands like the ones listed below to work with this variable. For example, the following R command fits a model for the `stackloss` dataset and stores it in the variable `m`:

```
m <- lm(stack.loss ~ ., data=stackloss)
```

Many operations are available to use with this object `m`:

- Printing `m` to the screen:

  ```
  m
  ```

  ```
  #
  # Call:
  # lm(formula = stack.loss ~ ., data = stackloss)
  #
  # Coefficients:
  # (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
  #    -39.9197       0.7156       1.2953      -0.1521
  ```

  This shows the estimated values for the regression coefficient.

- The command `summary()` can be used to print additional information about the fitted model:

  ```
  summary(m)
  ```

  ```
  #
  # Call:
  # lm(formula = stack.loss ~ ., data = stackloss)
  #
  # Residuals:
  #     Min      1Q  Median      3Q     Max
  # -7.2377 -1.7117 -0.4551  2.3614  5.6978
  #
  # Coefficients:
  #             Estimate Std. Error t value Pr(>|t|)
  # (Intercept) -39.9197    11.8960  -3.356  0.00375 **
  # Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
  # Water.Temp    1.2953     0.3680   3.520  0.00263 **
  # Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
  # ---
  # Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
  #
  # Residual standard error: 3.243 on 17 degrees of freedom
  # Multiple R-squared:  0.9136,  Adjusted R-squared:  0.8983
  # F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
  ```

  We will learn over the course of this module how to interpret all of this output.

- The coefficient vector β can be obtained using `coef(m)`:

```
coef(m)
```

```
# (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
# -39.9196744   0.7156402   1.2952861  -0.1521225
```

- The fitted values $\hat{y}_i$ can be obtained using the command `fitted(m)`:

```
fitted(m)
```

```
#         1         2         3         4         5         6         7         8
# 38.765363 38.917485 32.444467 22.302226 19.711654 21.006940 21.389491 21.389491
#         9        10        11        12        13        14        15        16
# 18.144379 12.732806 11.363703 10.220540 12.428561 12.050499  5.638582  6.094949
#        17        18        19        20        21
#  9.519951  8.455093  9.598257 13.587853 22.237713
```

- The estimated residuals $\hat{\varepsilon}_i = y_i - \hat{y}_i$ can be obtained using the command `resid(m)`:

```
resid(m)
```

```
#           1           2           3           4           5           6
#  3.23463723 -1.91748529  4.55553300  5.69777417 -1.71165358 -3.00693970
#           7           8           9          10          11          12
# -2.38949071 -1.38949071 -3.14437890  1.26719408  2.63629676  2.77946036
#          13          14          15          16          17          18
# -1.42856088 -0.05049929  2.36141836  0.90505080 -1.51995059 -0.45509295
#          19          20          21
# -0.59825656  1.41214728 -7.23771286
```

- The design matrix X can be found using `model.matrix(m)`:

```
model.matrix(m)
```

```
#    (Intercept) Air.Flow Water.Temp Acid.Conc.
# 1            1       80         27         89
# 2            1       80         27         88
# 3            1       75         25         90
# 4            1       62         24         87
# 5            1       62         22         87
# 6            1       62         23         87
# 7            1       62         24         93
# 8            1       62         24         93
# 9            1       58         23         87
# 10           1       58         18         80
# 11           1       58         18         89
# 12           1       58         17         88
# 13           1       58         18         82
# 14           1       58         19         93
# 15           1       50         18         89
# 16           1       50         18         86
# 17           1       50         19         72
# 18           1       50         19         79
# 19           1       50         20         80
# 20           1       56         20         82
# 21           1       70         20         91
# attr(,"assign")
# [1] 0 1 2 3
```

Instead of giving a fitted model `m`, we can also just give the data and formula as

in the call to `lm()`, *e.g.* `model.matrix(y ~ x1 + x2 + x3)`.

Making Predictions

One of the main aims of fitting a linear model is to use the model to make predictions for new, not previously observed x-values, *i.e.* to compute $y_{\text{new}} = X_{\text{new}}\hat{\beta}$. The general form of the command for prediction is `predict(m, newdata)`, where `m` is the model previously fitted using `lm()`, and `newdata` specifies the new x-values to predict responses for. The argument `new.data` should be a `data.frame` and for each variable in the original model there should be a column in `newdata` which has the name of the original variable and contains the new values. For example, if the model was fitted using

```
m <- lm(y ~ x + I(x^2))
```

and if the new samples are stored in `x.new`, then responses for the x-values in `x.new` can be predicted using the following command:

```
predict(m, data.frame(x=x.new))
```

As a second example, for the `stackloss` dataset, the following commands can be used to predict `stack.loss` for two new x-values:

```
m <- lm(stack.loss ~ ., data = stackloss)
new.data <- data.frame(Air.Flow=c(70, 73), Water.Temp=c(25,24), Acid.Conc.=c(78,90))
predict(m, new.data)
```

```
#        1        2
# 30.69174 29.71790
```

More information about `predict()` can be found by reading the output of `help(predict.lm)`.

Problem Sheet 1

You should attempt all these questions and write up your solutions in advance of your workshop in week 2 where the answers will be discussed.

1. Consider the simple linear regression model $y_i = \beta_0 + x_i\beta_1 + \varepsilon_i$ for $i \in \{1, 2, \dots, n\}$ and let X be the design matrix.

a. Show that $X^\top X = \begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix} \in \mathbb{R}^{2\times 2}$.

Solution: For simple linear regression, we have $p = 1$ and the design matrix is

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}.$$

Thus we have

$$\begin{aligned} X^\top X &= \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n 1 \cdot x_i \\ \sum_{i=1}^n x_i \cdot 1 & \sum_{i=1}^n x_i^2 \end{pmatrix} \\ &= \begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix}. \end{aligned}$$

b. Using the formula

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix},$$

find $(X^\top X)^{-1}$.

Solution: Using the formula for the inverse of a 2×2-matrix, we find

$$\begin{aligned} (X^\top X)^{-1} &= \begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix}^{-1} \\ &= \frac{1}{n\sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i\right)^2} \begin{pmatrix} \sum_{i=1}^n x_i^2 & -\sum_{i=1}^n x_i \\ -\sum_{i=1}^n x_i & n \end{pmatrix}. \end{aligned}$$

c. Find $X^\top y$ and use this to derive an explicit formula for the least squares estimate $\hat{\beta} = (X^\top X)^{-1} X^\top y$.

Solution: Omitting the indices in the sums for brevity, we have

$$X^\top y = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix}$$

and thus

$$
\begin{aligned}
\hat{\beta} &= (X^\top X)^{-1} X^\top y \\
&= \frac{1}{n\sum x_i^2 - \left(\sum x_i\right)^2} \begin{pmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & n \end{pmatrix} \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix} \\
&= \frac{1}{n\sum x_i^2 - \left(\sum x_i\right)^2} \begin{pmatrix} \sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i \\ -\sum x_i \sum y_i + n \sum x_i y_i \end{pmatrix} \\
&= \frac{1}{\frac{1}{n}\sum_{i=1}^n x_i^2 - \left(\frac{1}{n}\sum_{i=1}^n x_i\right)^2} \begin{pmatrix} \frac{1}{n}\sum_{i=1}^n x_i^2 \cdot \frac{1}{n}\sum_{i=1}^n y_i - \frac{1}{n}\sum_{i=1}^n x_i \cdot \frac{1}{n}\sum_{i=1}^n x_i y_i \\ \frac{1}{n}\sum_{i=1}^n x_i y_i - \frac{1}{n}\sum_{i=1}^n x_i \cdot \frac{1}{n}\sum_{i=1}^n y_i \end{pmatrix}.
\end{aligned}
$$

This completes the answer.

Inspection of the final result shows that we have recovered the traditional formula for the coefficients in simple linear regression, only written in slightly unusual form. For example, the term $\frac{1}{n}\sum_{i=1}^n x_i^2 - \left(\frac{1}{n}\sum_{i=1}^n x_i\right)^2$ equals the sample variance of the x_i (up to a factor of $n/(n-1)$). The algebra in this answer could be slightly simplified by changing to new coordinates $\tilde{x}_i = x_i - \bar{x}$ and $\tilde{y}_i = y_i - \bar{y}$ before fitting the regression model.

2. Let X be the design matrix of a model including an intercept, and let $H = X(X^\top X)^{-1}X^\top \in \mathbb{R}^{n\times n}$ be the hat matrix. Finally, let $\mathbf{1} = (1, 1, \dots, 1) \in \mathbb{R}^n$. Show that $H\mathbf{1} = \mathbf{1}$.

Solution: We already know that $HX = X(X^\top X)^{-1}X^\top X = X$. Since the first column of X equals $\mathbf{1}$, the first column of the matrix equation $HX = X$ is $H\mathbf{1} = \mathbf{1}$. This completes the proof.

3. For the `stackloss` dataset built into R, predict a value for `stack.loss` when the inputs are `Air.Flow = 60`, `Water.Temp = 21` and `Acid.Conc = 87`.

Solution: We can fit the model using `lm()` as usual:

```
m <- lm(stack.loss ~ ., data = stackloss)
m
```

```
#
# Call:
# lm(formula = stack.loss ~ ., data = stackloss)
#
# Coefficients:
# (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
#    -39.9197       0.7156       1.2953      -0.1521
```

To predict a new value, we use the `predict()` command. Note that `Acid.Conc.` is spelled with a trailing dot, which we need to include in the name when we supply the new input values here.

```
predict(m, data.frame(Air.Flow = 60, Water.Temp = 21, Acid.Conc. = 87))
```

```
#        1
# 16.98509
```

Thus, the predicted value for `stack.loss` is 16.98509.

4. Let $\varepsilon_1, \dots, \varepsilon_n \sim \mathcal{N}(\mu, \sigma^2)$ be independent. Then $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)$ is a random vector. Determine $\mathbb{E}(\varepsilon)$, $\mathrm{Cov}(\varepsilon)$ and $\mathbb{E}(\|\varepsilon\|^2)$.

Solution: We immediately find $\mathbb{E}(\varepsilon) = \mu\mathbf{1}$ where $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^n$. Since the ε_i are independent, we have $\mathrm{Cov}(\varepsilon) = \sigma^2 I$. Finally we have

$$\begin{aligned}\mathbb{E}(\|\varepsilon\|^2) &= \mathbb{E}(\sum_{i=1}^{n} \varepsilon_i^2) \\ &= \sum_{i=1}^{n} E(\varepsilon_i^2) \\ &= n\mathbb{E}(\varepsilon_1^2).\end{aligned}$$

Since $\mathbb{E}(\varepsilon_1^2) = \mathrm{Var}(\varepsilon_1) + \mathbb{E}(\varepsilon_1)^2 = \sigma^2 + \mu^2$ we get $\mathbb{E}(\|\varepsilon\|^2) = n(\sigma^2 + \mu^2)$.

3 Random Vectors and Covariance

Like in the one-dimensional case, we can build a **statistical model** for the data where we assume that the errors are random. More precisely we will assume

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \varepsilon_i \tag{12}$$

for all $i \in \{1, 2, \dots, n\}$, where $\varepsilon_1, \dots, \varepsilon_n$ are now independent and identically distributed (i.i.d.) random variables with $\mathbb{E}(\varepsilon_i) = 0$ and $\mathrm{Var}(\varepsilon_i) = \sigma^2$. As in (7), the statistical model can be written in vector form as

$$Y = X\beta + \varepsilon. \tag{13}$$

This is a vector-valued equation which contains two "random vectors", Y and ε.

A **random vector** is a vector $Z = (Z_1, \dots, Z_n)$ where each component Z_i is a random variable.

3.1 Expectation

The expectation of a random vector is taken for each component separately. This is formalised in the following definition.

Definition 3.1. Let $Z = (Z_1, \dots, Z_n) \in \mathbb{R}^n$ be a random vector. Then the expectation of Z is the (non-random) vector

$$\mathbb{E}(X) = \begin{pmatrix} \mathbb{E}(Z_1) \\ \vdots \\ \mathbb{E}(Z_n) \end{pmatrix} \in \mathbb{R}^n.$$

The same convention is sometimes used for random matrices M, as $\mathbb{E}(M)_{ij} = \mathbb{E}(M_{ij})$.

Example 3.1. The random vector ε in (13) has

$$\mathbb{E}(\varepsilon)_i = \mathbb{E}(\varepsilon_i) = 0$$

for all $i \in \{1, \dots, n\}$ and thus $\mathbb{E}(\varepsilon) = 0 \in \mathbb{R}^n$, where 0 here denotes the zero-vector $(0, \dots, 0) \in \mathbb{R}^n$.

Since the expectation of a random vector is defined in term of the usual expectation, most rules we know for expectations still hold. For example, if Y and Z are two random vectors, we have $\mathbb{E}(Y + Z) = \mathbb{E}(Y) + \mathbb{E}(Z)$.

Example 3.2. The random vector Y in (13) has

$$\mathbb{E}(Y)_i = \mathbb{E}(Y_i) = \mathbb{E}((X\beta)_i + \varepsilon_i) = (X\beta)_i + \mathbb{E}(\varepsilon_i) = (X\beta)_i$$

for all $i \in \{1, \dots, n\}$ and thus $\mathbb{E}(Y) = X\beta \in \mathbb{R}^n$. We often will write the above derivation in vector form as

$$\mathbb{E}(Y) = \mathbb{E}(X\beta + \varepsilon) = X\beta + \mathbb{E}(\varepsilon) = X\beta.$$

Example 3.3. If $A \in \mathbb{R}^{m \times n}$ is a matrix and $Z \in \mathbb{R}^n$ is a random vector, then we find the expectation of $AZ \in \mathbb{R}^m$ as

$$\mathbb{E}(AZ)_i = \mathbb{E}(AZ_i) = \mathbb{E}(\sum_{j=1}^{n} a_{ij} Z_j) = \sum_{j=1}^{n} \mathbb{E}(a_{ij} Z_j) = \sum_{j=1}^{n} a_{ij} \mathbb{E}(Z_j) = \sum_{j=1}^{n} a_{ij} \mathbb{E}(Z)_j$$

for all $i \in \{1, \dots, m\}$ and thus we have $\mathbb{E}(AZ) = A\mathbb{E}(Z)$.

3.2 Covariance Matrix

The variance of random variables is replaced with the concept of a "covariance matrix" for random vectors.

Definition 3.2. Let $Z = (Z_1, \dots, Z_n) \in \mathbb{R}^n$ be a random vector. Then the covariance matrix of Z is the matrix $\mathrm{Cov}(Z) \in \mathbb{R}^{n \times n}$ given by

$$\mathrm{Cov}(Z)_{ij} = \mathrm{Cov}(Z_i, Z_j),$$

for all $i, j \in \{1, \dots, n\}$, where $\mathrm{Cov}(Z_i, Z_j)$ denotes the usual covariance between random variables.

We collect some basic properties of covariance matrices here. Most of these arguments use concepts and rules from linear algebra, as summarised in section A in the appendix.

- Since $\mathrm{Cov}(Z_i, Z_j) = \mathrm{Cov}(Z_j, Z_i)$, covariance matrices are symmetric.
- The diagonal elements of $\mathrm{Cov}(Z)$ are

$$\mathrm{Cov}(Z)_{ii} = \mathrm{Cov}(Z_i, Z_i) = \mathrm{Var}(Z_i). \tag{14}$$

- If the elements Z_i of Z are (statistically) independent, we have $\mathrm{Cov}(Z_i, Z_j) = 0$ and thus $\mathrm{Cov}(Z)_{ij} = 0$ for $i \neq j$. Thus, if Z is a vector of independent random variables, the covariance matrix of Z is diagonal.
- Let $\mu = \mathbb{E}(Z) \in \mathbb{R}^n$. If we interpret μ as a column vector, then $M = (Z-\mu)(Z-\mu)^\top$ is an $n \times n$ matrix and we have

$$M_{ij} = \left((Z-\mu)(Z-\mu)^\top\right)_{ij} = (Z-\mu)_i (Z-\mu)_j.$$

  Taking expectations gives $\mathbb{E}(M_{ij}) = E((Z-\mu)_i (Z-\mu)_j) = \mathrm{Cov}(Z_i, Z_j)$ and thus we can write

$$\mathrm{Cov}(Z) = \mathbb{E}((Z-\mu)(Z-\mu)^\top). \tag{15}$$

- Covariance matrices are positive semi-definite. To see this, let $C = \mathrm{Cov}(Z)$ and $u \in \mathbb{R}^n$ be a vector. We have to show that $u^\top C u \geq 0$. Writing $\bar{Z} := Z - \mathbb{E}(Z)$ as an abbreviation, we get

$$\begin{aligned} u^\top C u &= u^\top \mathbb{E}(\bar{Z}\bar{Z}^\top) u \\ &= \mathbb{E}(u^\top \bar{Z}\bar{Z}^\top u) \\ &= \mathbb{E}((\bar{Z}^\top u)^\top \bar{Z}^\top u) \\ &= \mathbb{E}(\|\bar{Z}^\top u\|^2), \end{aligned}$$

  where $\|\bar{Z}^\top u\|$ denotes the Euclidean length of the vector $\bar{Z}^\top u$. Since $\|\bar{Z}^\top u\|^2 \geq 0$ we find $u^\top C u \geq 0$. This shows that the covariance matrix C is positive semi-definite. (Note that, nevertheless, individual *elements* of the matrix C can be negative numbers.)

Example 3.4. The random vector ε in equation (13) has $\mathbb{E}(\varepsilon) = 0$. We have $\mathrm{Cov}(\varepsilon)_{ii} = \mathrm{Var}(\varepsilon_i) = \sigma^2$ for all $i \in \{1, \dots, n\}$. Since we assumed the ε_i to be independent, the covariance matrix is diagonal and we find

$$\mathrm{Cov}(\varepsilon) = \sigma^2 I,$$

where I is the $n \times n$ identity matrix.

An important results about covariance matrices is given in the following lemma, which describes how the covariance matrix changes under affine transformations.

Lemma 3.1. *Let $Z \in \mathbb{R}^n$ be a random vector, $A \in \mathbb{R}^{m\times n}$ a matrix and $b \in \mathbb{R}^m$ a vector. Then*

$$\operatorname{Cov}(AZ + b) = A \operatorname{Cov}(Z) A^\top.$$

Proof. As in equation (15), we can write $\operatorname{Cov}(AZ + b)$ as

$$\operatorname{Cov}(AZ + b) = \mathbb{E}((AZ + b - \mu)(AZ + b - \mu)^\top),$$

where $\mu = \mathbb{E}(AZ + b) = \mathbb{E}(AZ) + b$. Thus, $AZ + b - \mu = AZ - \mathbb{E}(AZ)$ and we find

$$\begin{aligned}\operatorname{Cov}(AZ + b) &= \mathbb{E}((AZ - \mathbb{E}(AZ))(AZ - \mathbb{E}(AZ))^\top) \\ &= \operatorname{Cov}(AZ).\end{aligned}$$

This shows that the covariance matrix ignores non-random shifts.

Furthermore, we have $AZ - \mathbb{E}(AZ) = AZ - A\mathbb{E}(Z) = A(Z - \mathbb{E}(Z))$. Using equation (15) again, we find

$$\begin{aligned}\operatorname{Cov}(AZ) &= \mathbb{E}\Big((AZ - \mathbb{E}(AZ))(AZ - \mathbb{E}(AZ))^\top\Big) \\ &= \mathbb{E}\Big(A(Z - \mathbb{E}(Z))(Z - \mathbb{E}(Z))^\top A^\top\Big) \\ &= A\mathbb{E}\Big((Z - \mathbb{E}(Z))(Z - \mathbb{E}(Z))^\top\Big)A^\top \\ &= A \operatorname{Cov}(Z) A^\top.\end{aligned}$$

This completes the proof. □

3.3 The Multivariate Normal Distribution

Since we assume that the random errors ε_i are normally distributed, we will need to understand how vectors of normal distributed random variables behave.

Definition 3.3. A random vector $Z \in \mathbb{R}^n$ follows a **multivariate normal distribution**, if $u^\top Z$ is normally distributed or constant for every vector $u \in \mathbb{R}^n$.

This definition is takes its slightly surprising form to avoid some boundary cases which I will discuss in an example, below. To understand the definition, a good start is to consider the cases where u is one of the standard basis vectors, say $u_i = 1$ and $u_j = 0$ for all $j \neq i$. In this case we have

$$u^\top Z = \sum_{k=1}^{n} u_k Z_k = Z_i.$$

Thus, if Z follows a multivariate normal distribution, each of the components Z_i is normally distributed. Example 3.8, below, shows that the converse is not true.

One can show that a multivariate normal distribution is completely determined by the mean $\mu = \mathbb{E}(Z)$ and the covariance $\Sigma = \operatorname{Cov}(Z)$. The distribution of such a Z is denoted by $\mathcal{N}(\mu, \Sigma)$. Also, for every $\mu \in \mathbb{R}^n$ and every positive semi-definite matrix $\Sigma \in \mathbb{R}^{n\times n}$ there is a random vector Z which follows a multivariate normal distribution with this mean and covariance.

Example 3.5. Consider the vector ε from the model (13). This vector has components $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ and by assumption, the componens ε_i are independent. For $u \in \mathbb{R}^n$ we have

$$u^\top \varepsilon = \sum_{i=1}^{n} u_i \varepsilon_i.$$

Since this is a sum of independent, one-dimensional, normally distributed random variables, $u^\top \varepsilon$ is also normally distribution, for every u. (The independence of the ε_i is important in this argument.) Thus, ε is a normally distributed random vector. We have already seen $\mathbb{E}(\varepsilon) = 0$ and $\operatorname{Cov}(\varepsilon) = \sigma^2 I$, and thus $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$.

Without proof we state here some properties of the multivariate normal distribution:

- If $Z \sim \mathcal{N}(\mu, \Sigma)$ and $a \in \mathbb{R}^n$, then $Z + a \sim \mathcal{N}(\mu + a, \Sigma)$.
- If $Z \sim \mathcal{N}(\mu, \Sigma)$ and $A \in \mathbb{R}^{m \times n}$, then $AZ \sim \mathcal{N}(A\mu, A\Sigma A^\top)$.
- If $Z_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $Z_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$ are independent, then $Z_1 + Z_2 \sim \mathcal{N}(\mu_1 + \mu_2, \Sigma_1 + \Sigma_2)$.

Example 3.6. Let $Z = (Z_1, Z_2)$ where Z_1 and Z_2 are independently standard normal distributed. Let

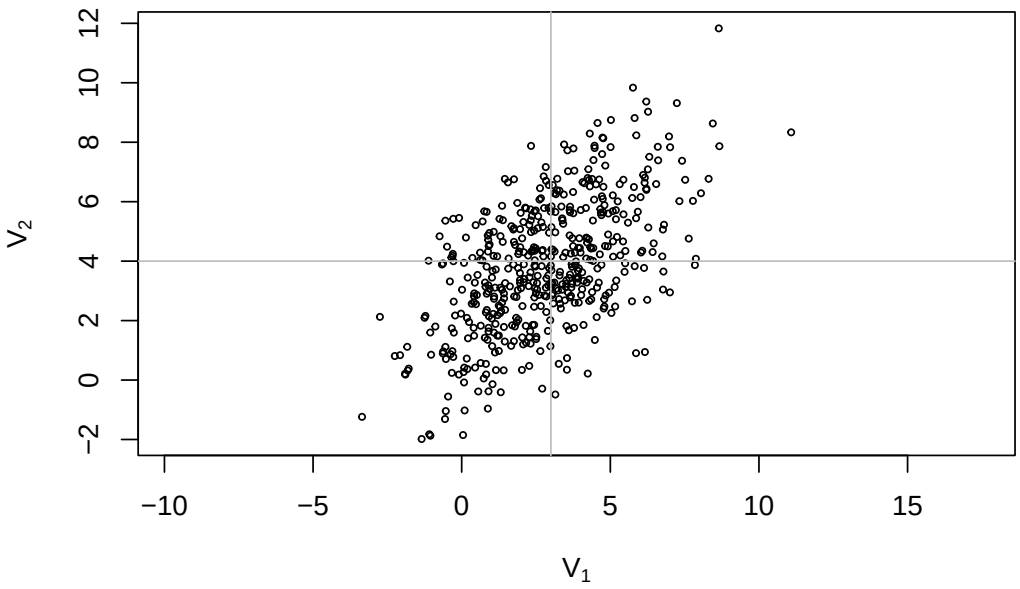
$$A := \begin{pmatrix} 2 & -1 \\ 2 & 1 \end{pmatrix} \qquad \text{and} \qquad b := \begin{pmatrix} 3 \\ 4 \end{pmatrix}.$$

Then $AZ + b \sim \mathcal{N}(b, \Sigma)$ where

$$\Sigma = A \operatorname{Cov}(Z) A^\top = \begin{pmatrix} 2 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 2 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 5 & 3 \\ 3 & 5 \end{pmatrix}$$

We can use R to plot a sample of this two-dimensional normal distribution. (The grey cross indicates the mean.)

```
N <- 500
Z <- rbind(rnorm(N), rnorm(N))
A <- matrix(c(2, 2, -1, 1), 2, 2)
b <- c(3, 4)
V <- A %*% Z + b
plot(V[1,], V[2,], asp = 1, cex = .5,
     xlab = expression(V[1]),
     ylab = expression(V[2]))
abline(v = 3, col = "grey")
abline(h = 4, col = "grey")
```



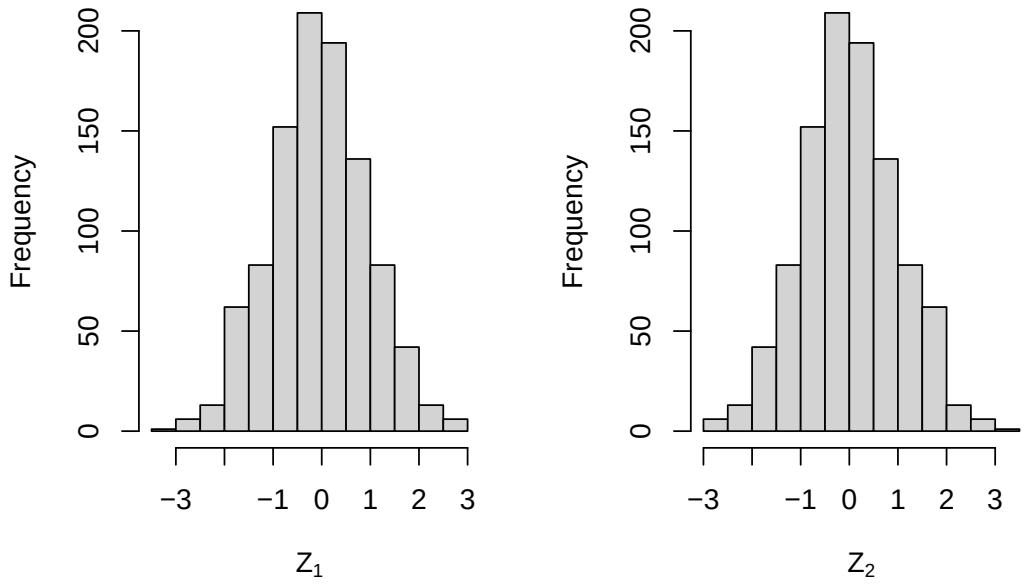
Example 3.7. The random vector Y in (13) satisfies $Y = X\beta + \varepsilon$ and since $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ we have $Y \sim \mathcal{N}(X\beta, \sigma^2 I)$.

Example 3.8. Let $Y \sim \mathcal{N}(0, 1)$ be a random variable. Define a random vector $Z = (Z_1, Z_2)$ as $Z_1 = Y$ and

$$Z_2 = \begin{cases} Y & \text{if } |Y| < 1, \text{ and} \\ -Y & \text{otherwise.} \end{cases}$$

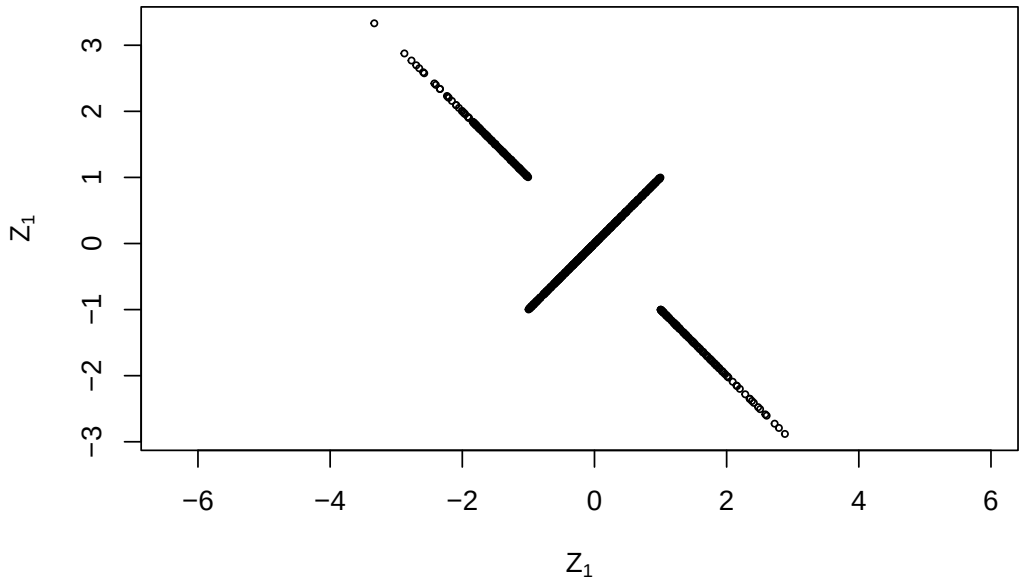
Clearly Z_1 is standard normally distributed. Since $\mathcal{N}(0,1)$ is symmetric, both Y and $-Y$ are standard normally distributed and it follows that Z_2 is also standard normally distributed. Nevertheless, the random vector Z does not follow a multivariate normal distribution. Instead of giving a proof of this fact, we illustrate this here using an R experiment. We start by verifying that Z_1 and Z_2 are normally distributed.

```
N <- 1000
Y <- rnorm(N)
Z1 <- Y
Z2 <- ifelse(abs(Y)<1, Y, -Y)

par(mfrow=c(1,2))
hist(Z1, main=NULL, xlab=expression(Z[1]))
hist(Z2, main=NULL, xlab=expression(Z[2]))
```



The histograms make it plausible that the components are indeed normally distributed. Now we use a scatter plot to show the joint distribution of Z_1 and Z_2:

```
plot(Z1, Z2, cex=0.5, asp=1,
     xlab=expression(Z[1]),
     ylab=expression(Z[1]))
```



This plot looks peculiar! Most people would not call this a normal distribution and the formal definition of a multivariate normal distribution is made to exclude cases like this.

Summary

- We learned the rules for computing the expectation of a random vector.
- The covariance matrix of random vectors plays the role of the variance for numeric random variables.
- We learned about the definition of the multivariate normal distribution.

4 Properties of the Least Squares Estimate

Like in the one-dimensional case, we can build a **statistical model** for the data. Here we assume that the residuals are random. More precisely we have

$$Y = X\beta + \varepsilon. \tag{16}$$

where $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)$ is a random error. The individual errors $\varepsilon_1, \dots, \varepsilon_n$ are now assumed to be i.i.d. random variables with $\mathbb{E}(\varepsilon_i) = 0$ and $\mathrm{Var}(\varepsilon_i) = \sigma^2$.

- Again, we assume that the x-values are fixed and known. The only random quantities in the model are ε_i and Y_i.
- The parameters in this model are now the regression coefficients $\beta = (\beta_0, \beta_1, \cdots, \beta_p) \in \mathbb{R}^{p+1}$ and the error variance σ^2.

The usual approach in statistics to quantify how well an estimator works is to apply it to random samples from the statistical model, where we can assume that we know the parameters, and then to study how well the parameters are reconstructed by the estimators. Since this approach uses random samples as input the the estimator, we obtain random estimates and we need to use statistical methods to quantify how close the estimate is to the truth.

4.1 Mean and Covariance

The bias of an estimator is the difference between the expected value of the estimate and the truth. For the least squares estimator we have

$$\mathrm{bias}(\hat{\beta}) = \mathbb{E}(\hat{\beta}) - \beta,$$

where

$$\hat{\beta} = (X^\top X)^{-1} X^\top Y$$

and Y is the random vector from (16).

Lemma 4.1. *We have*

1) $\hat{\beta} = \beta + (X^\top X)^{-1} X^\top \varepsilon$ *and*

2) $\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2 (X^\top X)^{-1})$.

Proof. From lemma 2.1 we know

$$\hat{\beta} = (X^\top X)^{-1} X^\top Y,$$

Using the definition of Y we can write this as

$$\begin{aligned}
\hat{\beta} &= (X^\top X)^{-1} X^\top Y \\
&= (X^\top X)^{-1} X^\top (X\beta + \varepsilon) \\
&= (X^\top X)^{-1} X^\top X\beta + (X^\top X)^{-1} X^\top \varepsilon \\
&= \beta + (X^\top X)^{-1} X^\top \varepsilon.
\end{aligned}$$

This proves the first claim.

Since ε follows a multi-variate normal distribution, $\beta + (X^\top X)^{-1} X^\top \varepsilon$ is also normally distributed. Taking expectations we get

$$\begin{aligned}
\mathbb{E}(\hat{\beta}) &= \mathbb{E}(\beta + (X^\top X)^{-1} X^\top \varepsilon) \\
&= \beta + (X^\top X)^{-1} X^\top \mathbb{E}(\varepsilon) \\
&= \beta,
\end{aligned}$$

since $\mathbb{E}(\varepsilon) = 0$.

For the covariance we find

$$\begin{aligned}\mathrm{Cov}(\hat{\beta}) &= \mathrm{Cov}(\beta + (X^\top X)^{-1} X^\top \varepsilon) \\ &= \mathrm{Cov}((X^\top X)^{-1} X^\top \varepsilon) \\ &= (X^\top X)^{-1} X^\top \,\mathrm{Cov}(\varepsilon)((X^\top X)^{-1} X^\top)^\top \\ &= (X^\top X)^{-1} X^\top \,\mathrm{Cov}(\varepsilon) X (X^\top X)^{-1}.\end{aligned}$$

Since $\mathrm{Cov}(\varepsilon) = \sigma^2 I$, this simplifies to

$$\begin{aligned}\mathrm{Cov}(\hat{\beta}) &= (X^\top X)^{-1} X^\top \sigma^2 I X (X^\top X)^{-1} \\ &= \sigma^2 (X^\top X)^{-1} X^\top X (X^\top X)^{-1} \\ &= \sigma^2 (X^\top X)^{-1}.\end{aligned}$$

This completes the proof. □

The lemma implies that $\mathbb{E}(\hat{\beta}) = \beta$, *i.e.* the estimator $\hat{\beta}$ is unbiased. Note that for this statement we only used $\mathbb{E}(\varepsilon) = 0$ to compute the expectation of $\hat{\beta}$. Thus, the estimator will still be unbiased for correlated noise or for noise which is not normally distributed.

We have seen earlier that the diagonal elements of a covariance matrix give the variances of the elements of the random vector. Setting $C := (X^\top X)^{-1}$ as a shorthand here, we find that the individual estimated coefficients $\hat{\beta}_i$ satisfy

$$\hat{\beta}_i \sim \mathcal{N}(\beta, \sigma^2 C_{ii}) \tag{17}$$

for $i \in \{1, \dots, n\}$.

These results about the (co-)variances of the estimator are not very useful in practice, because the error variance σ^2 is usually unknown. To derive more useful results, we will consider how to estimate this variance.

4.2 Properties of the Hat Matrix

In this and the following sections we will use various properties of the hat matrix $H = X(X^\top X)^{-1} X^\top$.

Lemma 4.2. *The hat matrix H has the following properties:*

1) H is symmetric, i.e. $H^\top = H$.
2) H is idempotent, i.e. $H^2 = H$.

Proof. For the first statement we have

$$\begin{aligned}H^\top &= (X(X^\top X)^{-1} X^\top)^\top \\ &= (X^\top)^\top ((X^\top X)^{-1})^\top X^\top \\ &= X(X^\top X)^{-1} X^\top \\ &= H,\end{aligned}$$

where we used that the inverse of a symmetric matrix is symmetric. The second statement follow from

$$\begin{aligned}H^2 &= (X(X^\top X)^{-1} X^\top)(X(X^\top X)^{-1} X^\top) \\ &= X((X^\top X)^{-1} X^\top X)(X^\top X)^{-1} X^\top \\ &= X(X^\top X)^{-1} X^\top.\end{aligned}$$

This completes the proof. □

Both properties from the lemma carry over from H to $I - H$: we have $(I - H)^\top = I^\top - H^\top = I - H$ and

$$\begin{aligned}(I-H)^2 &= (I-H)(I-H)\\ &= I^2 - HI - IH + H^2\\ &= I - H - H + H\\ &= I - H.\end{aligned}$$

For future reference we also state two simpler results: we have

$$HX = X(X^\top X)^{-1}X^\top X = X \tag{18}$$

and

$$(I-H)X = IX - HX = X - X = 0. \tag{19}$$

Finally, if we have a vector $v \in \mathbb{R}^n$ we can write v as

$$\begin{aligned}v &= (H + I - H)v\\ &= Hv + (I-H)v.\end{aligned}$$

The inner product between these two components is

$$\begin{aligned}(Hv)^\top(I-H)v &= v^\top H^\top (I-H)v\\ &= v^\top H(I-H)v\\ &= v^\top(H - H^2)v\\ &= v^\top(H-H)v\\ &= 0,\end{aligned}$$

so the two vectors are orthogonal. As a result we get

$$\begin{aligned}\|v\|^2 &= v^\top v\\ &= \big(Hv + (I-H)v\big)^\top \big(Hv + (I-H)v\big)\\ &= (Hv)^\top Hv + 2(Hv)^\top(I-H)v + \big((I-H)v\big)^\top(I-H)v\\ &= \|Hv\|^2 + \|(I-H)v\|^2\end{aligned}$$

for any vector $v \in \mathbb{R}^n$. (This is Pythagoras' theorem in $\mathbb{R}^n$.) Since $\hat{y} = Hy$ and $\hat{\varepsilon} = (I-H)y$, we can apply this idea to the vector $v = y$ to get

$$\|y\|^2 = \|\hat{y}\|^2 + \|\hat{\varepsilon}\|^2. \tag{20}$$

We note without proof that geometrically, H can be interpreted as the orthogonal projection onto the subspace of $\mathbb{R}^n$ which is spanned by the columns of X. This subspace contains the possible output vectors of the linear system and the least squares procedure finds the point $\hat{y}$ in this subspace which is closest to the observed data $y \in \mathbb{R}^n$.

Some authors define:

- $\mathrm{SS}_\mathrm{T} = \sum_{i=1}^n y_i^2$ (where "T" stands for "total")
- $\mathrm{SS}_\mathrm{R} = \sum_{i=1}^n \hat{y}_i^2$ (where "R" stands for "regression")
- $\mathrm{SS}_\mathrm{E} = \sum_{i=1}^n \hat{\varepsilon}_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ (where "E" stands for "error")

Using this notation, our equation $\|y\|^2 = \|\hat{y}\|^2 + \|\hat{\varepsilon}\|^2$ turns into

$$\mathrm{SS}_\mathrm{T} = \mathrm{SS}_\mathrm{R} + \mathrm{SS}_\mathrm{E}.$$

4.3 Cochran's theorem

Our main tool in this and the following section will be a simplified version of Cochran's theorem:

Theorem 4.1. *The following statements are true:*

1) $\frac{1}{\sigma^2}\varepsilon^\top H\varepsilon \sim \chi^2(p+1)$

2) $\frac{1}{\sigma^2}\varepsilon^\top (I-H)\varepsilon \sim \chi^2(n-p-1)$

3) $H\varepsilon$ *and* $(I-H)\varepsilon$ *are independent.*

Proof. Since H is symmetric, we can diagonalise H (see A.2 in the appendix): there is an orthogonal matrix U such that $D := UHU^\top$ is diagonal, and the diagonal elements of D are the eigenvalues of H. Since H is idempotent, these diagonal elements can only be 0 or 1. Also, since U is orthogonal, we have $U^\top U = I$ and thus

$$U^\top DU = U^\top UHU^\top U = H.$$

The same matrix U also diagonalises $I-H$, since $U(I-H)U^\top = UU^\top - UHU^\top = I-D$. Exactly one of the diagonal elements D_{ii} and $(I-D)_{ii}$ is 1 and the other one is 0 for every i.

Since $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ we find that $\eta := U\varepsilon$ is normally distributed with mean $U0 = 0$ and covariance matrix $\sigma^2 UIU^\top = \sigma^2 UU^\top = \sigma^2 I$. Thus η has the same distribution as ε does: $\eta \sim \mathcal{N}(0, \sigma^2 I)$ and the components η_i are independent of each other. We have

$$H\varepsilon = U^\top DU\varepsilon = U^\top D\eta.$$

and

$$(I-H)\varepsilon = U^\top (I-D)U\varepsilon = U^\top (I-D)\eta.$$

Since $(D\eta)_i = 0$ if $D_{ii} = 0$ and $\big((I-D)\eta\big)_i = 0$ otherwise, each component of η contributes to exactly one of the two vectors $D\eta$ and $(I-D)\eta$. Thus, $D\eta$ and $(I-D)\eta$ are independent, and thus $H\varepsilon$ and $(I-H)\varepsilon$ are also independent. This proves the third statement of the theorem.

For the first statement, we note that

$$\begin{aligned}\varepsilon^\top H\varepsilon &= \varepsilon^\top U^\top DU\varepsilon \\ &= \eta^\top D\eta \\ &= \sum_{\substack{i=1 \\ D_{ii}=1}}^{n} \eta_i^2.\end{aligned}$$

Since $(X^\top X) \in \mathbb{R}^{(p+1)\times(p+1)}$ is invertible, one can show that $\operatorname{rank}(H) = p+1$ and thus that there are $p+1$ terms contributing to the sum (we skip the proof of this statement here). Thus,

$$\frac{1}{\sigma^2}\varepsilon^\top H\varepsilon = \sum_{\substack{i=1 \\ D_{ii}=1}}^{n} (\eta_i/\sigma)^2$$

is the sum of the squares of $p+1$ independent standard normals, and thus is $\chi^2(p+1)$ distributed. This completes the proof of the first statement.

Finally, the second statement follows in much of the same way as the first one, except that H is replaced with $I-H$ and the sum is over the $n-p-1$ indices i where $D_{ii} = 0$. This completes the proof. □

Expressions of the form $x^\top Ax$ for $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ are called **quadratic forms**.

While the theorem as written only states that $H\varepsilon$ and $(I-H)\varepsilon$ are independent of each other, we can replace one or both of these terms the corresponding quadratic forms as still keep the independence. Since $(H\varepsilon)^\top(H\varepsilon) = \varepsilon^\top H^\top H\varepsilon = \varepsilon^\top H\varepsilon$, the quadratic form $\varepsilon^\top H\varepsilon$ is a function of $H\varepsilon$ and a similar statement holds with $H - I$ instead of H.

4.4 Estimating the Error Variance

So far we have only considered how to estimate the parameter vector β and we have ignored the parameter σ^2. We will see that an unbiased estimator for σ^2 is given by

$$\hat{\sigma}^2 := \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{21}$$

where $\hat{y}_i$ are the fitted values from equation (10). As for the one-dimensional case in (4), the estimate does not have the prefactor $1/n$, which one might naively expect, but the denominator is decreased by one for each component of the vector β. Using Cochran's theorem, we can now show that the estimator $\hat{\sigma}^2$ is unbiased.

We first note that

$$\begin{aligned}(n-p-1)\hat{\sigma}^2 &= (y-\hat{y})^\top(y-\hat{y})\\ &= (y-Hy)^\top(y-Hy)\\ &= y^\top(I-H)^\top(I-H)y\\ &= y^\top(I-H)y.\end{aligned}$$

To determine the bias, we need to use $Y = X\beta + \varepsilon$ in place of the data. This gives

$$\begin{aligned}(n-p-1)\hat{\sigma}^2 &= Y^\top(I-H)Y\\ &= (X\beta+\varepsilon)^\top(I-H)(X\beta+\varepsilon)\\ &= \beta^\top X^\top(I-H)X\beta + 2\varepsilon^\top(I-H)X\beta + \varepsilon^\top(I-H)\varepsilon\\ &= \varepsilon^\top(I-H)\varepsilon,\end{aligned}$$

where we used equation (19) to see that the first two terms in the sum equal zero.

Now we can apply Cochran's theorem. This shows that

$$\frac{1}{\sigma^2}(n-p-1)\hat{\sigma}^2 = \frac{1}{\sigma^2}\varepsilon^\top(I-H)\varepsilon \sim \chi^2(n-p-1). \tag{22}$$

Since the expectation of a $\chi^2(\nu)$ distribution equals ν (see appendix B.2), we find

$$\frac{1}{\sigma^2}(n-p-1)\mathbb{E}(\hat{\sigma}^2) = n-p-1$$

and thus

$$\mathbb{E}(\hat{\sigma}^2) = \sigma^2.$$

This proves that $\hat{\sigma}^2$ is an unbiased estimator for σ^2.

Summary

- The least squares estimator for the regression coefficients is unbiased.
- The hat matrix is idempotent and symmetric.
- Cochran's theorem allows to understand the distribution of some quadratic forms involving the hat matrix.
- $\hat{\sigma}^2$ is an unbiased estimator for σ^2.

5 Uncertainty for Individual Regression Coefficients

In this section we will consider different ways to study the uncertainty in the estimates $\hat{\beta}_i$ for the regression coefficient β_i, individually. In the following sections we will then consider the problem of simultaneously estimating several or all coefficients.

5.1 Measuring the Estimation Error

We have seen that $\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2 C)$, where $C := (X^\top X)^{-1}$. Restricting this to a single coefficient, we find

$$\hat{\beta}_i \sim \mathcal{N}(\beta_i, \sigma^2 C_{ii}),$$

since the diagonal elements of the covariance matrix contains the variances of the elements of a random vector. In practice we will not know the value of σ^2, so we have to estimate this from data, using the estimator

$$\hat{\sigma}^2 = \frac{1}{n-p-1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

from equation (21). As a first application of Cochran's theorem we showed in equation (22) that

$$\frac{1}{\sigma^2}(n-p-1)\hat{\sigma}^2 \sim \chi^2(n-p-1).$$

Note that in the equations above, we index the rows and columns of C using $i, j \in \{0, 1, \dots, p\}$, *i.e.* the first row and column are using the index 0 each. This is to match the convention for the components of $\beta = (\beta_0, \beta_1, \dots, \beta_p)$.

Lemma 5.1. *The random vector $\hat{\beta}$ and the random number $\hat{\sigma}^2$ are independent of each other.*

Proof. We will show that $\hat{\beta}$ can be written as a function of $H\varepsilon$ and that $\hat{\sigma}^2$ can be written as a function of $(I-H)\varepsilon$. The result then follows from Cochran's theorem.

From lemma 4.1 we know that the least squares estimate $\hat{\beta}$ can be written as

$$\hat{\beta} = \beta + (X^\top X)^{-1} X^\top \varepsilon$$

and that $H = X(X^\top X)^{-1} X^\top$. Thus we can write $\hat{\beta}$ as

$$\begin{aligned} \hat{\beta} &= \beta + (X^\top X)^{-1} X^\top X \, (X^\top X)^{-1} X^\top \varepsilon \\ &= \beta + (X^\top X)^{-1} X^\top H \varepsilon, \end{aligned}$$

which is a function of $H\varepsilon$.

Similar to the argument at the end of the previous section, we can write $\hat{\sigma}^2$ as

$$\begin{aligned} (n-p-1)\hat{\sigma}^2 &= (Y - \hat{Y})^\top (Y - \hat{Y}) \\ &= (Y - HY)^\top (Y - HY) \\ &= Y^\top (I-H)^\top (I-H) Y \\ &= \big\| (I-H)Y \big\|^2. \end{aligned}$$

Since $Y = X\beta + \varepsilon$ and since we know $(I-H)X = 0$ from equation (19), we find

$$\begin{aligned} \hat{\sigma}^2 &= \frac{1}{n-p-1} \big\| (I-H)(X\beta + \varepsilon) \big\| \\ &= \frac{1}{n-p-1} \big\| (I-H)\varepsilon \big\|, \end{aligned}$$

which is a function of $(I-H)\varepsilon$.

From Cochran's theorem we know that $H\varepsilon$ and $(I-H)\varepsilon$ are independent and thus we can conclude that $\hat{\beta}$ and $\hat{\sigma}^2$ are also independent of each other. This completes the proof. □

We now construct a quantity T which measures the distance between the estimated value $\hat{\beta}_i$ and the unknown true value β_i:

$$T := \frac{\hat{\beta}_i - \beta_i}{\sqrt{\hat{\sigma}^2 C_{ii}}}. \tag{23}$$

While there are many ways to measure this distance, the T constructed here has two main advantages:

- The value of T can be computed from the given data, without any reference to unknown quantities (except for β_i).
- Below, we will be able to find the distribution of T. This will allow us to use T to construct confidence intervals and statistical tests.

Lemma 5.2. *Assume that the data follows the model* (16)*. Then* $T \sim t(n-p-1)$, i.e. T *follows a t-distribution with* $n-p-1$ *degrees of freedom (see appendix B.3).*

Proof. We have

$$\begin{aligned} T &= \frac{(\hat{\beta}_i - \beta_i)/\sqrt{C_{ii}}}{\sqrt{\hat{\sigma}^2}} \\ &= \frac{(\hat{\beta}_i - \beta_i)/\sqrt{\sigma^2 C_{ii}}}{\sqrt{\hat{\sigma}^2/\sigma^2}} \\ &= \frac{(\hat{\beta}_i - \beta_i)/\sqrt{\sigma^2 C_{ii}}}{\sqrt{((n-p-1)\hat{\sigma}^2/\sigma^2)/(n-p-1)}} \\ &=: \frac{Z}{\sqrt{Y/(n-p-1)}}, \end{aligned}$$

where $Z = (\hat{\beta}_i - \beta_i)/\sqrt{\sigma^2 C_{ii}} \sim \mathcal{N}(0,1)$ and $Y = (n-p-1)\hat{\sigma}^2/\sigma^2 \sim \chi^2(n-p-1)$ are independent, by lemma 5.1. Thus, $T \sim t(n-p-1)$ as required. □

The quantity $\sqrt{\sigma^2 C_{ii}}$ is sometimes called the **standard error** of the estimator $\hat{\beta}_i$, denoted by $\mathrm{se}(\hat{\beta}_i)$.

5.2 Confidence Intervals

Using the scaled distance T, it is easy to construct a confidence interval for $\hat{\beta}_i$: For $\alpha \in (0,1)$, say $\alpha = 5\%$, lemma 5.2 shows that

$$P\Big(T \in [-t_{n-p-1}(\alpha/2), +t_{n-p-1}(\alpha/2)]\Big) = 1-\alpha,$$

where $t_{n-p-1}(\alpha/2)$ is the $(1-\alpha/2)$-quantile of the $t(n-p-1)$-distribution. Rewriting this expression as a condition on $\hat{\beta}_i$ instead of on T gives a confidence interval for β_i.

Lemma 5.3. *The interval*

$$[U,V] := \Big[\hat{\beta}_i - \sqrt{\hat{\sigma}^2 C_{ii}}\, t_{n-p-1}(\alpha/2), \hat{\beta}_i + \sqrt{\hat{\sigma}^2 C_{ii}}\, t_{n-p-1}(\alpha/2)\Big]$$

is a $(1-\alpha)$*-confidence interval for* β_i*.*

Proof. We have to show that $P(\beta_i \in [U, V]) \geq 1 - \alpha$. We have

$$\begin{aligned}
&\beta_i \in [U, V] \\
&\quad \iff \left|\hat{\beta}_i - \beta_i\right| \leq \sqrt{\hat{\sigma}^2 C_{ii}} t_{n-p-1}(\alpha/2) \\
&\quad \iff \left|\frac{\hat{\beta}_i - \beta_i}{\sqrt{\hat{\sigma}^2 C_{ii}}}\right| \leq t_{n-p-1}(\alpha/2) \\
&\quad \iff T \in [-t_{n-p-1}(\alpha/2), +t_{n-p-1}(\alpha/2)]
\end{aligned}$$

and thus $P(\beta_i \in [U, V]) = 1 - \alpha$. This completes the proof. □

5.3 Hypthesis Tests

Very similar to the argument for confidence intervals, we can derive a hypothesis test to test the hypothesis

$$H_0 \colon \beta_i = b$$

against the alternative

$$H_1 \colon \beta_i \neq b.$$

If we use $b = 0$ in the test, we can test whether $\beta_i = 0$. If $\beta_i = 0$ is true, the corresponding input x_i has no influence on the output.

Here we redefine T as

$$T := \frac{\hat{\beta}_i - b}{\sqrt{\hat{\sigma}^2 C_{ii}}}, \tag{24}$$

using b in place of the β_i above. When H_0 is true, this new defintion of T is the same as (23).

Lemma 5.4. *The test which rejects H_0 if and only if $|T| > t_{n-p-1}(\alpha/2)$ has significance level α.*

Proof. We have to show that the probability of type I errors (*i.e.* of wrongly rejecting H_0 when it is true) is less than or equal to α. Assume that H_0 is true. Then we have $\beta_i = b$ and thus the T defined in this section coincides with the expression from equation (23). From lemma 5.2 we know that $T \sim t(n - p - 1)$. Thus we have

$$\begin{aligned}
P(\text{type I error}) &= P(|T| > t_{n-p-1}(\alpha/2)) \\
&= P(T < -t_{n-p-1}(\alpha/2)) + P(T > t_{n-p-1}(\alpha/2)) \\
&= 2P(T > t_{n-p-1}(\alpha/2)) \\
&= 2P(T > t_{n-p-1}(\alpha/2)) \\
&= 2\frac{\alpha}{2} \\
&= \alpha.
\end{aligned}$$

This completes the proof. □

As usual with statistical tests, one needs to be extremely careful when performing several tests on the same data. In particular, it would be unwise to test more than one component of β using this procedure for the same data. Instead, in the next section we will consider how to perform tests for several components of β simultaneously. Before we do this, we will perform some experiments with R.

5.4 R Experiments

5.4.1 Fitting the model

```
m <- lm(stack.loss ~ ., data = stackloss)
summary(m)
```

```
#
# Call:
# lm(formula = stack.loss ~ ., data = stackloss)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -7.2377 -1.7117 -0.4551  2.3614  5.6978
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) -39.9197    11.8960  -3.356  0.00375 **
# Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
# Water.Temp    1.2953     0.3680   3.520  0.00263 **
# Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 3.243 on 17 degrees of freedom
# Multiple R-squared:  0.9136,  Adjusted R-squared:  0.8983
# F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

5.4.2 Estimating the Variance of the Error

We can get the design matrix X and the covariance matrix C as follows:

```
X <- model.matrix(m)
C <- solve(t(X) %*% X)
round(C, 4)
```

```
#             (Intercept) Air.Flow Water.Temp Acid.Conc.
# (Intercept)     13.4527   0.0273    -0.0620    -0.1594
# Air.Flow         0.0273   0.0017    -0.0035    -0.0007
# Water.Temp      -0.0620  -0.0035     0.0129     0.0000
# Acid.Conc.      -0.1594  -0.0007     0.0000     0.0023
```

Next we need to estimate the variance σ^2:

```
y <- stackloss$stack.loss
n <- nrow(stackloss)
p <- ncol(stackloss) - 1
hat.sigma2 <- sum((y - fitted(m))^2) / (n - p - 1)
hat.sigma2
```

```
# [1] 10.51941
```

The square root of this number, so the estimated standard deviation of the ε_i is shown as `Residual standard error` in the summary output above. We check that we get the same result:

```
sqrt(hat.sigma2)
```

```
# [1] 3.243364
```

This result is also listed as the `Residual standard error` near the bottom of the `summary(m)` output, above.

5.4.3 Estimating the Standard Errors

We can estimate the standard errors, *i.e.* the standard deviations stdev$(\hat{\beta})_i$ as $\sqrt{\hat{\sigma}^2 C_{ii}}$:

```
se <- sqrt(hat.sigma2 * diag(C))
se
```

```
# (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
#  11.8959969   0.1348582   0.3680243   0.1562940
```

These values are also listed in the `Std. Error` column of the `summary(m)` output.

5.4.4 Hypothesis tests

Let us now test the hypothesis $H_0 : \beta_i = 0$. The test statistic for this case is the following:

```
T <- coef(m) / se
T
```

```
# (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
#  -3.3557234   5.3066130   3.5195672  -0.9733098
```

These values are also listed in the `t value` column of the `summary(m)` output.

Before we can perform the test, we need to choose α and to find the corresponding critical value:

```
alpha <- 0.05
t <- qt(1 - alpha/2 , n - p - 1)
t
```

```
# [1] 2.109816
```

Using the critical value t we can decided whether H_0 should be accepted or rejected. For example, looking at the intercept β_0, we find $|T_0| = |-3.3557234| > 2.109816 = t_{n-p-1}(1-\alpha/2)$ and thus we can reject the hypothesis $H_0 : \beta_0 = 0$. This means that the intercept is significantly different from 0.

5.4.5 Confidence Intervals

Using the quantile `t` we can also get confidence intervals. Here we only show the confidence interval for the intercept β_0:

```
c(coef(m)[1] - se[1] * t, coef(m)[1] + se[1] * t)
```

```
# (Intercept) (Intercept)
#   -65.01803   -14.82131
```

Confidence intervals for the remaining coefficients can be obtained simimlarly.

Summary

- We know how to scale the distance between individual parameter estimates and the truth.
- We have seen how to construct confidence intervals for β_i.
- We have seen how to construct statistical tests for β_i.
- We have understood some more of the summary output for the `lm()` function in R.

6 Estimating Coefficients Simultaneously

In this section we will study how to assess the uncertainty in two or more regression coefficients simultaneously. This is needed since the estimates for different coefficients will usually not be independent.

6.1 Linear Combinations of Coefficients

As a general setup which allows to describe which coefficients we are interested in, we consider the image $K\beta$, where $K \in \mathbb{R}^{k\times(p+1)}$ with $k \leq p+1$.

Example 6.1. If $p = 3$ and

$$K = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

then

$$K\beta = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}.$$

Thus, this choice of K would be appropriate if we are interested in β_1 and β_2 only.

The setup allows for more general questions than just selecting components of β. We can also derive statements about linear combination of the β_i, *e.g.* we will be able to derive confidence intervals for quantities like $\beta_1 - \beta_2$.

Since $\hat{\beta}$ is an estimator for β, we can use $K\hat{\beta}$ as an estimator for $K\beta$. From lemma 4.1 we know that $\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2(X^\top X)^{-1})$ and thus we get

$$K\hat{\beta} \sim \mathcal{N}(K\beta, \sigma^2 K(X^\top X)^{-1}K^\top).$$

This shows that the estimator $K\hat{\beta}$ is unbiased.

Given an invertible matrix Q, we define the shorthand notation

$$\|v\|_Q^2 := v^\top Q^{-1} v.$$

Using this notation for $Q = K(X^\top X)^{-1}K^\top$, we define

$$F := \frac{\left\|K\hat{\beta} - K\beta\right\|^2_{K(X^\top X)^{-1}K^\top}}{k\hat{\sigma}^2} \tag{25}$$

as a measure for the distance between $K\hat{\beta}$ and $K\beta$. This quantity plays the role of T (or more precisely, T^2) from the previous section. We also need to introduce the F-distribution, which will take the place of the t-distribution from the previous section.

Definition 6.1. The F-distribution with ν_1 and ν_2 degrees of freedom is the distribution of

$$X = \frac{S_1/\nu_1}{S_2/\nu_2},$$

where S_1 and S_2 are independent random variables with chi-square distributions with ν_1 and ν_2 degrees of freedom, respectively.

With these preparations in place, we can state the main result.

Lemma 6.1. *Assume that the data follows the model* (16) *and that* $Q := K(X^\top X)^{-1}K^\top$ *is invertible. Then* $F \sim F_{k,n-p-1}$, i.e. F *follows a* F*-distribution with* k *and* $n-p-1$ *degrees of freedom.*

Proof. We have

$$\begin{aligned}
&\left\|K\hat{\beta} - K\beta\right\|_Q^2 \\
&= \left\|K(\hat{\beta} - \beta)\right\|_Q^2 \\
&= \left\|K(X^\top X)^{-1}X^\top \varepsilon\right\|_Q^2 \\
&= \varepsilon^\top X(X^\top X)^{-1}K^\top Q^{-1}K(X^\top X)^{-1}X^\top \varepsilon \\
&=: \varepsilon^\top R\varepsilon.
\end{aligned}$$

It is tedious but easy to check that R is idempotent:

$$\begin{aligned}
R^2 &= \Big(X(X^\top X)^{-1}K^\top Q^{-1}K(X^\top X)^{-1}X^\top\Big)\Big(X(X^\top X)^{-1}K^\top Q^{-1}K(X^\top X)^{-1}X^\top\Big) \\
&= X(X^\top X)^{-1}K^\top Q^{-1}K(X^\top X)^{-1}\Big(X^\top X(X^\top X)^{-1}\Big)K^\top Q^{-1}K(X^\top X)^{-1}X^\top \\
&= X(X^\top X)^{-1}K^\top \Big(Q^{-1}K(X^\top X)^{-1}K^\top\Big)Q^{-1}K(X^\top X)^{-1}X^\top \\
&= X(X^\top X)^{-1}K^\top Q^{-1}K(X^\top X)^{-1}X^\top \\
&= R.
\end{aligned}$$

When we checked in the proof of Cochran's theorem that $\varepsilon^\top H\varepsilon$ was chi-squared distributed, the only property of H we used was that H is idempotent. (If you don't remember the details, it would be a good idea to re-read the proof before continuing.) Thus, the same argument gives that $\varepsilon^\top R\varepsilon/\sigma^2$ is chi-squared distributed, and as before the number of degrees of freedom of this chi-squared distribution equals the rank of R. Using the assumption that Q is invertible, one can show (we skip this part of the proof again) that the rank of Q equals $\min(k, p+1) = k$ and thus we find that

$$S_1 := \frac{1}{\sigma^2}\left\|K\hat{\beta} - K\beta\right\|_Q^2 \sim \chi^2(k).$$

Similarly, from the direct application of Cochran's theorem in equation (22), we know

$$S_2 := \frac{1}{\sigma^2}(n-p-1)\hat{\sigma}^2 \sim \chi^2(n-p-1).$$

Since S_1 is a function of $\hat{\beta}$ and S_2 is a function of $\hat{\sigma}^2$, we can use lemma 5.1 to conclude that S_1 and S_2 are independent. Combining these results we find

$$\begin{aligned}
F &= \frac{\left\|K\hat{\beta} - K\beta\right\|^2_{K(X^\top X)^{-1}K^\top}}{k\hat{\sigma}^2} \\
&= \frac{\frac{1}{\sigma^2}\left\|K\hat{\beta} - K\beta\right\|^2_{K(X^\top X)^{-1}K^\top}/k}{\frac{1}{\sigma^2}(n-p-1)\hat{\sigma}^2/(n-p-1)} \\
&= \frac{S_1/k}{S_2/(n-p-1)} \\
&\sim F_{k,n-p-1}.
\end{aligned}$$

This completes the proof. □

6.2 Confidence Regions

Using F as a distance between the unknown true $K\beta$ and the estimator $\hat{\beta}$, it is easy to find a region of $\mathbb{R}^k$ which covers $K\beta$ with high probability. Since we now have an k-dimensional parameter vector, this region will no longer be an interval. Instead, we will get an k-dimensional ellipsoid.

6.2.1 Result

Define

$$E := \left\{ z \in \mathbb{R}^k \;\middle|\; \|z - K\hat{\beta}\|_{K(X^\top X)^{-1}K^\top} \leq \sqrt{k\hat{\sigma}^2 f_{k,n-p-1}(\alpha)} \right\},$$

where $f_{k,n-p-1}(\alpha)$ is the $(1-\alpha)$-quantile of the $F_{k,n-p-1}$-distribution. This is a "ball" around $K\hat{\beta}$ in $\mathbb{R}^k$, where distance is measured using the norm $\|\cdot\|_{K(X^\top X)^{-1}K^\top}$ introduced above. The following lemma shows that $\sqrt{k\hat{\sigma}^2 f_{k,n-p-1}(\alpha)}$ is the correct choice of "radius" to make the ball cover the true value $K\beta$ with probability $1-\alpha$.

Lemma 6.2. *We have*

$$P(K\beta \in E) = 1 - \alpha,$$

i.e. *the set E is a $(1-\alpha)$-confidence region for $K\beta$.*

Proof. We have

$$\begin{aligned}
& K\beta \in E \\
& \quad \iff \|K\beta - K\hat{\beta}\|_{K(X^\top X)^{-1}K^\top} \leq \sqrt{k\hat{\sigma}^2 f_{k,n-p-1}(\alpha)} \\
& \quad \iff \|K\hat{\beta} - K\beta\|^2_{K(X^\top X)^{-1}K^\top} \leq k\hat{\sigma}^2 f_{k,n-p-1}(\alpha) \\
& \quad \iff \frac{\|K\hat{\beta} - K\beta\|^2_{K(X^\top X)^{-1}K^\top}}{k\hat{\sigma}^2} \leq f_{k,n-p-1}(\alpha) \\
& \quad \iff F \leq f_{k,n-p-1}(\alpha)
\end{aligned}$$

Now the claim follows, since $f_{k,n-p-1}(\alpha)$ is the $(1-\alpha)$-quantile of F. □

6.2.2 Numerical Experiments

We start by fitting a linear model to the `stackloss` dataset as before:

```
m <- lm(stack.loss ~ ., data = stackloss)
summary(m)
```

```
#
# Call:
# lm(formula = stack.loss ~ ., data = stackloss)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -7.2377 -1.7117 -0.4551  2.3614  5.6978
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) -39.9197    11.8960  -3.356  0.00375 **
# Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
# Water.Temp    1.2953     0.3680   3.520  0.00263 **
# Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 3.243 on 17 degrees of freedom
# Multiple R-squared:  0.9136,  Adjusted R-squared:  0.8983
# F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

Here I want to consider the two regressions coefficients β_1 (`Air.Flow`) and β_2 (`Water.Temp`). For this we need to construct a matrix K with two rows, where each row selects one of the two coefficients:

```
X <- model.matrix(m)
n <- nrow(X)
p <- ncol(X) - 1

K <- matrix(c(0, 1, 0, 0,    # indices in R start at 1, so beta_1 is col. 2
              0, 0, 1, 0),   # ... and beta_2 is column 3.
            byrow = TRUE,
            nrow = 2, ncol = 4)
k <- nrow(K)
K.beta.hat <- as.vector(K %*% coef(m))
```

As we have seen in the previous section, for this K the covariance matrix for the ellipse is $Q = K(X^\top X)^{-1}K^\top$:

```
Q <- K %*% solve(t(X) %*% X) %*% t(K)
```

6.2.2.1 A Single Point To try out the method, we first consider the test point $m = K\beta = (1, 1)$ and we compute the F value for this point to see whether the point is inside or outside the ellipse:

```
a <- c(1, 1)

sigma.hat <- summary(m)$sigma
d <- a - K.beta.hat
F <- t(d) %*% solve(Q, d) / (k * sigma.hat^2)
F
```

```
#          [,1]
# [1,] 2.834083
```

The F value, measuring the distance from the centre of the ellipse is 2.834 for this test point. This has to be compared to the critical value:

```
alpha <- 0.05
f.crit <- qf(1 - alpha, k, n - p - 1)
f.crit
```

```
# [1] 3.591531
```

Since the F value is less than the critical value, the point $(1, 1)$ is inside the ellipse. As the next step, we will plot a picture of the full ellipse.

6.2.2.2 Points on a Grid An easy way to show the ellipse is to repeat the above procedure for all points on a rectangular grid, and then colour the points depending on whether they are inside or outside the ellipse. We start by making a list of grid points.

```
x.min <- -1
x.max <- 3
y.min <- -0.5
y.max <- 3
L <- 200

xx <- seq(x.min, x.max, length.out = L)
yy <- seq(y.min, y.max, length.out = L)

Z <- as.matrix(expand.grid(x = xx - K.beta.hat[1],
```

```
                              y = yy - K.beta.hat[2],
                              KEEP.OUT.ATTRS = FALSE))
dim(Z)
```

```
# [1] 40000     2
```

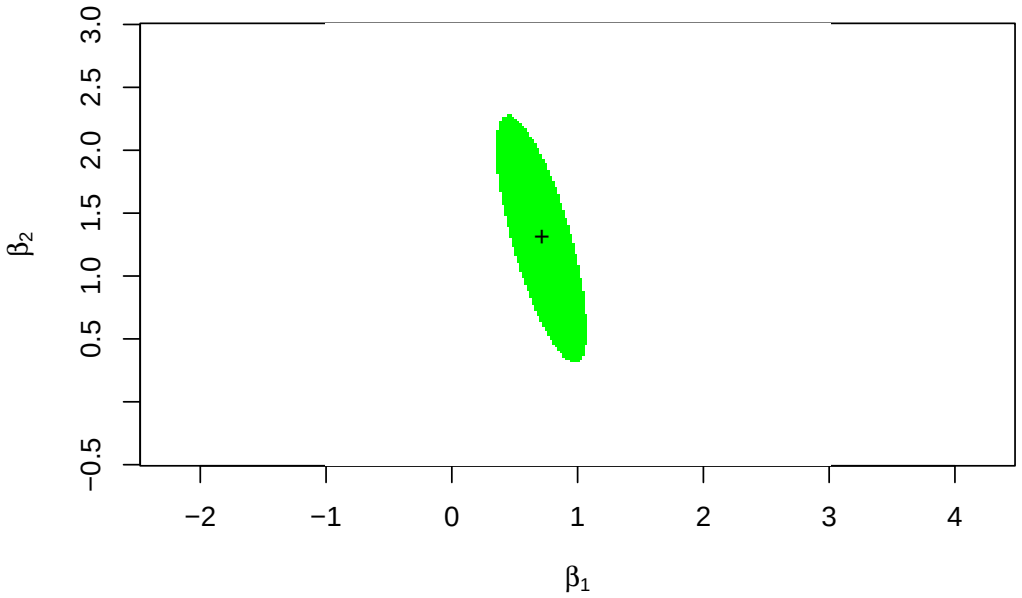
The matrix Z now has two columns, containing the x and y coordinates respectively of the 200×200 points in our grid. Now we need to compute the F value for every grid point:

```
F <- rowSums(Z * t(solve(Q, t(Z)))) / (k * sigma.hat^2)
F <- matrix(F, byrow = TRUE, L, L)
dim(F)
```

```
# [1] 200 200
```

The resulting matrix contains the F value for every grid point. Finally, we can mark all the points where F exceeds the critical value in a plot:

```
image(x = xx, y = yy, t(F > f.crit), asp = 1,
      col = c("green", "white"),
      xlab = expression(beta[1]), ylab = expression(beta[2]))
points(K.beta.hat[1], K.beta.hat[2], pch = "+")
```



The green ellipse in this plot is the 95% confidence ellipse for the vector (β_1, β_2).

6.2.2.3 Coordinates of the Outline This sub-section is non-examinable (but hopefully interesting).

Using some more linear algebra, we can find a formula for the coordinates of the points on the ellipse which forms the boundary of the confidence region. For this, we use the Singular Value Decomposition of the matrix Q. This allows to write Q as

$$Q = UDV^\top$$

where U and V are orthogonal matrices and D is a diagonal matrix:

```
svd <- La.svd(Q)
svd
```

```
# $d
# [1] 0.0138678012 0.0007364967
#
# $u
#            [,1]      [,2]
# [1,] -0.2749061 0.9614711
# [2,]  0.9614711 0.2749061
#
# $vt
#            [,1]      [,2]
# [1,] -0.2749061 0.9614711
# [2,]  0.9614711 0.2749061
```

The R output shows the diagonal elements d_{ii} of D and the matrices U and $V^\top$. Since Q is symmetric, we have $U = V$ and thus $Q = UDU^\top$ in this case, *i.e.* we have found a diagonalisation of Q.

Using the SVD we can simplify the norm used in the definition of the ellipse. We write $D^{-1/2}$ for the matrix which has $1/\sqrt{d_{ii}}$ on the diagonal. Then we have

$$\begin{aligned} Q(D^{-1/2}U^\top)^\top(D^{-1/2}U^\top) &= QUD^{-1/2}D^{-1/2}U^\top \\ &= UDU^\top UD^{-1}U^\top \\ &= UDD^{-1}U^\top \\ &= UU^\top \\ &= I. \end{aligned}$$

Thus, $(D^{-1/2}U^\top)^\top(D^{-1/2}U^\top)$ is the inverse of Q and we get

$$\begin{aligned} \|z - K\hat{\beta}\|_Q^2 &= (z - K\hat{\beta})^\top Q^{-1}(z - K\hat{\beta}) \\ &= (z - K\hat{\beta})^\top (D^{-1/2}U^\top)^\top (D^{-1/2}U^\top)(z - K\hat{\beta}) \\ &= \left\| D^{-1/2}U^\top (z - K\hat{\beta}) \right\|^2, \end{aligned}$$

where the norm on the right-hand side is the usual Euclidean norm. Thus, the boundary of the ellipse consists of all points of the form $K\hat{\beta} + d$, where $e := D^{-1/2}U^\top d$ is a vector of length

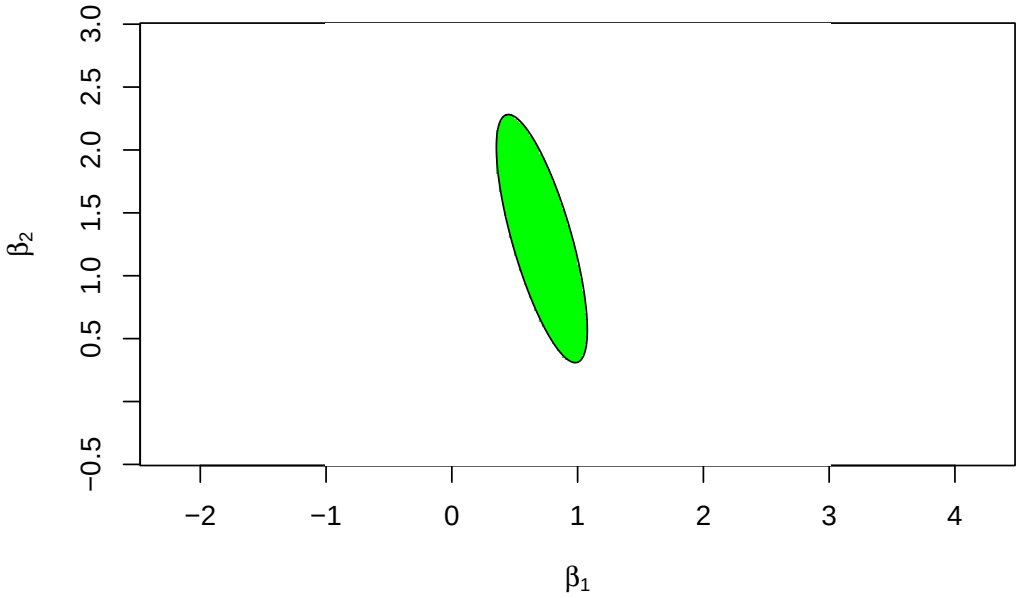
$$\|e\| = \sqrt{k\hat{\sigma}^2 f_{k,n-p-1}(\alpha)}.$$

Finally, using polar coordinates, we find the points on the boundary to be

$$d = UD^{1/2}e = UD^{1/2}\sqrt{k\hat{\sigma}^2 f_{k,n-p-1}(\alpha)} \begin{pmatrix} \cos(\varphi) \\ \sin(\varphi) \end{pmatrix}$$

with $\varphi \in [0, 2\pi]$. This allows to plot the boundary line in R.

```
phi <- seq(0, 2*pi, length.out = 201)
circ <- rbind(cos(phi), sin(phi)) * sqrt(f.crit * k * sigma.hat^2)

ellipse <- svd$u %*% (circ * sqrt(svd$d)) + K.beta.hat

image(x = xx, y = yy, t(F > f.crit), asp = 1,
      col = c("green", "white"),
      xlab = expression(beta[1]), ylab = expression(beta[2]))
lines(ellipse[1,], ellipse[2,])
```



To verify that everything is consistent, we have plotted the line on top of the image from the previous subsection. Since the black line perfectly surrounds the green area, both approaches are consistent.

6.3 Hypothesis Tests

We can easily derive a hypothesis test to test the hypothesis

$$H_0 \colon K\beta = m$$

against the alternative

$$H_1 \colon K\beta \neq m,$$

where $m \in \mathbb{R}^k$.

We redefine F as

$$F := \frac{\left\|K\hat{\beta} - m\right\|^2_{K(X^\top X)^{-1}K^\top}}{k\hat{\sigma}^2}$$

using m in place of the $K\beta$ above. Then the new definition of F is the same as (25) if H_0 is true.

Lemma 6.3. *The test which rejects H_0 if and only if $|F| > f_{k,n-p-1}(\alpha)$ has significance level α.*

Proof. Assume that H_0 is true. Then we have $K\beta = m$ and thus the F defined in this section coincides with the expression from equation (25). From lemma 6.1 we know that $F \sim F_{k,n-p-1}$. Thus we have

$$\begin{aligned} P(\text{type I error}) &= P(F > f_{k,n-p-1}(\alpha)) \\ &= 1 - P(F \leq f_{k,n-p-1}(\alpha)) \\ &= 1 - (1-\alpha) \\ &= \alpha. \end{aligned}$$

This completes the proof. □

Summary

- We have learned how to measure the distance between a subset of coefficients and the corresponding estimates.
- We have introduced the F-distribution.
- We have learned how to compute confidence regions for subsets of the regression coefficients.
- We have learned how to perform hypothesis tests for subsets of the regression coefficients.

Interlude: Loading Data into R

Before we can analyse data using R, we have to "import" the data into R. How exactly this is done depends on how the data is stored, and more specifically on which file format is used. Here we consider two commonly used formats: comma-separated values (`.csv`) and Microsoft Excel files (`.xls` or `.xlsx`).

Importing CSV Files

The `read.csv()` command can be used to import `.csv` files into R: if we use the command

```
x <- read.csv("file.csv")
```

then the contents of the file `file.csv` will be stored in the variable `x`. Optional arguments to `read.csv()` can be used to specify whether or not the file includes column names, and allow to deal with variations of how the data may be stored in the file. Some of these are explained below.

The most important things to know about the function `read.csv()` are:

- The filename can either denote a file on the local computer, or a file available for download from the internet. If you want R to read the file directly from the internet, replace the file name with the web address (starting with `http://` or `https://`).
- If the input file is on the local computer, you may need to change R's current directory to the directory where the file is stored before calling `read.csv()`. In RStudio you can use the menu "Session > Set Working Directory > Choose Directory …" to do this. If you are not sure what the filename for a given input file is, you can use the command `file.choose()` in place of the file name, to choose a file interactively.
- By default, R uses the first row of the `.csv` file to set the column names of the data frame and assumes that the actual data starts in row 2 of the `.csv` file. If the file does not contain column names, you can use the `header=FALSE` option with `read.csv()` to tell R that the column names are not included in the data:

  ```
  x <- read.csv("file.csv", header=FALSE)
  ```

  You can see whether this option is needed by opening the file in Excel and looking whether the first row contains headers or not. Alternatively you can inspect the column names and the contents of the first data row in R to see whether everything looks right after importing the data.
- R uses a special data type called `Factor` to represent categorical data. If the `.csv` file contains such data, the option `stringsAsFactors=TRUE` can be used to automatically convert text to factors. (This option used to be the default for R versions before 4.0.0.)
- Sometimes, the columns in a `.csv` file are separated not by a comma, but using a semicolon instead. In this case you need to use the option `sep=";"` when you import the data:

  ```
  x <- read.csv("file.csv", sep=";")
  ```
- Missing values should be represented by empty cells in the `.csv` file and are represented as special `NA` values in the data frame. If the `.csv` file uses a different encoding for missing values, the `na.strings` option can be used to tell `read.csv()` which cell values should be interpreted as missing values. For example, `read.csv("file.csv", na.strings=c("", "-"))` can be used for a file

where missing values are indicated by either empty cells or cells containing a hyphen.

Further details about the function `read.csv()` can be found using the command `help(read.csv)` in R.

Example 6.2. In the 2016 version of the MATH1712 module, I performed a small questionnaire in the first lecture. The following R command can be used to load the data from the questionnaire into R

```
x <- read.csv("https://www1.maths.leeds.ac.uk/~voss/rintro/Q2016.csv")
```

The variable `x` now contains the questionnaire results. Instead of directly downloading the data from the internet into R, you can alternatively first download the data using a web browser, and then import the data directly from your computer:

```
x <- read.csv("Q2016.csv")
```

Both approaches give the same result.

Importing Microsoft Excel Files

The easiest way to import Excel files into R is to first convert these files to `.csv` format. To do this:

- Open the file with the data in Excel.
- Open a new, empty file (choosing "Blank workbook" in Excel).
- Copy and paste the relevant cells into the empty file. It is important to just copy the required data and to leave out any explanatory text and and empty rows/columns. The data must form a tidy rectangle, with one sample per row and one variate per column. Optionally, you can put column headers into the first row.
- Save the new file in `.csv` format in a folder where you will find it again.
- Read the resulting `.csv` into R as explained above.

Alternatively, you can try the `read_excel()` function from the `readxl` package. (You may need to install this R package first.) You can learn how to use this function at https://readxl.tidyverse.org/ or using the command `help(read_excel)`. Note that these functions return a "tibble" instead of a data frame, so some additional knowledge is required to use the result.

Checking the Imported Data

The following commands can be used to get a quick overview over the data:

- `dim(x)` gives the number of rows and columns of the data frame. Similarly, `nrow(x)` shows the number of rows and `ncol(x)` shows the number of columns in the data frame.
- `str(x)` shows the structure of any R object. This command is often an excellent way to understand the nature of any problems one may encounter while importing data into R.
- `summary(x)` prints, for each variate, the values of various summary statistics. For variates with numeric values, these are the minimum, first quartile, median, mean, third quartile, maximum, and the number of missing values. For attribute data this gives the counts for each observed value.
- `head(x)` shows the first few rows of the data.

Every time you import a dataset into R, you should use some of these commands to check that everything went well. In case you discover problems, you should either fix the data file (*e.g.* using Microsoft Excel) or by using the correct options for the `read.csv()` command.

Example 6.3. Continuing with the dataset from example above, we can try the following commands: We first check that the data has plausible dimensions:

```
dim(x)
```

```
# [1] 220   5
```

This tells us that the data has $n = 220$ rows and $p = 5$ columns, as expected. Next, we get some details about the five columns of the data frame:

```
str(x)
```

```
# 'data.frame': 220 obs. of  5 variables:
#  $ gender     : chr  "M" "M" "M" "M" ...
#  $ height     : num  180 183 183 182 164 ...
#  $ handedness : chr  "R" "R" "R" "R" ...
#  $ travel.time: int  20 25 20 12 5 20 15 10 11 7 ...
#  $ R.skills   : chr  "basic" "basic" "basic" "basic" ...
```

This shows, for each column, the name, the "data type" and the first few values. The "data type" is a good indicator to detect problems; it should read `int` for integer valued numeric data, `num` for all other numeric data, and `Factor` for attribute data. For attribute data, the range of observed values is also shown. For our dataset, we see that the fields `$gender`, `$handedness` and `$R.skills` are represented as text data (type `chr`) instead of categorical data. To fix this, we re-import the data using the `stringsAsFactors=TRUE` option:

```
x <- read.csv("data/Q2016.csv", stringsAsFactors=TRUE)
str(x)
```

```
# 'data.frame': 220 obs. of  5 variables:
#  $ gender     : Factor w/ 2 levels "F","M": 2 2 2 2 1 1 1 1 1 2 ...
#  $ height     : num  180 183 183 182 164 ...
#  $ handedness : Factor w/ 3 levels "L","R","both": 2 2 2 2 2 2 2 2 2 2 ...
#  $ travel.time: int  20 25 20 12 5 20 15 10 11 7 ...
#  $ R.skills   : Factor w/ 4 levels "basic","good",..: 1 1 1 1 1 1 1 3 1 1 ...
```

Now the output is as expected, and the correct "levels" are shown for the categorical variables.

Finally, we can print summary statistics for all columns:

```
summary(x)
```

```
#  gender      height      handedness  travel.time       R.skills
#  F:102   Min.   :147.0   L   : 18   Min.   :  0.00   basic :157
#  M:118   1st Qu.:167.0   R   :201   1st Qu.:  5.00   good  :  4
#          Median :175.0   both:  1   Median : 15.00   medium: 28
#          Mean   :173.7              Mean   : 19.15   none  : 31
#          3rd Qu.:180.3              3rd Qu.: 25.00
#          Max.   :195.6              Max.   :150.00
#          NA's   :23
```

None of these commands showed any remaining problems, so we are now ready to use the data.

Common Problems

There are many things which can go wrong when importing data. Some commonly encountered problems are the following:

- If the data file contains no line with headers, but the `header=FALSE` option for `read.csv()` has been omitted, the first row of data will be used in place of the column names. This can, for example, be seen in the `str()` output. The solution to this problem is to correctly use the `header=FALSE` option.
- If the data in a `.csv` file is not separated by commas but by some other character like a semicolon, R will be unable to separate the columns. When this is the case, the imported data will appear to only have one column, where each entry shows as some garbled version of the data for the whole row. The solution to this problem is to use the `sep=...` option.
- If attribute values are encoded inconsistently, *e.g.* if a mix of `m` and `M` is used to encode the gender "male", this will be visible in the `str()` output. One solution to this problem is to fix the `.csv` file using Microsoft Excel, before trying to import it into R again.
- If a numeric column in the input file contains one or more entries which are neither numbers nor empty, R will interpret the whole column as attribute data. This problem can be detected in the `str()` output, when a numeric column is listed as having data type `Factor`. One solution to this problem is to use Microsoft Excel to remove or fix the offending entry.

Problem Sheet 2

You should attempt all these questions and write up your solutions in advance of the workshop in week 4 where the answers will be discussed.

5. Assume that $X \sim \mathcal{N}(0,1)$, $Y \sim \chi^2(2)$ and $Z \sim \chi^2(3)$ are independent. What are the distributions of the following random variables?

- X^2,
- $X^2 + Y + Z$,
- $\dfrac{\sqrt{2}X}{\sqrt{Y}}$,
- $\dfrac{2X^2}{Y}$, and
- $1.5\,Y/Z$.

Solution:

The distributions involved in this question are the normal distribution, the χ^2-dis-tri-bu-tion, the t-distribution and the F-distribution. The general rules are: (1) the sum of squares of d independent, standard normally distributed random variables follows a $\chi^2(d)$-distribution. (2) If $Z \sim \mathcal{N}(0,1)$ and $V \sim \chi^2(d)$ are independent, then $Z/\sqrt{V/d} \sim t(d)$, and finally (3) If $V_1 \sim \chi^2(d_1)$ and $V_2 \sim \chi^2(d_2)$ are independent, then $(V_1/d_1)/(V_2/d_2) \sim F(d_1, d_2)$. Using these rules:

- $X^2 \sim \chi^2(1)$,
- $X^2 + Y + Z \sim \chi^2(1+2+3) = \chi^2(6)$,
- $\dfrac{\sqrt{2}X}{\sqrt{Y}} = \dfrac{X}{\sqrt{Y/2}} \sim t(2)$,
- $\dfrac{2X^2}{Y} = \dfrac{X^2/1}{Y/2} \sim F(1,2)$,
- $1.5\,Y/Z = \dfrac{Y/2}{Z/3} \sim F(2,3)$.

6. Let $\mathbf{1} = (1, 1, \ldots, 1) \in \mathbb{R}^n$ and let $X \sim \mathcal{N}(\mu, \sigma^2 I)$ be a normally-distributed random vector, where I is the $n \times n$ identity matrix. Define $A = \frac{1}{n}\mathbf{1}\mathbf{1}^\top \in \mathbb{R}^{n\times n}$.

- Show that $(AX)_i = \bar{X}$ for all $i \in \{1, \ldots, n\}$, where $\bar{X} = \frac{1}{n}\sum_{j=1}^n X_j$.
- Show that A is symmetric.
- Show that A is idempotent.
- Use results from lectures to conclude that the random variables AX and $X^\top(I - A)X$ are independent.
- Using the previous parts of the question, show that $\bar{X}$ and $\frac{1}{n-1}\sum_{i=1}^n (X_i - \bar{X})^2$ are independent.

Solution:

The matrix $A = (a_{ij})$ has entries $a_{ij} = 1/n$ for all $i, j \in \{1, \ldots, n\}$. Using this observation, the statements of the first few questions follow easily:

- We have

$$(Ax)_i = \sum_{j=1}^n a_{ij}x_j = \sum_{j=1}^n \frac{1}{n}x_j = \frac{1}{n}\sum_{j=1}^n x_j = \bar{x}$$

for all $i \in \{1, \dots, n\}$.

- We have
$$a_{ij} = \frac{1}{n} = a_{ji}$$
for all $i, j \in \{1, \dots, n\}$ and thus A is symmetric.

- We have $\mathbf{1}^\top \mathbf{1} = \sum_{i=1}^n 1 \cdot 1 = n$ and thus
$$A^2 = (\frac{1}{n}\mathbf{1}\mathbf{1}^\top)(\frac{1}{n}\mathbf{1}\mathbf{1}^\top) = \frac{1}{n^2}\mathbf{1}(\mathbf{1}^\top\mathbf{1})\mathbf{1}^\top = \frac{1}{n}\mathbf{1}\mathbf{1}^\top = A.$$
(Alternatively, we could use the fact that we know the entries of A and compute A^2 explicitly.)

- For the remaining statements we will use the fact that X is normally distributed. We will repeatedly use the result that if two random variables X and Y are independent, that then any function of X is independent of any function of Y.

  In lectures we learned that, if A is symmetric and idempotent and if $\varepsilon \sim \mathcal{N}(0, I)$, then $A\varepsilon$ and $(I - A)\varepsilon$ are independent. Applying this result with $\varepsilon = (X - \mu\mathbf{1})/\sigma$ we find that $AX = \sigma A\varepsilon + \mu A\mathbf{1}$ and $(I - A)X = \sigma(I - A)\varepsilon + \mu(I - A)\mathbf{1}$ are independent, since they are functions of $A\varepsilon$ and $(I - A)\varepsilon$.

  Since $(I - A)^\top(I - A) = I^2 - IA - AI + A^2 = I - A - A + A = I - A$ we have
$$X^\top(I - A)X = X^\top(I - A)^\top(I - A)X = \Big((I - A)X\Big)^\top\Big((I - A)X\Big).$$
Thus, $X^\top(I - A)X$ is a function of $(I - A)X$ and as such is also independent of AX.

- We have seen $(AX)_i = \bar{X}$ and thus $\big((I - A)X\big)_i = X_i - \bar{X}$. This gives
$$\begin{aligned} X^\top(I - A)X &= \Big((I - A)X\Big)^\top\Big((I - A)X\Big) \\ &= \sum_{i=1}^n \Big((I - A)X\Big)_i^2 \\ &= \sum_{i=1}^n (X_i - \bar{X})^2 \end{aligned}$$
Thus we can write the sample mean as $\bar{X} = (AX)_i$, and the sample variance as $s_X^2 = \frac{1}{n-1}\sum_{i=1}^n (X_i - \bar{X})^2 = X^\top(I - A)X/(n-1)$, and if X is normally distributed the two values $\bar{X}$ and s_X^2 are independent.

7. Consider the following R commands:

```
m <- lm(stack.loss ~ ., data=stackloss)
summary(m)
```

```
#
# Call:
# lm(formula = stack.loss ~ ., data = stackloss)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -7.2377 -1.7117 -0.4551  2.3614  5.6978
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
```

```
# (Intercept) -39.9197    11.8960  -3.356  0.00375 **
# Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
# Water.Temp    1.2953     0.3680   3.520  0.00263 **
# Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 3.243 on 17 degrees of freedom
# Multiple R-squared:  0.9136,  Adjusted R-squared:  0.8983
# F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

Either using this output, or using R to further inspect the built-in `stackloss` dataset, find a 99%-confidence interval for the parameter $\beta_{\texttt{Acid.Conc.}}$.

Solution: From lectures, we know that a confidence interval for a single coefficient β_j is given by

$$[U, V] = \left[\hat{\beta}_j - t_{n-p-1}(\alpha/2)\sqrt{\widehat{\sigma^2}C_{jj}}, \hat{\beta}_j + t_{n-p-1}(\alpha/2)\sqrt{\widehat{\sigma^2}C_{jj}}\right],$$

where t_{n-p-1} is the $(1-\alpha/2)$-quantile of the t-distribution, $C_{jj} = (X^\top X)^{-1}_{jj}$, and X is the design matrix.

We can read off the required values for computing the confidence interval from the output of `summary(m)`: the centre of the confidence interval can be found in the column `Estimate`, the standard error $\sqrt{\widehat{\sigma^2}C_{jj}}$ is given in column `Std. Error`, and the value $n-p-1$ for the t-quantile can be found as the `degrees of freedom` for the residual standard error, near the bottom of the output. Using these values, we can find a 95%-confidence interval for `Acid.Conc.` as follows:

```
c(-0.1521 - qt(0.995, 17) * 0.1563, -0.1521 + qt(0.995, 17) * 0.1563)
```

```
# [1] -0.6050934  0.3008934
```

Thus, the confidence interval is $[-0.605, 0.301]$.

8. The file `20211020.csv` from

- https://www1.maths.leeds.ac.uk/~voss/data/20211020.csv

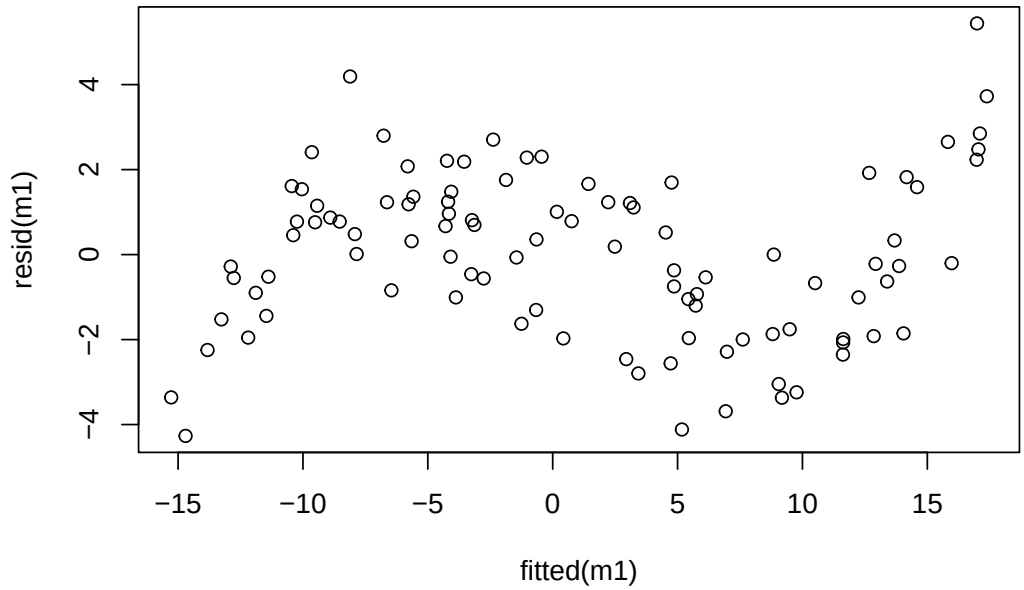
contains pairs of (x, y) values. Fit the following models to the data:

- $y = \beta_0 + \beta_1 x + \varepsilon$
- $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$
- $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \varepsilon$

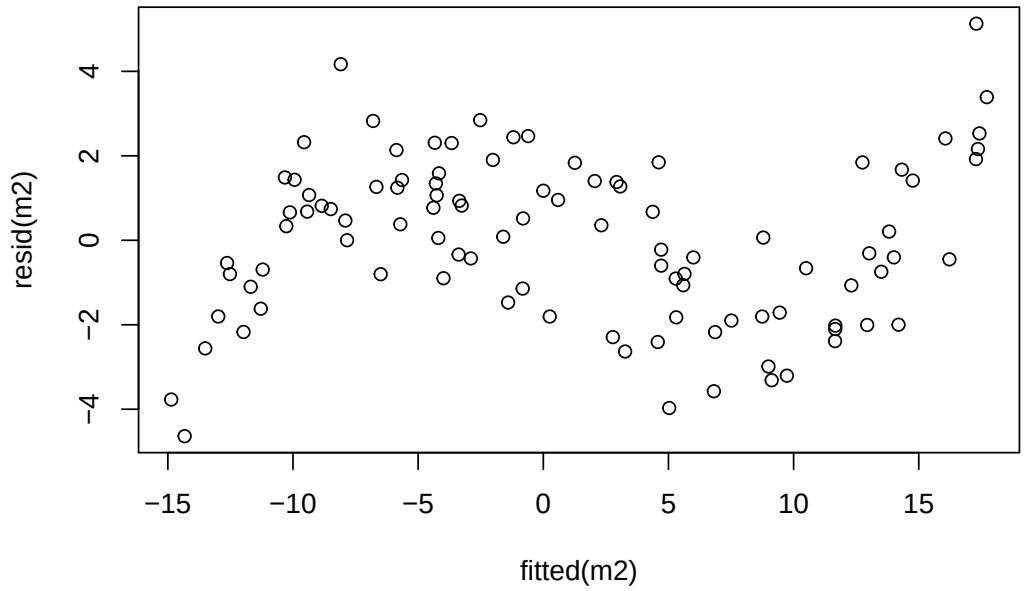
For each model, create a "residual plot", *i.e.* a scatter plot which has the fitted values on the horizontal axis and the residuals on the vertical axis. Comment on the resulting plots.

Solution:

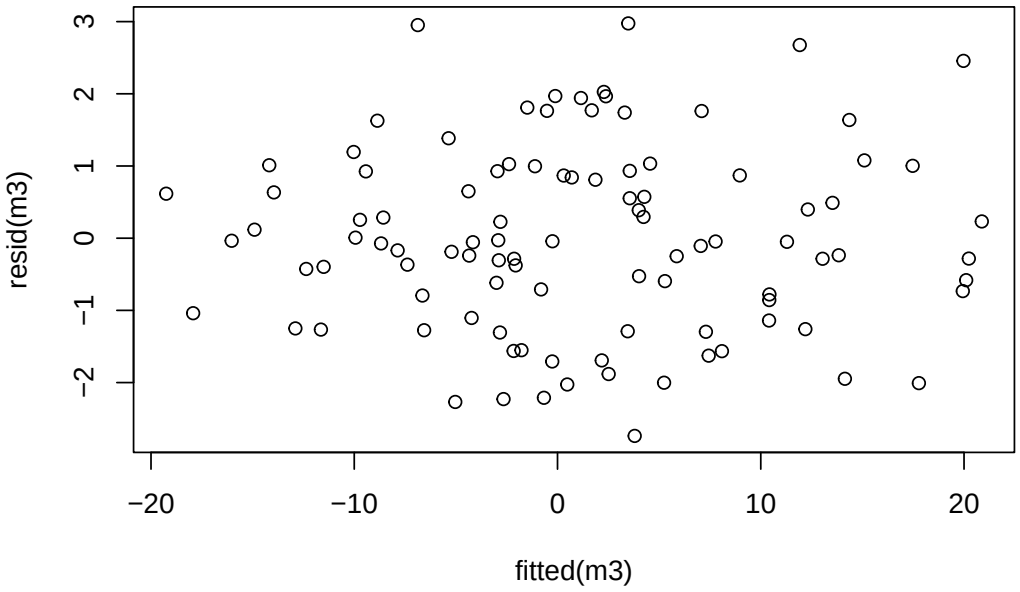
```
# data from https://www1.maths.leeds.ac.uk/~voss/data/20211020.csv
d <- read.csv("data/20211020.csv")

m1 <- lm(y ~ x, data = d)
plot(fitted(m1), resid(m1))
```




```
m2 <- lm(y ~ x + I(x^2), data = d)
plot(fitted(m2), resid(m2))
```




```
m3 <- lm(y ~ x + I(x^2) + I(x^3), data = d)
plot(fitted(m3), resid(m3))
```


resid(m3)
3
2
1
0
−1
−2
−20
−10
0
10
20
fitted(m3)

7 Examples

To illustrate the results of the last two sections, we consider a series of examples.

7.1 Simple Confidence Interval

In this subsection we will illustrate how to compute a simple confidence interval for a single regression coefficient. We use a dataset about toxicity of chemicals towards aquatic life from the from the UCI Machine Learning Repository. The data is available at

- https://archive.ics.uci.edu/ml/datasets/QSAR+aquatic+toxicity

There are nine different variables. (The web page explains the interpretation of these variables.) We will fit a model which describes `LC50` as a function of the remaining variables:

```
# data from https://archive.ics.uci.edu/ml/datasets/QSAR+aquatic+toxicity
qsar <- read.csv("data/qsar_aquatic_toxicity.csv", sep = ";", header = FALSE)
names(qsar) <- c(
    "TPSA",
    "SAacc",
    "H050",
    "MLOGP",
    "RDCHI",
    "GATS1p",
    "nN",
    "C040",
    "LC50"
)
m <- lm(LC50 ~ ., data = qsar)
summary(m)
```

```
#
# Call:
# lm(formula = LC50 ~ ., data = qsar)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -4.4934 -0.7579 -0.1120  0.5829  4.9778
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept)  2.698887   0.244554  11.036  < 2e-16 ***
# TPSA         0.027201   0.002661  10.220  < 2e-16 ***
# SAacc       -0.015081   0.002091  -7.211 1.90e-12 ***
# H050         0.040619   0.059787   0.679 0.497186
# MLOGP        0.446108   0.063296   7.048 5.60e-12 ***
# RDCHI        0.513928   0.135565   3.791 0.000167 ***
# GATS1p      -0.571313   0.153882  -3.713 0.000227 ***
# nN          -0.224751   0.048301  -4.653 4.12e-06 ***
# C040         0.003194   0.077972   0.041 0.967340
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 1.203 on 537 degrees of freedom
# Multiple R-squared:  0.4861,  Adjusted R-squared:  0.4785
# F-statistic:  63.5 on 8 and 537 DF,  p-value: < 2.2e-16
```

Our aim is to find a 95% confidence interval for `GATS1p`.

7.1.1 From First Principles

To compute the confidence interval from first principles, we can use Lemma 5.3. The centre of the confidence interval is given by $\widehat{\beta}_i$:

```
mid <- coef(m)[7]
mid
```

```
#     GATS1p
# -0.5713131
```

To find the half-width of the interval, we need to first construct the covariance matrix C and the estimated residual variance $\hat{\sigma}^2$. We will also need the corresponding quantile of the t-distribution.

```
X <- model.matrix(m)
y <- qsar$LC50
n <- nrow(X)
p <- ncol(X) - 1

Cii <- solve(t(X) %*% X)[7, 7]
cat("Cii =", Cii, "\n")
```

```
# Cii = 0.01637454
```

```
sigma.hat.squared <- sum((fitted(m) - y)^2) / (n - p - 1)
cat("hat.sigma.square =", sigma.hat.squared, "\n")
```

```
# hat.sigma.square = 1.446134
```

```
alpha <- 0.05
t <- qt(1 - alpha/2, n - p - 1)
cat("t =", t, "\n")
```

```
# t = 1.964391
```

With this information in place, we can easily find the confidence interval:

```
d <- sqrt(Cii * sigma.hat.squared) * t
cat("[", mid - d, ", ", mid + d, "]\n", sep = "")
```

```
# [-0.8735983, -0.2690279]
```

7.1.2 From the `lm()` Output

We can obtain the same result using only the quantile t and numbers shows in the `summary(m)` output. We focus on the row corresponding to `GATS1p` in the output shown above:

```
GATS1p       -0.571313   0.153882  -3.713 0.000227 ***
```

The first number in this row is the centre of the confidence interval, the second number corresponds to $\sqrt{\hat{\sigma}^2 C_{ii}}$. To have everything in one place, we copy the commands for computing t:

```
alpha <- 0.05
t <- qt(1 - alpha/2, n - p - 1)
cat("[", -0.571313 - 0.153882*t, ", ", -0.571313 + 0.153882*t, "]\n", sep = "")
```

```
# [-0.8735975, -0.2690285]
```

This is the same result as above.

7.2 Confidence Intervals for the Mean

So far we have only seen how to compute confidence intervals for the regression coefficients β_i. Using the same techniques we can also compute confidence intervals for the model mean corresponding to an input $(\tilde{x}_1, \dots, \tilde{x}_p)$. If we write

$$\tilde{x} = (1, \tilde{x}_1, \dots, \tilde{x}_p),$$

then the model mean corresponding to this input is

$$\tilde{y} = \beta_0 + \tilde{x}_1\beta_1 + \cdots + \tilde{x}_p\beta_p = \tilde{x}^\top\beta,$$

To derive a confidence interval for this quantity, we follow the same steps as we did when we derived a confidence interval for $\hat{\beta}_i$.

Our estimator for this $\tilde{y}$ is

$$\hat{y} = \tilde{x}^\top\hat{\beta} \sim \mathcal{N}(\tilde{x}^\top\beta, \sigma^2\tilde{x}^\top(X^\top X)^{-1}\tilde{x}).$$

Thus,

$$\begin{aligned}\tilde{T} &:= \frac{\tilde{x}^\top\hat{\beta} - \tilde{x}^\top\beta}{\sqrt{\hat{\sigma}^2\tilde{x}^\top(X^\top X)^{-1}\tilde{x}}} \\ &= \frac{\frac{1}{\sqrt{\sigma^2\tilde{x}^\top(X^\top X)^{-1}\tilde{x}}}(\tilde{x}^\top\hat{\beta} - \tilde{x}^\top\beta)}{\sqrt{\frac{1}{\sigma^2}\hat{\sigma}^2}}\end{aligned}$$

is $t(n-p-1)$-distributed, since the numerator is $\mathcal{N}(0,1)$, the denominator is $\sqrt{\chi^2(n-p-1)/(n-p-1)}$ and $\hat{\beta}$ is independent of $\hat{\sigma}^2$. Let $t_{n-p-1}(\alpha/2)$ be the $(1-\alpha/2)$-quantile of the $t(n-p-1)$-distribution. Then we can solve the inequality $|T| \leq t_{n-p-1}(\alpha/2)$ for $\tilde{x}^\top\beta$ to find the required confidence interval:

$$\left[\tilde{x}^\top\hat{\beta} - \sqrt{\hat{\sigma}^2\tilde{x}^\top(X^\top X)^{-1}\tilde{x}}t_{n-p-1}(\alpha/2), \tilde{x}^\top\hat{\beta} + \sqrt{\hat{\sigma}^2\tilde{x}^\top(X^\top X)^{-1}\tilde{x}}t_{n-p-1}(\alpha/2)\right].$$

We illustrate this using a numerical example. For the `stackloss` dataset with input values `Air.Flow = 60`, `Water.Temp = 21` and `Acid.Conc = 87`, we can get the following results:

```
m <- lm(stack.loss ~ ., data = stackloss)
X <- model.matrix(m)
n <- nrow(X)
p <- ncol(X) - 1

tilde.x <- c(1, 60, 21, 87)
hat.beta <- coef(m)

c <- t(tilde.x) %*% solve(t(X) %*% X, tilde.x)
sigma.hat.squared <- sum((fitted(m) - stackloss$stack.loss)^2) / (n - p - 1)
t <- qt(0.975, n - p - 1)

mid <- t(tilde.x) %*% hat.beta
d <- sqrt(c * sigma.hat.squared) * t
cat("[", mid - d, ", ", mid + d, "]\n", sep = "")
```

```
# [15.46463, 18.50554]
```

7.3 Testing a Single Coefficient

Continuing with the dataset from the previous example, we can test whether the regression coefficient corresponding to the water temperature might equal zero. We will perform a test at 5%-level.

7.3.1 From First Principles

We can compute the test statistic manually, using the formula from equation (24) with $b = 0$:

```
T <- hat.beta[3] / sqrt(sigma.hat.squared * solve(t(X) %*% X)[3, 3])
T
```

```
# Water.Temp
#   3.519567
```

```
t.crit <- qt(0.975, n - p - 1)
abs(T) > t.crit
```

```
# Water.Temp
#       TRUE
```

Since the test statistic is larger than the critical value, we reject the hypothesis that $\beta_2 = 0$.

7.3.2 Using the `lm()` Output, I

We can also find the test statistic in the `lm()` output:

```
summary(m)
```

```
#
# Call:
# lm(formula = stack.loss ~ ., data = stackloss)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -7.2377 -1.7117 -0.4551  2.3614  5.6978
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) -39.9197    11.8960  -3.356  0.00375 **
# Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
# Water.Temp    1.2953     0.3680   3.520  0.00263 **
# Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 3.243 on 17 degrees of freedom
# Multiple R-squared:  0.9136,  Adjusted R-squared:  0.8983
# F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

The column `t value` shows the value of the test statistic T for the hypothesis that $\beta_i = 0$. For β_2 we find the value 3.520, which is a rounded version of the 3.519567 we found above.

Again, we can use the command `qt(0.975, 17)` to get the critical value, where we can find the degrees of freedom in the `Residual standard error` row near the bottom. Alternatively, we can use the so called "p-value" listed in the `Pr(>|t|)` column. This value is the smallest significance level at which H_0 is still rejected. Here we find 0.00263 and since we have $\alpha = 0.05 > 0.00263$, we can see that H_0 is rejected.

If we wanted to test $\beta_2 = 1$ instead, we could use the value $\hat{\beta}_2 = 1.2953$ from the `Estimate` column and $\sqrt{\hat{\sigma}^2 C_{ii}} = 0.3680$ from the `Std. Error` column, to get $T = (1.2953 - 1)/0.3680$ and then we would accept or reject the hypothesis $\beta_2 = 1$ by comparing the result to the critical value `qt(0.975, 17)`.

7.3.3 Using the `lm()` Output, II

For the hypotheses $\beta_i = 0$, we can also use the stars (or sometimes dots) shown in the right margin to perform the test. Since the `Water.Temp` row ends in `**`, we would reject $\beta_2 = 0$ even at the stricter $\alpha = 0.01$ level, and thus we would also reject this hypothesis at $\alpha = 0.05$ level.

7.4 Testing Multiple Coefficents

This section illustrates how a group of coefficients can be tested simultaneously. We consider a small, synthetic dataset:

```
# data from https://www1.maths.leeds.ac.uk/~voss/data/small/small.csv
small <- read.csv("data/small.csv")
m <- lm(y ~ ., data = small)
summary(m)
```

```
#
# Call:
# lm(formula = y ~ ., data = small)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -0.64664 -0.09369 -0.03420  0.07768  0.63353
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  5.53329    0.12360  44.768 6.92e-12 ***
# x1           0.18903    0.12059   1.568   0.1514
# x2          -0.30731    0.18011  -1.706   0.1221
# x3          -0.12382    0.05639  -2.196   0.0557 .
# x4          -0.11731    0.12619  -0.930   0.3768
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 0.3822 on 9 degrees of freedom
# Multiple R-squared:  0.6404,  Adjusted R-squared:  0.4806
# F-statistic: 4.007 on 4 and 9 DF,  p-value: 0.03891
```

From the R output it is clear that the intercept is non-zero, but none of the remaining regression coefficients are significantly different from zero when testing at 5% level. This poses the question whether the inputs have any effect on the output at all. To answer this question, we test the hypothesis

$$H_0 \colon \beta_1 = \beta_2 = \beta_3 = \beta_4 = 0,$$

omitting only β_0 in the hypothesis. We use the method from section 6.3 to perform this test.

7.4.1 From First Principles

The matrix K which selects the regression coefficents is

$$K = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{4\times 5}.$$

The test statistic is

$$F = \frac{\|K\hat{\beta} - 0\|^2_{K(X^\top X)^{-1}K^\top}}{4\hat{\sigma}^2},$$

where $K\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_4)$ and $K(X^\top X)^{-1}K^\top$ is obtained by removing the first row and column from $(X^\top X)^{-1}$:

```
b <- coef(m)[2:5]
X <- model.matrix(m)
C <- solve(t(X) %*% X)[2:5, 2:5]
F <- t(b) %*% solve(C, b) / (4 * summary(m)$sigma^2)
F
```

```
#          [,1]
# [1,] 4.007393
```

From lemma 6.3 we know that under H_0 the test statistic is $F_{k,n-p-1}$ distributed, and the critical value is the $1-\alpha$ quantile of this distribution:

```
n <- nrow(X)       # 14
p <- ncol(X) - 1   # 4
k <- 4
f <- qf(0.95, k, n - p - 1)
f
```

```
# [1] 3.633089
```

Since the test statistic is larger than the critical value, we can reject H_0 and conclude at significance level 5% that not all regression coefficients can be zero.

7.4.2 Using the `lm()` output

The information required to perform this F test is contained in the last row of the `summary(m)` output:

```
F-statistic: 4.007 on 4 and 9 DF,  p-value: 0.03891
```

Here we can see that the test statistic is 4.007 (a rounded version of the value we found above) and the degreen of freedom for the F-test are $k = 4$ and $n - p - 1 = 9$. The "p-value" listed is the smallest significance level at which the test still rejects H_0. We see that H_0 is rejected at significance level 0.03891, and thus also at $\alpha = 0.05$.

Summary

- We have illustrated the results of sections 5 and 5 with the help of examples.
- We have learned how to use the `lm()` output to compute confidence intervals and to perform hypothesis tests.

8 Regression Diagnostics

The traditional principle of data analysis could be described by the following steps

1. Explore the data; identify a plausible model.
2. Estimate any parameters specified by the model.
3. Check that the model is adequate — often by examining the discrepancy between the observed data and that predicted by the model — if necessary go back a step with a refined/revised model.
4. See if the model can be simplified; if necessary go to step 2.

Here we consider techniques for step 3, the checking of the model.

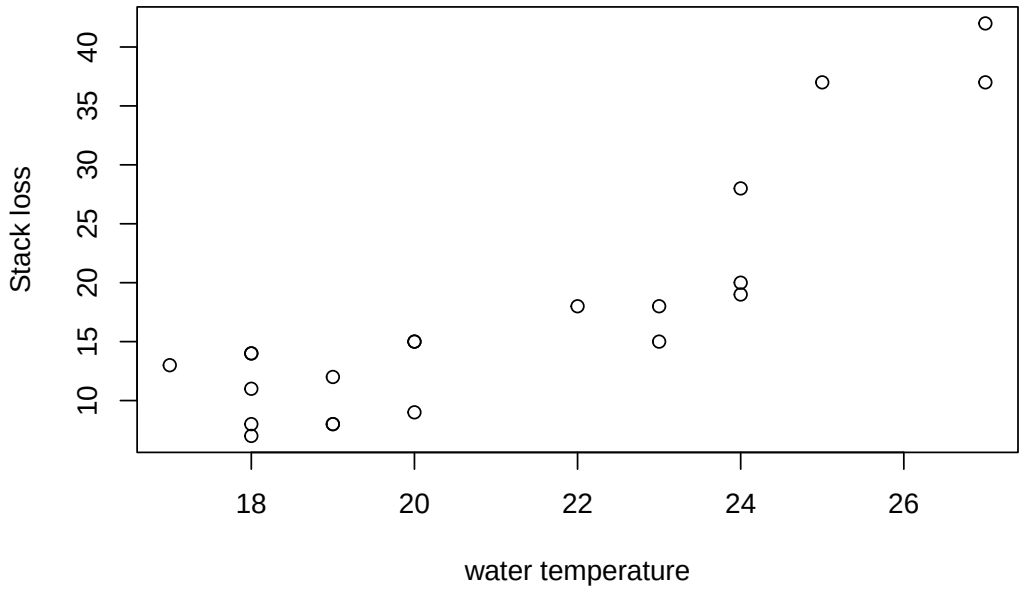
8.1 Diagnostic Plots

The regression model is often checked by examining the residuals. Various plots are used:

1. A scatter plot of the response y against each of the explanatory variables x_j for $j \in \{1, \dots, p\}$.
2. A scatter plot of $\hat{\varepsilon}$ against each of the explanatory variables x_j for $j \in \{1, \dots, p\}$.
3. A plot of $\hat{\varepsilon}_i$ *vs.* $\hat{y}_i$. If the variance of the residuals seems to increase with $\hat{y}$ then a transformation may be necessary. This type of plot is often called a "residual plot". The presence of a curvilinear relationship (in this plot, or the ones above) suggests a higher-order term, perhaps a quadratic in the explanatory variable, should be added to the model, or possibly a transformation should be applied to the response variable.
4. A Q-Q plot of the residuals can be used to assess whether the residuals are normally distributed. These plots plot quantiles of the data against quantiles of a normal distribution. If the residual are normally distributed, the points in a Q-Q plot will approximately lie on a straight line. The R command to produce Q-Q plots for the normal distribution is `qqnorm()`.
5. Data are often collected in time order. Even if time is not an explanatory variable, a plot of y *vs.* time can be of interest. It can reveal serial correlation in the data. Similarly, plotting the residuals *vs.* time.
6. A plot of x_j *vs.* x_k for $j \neq k$ can also be useful. If two or more regressors are highly correlated, we say that multicollinearity is present. When this occurs the least squares estimate $\hat{\beta}$ becomes numerically unstable. (We will learn more about this effect in chapter 10.)

Example 8.1. An example for the first type of diagnostic plot would be to plot `stack.loss` against `Water.Temp` for the `stackloss` dataset:

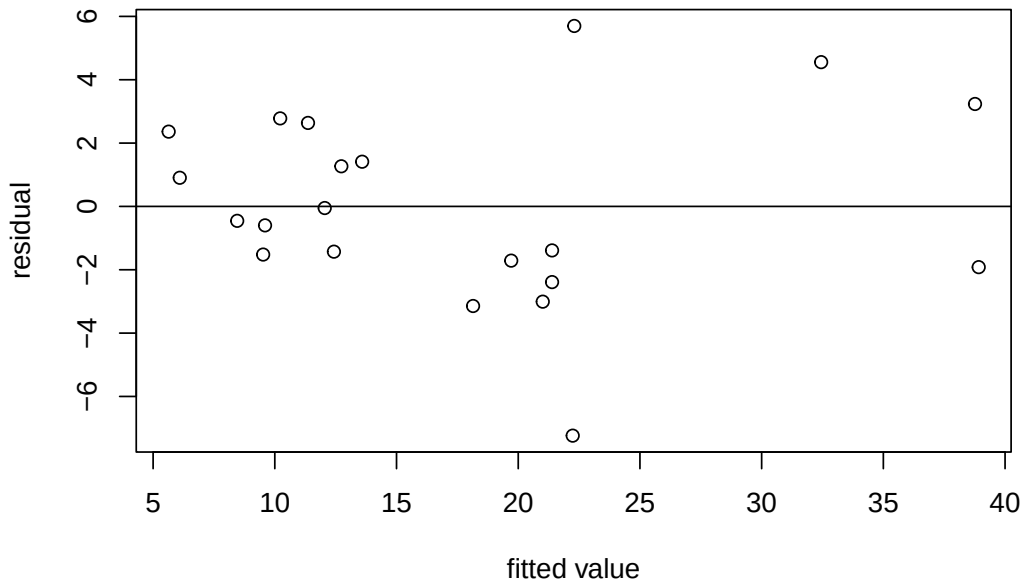
```
plot(stackloss$Water.Temp, stackloss$stack.loss,
     xlab = "water temperature", ylab = "Stack loss")
```



Clearly the stack loss increases with water temperature, but the relationship may not be linear. It may make sense to include the square of the water temperature as an additional input variable in the model.

Example 8.2. To produce a "residual plot", we first need to fit a model to the data. For the `stackloss` dataset this can be done as follows:

```
m <- lm(stack.loss ~ ., data = stackloss)
plot(fitted(m), resid(m),
     xlab = "fitted value", ylab = "residual")
abline(h = 0)
```



We can see that the residuals are clustered around zero as expected. Since there are no obvious patterns, the plot does not indicate any problems with model fit.

8.2 The Coefficient of Multiple Determination

In equation (20) we have seen that

$$\|y\|^2 = \|\hat{y}\|^2 + \|\hat{\varepsilon}\|^2.$$

The following lemma shows that a similar relation also holds after the sample means are subtracted.

Lemma 8.1. *Assume that the fitted model includes an intercept. Then we have*

$$\begin{aligned}\sum_{i=1}^{n}(y_i-\overline{y})^2 &= \sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2+\sum_{i=1}^{n}\hat{\varepsilon}_i^2\\ &= \sum_{i=1}^{n}(\hat{y}_i-\overline{\hat{y}})^2+\sum_{i=1}^{n}(\hat{\varepsilon}_i-\overline{\hat{\varepsilon}})^2,\end{aligned}$$

where $\overline{y}$, $\overline{\hat{y}}$ and $\overline{\hat{\varepsilon}}$ are the sample means of the three vectors y, $\hat{y}$ and $\hat{\varepsilon}$.

Proof. Define $\mathbf{1} = (1,\dots,1) \in \mathbb{R}^p$. From exercise 2 on problem sheet 1 we know that $H\mathbf{1} = \mathbf{1}$ and since H is symmetric we also have $\mathbf{1}^\top H = \mathbf{1}^\top$. This gives

$$\mathbf{1}^\top(I-H) = \mathbf{1}^\top - \mathbf{1}^\top H = \mathbf{1}^\top - \mathbf{1}^\top = 0. \tag{26}$$

As we have already seen previously, we have

$$\begin{aligned}\hat{y}^\top\hat{\varepsilon} &= (Hy)^\top(I-H)y\\ &= y^\top H(I-H)y\\ &= 0\end{aligned}$$

and using equation (26) we also get

$$\mathbf{1}^\top\hat{\varepsilon} = \mathbf{1}^\top(I-H)y = 0. \tag{27}$$

Thus, $(\hat{y}-\overline{y}\mathbf{1})^\top\hat{\varepsilon} = 0$ and using Pythagoras' theorem as in the section about Properties of the Hat Matrix, we find

$$\begin{aligned}\|y-\overline{y}\mathbf{1}\|^2 &= \|y-\hat{y}+\hat{y}-\overline{y}\mathbf{1}\|^2\\ &= \|\hat{\varepsilon}+\hat{y}-\overline{y}\mathbf{1}\|^2\\ &= \|\hat{\varepsilon}\|^2+\|\hat{y}-\overline{y}\mathbf{1}\|^2.\end{aligned}$$

This proves the first equality.

Using equation (26) again, we find

$$\sum_{i=1}^{n}\hat{\varepsilon}_i = \mathbf{1}^\top\hat{\varepsilon} = \mathbf{1}^\top(y-\hat{y}) = \mathbf{1}^\top(I-H)y = 0$$

and thus $\overline{\hat{\varepsilon}} = 0$. Since $y = \hat{y}+(y-\hat{y}) = \hat{y}+\hat{\varepsilon}$ we also have

$$\overline{y} = \overline{\hat{y}}+\overline{\hat{\varepsilon}} = \overline{\hat{y}}.$$

This completes the proof. □

We can express the terms in the statement of lemma 8.1 as sample variances. For example, we write

$$\mathrm{s}_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i-\bar{y})^2$$

for the sample variance of y. Using this notation, the statement of the lemma can be written as

$$\mathrm{s}_y^2 = \mathrm{s}_{\hat{y}}^2+\mathrm{s}_{\hat{\varepsilon}}^2.$$

Different from equation (20), the current relation is only true for models which include an intercept.

Some authors define

- $\mathrm{SS}_{\mathrm{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2$ (where "tot" stands for "total")
- $\mathrm{SS}_{\mathrm{reg}} = \sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2$ (where "reg" stands for "regression")
- $\mathrm{SS}_{\mathrm{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ (where "res" stands for "residual")

Using this notation, the statement of lemma 8.1 becomes

$$\mathrm{SS}_{\mathrm{tot}} = \mathrm{SS}_{\mathrm{reg}} + \mathrm{SS}_{\mathrm{res}}.$$

Definition 8.1. The **coefficient of multiple determination** is defined as

$$R^2 := 1 - \frac{\mathrm{s}^2_{\hat{\varepsilon}}}{\mathrm{s}^2_y} = 1 - \frac{\sum_{i=1}^{n}\hat{\varepsilon}_i^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}.$$

Using the result of lemma 8.1, for models which include an intercept we can also express R^2 as

$$R^2 = \mathrm{s}^2_{\hat{y}}/\mathrm{s}^2_y.$$

Since both numerator and denominator are positive, if is clear the $R^2 \geq 0$. Similarly, from the definition it is clear that $R^2 \leq 1$, so that we always have $R^2 \in [0, 1]$. A large value of R^2 indicates that the residuals are small compared to the sample variance of the y_i, and thus often indicates good model fit.

Example 8.3. We can easily compute the R^2 value for the `stackloss` dataset manually:

```
m <- lm(stack.loss ~ ., data = stackloss)
R.squared <- 1 - var(resid(m)) / var(stackloss$stack.loss)
R.squared
```

```
# [1] 0.9135769
```

We can also find this value near the bottom of the output of `summary(m)`, listed as `Multiple R-squared`:

```
summary(m)
```

```
#
# Call:
# lm(formula = stack.loss ~ ., data = stackloss)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -7.2377 -1.7117 -0.4551  2.3614  5.6978
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) -39.9197    11.8960  -3.356  0.00375 **
# Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
# Water.Temp    1.2953     0.3680   3.520  0.00263 **
# Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 3.243 on 17 degrees of freedom
# Multiple R-squared:  0.9136,  Adjusted R-squared:  0.8983
# F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

One problem with the R^2 value is, that it always increases when another input variable is added to the model. Thus, the R^2 value cannot be used to compare model fit for models with different numbers of variables. An attempt to compensate for this effect is the adjusted R^2 value:

Definition 8.2. The **adjusted R^2 value** is given by

$$R^2_{\text{adj}} := 1 - \frac{\frac{1}{n-p-1}\mathrm{S}^2_{\hat{\varepsilon}}}{\frac{1}{n-1}\mathrm{S}^2_y} = 1 - \frac{n-1}{n-p-1}(1-R^2).$$

The value R^2_{adj} is always smaller than the R^2 value, and R^2_{adj} can be (slightly) negative.

Example 8.4. The adjusted R^2 value for the stackloss dataset can be found as follows:

```
n <- nrow(stackloss)
p <- ncol(stackloss) - 1
R.squared.adj <- 1 - (n - 1) / (n - p - 1) * (1 - R.squared)
R.squared.adj
```

```
# [1] 0.8983258
```

We can also find this value near the bottom of the output of `summary(m)`, listed as `Adjusted R-squared`.

Example 8.5. As suggested in example 8.1, we may want to include a new quadratic term in the stackloss dataset:

```
m2 <- lm(stack.loss ~ . + I(Water.Temp^2), data = stackloss)
summary(m2)
```

```
#
# Call:
# lm(formula = stack.loss ~ . + I(Water.Temp^2), data = stackloss)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -4.1217 -1.5205 -0.3091  0.9753  6.0554
#
# Coefficients:
#                 Estimate Std. Error t value Pr(>|t|)
# (Intercept)     57.73727   46.08487   1.253  0.22826
# Air.Flow         0.53683    0.14708   3.650  0.00216 **
# Water.Temp      -7.62030    4.10441  -1.857  0.08188 .
# Acid.Conc.      -0.09697    0.14371  -0.675  0.50946
# I(Water.Temp^2)  0.21224    0.09738   2.179  0.04459 *
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 2.936 on 16 degrees of freedom
# Multiple R-squared:  0.9334,  Adjusted R-squared:  0.9167
# F-statistic: 56.02 on 4 and 16 DF,  p-value: 3.293e-09
```

As expected, the R^2 value has increased here (from 0.9136 to 0.9334). Since the adjusted R^2 value has also increased (from 0.8983 to 0.9167), the extended model may be better than the original model.

Summary

- Diagnostic plots can help to discover problems with model fit.
- The R^2 value is a numerical measure of the goodness of fit of a model.
- If models with different numbers of variables are compared, R^2_{adj} should be used instead of R^2.

Interlude: Understanding the `lm()` Output

We have seen that the `lm()` function returns an object which contains a lot of information about the fitted model. We can inspect this information by using the `summary()` function. The aim of this section is to help you understand how the output of `summary()` relates to the mathematical expressions we have considered in the previous sections.

We will use the `stackloss` dataset as an example:

```
m <- lm(stack.loss ~ ., data = stackloss)
summary(m)
```

```
Call:
lm(formula = stack.loss ~ ., data = stackloss)

Residuals:
    Min      1Q  Median      3Q     Max 
-7.2377 -1.7117 -0.4551  2.3614  5.6978 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -39.9197    11.8960  -3.356  0.00375 ** 
Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
Water.Temp    1.2953     0.3680   3.520  0.00263 ** 
Acid.Conc.   -0.1521     0.1563  -0.973  0.34405    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.243 on 17 degrees of freedom
Multiple R-squared:  0.9136,    Adjusted R-squared:  0.8983 
F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

Here I have marked different parts of the output using red shaded boxes and using the letters A to I. We discuss each of the marked sections in turn:

- The first section, part **A**, contains summary statistics for the fitted residuals $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_n$. The values shown are the minimum, first quartile, median, third quartile, and maximum of the residuals. This is the same information we can get using the command `summary(resid(m))`. The mean is omitted from A, since if always equals zero.
- Column **B** shows the estimated coefficient vector $\hat{\beta}$. This is computed using the formula from lemma 2.1.
- Column **C** shows the standard error of the estimated coefficients. The ith entry is the (estimated) standard deviation of $\hat{\beta}_i$, computed as $\sqrt{\hat{\sigma}^2 C_{ii}}$, where $C = (X^\top X)^{-1}$. This corresponds to the variance shown in equation (17), where the true variance is σ^2 replaced with the estimate $\hat{\sigma}^2$. These quantities are used to compute the t-test statistic in equation (24).
- Column **D** shows the t-test statistic for the coefficients. The values are computed using equation (24).
- Column **E** replicates the information from column **D** in different form, showing p-values instead of the test statistics.

- The field **F** shows the estimated standard deviation $\hat{\sigma}$. This is computed as the square root of $\hat{\sigma}^2$ from equation (21).
- Field **G** shows the value $n - p - 1$. This is the number of degrees of freedom in lemma 5.2. The value is needed when performing hypothesis tests and computing confidence intervals for individual coefficients.
- Field **H** shows the R^2 value and adjusted R^2 value. These are computed using definitions 8.1 and 8.2.
- Field **I** shows the F-test statistic for testing the hypothesis $H_i \colon \beta_1 = \cdots = \beta_p = 0$ (omitting the coeficient β_0 for the intercept). This value can be used to test the hypothesis that the inputs have no effect on the output. The F-test statistic is computed using equation (25), the degrees of freedom shown are $k = p$ and $n-p-1$ from lemma 6.1.

9 The Influence of Observations

In this section we consider the influence which individual samples have on the estimate. Samples with a large influence may be outliers, and are worth checking for validity.

9.1 Deleting Observations

The most direct way to check for the influence of an observation is to compute the parameter estimate $\widehat{\beta}$ both with and without the observation in question, and to see how much the two results differ.

Definition 9.1. For $I \subset \{1, \dots, n\}$ we write $\widehat{\beta}^{(I)}$ for the estimate where all observations $i \in I$ have been removed. As a shorthand we write $\widehat{\beta}^{(i)}$ instead of $\widehat{\beta}^{(\{i\})}$ for the case where only a single observation has been removed.

This approach is very different from what we used in the section about Estimating Coefficients Simultaneously. There, we selected a subset of regression coefficients, corresponding to columns in the design matrix X, whereas here we are considering a subset of the observations, corresponding to rows of X.

Using the formula for the least squares estimate, we know that

$$\widehat{\beta}^{(I)} = (X_{(I)}^\top X_{(I)})^{-1} X_{(I)}^\top y_{(I)}, \tag{28}$$

where now $y^{(I)}$ is y with all y_i for $i \in I$ removed and $X_{(I)}$ is X with all rows $i \in I$ removed. (We write the "(I)" as a subscript, to make space for the transpose sign.) Our first aim is to find an explicit formula for the difference between $\widehat{\beta}$ and $\widehat{\beta}^{(I)}$.

We have

$$\begin{aligned}(X_{(I)}^\top y_{(I)})_k &= \sum_{j \notin I} X_{jk} y_j \\ &= \sum_{j=1}^{n} X_{jk} y_j - \sum_{j \in I} X_{jk} y_j\end{aligned}$$

for all $k \in \{0, \dots, p\}$, and thus

$$X_{(I)}^\top y_{(I)} = X^\top y - X_I^\top y_I, \tag{29}$$

where X_I stands for the matrix which only contains the rows $i \in I$ of the matrix X, and $y_I = (y_i)_{i \in I}$. Before we substitute this result into equation (28), we first consider how we can write the the inverse $(X_{(I)}^\top X_{(I)})^{-1}$ in terms of $(X^\top X)^{-1}$.

Lemma 9.1. *Let $A \in \mathbb{R}^{(p+1)\times(p+1)}$ and let $U, V \in \mathbb{R}^{m\times(p+1)}$. Then*

$$(A - U^\top V)^{-1} = A^{-1} + A^{-1} U^\top (I - V A^{-1} U^\top)^{-1} V A^{-1}.$$

Proof. This statement is proved by multiplying the equation with $A - U^\top V$. Expanding all brackets and using the relation $AA^{-1} = I$, we find

$$(A - U^\top V)(A^{-1} + A^{-1} U^\top (I - V A^{-1} U^\top)^{-1} V A^{-1}) = \cdots = I.$$

This shows that we have indeed found the inverse of $A - U^\top V$. □

Using this result, we can now derive a formula for the change in $\widehat{\beta}$ when some observations are omitted.

Lemma 9.2. *We have*

$$\hat{\beta}^{(I)} - \hat{\beta} = -(X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} \hat{\varepsilon}_I,$$

where $H_{II} \coloneqq X_I (X^\top X)^{-1} X_I^\top = (h_{ij})_{i,j \in I}$ *is the matrix which contains the elements of the hat matrix* H *where both row and column are in* I*, and* $\hat{\varepsilon}_I = (\hat{\varepsilon}_i)_{i \in I} = (y_i - \hat{y}_i)_{i \in I}$ *is the vector of residuals at the omitted samples.*

The vector $\hat{\varepsilon}_I$ consists of a subset of the original residuals, computed using the original $\hat{y}$ and $\hat{\beta}$. It does not refer to the modified estimate $\hat{\beta}^{(I)}$ which was computed with some observations omitted. Similarly, H_{II} is a submatrix of the original hat matrix.

Proof. Similar to (29), we

$$(X_{(I)}^\top X_{(I)})_{ik} = \sum_{j \notin I} X_{ji} X_{jk} = \sum_{j=1}^{n} X_{ji} X_{jk} - \sum_{j \in I} X_{ji} X_{jk}$$

and thus

$$X_{(I)}^\top X_{(I)} = X^\top X - X_I^\top X_I.$$

Now we can use the lemma with $U = V = X_I$ to get

$$\begin{aligned}
(X_{(I)}^\top X_{(I)})^{-1} &= (X^\top X)^{-1} + (X^\top X)^{-1} X_I^\top (I - X_I (X^\top X)^{-1} X_I^\top)^{-1} X_I (X^\top X)^{-1} \\
&= (X^\top X)^{-1} + (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} X_I (X^\top X)^{-1}.
\end{aligned}$$

Using equations (28) and (29) and lemma 9.1 we get

$$\begin{aligned}
\hat{\beta}^{(I)} &= (X_{(I)}^\top X_{(I)})^{-1} X_{(I)}^\top y \\
&= (X_{(I)}^\top X_{(I)})^{-1} (X^\top y - X_I^\top y_I) \\
&= \Big((X^\top X)^{-1} + (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} X_I (X^\top X)^{-1} \Big) (X^\top y - X_I^\top y_I) \\
&= (X^\top X)^{-1} X^\top y \\
&\qquad + (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} X_I \hat{\beta} \\
&\qquad - (X^\top X)^{-1} X_I^\top y_I \\
&\qquad - (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} X_I (X^\top X)^{-1} X_I^\top y_I \\
&= \hat{\beta} + (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} X_I \hat{\beta} \\
&\qquad - (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} (I - H_{II}) y_I \\
&\qquad - (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} H_{II} y_I \\
&= \hat{\beta} + (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} X_I \hat{\beta} \\
&\qquad - (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} y_I \\
&= \hat{\beta} - (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} (y_I - X_I \hat{\beta}) \\
&= \hat{\beta} - (X^\top X)^{-1} X_I^\top (I - H_{II})^{-1} \hat{\varepsilon}_I.
\end{aligned}$$

This completes the proof. □

We can also find the change in estimated error variance, when samples are omitted from the estimate. The estimated variance in the modified model is

$$\begin{aligned}
\hat{\sigma}_{(I)}^2 &\coloneqq \frac{1}{n - m - p - 1} \sum_{i \notin I} \Big(y_i - (X \hat{\beta}^{(I)})_i \Big)^2 \\
&= \frac{1}{n - m - p - 1} \Big\| y_{(I)} - X_{(I)} \hat{\beta}^{(I)} \Big\|^2
\end{aligned}$$

where $m = |I|$ so that $n - m$ is the number of observations used in the estimate. We want to compare this quantity to the original estimate

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{n-p-1}\sum_{i=1}^{n}\Big(y_i - (X\hat{\beta})_i\Big)^2 \\ &= \frac{1}{n-p-1}\Big\|y - X\hat{\beta}\Big\|^2.\end{aligned}$$

Here we used again the Euclidean norm as a shorthand for the sum of squares used in these estimates. The following lemma shows how the residual sum of squares decreases when the observations I are omitted from the estimate.

Lemma 9.3. *We have*

$$\|y_{(I)} - X_{(I)}\hat{\beta}^{(I)}\|^2 - \|y - \hat{y}\|^2 = -\hat{\varepsilon}_I^\top (I - H_{II})^{-1}\hat{\varepsilon}_I.$$

The proof of this lemma is similar to the proof of lemma 9.2, but we omit the tedious calculations here.

Example 9.1. To illustrate the results of this subsection, we give a numerical example. In the example we consider the `stackloss` dataset, and study what happens when the first two observations are omitted:

```
m1 <- lm(stack.loss ~ ., data = stackloss)
m2 <- lm(stack.loss ~ ., data = stackloss[-c(1,2),])
coef(m2) - coef(m1)
```

```
# (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
#  0.86331016 -0.03780761 -0.02706305  0.02124533
```

Using the result of lemma 9.2, we can get the difference between the two estimates without using the model `m2`:

```
X <- model.matrix(m1)
H <- X %*% solve(t(X) %*% X, t(X))
XI <- X[1:2,]
hII <- H[1:2, 1:2]
hat.epsI <- resid(m1)[1:2]

-as.vector(solve(t(X) %*% X) %*% t(XI) %*% solve(diag(2) - hII, hat.epsI))
```

```
# [1]  0.86331016 -0.03780761 -0.02706305  0.02124533
```

This is the same result as we obtained above. (I used `as.vector()` to convert the result from a 4×1 matrix into a vector.)

Similarly, we can compare the residual sum of squares: A direct comparison gives

```
sum(resid(m2)^2) - sum(resid(m1)^2)
```

```
# [1] -15.41758
```

Using the result of lemma 9.3, we get the same result without requiring `m2`:

```
-t(hat.epsI) %*% solve(diag(2) - hII, hat.epsI)
```

```
#           [,1]
# [1,] -15.41758
```

Again, the results using both methods agree.

In the special case of $m = 1$, where $I = \{i\}$ for some $i \in \{1, \dots, n\}$, our results simplify to

$$\hat{\beta}^{(i)} - \hat{\beta} = -(X^\top X)^{-1} x_i \frac{\hat{\varepsilon}_i}{1 - h_{ii}} \tag{30}$$

and

$$\|y_{(i)} - x_i^\top \hat{\beta}^{(i)}\|^2 - \|y - \hat{y}\|^2 = -\frac{\hat{\varepsilon}_i^2}{1 - h_{ii}},$$

where $x_i \in \mathbb{R}^{p+1}$ is the ith row of X as a vector, $\hat{\varepsilon}_i$ is the usual ith residual, and h_{ii} is the ith diagonal element of the hat matrix. Using the correct prefactors the second result gives

$$(n - p - 2)\hat{\sigma}^2_{(i)} = (n - p - 1)\hat{\sigma}^2 - \frac{\hat{\varepsilon}_i^2}{1 - h_{ii}}.$$

We can find numerical values for these quantities using the function `influence()`:

```
x <- c(1, 2, 3, 10)
y <- c(0, 0, 1, 3)
m <- lm(y ~ x, data = stackloss)
influence(m)
```

```
# $hat
#    1    2    3    4
# 0.43 0.33 0.27 0.97
#
# $coefficients
#   (Intercept)            x
# 1  0.01719298 -0.002105263
# 2 -0.19582090  0.019104478
# 3  0.15369863 -0.009315068
# 4  0.30666667 -0.160000000
#
# $sigma
#         1         2         3         4
# 0.4682929 0.2591605 0.2482818 0.4082483
#
# $wt.res
#     1     2     3     4
#  0.02 -0.32  0.34 -0.04
```

The output lists the diagonal elements h_{ii} of the hat matrix as `$hat`, the vectors $\hat{\beta}^{(i)} - \hat{\beta}$ for each i as `$coefficients`, the standard deviations $\hat{\sigma}_{(i)}$ for each i as `$sigma`, and the residuals $\hat{\varepsilon}_i$ as `$wt.res`.

9.2 Cook's Distance

In the section about Estimating Coefficients Simultaneously we learned how to measure distances of estimated parameter vectors: To compare $K\hat{\beta}$ to a vector $K\beta$ we used the norm

$$\|K\hat{\beta} - K\beta\|^2_{K(X^\top X)^{-1}K^\top} = (K\hat{\beta} - K\beta)^\top (K(X^\top X)^{-1}K^\top)^{-1}(K\hat{\beta} - K\beta).$$

If we consider the full parameter vector, we can use $K = I$ to get

$$\|\hat{\beta} - \beta\|^2_{(X^\top X)^{-1}} = (\hat{\beta} - \beta)^\top X^\top X(\hat{\beta} - \beta).$$

From lemma 6.1 we know that

$$F = \frac{\|\hat{\beta} - \beta\|^2_{(X^\top X)^{-1}}}{(p+1)\hat{\sigma}^2}$$

follows an $F_{p+1,n-p-1}$-distribution. We can use this measure to quantify when the difference $\hat{\beta}^{(i)} - \hat{\beta}$ should be considered "large".

Definition 9.2. Cook's distance for observation i is defined as

$$D_i = \frac{(\hat{\beta}^{(i)} - \hat{\beta})^\top X^\top X(\hat{\beta}^{(i)} - \hat{\beta})}{(p+1)\hat{\sigma}^2}.$$

We will consider observation i to be **influential** if D_i is large. Since this "lives on the scale" of an $F_{p+1,n-p-1}$-distribution, we should compare D_i to typical values of such a distribution. For example, we could consider the median. Here is a sample of median values for different n and p:

```
n <- c(10, 30, 100)
p <- c(1, 3, 5)
xx <- expand.grid(n = n, p = p)
cbind(xx, median = qf(0.5, xx$p + 1, xx$n - xx$p - 1))
```

```
#     n p    median
# 1  10 1 0.7568285
# 2  30 1 0.7105929
# 3 100 1 0.6980730
# 4  10 3 0.9419133
# 5  30 3 0.8614530
# 6 100 3 0.8451308
# 7  10 5 1.0616689
# 8  30 5 0.9168687
# 9 100 5 0.8977754
```

In practice, some authors suggest that an observation is "influential" if $D_i > 1$, instead of using an exact quantile of the F-distribution.

The following lemma provides two additional ways to compute Cook's distance.

Lemma 9.4. *We have*

$$\begin{aligned} D_i &= \frac{(\hat{y}^{(i)} - \hat{y})^\top (\hat{y}^{(i)} - \hat{y})}{(p+1)\hat{\sigma}^2} \\ &= \frac{\hat{\varepsilon}_i^2}{(p+1)\hat{\sigma}^2} \cdot \frac{h_{ii}}{(1-h_{ii})^2}. \end{aligned}$$

where $\hat{y}^{(i)} := X\hat{\beta}^{(i)}$ is the fitted value which is predicted for observation i when this observation is not used to fit the model, $\hat{\varepsilon}_i$ are the usual residuals, and h_{ii} are the diagonal elements of the hat matrix.

Proof. We have $\hat{y}^{(i)} - \hat{y} = X\hat{\beta}^{(i)} - X\hat{\beta}$ and thus

$$(\hat{y}^{(i)} - \hat{y})^\top (\hat{y}^{(i)} - \hat{y}) = (\hat{\beta}^{(i)} - \hat{\beta})^\top X^\top X(\hat{\beta}^{(i)} - \hat{\beta}).$$

This proves the first equality. From equation (30) we know

$$\hat{\beta}^{(i)} - \hat{\beta} = -(X^\top X)^{-1} x_i \frac{\hat{\varepsilon}_i}{1-h_{ii}}$$

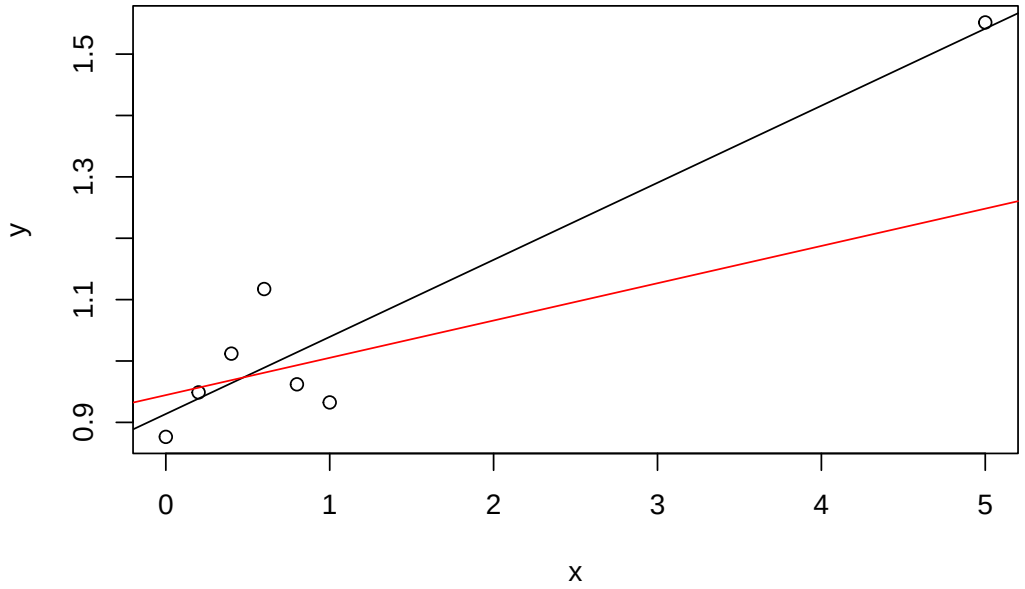
and thus we find

$$\begin{aligned} (\hat{\beta}^{(i)} - \hat{\beta})^\top X^\top X(\hat{\beta}^{(i)} - \hat{\beta}) &= \frac{\hat{\varepsilon}_i^2}{(1-h_{ii})^2} x_i^\top (X^\top X)^{-1} X^\top X (X^\top X)^{-1} x_i \\ &= \frac{\hat{\varepsilon}_i^2}{(1-h_{ii})^2} x_i^\top (X^\top X)^{-1} x_i \\ &= \frac{\hat{\varepsilon}_i^2}{(1-h_{ii})^2} h_{ii} \end{aligned}$$

and dividing by $(p+1)\hat{\sigma}^2$ completes the proof. □

Example 9.2. To illustrate how Cook's D is computed, we use a small dataset with simulated data. We include an x-space outlier:

```
set.seed(20211026)
x <- c(5, seq(0, 1, length.out = 6))
y <- 1 + 0.1 * x + 0.1 * rnorm(7)
plot(x, y)

m <- lm(y ~ x)
abline(m)

m2 <- lm(y[-1] ~ x[-1])
abline(m2, col = "red")
```



The black line in the plot is the regression line using all data, the red line is the regression line fitted using only the left-most six points and leaving out the "outlier" at $x = 5$. The difference between the two lines makes it clear that the point at $x = 5$ has a large effect on the estimated regression line. An easy way to compute the D-values is the second formula from lemma 9.4:

```
X <- model.matrix(m)
H <- X %*% solve(t(X) %*% X, t(X))
hii <- diag(H)
D <- resid(m)^2 * hii / 2 / summary(m)$sigma^2 / (1 - hii)^2
D
```

```
#           1           2           3           4           5           6
# 6.456405260 0.035318182 0.002129694 0.042504418 0.268374731 0.040765842
#           7
# 0.162580015
```

The results show as expected a very large value for the first observation, $D_1 = 6.456$, and small values for the remaining D_i. We can also find Cook's D-values in the output of the `influence.measures()` function:

```
influence.measures(m)
```

```
# Influence measures of
#    lm(formula = y ~ x) :
#
#    dfb.1_   dfb.x   dffit  cov.r  cook.d   hat inf
# 1 -0.7475  3.1083  3.3670 39.046 6.45641 0.967   *
# 2 -0.2441  0.1415 -0.2441  1.791 0.03532 0.215
# 3  0.0583 -0.0296  0.0585  1.920 0.00213 0.192
# 4  0.2674 -0.1142  0.2720  1.596 0.04250 0.173
# 5  0.9536 -0.3189  0.9959  0.348 0.26837 0.159
# 6 -0.2461  0.0560 -0.2681  1.512 0.04077 0.149
# 7 -0.5620  0.0577 -0.6512  0.687 0.16258 0.144
```

The table also shows the diagonal elements of the hat matrix, in the column `hat`, and some additional measures which we will not discuss here. A star in the last column, titled `inf`, marks influential observations. The star is placed if *any* of the measures shown in the table is significantly large.

Summary

- We can measure the influence of an observation by the amount the estimated parameters change when the observation is deleted.
- We learned how to determine the amount of change without having to fit n additional models.
- Cook's distance provides a numerical measure for the influence of each observation.

10 Multicollinearity

Definition 10.1. In linear regression, **Multicollinearity** denotes the situation when the columns of the design matrix X are (approximately) linearly dependent.

The columns of X are linearly dependent, if and only if there is a $v \in \mathbb{R}^{p+1}$ such that $Xv = 0$. The columns are approximately linearly dependent, if there is a vector v such that $Xv \approx 0$.

10.1 Consequences of Multicollinearity

In the derivation of the least squares estimator $\hat{\beta} = (X^\top X)^{-1} X^\top y$ we had to assume that the matrix $X^\top X$ is invertible. If the columns of X are linearly dependent, $X^\top X$ is no longer invertible and the least squares estimate is no longer uniquely defined.

Example 10.1. Consider the following data:

y	x_1	x_2
2	1	1
4	2	2
6	3	3

In this case, an exact match with no residuals can be achieved as $y = x_1 + x_2$, *i.e.* for $\beta_1 = \beta_2 = 1$. But this solution is not unique: we could just as well write $y = 2x_1$ or $y = 2x_2$. Any choice of β_1 and β_2 with $\beta_1 + \beta_2 = 2$ will lead to zero residuals.

The problem in the example above occurs, since the two input columns in the data are identical. The same problem, in a less obvious way, would occur in datasets with more inputs when one column of X can be written as a linear combination of other columns. This is a problem of the given input data, not of the responsees or of the statistical model.

In the case where there is only approximate linear dependency between the columns of X, the inverse $(X^\top X)^{-1}$ exists and an estimate $\hat{\beta}$ can be computed, but there will be huge uncertainties in some of the stimated coefficients. We illustrate this effect using a numerical example.

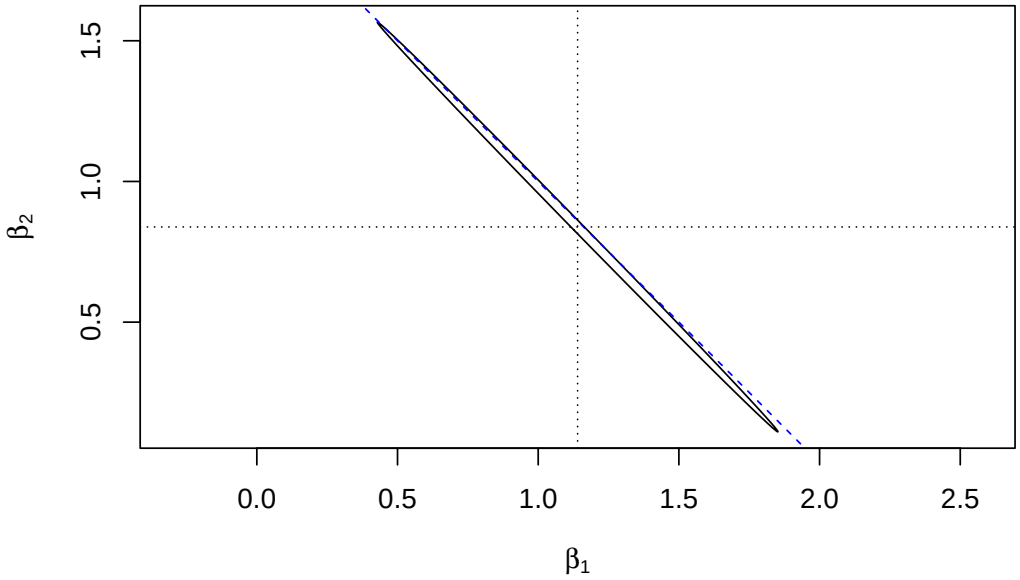
Example 10.2. Here we simulate data which has similar characteristics to the toy example above: we have very similar two inputs, the sum of which nearly equals the output. We fit a model without the intercept, so that there are only two coefficients to estimate and plotting these is no problem.

```
set.seed(20211101)
n <- 10
y <- 2 * (1:n)
x1 <- rnorm(10, 1:n, 0.15)
x2 <- rnorm(10, 1:n, 0.15)
m <- lm(y ~ 0 + x1 + x2)
```

Similar to the approach in section 6, we can plot a confidence ellipse. (Don't worry about the details of the commands used to plot the ellipse.)

```
sigma.hat <- summary(m)$sigma
X <- model.matrix(m)
svd <- La.svd(t(X) %*% X)
alpha <- 0.05
f.crit <- qf(1 - alpha, ncol(X), nrow(X) - ncol(X))
phi <- seq(0, 2*pi, length.out = 201)
```

```
circ <- rbind(cos(phi), sin(phi)) * sqrt(f.crit * ncol(X) * sigma.hat^2)
ellipse <- svd$u %*% (circ / sqrt(svd$d)) + coef(m)

plot(ellipse[1,], ellipse[2,], type = "l", asp = 1,
     xlab = expression(beta[1]),
     ylab = expression(beta[2]))
abline(2, -1, lty = "dashed", col = "blue")
abline(v = coef(m)[1], lty = "dotted")
abline(h = coef(m)[2], lty = "dotted")
```



We can see that, as in the toy example above, the confidence region places (β_1, β_2) close to the line where $\beta_1 + \beta_2 = 2$ (the diagonal dashed line), but there is considerable uncertainty about where on this line the coefficients are located. The effect gets more pronounced when the amount of noise in the definition of `x1` and `x2` is reduced.

There are two, slightly different effects of multicollinearity:

1. In the case of exact multicollinearity, $X^\top X$ is not invertible. If the columns of X are approximately linearly dependent, then $(X^\top X)^{-1}$ does exist, but small changes of X lead to large changes of $(X^\top X)^{-1}$ and numerical computation of the inverse is strongly affected by rounding errors.

2. If the columns of X are approximately linearly dependent, the computed value of the estimator $\hat{\beta}$ is strongly affected by small changes to the system: the noise in the model strongly affects $\hat{\beta}$ (as demonstrated in example 10.2, above), leaving out a single observation may lead to large changes in $\hat{\beta}$ and computation of $\hat{\beta}$ is sensitive to rounding errors.

While multicollinearity can make the regression coefficients ambiguous, the outputs y are not affected. Predictions made using a model where multicollinearity is present are still reliable.

10.2 Detecting Multicollinearity

While collinearity of two columns of X is easy to spot, for example in a pair scatter plot of the inputs x_i, multicollinearity which involves more than two columns can be harder to notice. The condition number of X is a quantitative measure of how close X is to multicollinearity.

Definition 10.2. The **condition number** of X is defined to be

$$\kappa(X) = \frac{\sigma_{\max}(X)}{\sigma_{\min}(X)},$$

where $\sigma_{\max}(X) = \sqrt{\lambda_{\max}(X^\top X)}$ is the largest singular value of X, computed as the square root of the largest eigenvalue of $X^\top X$, and similarly $\sigma_{\min}(X) = \sqrt{\lambda_{\min}(X^\top X)}$ is the smallest singular value of X, computed as the square root of the smallest eigenvalue of $X^\top X$.

As a rule of thumb, if $\kappa(X) < 10$ there are no significant problems with multicollinearity, and if $\kappa(X) > 30$ the regression problem sufferes severe problems with multicollinearity. In the case where the columns of X are exactly linearly dependent, we have $\sigma_{\min}(X) = 0$ and $\kappa(X) = \infty$.

In R, the condition number can be computed using the function `kappa()`. The function can either be called as `kappa(X, exact = TRUE)`, where `X` is the design matrix, or as `kappa(m, exact = TRUE)` where `m` is the object returned by `lm()`. If the optional argument `exact = TRUE` is omitted, only an approximate result is returned (using a faster algorithm).

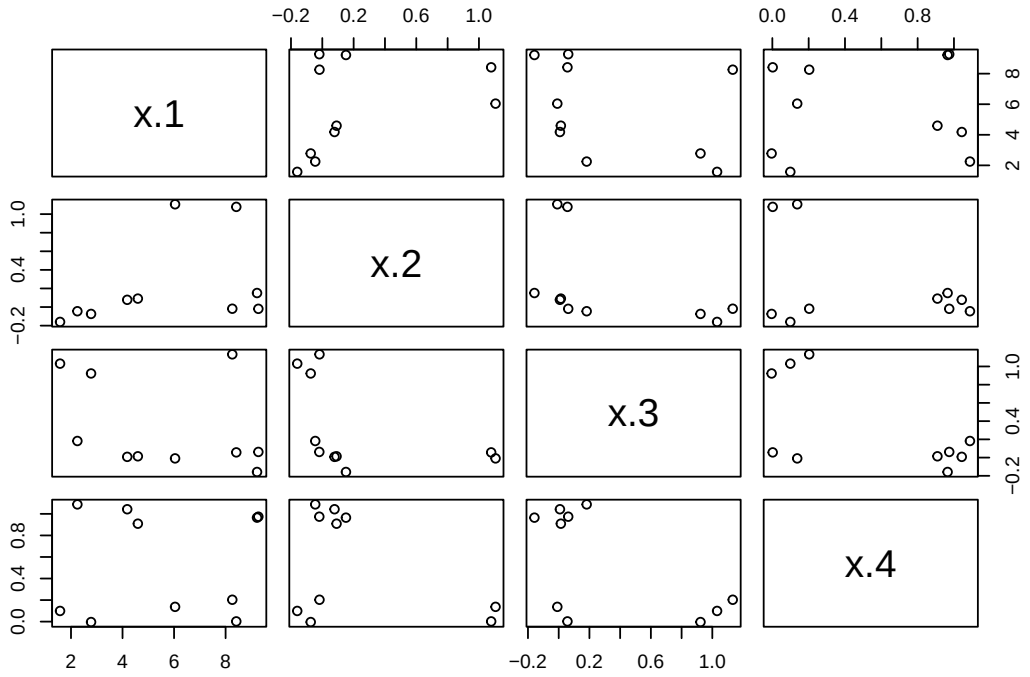
Example 10.3. Consider the following toy dataset:

```
data
```

```
#        y   x.1    x.2    x.3    x.4
# 1   7.37 4.182  0.078  0.008  1.043
# 2   9.86 6.039  1.106 -0.009  0.137
# 3  13.44 9.220  0.151 -0.158  0.965
# 4  12.95 8.260 -0.018  1.133  0.203
# 5  13.05 8.415  1.078  0.057  0.002
# 6  13.17 9.272 -0.019  0.062  0.974
# 7   5.28 1.575 -0.160  1.031  0.099
# 8   8.18 4.595  0.091  0.014  0.909
# 9   5.13 2.779 -0.074  0.923 -0.004
# 10  6.80 2.248 -0.045  0.182  1.088
```

To inspect these data, we use a pair scatter plot of the input variables, leaving out the responses in column 1:

```
pairs(data[,-1])
```



While there are visbile patterns, there are no clear signs of a linear dependency between any two columns. To see whether there are problems with multicollinearity, we fit a linear model and determine the condition number.

```
m <- lm(y ~ ., data = data)
kappa(m, exact = TRUE)
```

```
# [1] 97.38214
```

Since the condition number is larger than 30, we can say that the data shows severe multicollinearity.

To find out which variables are involved, we can use singular value decomposition. This allows to write X as $X = UDV^\top$, where $D \in \mathbb{R}^{(p+1)\times(p+1)}$ is a diagonal matrix and $U \in \mathbb{R}^{n\times(p+1)}$ and $V \in \mathbb{R}^{(p+1)\times(p+1)}$ are matrices with orthonormal columns.

```
X <- model.matrix(m)
s <- svd(X, nu = 0)
s
```

```
# $d
# [1] 20.3002516  2.0294781  1.8122508  1.2526330  0.2084597
#
# $v
#              [,1]        [,2]         [,3]        [,4]          [,5]
# [1,] -0.14024576 -0.5552821  0.04218055 -0.62161197 -0.5327403485
# [2,] -0.98554152  0.1088264 -0.04131956  0.12311216 -0.0009051211
# [3,] -0.04170949  0.4036310 -0.21603327 -0.75981456  0.4597323397
# [4,] -0.03395454 -0.6495911 -0.55665792  0.11926737  0.5027780067
# [5,] -0.07839927 -0.3081105  0.79998442 -0.08306065  0.5020431792
```

We used the optional argument `nu = 0` to tell R that we won't need the matrix U. The diagonal elements $\sigma_0(X), \dots, \sigma_p(X) \geq 0$, stored in `s$d`, are the singular values of X in decreasing order. We can find the condition number using these values:

```
s$d[1] / s$d[length(s$d)]
```

```
# [1] 97.38214
```

This agrees with the value for $\kappa(X)$ we found above.

Finally, the columns $v_1, \ldots, v_{p+1}$ of `s$v` are called the (right) singular vectors of X. We can use these vectors to identify which variables are involved in multicollinearity: It is easy to show that $\|Xv_k\| = \sigma_k(X)$, so every "small" $\sigma_k(X)$ corresponds to a vector v_k which describes an approximate linear dependency between the columns of X. Looking at $\sigma_4(X)$ (the 5th element of `s$d`) we get

```
round(s$v[,5], 3)
```

```
# [1] -0.533 -0.001  0.460  0.503  0.502
```

Rounding the numbers some more and remembering that the first value corresponds to the intercept, we get that

$$-0.5 \cdot 1 + 0.5 \cdot x_2 + 0.5 \cdot x_3 + 0.5 \cdot x_4 \approx 0$$

or, equivalently

$$x_2 + x_3 + x_4 \approx 1.$$

Looking back over the original numbers we see that this relation indeed holds.

10.3 Mitigations

There are various ways problems resulting from multicollinearity can be addressed:

- Sometimes columns are linearly dependent, because a redundant input is present. Removing one of the input variables in a group of linearly dependent inputs can solve this problem.
- The inputs can be transformed by choosing new variables as functions of the original inputs. For example, in example 10.2 one could try the input variables $\tilde{x}_1 = (x_1 + x_2)/2$ and $\tilde{x}_2 = (x_1 - x_2)/2$. The confidence ellipse shown above indicates that it should be possible to get good estimates for the coefficient corresponding to $\tilde{x}_1$.
- If n is small, the problem may be resolved by getting more data. In particular, for $n < p + 1$, we *always* have strict multicollinearity, so in this case we definitely need more data.
- Alternative estimation approaches may be used. The basic idea is to sacrifice the requirement that $\hat{\beta}$ is unbiased, in exchange for a large variance reduction. In a later section we will discuss "ridge regression" as an example of this approach.

Summary

- Multicollinearity can lead to numerical problems and large variances in the estimated regression coefficients.
- The condition number of the design matrix can be used to detect multicollinearity.
- Singular value decomposition can used to better understand linear dependencies in the inputs.

Problem Sheet 3

You should attempt all these questions and write up your solutions in advance of the workshop in week 6 where the answers will be discussed.

9. Consider the following dataset. Our aim is to predict y from the variables x_1, x_2 and x_3.

i	$x_{1,i}$	$x_{2,i}$	$x_{3,i}$	y_i
1	-2.17	-2.08	-2.16	4.47
2	-1.35	-0.50	-0.74	5.60
3	-1.22	-1.00	-1.78	4.16
4	-1.04	-0.32	-0.40	5.52
5	-0.87	-0.39	-0.67	5.27
6	-0.41	0.07	-0.66	4.70
7	0.07	0.74	0.37	5.50
8	0.25	0.35	0.02	4.84
9	0.87	1.28	0.52	4.92
10	1.53	2.30	1.35	5.35
11	2.46	2.55	1.77	4.86
12	5.00	5.04	4.05	5.09

These data are also available in machine-readable form at

- https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/P03Q09.csv

Using these data:

a. Fit a linear model of the form $y_i = \beta_0 + x_{1,i}\beta_1 + x_{2,i}\beta_2 + x_{3,i}\beta_3 + \varepsilon_i$.

Solution:

To fit the required model we can use the following R commands:

```
url <- "https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/P03Q09.csv"
x <- read.csv(url)
m <- lm(y ~ x1 + x2 + x3, data=x)
```

The fitted model has the following coefficients:

```
summary(m)
```

```
#
# Call:
# lm(formula = y ~ x1 + x2 + x3, data = x)
#
# Residuals:
#       Min        1Q    Median        3Q       Max
# -0.104928 -0.032723 -0.001836  0.024802  0.142993
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  4.97725    0.06026  82.598 5.15e-13 ***
# x1          -1.11954    0.07427 -15.074 3.71e-07 ***
# x2           0.28176    0.12023   2.344   0.0472 *
# x3           1.06621    0.09492  11.232 3.54e-06 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
```

```
# Residual standard error: 0.07109 on 8 degrees of freedom
# Multiple R-squared:  0.9817,  Adjusted R-squared:  0.9748
# F-statistic: 142.8 on 3 and 8 DF,  p-value: 2.76e-07
```

b. Determine a 95% confidence interval for the coefficient β_2.

Solution: From lectures we know that a confidence interval for a single coefficient β_j is given by

$$[U, V] = \left[\hat{\beta}_j - t_{n-p-1}(\alpha/2)\sqrt{\widehat{\sigma^2}C_{jj}}, \hat{\beta}_j + t_{n-p-1}(\alpha/2)\sqrt{\widehat{\sigma^2}C_{jj}}\right],$$

where t_{n-p-1} is the $(1-\alpha/2)$-quantile of the t-distribution, $C_{jj} = (X^\top X)^{-1}_{jj}$, and X is the design matrix.

We can read off the required values for computing the confidence interval from the output of `summary(m)`: the centre of the confidence interval can be found in the column `Estimate`, the standard error $\sqrt{\widehat{\sigma^2}C_{jj}}$ is given in column `Std. Error`, and $n-p-1 = 12-3-1 = 8$. Using these values, we can find a 95%-confidence interval for β_2 as follows:

```
c(0.28176 - qt(0.975, 8) * 0.12023, 0.28176 + qt(0.975, 8) * 0.12023)
```

```
# [1] 0.004509123 0.559010877
```

Thus, the confidence interval is $[0.0045, 0.5590]$.

c. Perform a hypothesis test, at the 95%-level, for the hypothesis $H_0 \colon \beta_2 = 0$ with alternative $H_1 \colon \beta_2 \neq 0$.

Solution: The test statistic is

$$T = \frac{|\hat{\beta}_2|}{\sqrt{\widehat{\sigma^2}C_{jj}}} = \frac{0.28176}{0.12023} = 2.344.$$

(This value is also shown in the column `t value` of the R summary output.) The critical value is

$$t = t_{n-p-1}(\alpha/2) = 2.306.$$

Since $T > t$, we can reject the hypothesis $\beta_2 = 0$ at the 95%-level.

10. Let $x_1, \ldots, x_n, y_1, \ldots, y_n \in \mathbb{R}$ be given. Assume that it is known that the y-values satisfy $y \approx 1 - x + cx^2 - dx^3$, where c and d are unknown constants.

a. Explain how linear regression can be used to estimate the parameters c and d from the given data.

Solution: We can rewrite the equation for y as

$$y + x - 1 = cx^2 - dx^3,$$

where c and d are the unknown quantities, and x and y are given. Thus we can estimate c and d by fitting a linear model with two inputs, x_i^2 and x_i^3 and one output $y_i + x_i - 1$. The model does not include an intercept.

b. What is the design matrix in this situation?

Solution: Since there is no intercept, the design matrix is

$$X = \begin{pmatrix} x_1^2 & x_1^3 \\ x_2^2 & x_2^3 \\ \vdots & \vdots \\ x_n^2 & x_n^3 \end{pmatrix} \in \mathbb{R}^{n \times 2}.$$

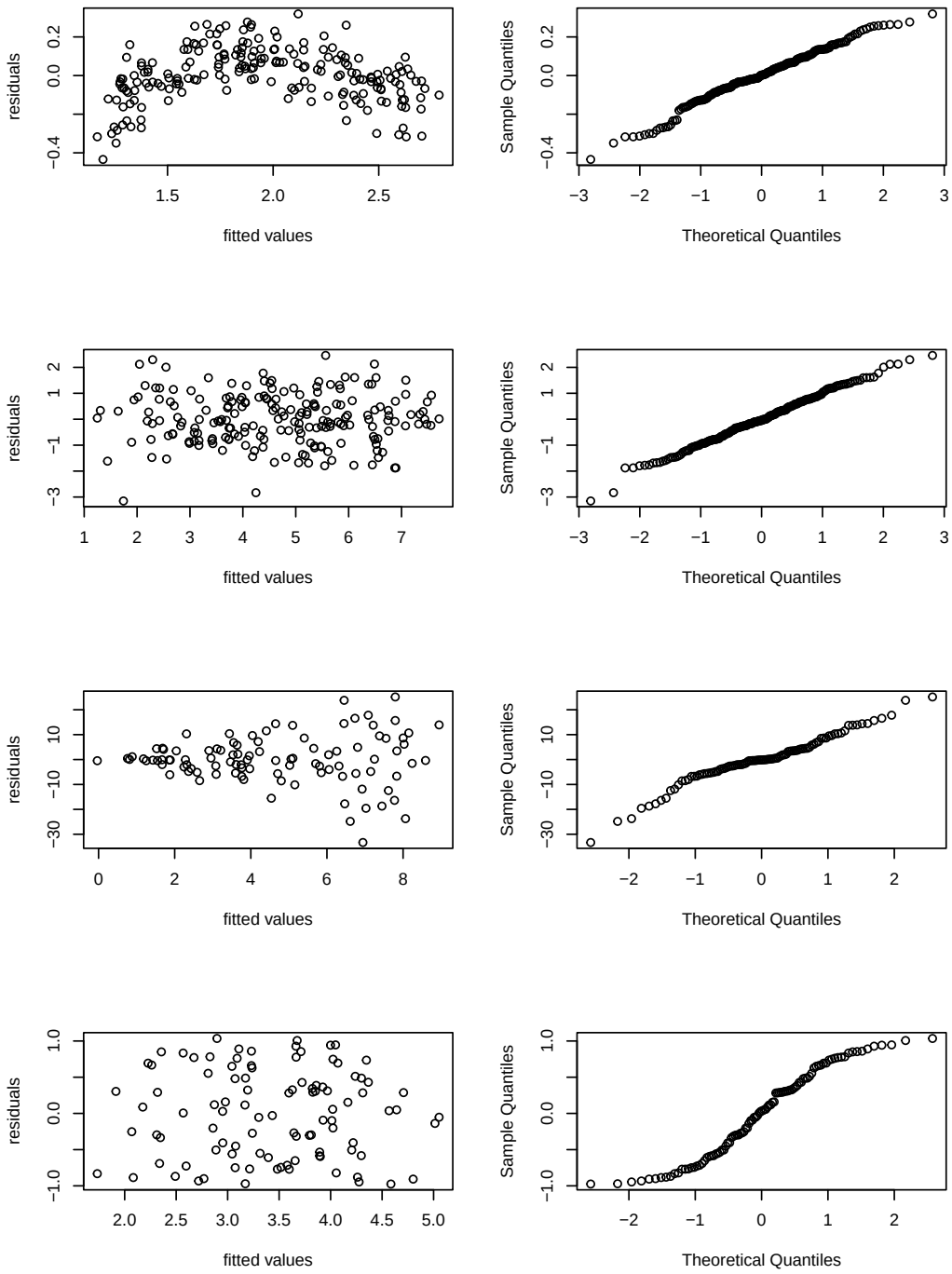
c. Why can *linear* regression be used, despite the presence of the *non-linear* terms x^2 and x^3?

Solution: A linear model is appropriate, because the response depends on the unknown coefficients c and d in a linear way: c and d are only multiplied by known constants, and there are no non-linear functions of c and d in the model.

11. In this question we consider four different datasets, given by inputs x_i and responses y_i for $i \in \{1, 2, 3, 4\}$.

a. Based on the following plots, discuss model fit of each model. Describe all relevant features of the plots. Describe any problems with the model fit you find.

```
par(mfrow=c(4,2))

m1 <- lm(y1 ~ x1)
plot(fitted(m1), resid(m1), xlab="fitted values", ylab="residuals")
qqnorm(resid(m1), main=NULL)

m2 <- lm(y2 ~ x2)
plot(fitted(m2), resid(m2), xlab="fitted values", ylab="residuals")
qqnorm(resid(m2), main=NULL)

m3 <- lm(y3 ~ x3)
plot(fitted(m3), resid(m3), xlab="fitted values", ylab="residuals")
qqnorm(resid(m3), main=NULL)

m4 <- lm(y4 ~ x4)
plot(fitted(m4), resid(m4), xlab="fitted values", ylab="residuals")
qqnorm(resid(m4), main=NULL)
```



Solution:

We first note that the given R commands generate the plots along rows, so the first row of plots corresponds to model `m1`, the second row to model `m2`, and so on. The plots shown are residual plots in the left column, and Q-Q-plots in the right column.

- **m1:** Samples in the residual plot on the left seem to follow an upside down parabola indicating a non-linear relationship between x and y. Samples in the QQ-plot on the right seem to (approximately?) follow a straight line, so residuals seem to be (approximately?) normally distributed. No outliers are visible.
- **m2:** Samples in the residual plot form a band centred at zero. Possibly very negative residuals are less frequent for fitted values $\hat{y}_i < 4$? No outliers are apparent.

The samples in the QQ-plot seem to clearly fall on a straight line. Overall, model fit seems very good.

- **m3:** The samples in the residual plot are centred around $\hat{\varepsilon} = 0$, but the width of the spread increases as the fitted values increase, leasing to a triangular shape. This indicates that the variance of the residuals is not constant but increases with y. Possibly, a model with multiplicative noise would be appropriate for these data. The QQ-plot shows an approximately straight line, but fit is not as good as for model `m2`.
- **m4:** The samples in the residual plot form a horizontal band centred at $\hat{\varepsilon} = 0$, no outliers are visible. The QQ-plot shows clearly that the residuals are not normally distributed: the samples form an "S-shaped" line, indicating that the errors have lighter tails than we would expect for a normal distribution.

b. Which of the four models has the best fit?

Solution: The data in `m1` seems to have a non-linear relationship between x and y, the data in `m3` seems to be better modelled using multiplicative noise, and the residuals in `m4` seem not to be normally distributed. Thus, the model with the best fit is `m2`, because it is the only model which does not show any obvious problems.

12. Using singular value decomposition, we can write a design matrix X as $X = UDV^\top$, where $D \in \mathbb{R}^{(p+1)\times(p+1)}$ is a diagonal matrix and $U \in \mathbb{R}^{n\times(p+1)}$ and $V \in \mathbb{R}^{(p+1)\times(p+1)}$ are matrices with orthonormal columns. Denote the diagonal elements of D by $\sigma_0(X), \dots, \sigma_p(X) \geq 0$, and the columns of V by $v_0, \dots, v_p$. Show that $\|Xv_k\| = \sigma_k(X)$. (We used this fact in example 10.3.)

Solution: Using the singular value decomposition of X, we find

$$\|Xv_k\| = \|UDV^\top v_k\|.$$

For every vector x we have $\|Ux\|^2 = x^\top U^\top U x = x^\top x = \|x\|^2$. Thus, multiplying a vector x by U does not change its length. Using this rule we find

$$\|Xv_k\| = \|DV^\top v_k\|.$$

Since the columns of V are orthonormal, we have $V^\top v_k = e_k$, where $e_k = (0, \dots, 1, \dots, 0)$ is the kth standard basis vector. Thus we get

$$\|Xv_k\| = \|De_k\| = \|\sigma_k(X)e_k\| = \sigma_k(X)\|e_k\| = \sigma_k(X)$$

as required. This completes the proof.

11 Improving the Model Fit

When we developed the theory for linear models, we used the following assumptions:

- Linear relationship between inputs and outputs: $y \approx x^\top \beta$
- Independence of errors: the $\varepsilon_i = y_i - (X\beta)_i$ are independent of each other.
- Normally distributed errors: $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ for all i. In particular, the variance of the ε_i does not depend on i.

The diagnostic plots mentioned in section 8.1 can often reveal if the data do not fit these modelling assumptions. In cases where the assumptions are violated, sometimes a linear model can still be used if the data is transformed first.

11.1 Linearising the Mean

If the model mean is a non-linear function of the inputs, we can sometimes transform the variables to achieve a linear relationship. We list some examples of non-linear models which can be transformed to linear models:

nonlinear model	transformation	linear Model
$y \approx \beta_0 x^{\beta_1}$	$x' = \log(x)$, $y' = \log(y)$	$y' \approx \log(\beta_0) + \beta_1 x'$
$y \approx \beta_0 e^{\beta_1 x}$	$y' = \log y$	$y' \approx \log \beta_0 + \beta_1 x$
$y \approx \beta_0 + \beta_1 \log x$	$x' = \log x$	$y \approx \beta_0 + \beta_1 x'$
$y \approx \frac{x}{\beta_0 x - \beta_1}$	$x' = 1/x$, $y' = 1/y$	$y' \approx \beta_0 - \beta_1 x'$

In all such cases we also would need to check the residuals of the transformed models, to see whether linear regression can be used for the transformed model.

11.2 Stabilising the Variance

The assumption of constant variance is a basic requirement of regression analysis. A common reason for the violation of this assumption is for the response variable y to follow a distribution in which the variance depends on y or $\mathbb{E}(y)$ and thus on x.

Example 11.1. The error in our model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon = x^\top \beta + \varepsilon$$

is sometimes called **additive error**, since it is added to the model mean $x^\top \beta$. Sometimes the error is instead given in percentages of the quantity of interest. In these cases we speak of **multiplicative error**. This can, for example, be modelled as

$$Y = (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p) \exp(\varepsilon) = x^\top \beta \exp(\varepsilon).$$

For $\varepsilon = 0$ we have $\exp(0) = 1$ and thus $Y = x^\top \beta$. Similarly, for small ε we have $Y \approx x^\top \beta$, but the variance is now proportional to $(x^\top \beta)^2$ instead of being constant. Also, since the exponential is nonlinear we only have $\mathbb{E}(Y) \approx x^\top \beta$ instead of strict equality.

Some commonly-used variance stabilising transformations are:

variance	transformation
$\sigma^2 = \text{constant}$	no transformation
$\sigma^2 \propto y$	$y' = \sqrt{y}$
$\sigma^2 \propto y^2$	$y' = \log y$
$\sigma^2 \propto y^3$	$y' = \frac{1}{\sqrt{y}}$

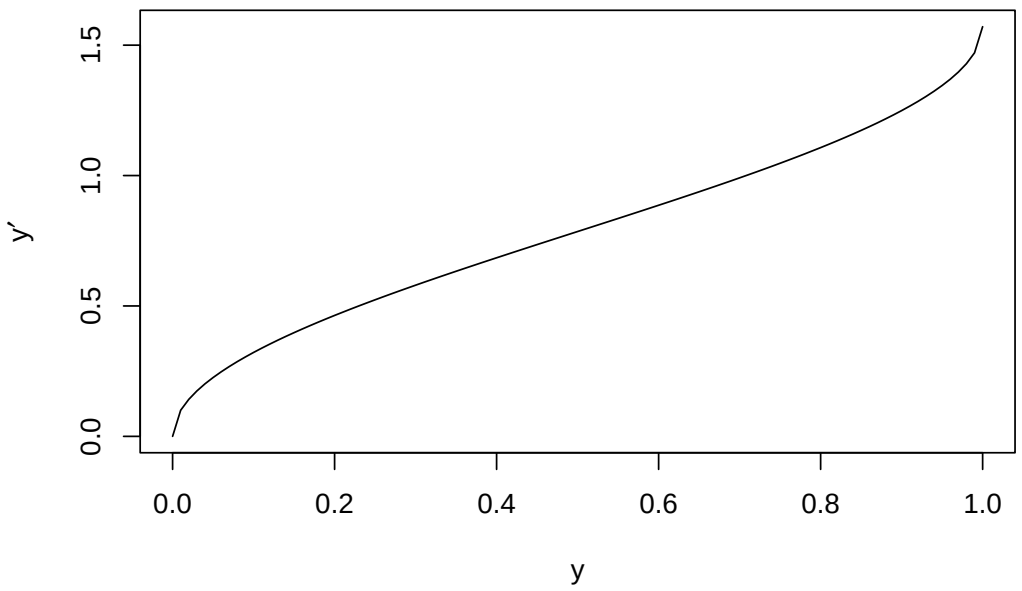
variance	transformation
$\sigma^2 \propto y^4$	$y' = \frac{1}{y}$
$\sigma^2 \propto y(1-y)$ where $y \in [0,1]$	$y' = \arcsin(\sqrt{y})$

Of course we do not have accurate knowledge of the relationship, but it can be diagnosed from the residual plots and transformations can be selected by experimenting with different choices. Any of these transformations will also affect the mean and we need to check the model fit for the transformed data, to see whether the transformed data can still reasonably be described by a linear model.

Example 11.2. The last transformation in the table above corresponds to the case of binomial sampling: If $x = p$ and $Y \sim B(n,p)/n$ then we have $\mathbb{E}(Y) = np/n = x$ and a linear model may be appropriate. But we also have $\mathrm{Var}(Y) = p(1-p)/n \propto \mathbb{E}(Y)(1-\mathbb{E}(Y))$, so the assumption of constant variance is violated.

We try to apply the transformation suggested in the table. The function arcsin is the inverse of the sine function. In R this function is available as `asin()`. To get some intuition about this transformation, we plot the function $y \mapsto \arcsin(\sqrt{y})$:

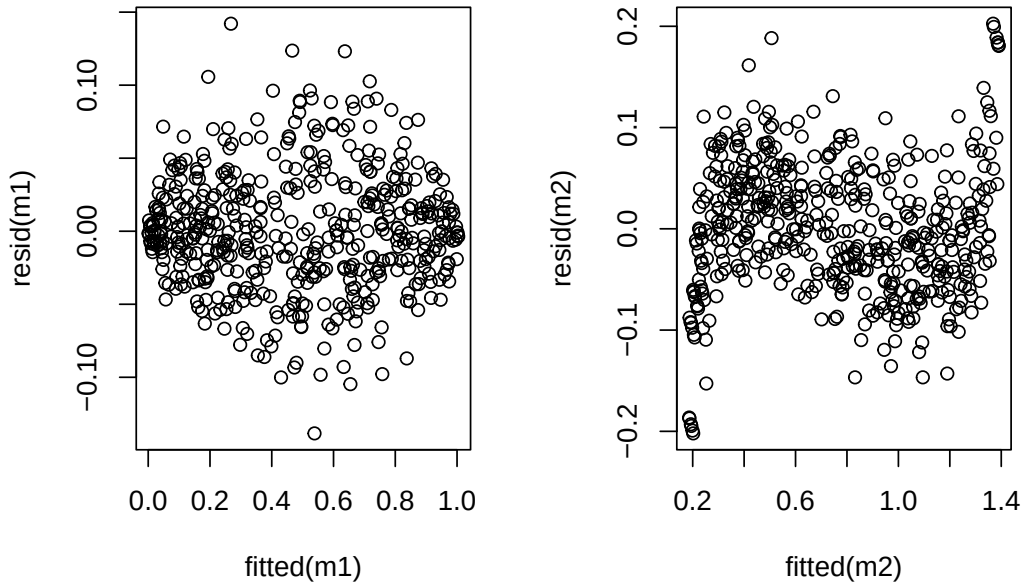
```
y <- seq(0, 1, l=101)
plot(y, asin(sqrt(y)), type = "l",
     ylab = expression(y * minute))
```



We can see that the transformation is approximately linear for most of the interval, but has a steeper slope near the edges. The effect of this is to increase the size of fluctations for small and large y-values. We now consider residual plots for the original and transformed data, for a simulated dataset:

```
n <- 500
x <- runif(n, 0, 1)
y <- rbinom(n, 100, x) / 100

par(mfrow = c(1, 2))

m1 <- lm(y ~ x)
plot(fitted(m1), resid(m1))

y.prime <- asin(sqrt(y))
```

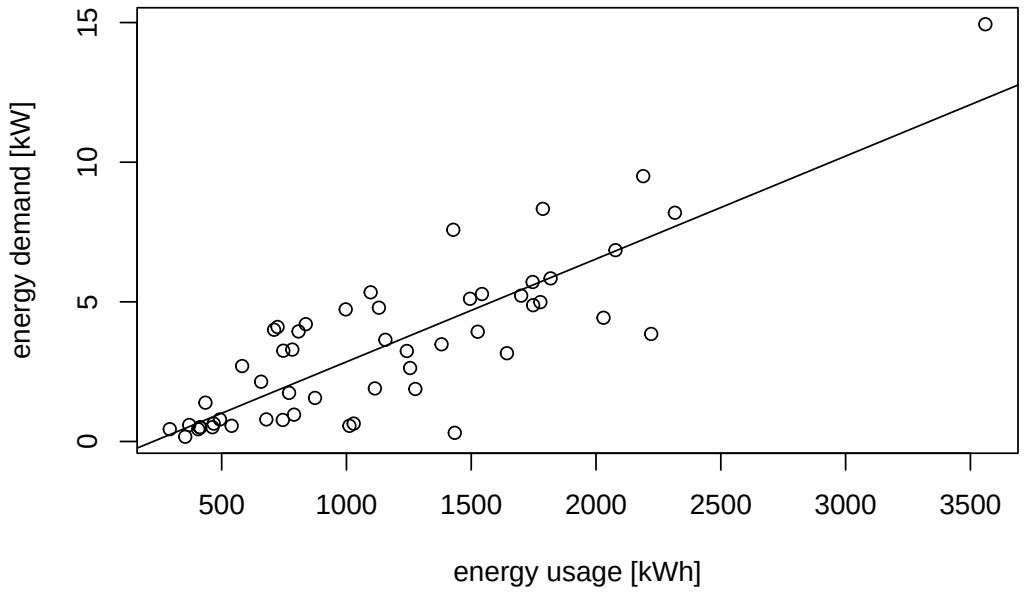
```
m2 <- lm(y.prime ~ x)
plot(fitted(m2), resid(m2))
```



The plot shows that the variance has indeed improved, while linearity has suffered (an S-shaped curve is now visible). Neither model is perfect and whether the transformation is beneficial or not depends on the particular circumstances.

Example 11.3. An electric utility company is interested in developing a model relating the peak hour demand (y, measured in kW) to total energy usage during the month (x, in kWh). A scatter plot of the data is shown below:

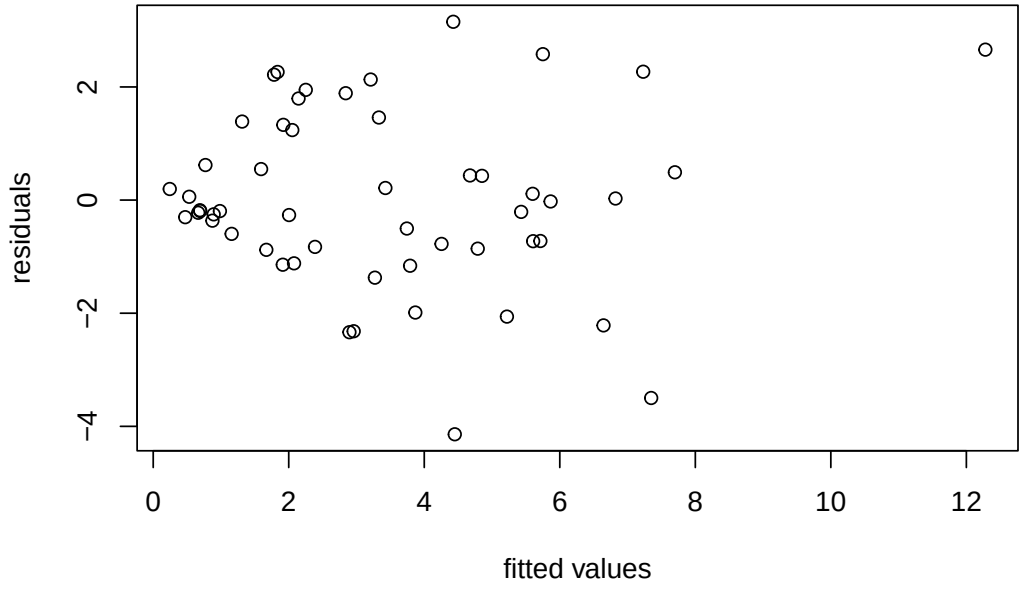
```
# data at https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/electric.dat
d <- read.table("data/electric.dat", header=FALSE)
plot(d$V1, d$V2,
     xlab="energy usage [kWh]", ylab="energy demand [kW]")
m1 <- lm(V2 ~ ., data = d)
abline(m1)
```



A linear relationship looks plausible, but we can see that the spread of points around

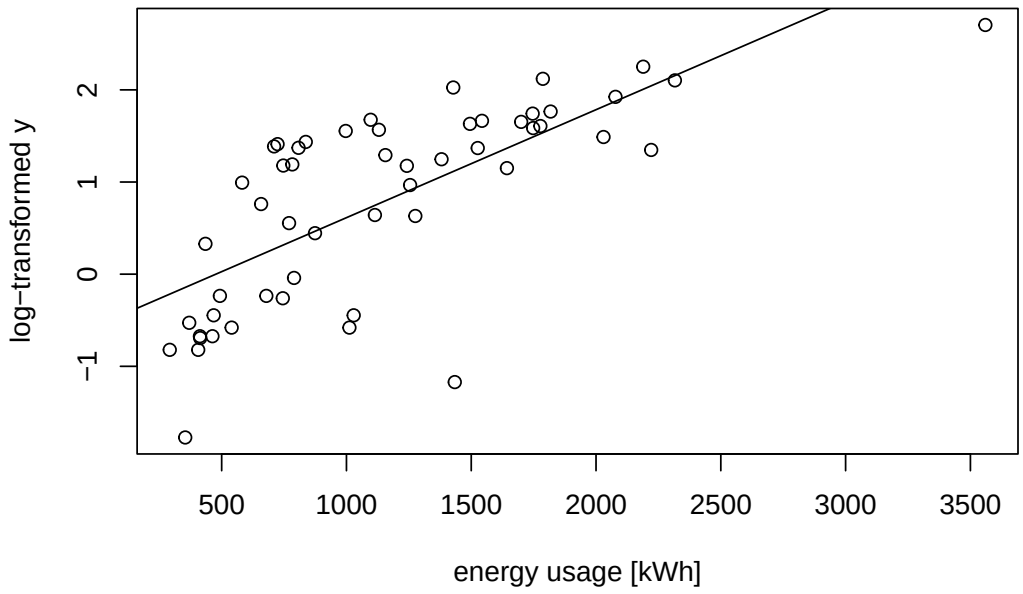
the regression line widens as energy usage increases. this is more clearly visible in a residual plot:

```
plot(fitted(m1), resid(m1),
     xlab = "fitted values", ylab = "residuals")
```

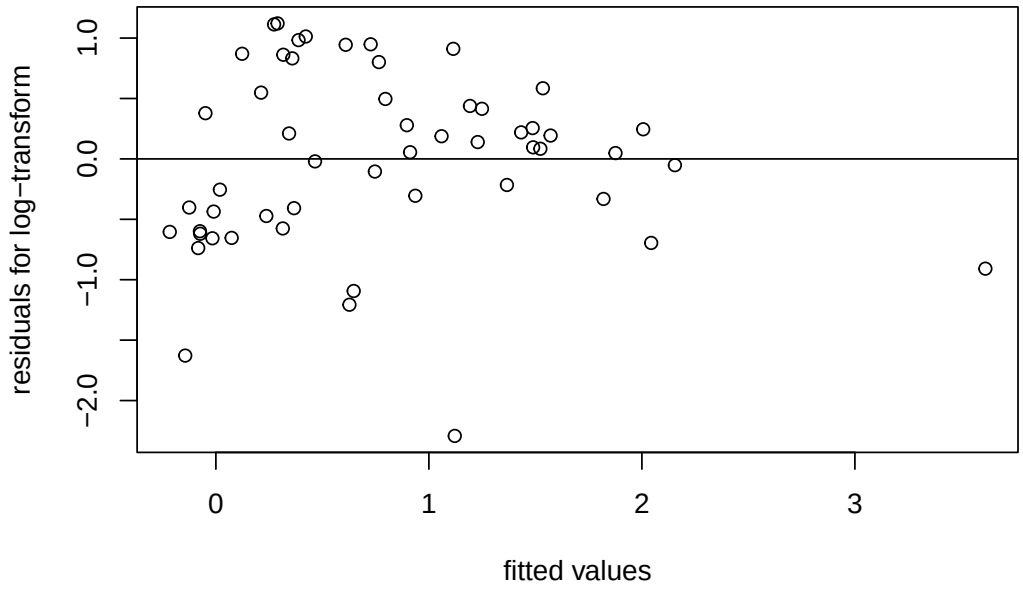


We try a transformation of the data to stabilise the variance. From the "wedge" shape on the left hand side of the plot, it is clear that the variance σ^2 increases as y increases. Thus we can try transformations like $y' = \log(y)$ or $y' = \sqrt{y}$. Taking the logarithm for illustration, we get

```
m2 <- lm(log(V2) ~ ., data = d)
plot(d$V1, log(d$V2),
     xlab="energy usage [kWh]", ylab="log-transformed y")
abline(m2)
```




```
plot(fitted(m2), resid(m2),
     xlab = "fitted values", ylab = "residuals for log-transform")
abline(h = 0)
```



The spread of residuals now looks somewhat more reasonable.

11.3 The Power Transform

A family of transformations for the response variable y is given by the **power transform**. These transformations only apply to strictly positive data, *i.e.* $y > 0$, and are defined by

$$y^{(\lambda)} = \begin{cases} \dfrac{y^\lambda - 1}{\lambda g^{\lambda-1}} & \text{if } \lambda \neq 0 \\ g \log y & \text{if } \lambda = 0 \end{cases}$$

where $g = (\prod_{i=1}^n y_i)^{1/n}$ is the geometric mean.

The geometric mean is a constant (it does not depend on i) and is not needed for the power transform, but it is usually included to make values for different λ more comparable. For $\lambda = 1$ the transform is $y' = y - 1$. This is a simple shift which has no effect on the fitted model, it just decreases the intercept by 1 and leaves the residuals unchanged. Thus, this case is that same as applying no transformation.

Using Taylor approximation on the numerator in the definition of $y^{(\lambda)}$ we get

$$y^\lambda = \exp(\log(y^\lambda)) = \exp(\lambda \log(y)) \approx 1 + \lambda \log(y) + O\Big((\lambda \log(y))^2\Big),$$

where the $O(\cdots)$ stands for terms which are negligible as λ converges to 0. Thus the formula given for $\lambda \neq 0$ converges to the case for $\lambda = 0$ as λ converges to 0. This makes the transformation continuous as a function of λ.

A heuristic rule to find a range of λ for which the power transform is appropriate is based on how the the residual sum of squares changes with λ: Let

$$r(\lambda) = \sum_{i=1}^n (y_i^{(\lambda)} - \hat{y}_i^{(\lambda)})^2,$$

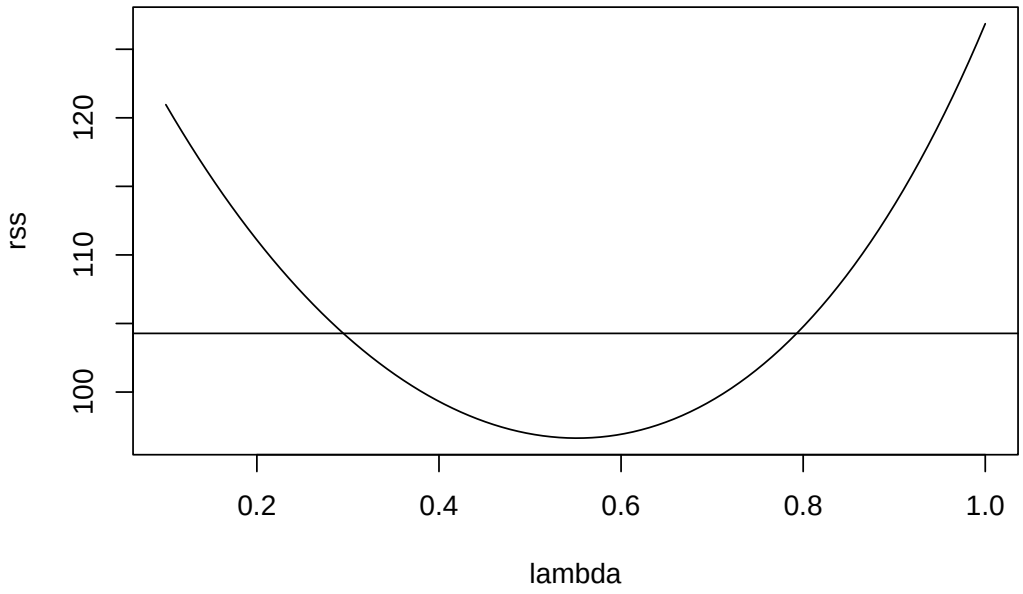
where $\hat{y}_i^{(\lambda)}$ denotes the fitted value for the model using the transformed data $y_i^{(\lambda)}$. It is easy to plot this function numerically. We want to choose λ close to where the function has its minimum. The heuristic rule is to consider all values of λ with

$$r(\lambda) \leq r(\lambda_{\min})\Big(1 + \frac{t_{n-p-1}(\alpha/2)^2}{n-p-1}\Big), \tag{31}$$

where λ_{min} is the value of λ where the residual sum of squares has its minimum and $t_{n-p-1}(\alpha/2)$ is the $(1-\alpha/2)$-quantile of the $t(n-p-1)$-distribution. One can show that this is an approximate $(1-\alpha)$ confidence interval for λ. If we want to interpret the model, it is usually better to select a "simple" λ, *e.g.* $\lambda = 0.5$ rather than using the "optimal" value λ_{min}.

Example 11.4. Continuing from example 11.3 above, we can try a power transform for the data. We start by plotting $r(\lambda)$ as a function of λ, together with the cutoff suggested by equation (31):

```
x <- d$V1
y <- d$V2
gm <- exp(mean(log(y))) # more stable way to compute geometric mean
lambda <- seq(0.1, 1, length.out = 101)
rss <- numeric(length(lambda))
for (i in seq_along(lambda)) {
    li <- lambda[i]
    y.prime <- (y^li - 1) / (li * gm^(li-1))
    mi <- lm(y.prime ~ x)
    rss[i] <- sum(resid(mi)^2)
}
plot(lambda, rss, type="l")

n <- nrow(d) # 53
p <- 1
cutoff <- min(rss) * (1 + qt(0.975, n - p - 1)^2 / (n - p - 1))
abline(h = cutoff)
```



This suggests the range of reasonable λ values to be

```
range(lambda[which(rss <= cutoff)])
```

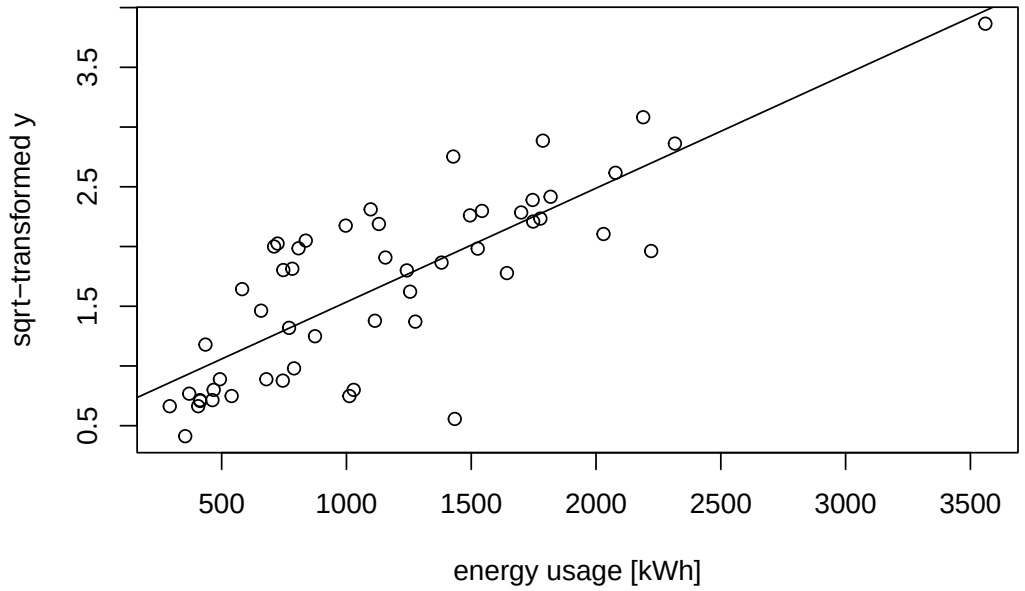
```
# [1] 0.298 0.784
```

The value $\lambda = 1$ (no transformation) is not contained in the interval, suggesting that a transformation may be helpful. Choosing a "simple" value inside the interval and close to the minimum, we try $\lambda = 0.5$. This leads to the following transformation:

$$y' = 2\sqrt{g}(\sqrt{y} - 1).$$

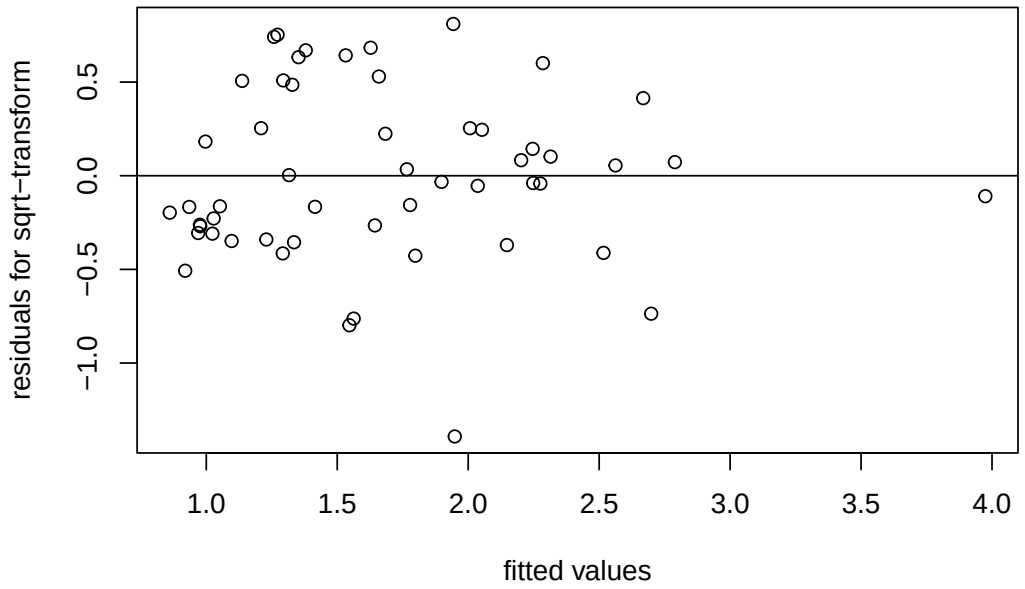
Up to the shift and scaling, this just takes the square root of the data.

```
y.prime <- sqrt(y)
plot(x, y.prime,
     xlab="energy usage [kWh]", ylab="sqrt-transformed y")
m3 <- lm(y.prime ~ x)
abline(m3)
```



To see whether the variance of the residuals has improved, we consider a residual plot:

```
plot(fitted(m3), resid(m3),
     xlab = "fitted values", ylab = "residuals for sqrt-transform")
abline(h = 0)
```



We see that the spread of the residuals has indeed improved, when compared to fitting the original data.

11.4 Orthogonal Inputs

So far, this chapter has been concerned with what to do when there are problems with a given model. In some cases we have the chance to "plan ahead" so that problems are less likely to occur.

Collinearity occurs when there is high correlation amongst the explanatory variables. In some situations we have the opportunity to *choose* the design space X. In these cases, there are advantages in creating design variables which are orthogonal, thus avoiding or reducing problems caused by collinearity.

Example 11.5. Suppose we have observations (x_i, y_i) for $i \in \{1, \dots, n\}$ and we want to fit a polynomial model of the form

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_p x_i^p + \varepsilon_i.$$

If we use the actual inputs x to create new variables, say $x_j = x^j$, then severe multicollinearity can arise. For example, for

$$x = (10000, 10001, \dots, 10010),$$

we have

$$x^2 = 10^8 + (0, 20001, 40004, \dots, 200100)$$

and $\text{cor}(x, x^2) \approx 1 - 9.74 \times 10^{-9}$:

```
x <- seq(10000, 10010)
cor(x, x^2)
```

```
# [1] 1
```

```
1 - cor(x, x^2)
```

```
# [1] 9.740257e-09
```

```
X <- cbind(1, x, x^2)
kappa(t(X) %*% X, exact = TRUE)
```

```
# [1] 1.31431e+24
```

The first result, for the correlation, is shown as 1 due to rounding. The internal number representation of the correlation is actually less than 1 and we can print the difference `1 - cor(x, x^2)` to get the result stated above. Since this difference is close to 0 (instead of close to 1), R can now use scientific notation to show the very small result. The final line of the output shows that the condition number is greater than 10^{23}, so the problem indeed suffers severe multicollinearity.

We can greatly improve the conditioning of the problem by using the centred data $x_i^* = x_i - \bar{x}$ before taking powers:

```
x.star <- x - mean(x)
cor(x.star, x.star^2)
```

```
# [1] 0
```

```
X.star <- cbind(1, x.star, x.star^2)
kappa(t(X.star) %*% X.star, exact = TRUE)
```

```
# [1] 408.7796
```

Now the correlation between x^* and $(x^*)^2$ is exactly zero, and the condition number is much improved.

A more systematic approach to choose input variables for polynomial regression is to use **orthogonal polynomials** of the form

$$\begin{aligned}\psi_0(x) &= 1\\ \psi_1(x) &= \alpha_{1,1}x + \alpha_{1,0}\\ \psi_2(x) &= \alpha_{2,2}x^2 + \alpha_{2,1}x + \alpha_{2,0}\\ &\vdots\end{aligned}$$

where the coefficients $\alpha_{j,k}$ are chosen such that

$$\sum_{i=1}^{n} \psi_j(x_i)\psi_k(x_i) = 0$$

for all $j, k \in \{0, \dots, p\}$ with $j \neq k$. We can then use input variables

$$x_j = \psi_j(x)$$

for $j \in \{0, \dots, p\}$ to get a design matrix X such that

$$\begin{aligned}(X^\top X)_{jk} &= \sum_{i=1}^{n} X_{ij}X_{ik}\\ &= \sum_{i=1}^{n} \psi_j(x_i)\psi_k(x_i)\\ &= 0\end{aligned}$$

for all $j, k \in \{0, \dots, p\}$ with $j \neq k$. In this case the columns of X are orthogonal and there is no multicollinearity. Our model is now

$$\begin{aligned}y_i &= \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \varepsilon_i\\ &= \beta_0\psi_0(x_i) + \beta_1\psi_1(x_i) + \beta_2\psi_2(x_i) + \cdots + \beta_p\psi_p(x_i) + \varepsilon_i.\end{aligned}$$

Since $X^\top X$ is diagonal, the inverse of this matrix is trivial to compute. If we denote the diagonal elements by

$$A_j = (X^\top X)_{jj},$$

then $X^\top X = \operatorname{diag}(A_0, \dots, A_p)$ and $(X^\top X)^{-1} = \operatorname{diag}(1/A_0, \dots, 1/A_p)$. Thus we get

$$\begin{aligned}\hat{\beta}_j &= \left((X^\top X)^{-1}X^\top y\right)_j\\ &= \frac{1}{A_j}\sum_{i=1}^{n} \psi_j(x_i)y_i\end{aligned}$$

and similarly, using equation (17), we find

$$\operatorname{Var}(\hat{\beta}_j) = \frac{\sigma^2}{A_j}.$$

An additional advantage of this approach is, that we can add another term, $\psi_{p+1}(x)$, without changing the previous estimates of the regression coefficients. Since $X^\top X$ is diagonal, the change will not affect the estimates $\hat{\beta}_j$ for $j \in \{0, \dots, p\}$ (but the change will affect σ^2).

Example 11.6. In R we can create specific orthogonal entries for polynomial regression using the command `poly(x, p)`. The output of this function is a matrix with elements $\psi_j(x_i)$ for $j \in \{1, \dots, p\}$ (omitting the intercept) and $i \in \{1, \dots, n\}$. Continuing from the example above we find

```
x <- seq(10000, 10010)
X <- cbind(1, poly(x, 2)) # manually prepend a column of ones
round(t(X) %*% X, 5)
```

```
#       1 2
#   11 0 0
# 1  0 1 0
# 2  0 0 1
```

Different from our previous approach, the last two columns are now also orthogonal to the columns of ones. This further improves the conditioning of the design matrix:

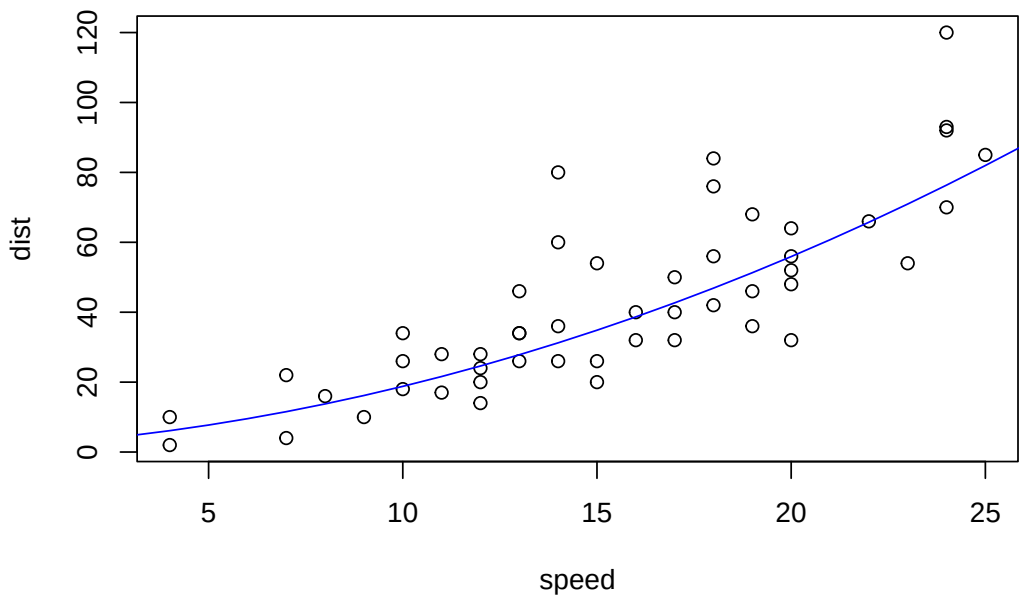
```
kappa(X, exact = TRUE)
```

```
# [1] 3.316625
```

We see that the use of orthogonal polynomials entirely eliminated any problems with multicollinearity.

Example 11.7. To illustrate the use of `poly()` for fitting a polynomial model to real data, we use the built-in `cars` dataset in R. This dataset contains historic data (from the 1920s) about the speed of cars (in mph) and the distances taken to stop (in ft). We first fit a second order polynomial to the data using the naive approach:

```
m1 <- lm(dist ~ speed + I(speed^2), data = cars)
plot(cars)
lines(predict(m1, newdata = data.frame(speed=seq(0, 30, l=31))),
      col="blue")
```



Checking the condition number, we see that there is severe multicollinearity:

```
kappa(m1, exact = TRUE)
```

```
# [1] 2160.976
```

We can also see that none of the coefficients is significantly different from zero, while the F-test rejects the hypothesis that the vector of all regression coefficients is zero. We have seen previously that this effect can be an indication of multicollinearity.

```
summary(m1)
```

```
#
# Call:
# lm(formula = dist ~ speed + I(speed^2), data = cars)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -28.720  -9.184  -3.188   4.628  45.152
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  2.47014   14.81716   0.167    0.868
# speed        0.91329    2.03422   0.449    0.656
# I(speed^2)   0.09996    0.06597   1.515    0.136
#
# Residual standard error: 15.18 on 47 degrees of freedom
# Multiple R-squared:  0.6673,  Adjusted R-squared:  0.6532
# F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

We can fit an improved model using orthogonal polynomials. Since `lm()` automatically adds the intercept, we don't need to prepend a column of ones to the `poly()` output here.

```
m2 <- lm(dist ~ poly(speed, 2), data = cars)
kappa(m2)
```

```
# [1] 6.670129
```

We see that in the new model there are no problems with multicollinearity any more. The new model also has coefficients which are significantly different from zero:

```
summary(m2)
```

```
#
# Call:
# lm(formula = dist ~ poly(speed, 2), data = cars)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -28.720  -9.184  -3.188   4.628  45.152
#
# Coefficients:
#                 Estimate Std. Error t value Pr(>|t|)
# (Intercept)       42.980      2.146  20.026  < 2e-16 ***
# poly(speed, 2)1  145.552     15.176   9.591 1.21e-12 ***
# poly(speed, 2)2   22.996     15.176   1.515    0.136
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 15.18 on 47 degrees of freedom
# Multiple R-squared:  0.6673,  Adjusted R-squared:  0.6532
# F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

Summary

- We have seen different transformations which can be used to transform a nonlinear model into a linear one.
- We have discussed transformations which can stabilise the variance of the errors in the model.

- We have seen how orthogonal polynomials can be used to avoid multicollinearity in polynomial regression.

12 Ridge Regression

In cases where X shows approximate multicollinearity, the parameter estimates $\hat{\beta}$ have coefficients with large variance. In case of exact multicollinearity, $\hat{\beta}$ is no longer uniquely defined, and the minimum of the residual sum of squares is attained on an (unbounded) subspace of the parameter space. In both cases, a more stable estimate can be obtained by changing the estimation procedure to include an additional penalty term which discourages "large" values of $\hat{\beta}$. This is the idea of ridge regression. In this section we will see that the resulting estimates are biased, but sometimes have smaller mean squared error than the least squares estimator.

12.1 Definition the Estimator

Definition 12.1. In multiple linear regression, the **ridge regression** estimate for the parameter vector β is given by

$$\hat{\beta}^{(\lambda)} = \arg\min_{\beta} \Big(\sum_{i=1}^{n} (y_i - (X\beta)_i)^2 + \lambda \|\beta\|^2 \Big). \tag{32}$$

The parameter $\lambda \geq 0$ is called the **regularisation parameter**.

Here we write $\arg\min_{\beta}(\cdots)$ to denote the value of β which minimises the expression on the right-hand side, and $\|\beta\|$ denotes the Euclidean norm of the vector β, *i.e.*

$$\|\beta\|^2 = \sum_{j=0}^{p} \beta_j^2.$$

The term being minimised is

$$r^{(\lambda)}(\beta) = \sum_{i=1}^{n} (y_i - (X\beta)_i)^2 + \lambda \|\beta\|^2,$$

consisting of the residual sum of squares plus a penalty term. The regularisation parameter λ controls the relative weight of these two terms. For large λ, the main objective to minimise the norm $\|\beta\|$ and one can show that $\lim_{\lambda \to \infty} \hat{\beta}^{(\lambda)} = 0$. For $\lambda \downarrow 0$ only the residual sum of squares is left and if $X^\top X$ is invertible we get $\hat{\beta}^{(0)} = \hat{\beta}$, *i.e.* here the ridge regression estimate coincides with the least squares estimate.

As in lemma 2.1 we can find an explicit formula for the ridge regression estimate. For this we write $r^{(\lambda)}$ as

$$r^{(\lambda)} = (y - X\top\beta)^\top (y - X\top\beta) + \lambda \beta^\top \beta$$

and then take derivatives to find the minimum. The result is given by

$$\hat{\beta}^{(\lambda)} = (X^\top X + \lambda I)^{-1} X^\top y. \tag{33}$$

Example 12.1. To illustrate the method, we compute the ridge regression estimate for a badly conditioned toy dataset.

```
# fix the true parameters for our simulated data:
beta <- c(0, .5, .5)
sigma <- 0.01

set.seed(20211112)
x1 <- c(1.01, 2.00, 2.99, 4.02, 5.01, 5.99)
x2 <- c(0.98, 1.99, 3.00, 4.00, 4.99, 6.00)
```

```r
# beta[1] in R corresponds to \beta_0, beta[2] = \beta_1, beta[3] = \beta_2
y <- beta[1] + beta[2] * x1 + beta[3] * x2 + rnorm(6, 0, sigma)

lambda <- 0.01
X <- model.matrix(y ~ x1 + x2)
beta.ridge <- solve(t(X) %*% X + lambda * diag(ncol(X)), t(X) %*% y)
beta.ridge[,1] # get first column instead of n × 1 matrix, for tidiness
```

```
# (Intercept)          x1          x2
# -0.01620501  0.52739367  0.47343531
```

For comparison, we also compute the least squares estimate:

```r
m <- lm(y ~ x1 + x2)
coef(m)
```

```
# (Intercept)          x1          x2
# -0.02935851  1.03064453 -0.02749857
```

We see that despite the small value of λ there is a considerable difference between the estimated values $\hat{\beta}$ and $\hat{\beta}^{(\lambda)}$ in this example. Experimenting with different values of the seed also shows that the variance of $\hat{\beta}^{(\lambda)}$ is much smaller than the variance of $\hat{\beta}$. In the next section we will see a theoretical explanation for this fact.

For $\lambda > 0$ and $v \neq 0$ we have $\|v\| > 0$ and thus

$$\begin{aligned}
\|(X^\top X + \lambda I)v\|^2 &= v^\top (X^\top X + \lambda I)^\top (X^\top X + \lambda I)v \\
&= v^\top X^\top X X^\top X v + 2\lambda v^\top X^\top X v + \lambda^2 v^\top v \\
&= \|X X^\top X v\|^2 + 2\lambda \|X v\|^2 + \lambda^2 \|v\|^2 \\
&\geq \lambda^2 \|v\|^2 \\
&> 0.
\end{aligned}$$

This shows, by theorem A.1, that the matrix $X^\top X + \lambda I$ is invertible for every $\lambda > 0$. Thus the ridge regression estimate for $\lambda > 0$ is always uniquely defined, even in cases where the least squares estimate is *not* uniquely defined.

There are variants of the method where the penalty term $\|\beta\|^2$ is is replaced with a different penalty term. One example of this is Lasso (least absolute shrinkage and selection operator) regression, which uses $\|\beta\|_1 = \sum_{j=0}^{p} |\beta_j|$ as the penalty term.

Outside statistics, ridge regression is known as **Tikhonov regularisation**.

12.2 Properties of the Estimate

Using the statistical model

$$Y = X\beta + \varepsilon$$

as before, we have

$$\begin{aligned}
\hat{\beta}^{(\lambda)} &= (X^\top X + \lambda I)^{-1} X^\top Y \\
&= (X^\top X + \lambda I)^{-1} X^\top (X\beta + \varepsilon) \\
&= (X^\top X + \lambda I)^{-1} (X^\top X + \lambda I)\beta \\
&\qquad - (X^\top X + \lambda I)^{-1} \lambda I \beta \\
&\qquad + (X^\top X + \lambda I)^{-1} X^\top \varepsilon
\end{aligned}$$

and simplifying the first term on the right-hand side we get

$$\hat{\beta}^{(\lambda)} = \beta - \lambda (X^\top X + \lambda I)^{-1} \beta + (X^\top X + \lambda I)^{-1} X^\top \varepsilon. \tag{34}$$

12.2.1 Bias

Using the formula for $\hat{\beta}^{(\lambda)}$ we get

$$\mathbb{E}(\hat{\beta}^{(\lambda)}) = \beta - \lambda(X^\top X + \lambda I)^{-1}\beta + 0$$

and thus

$$\text{bias}(\hat{\beta}^{(\lambda)}) = -\lambda(X^\top X + \lambda I)^{-1}\beta.$$

Since this term in general is non-zero, we see that $\hat{\beta}^{(\lambda)}$ is a biased estimate. The amount of bias depends on the unknown, true coefficient vector β.

12.2.2 Variance

The covariance matrix of the vector $\hat{\beta}^{(\lambda)}$ is given by

$$\begin{aligned}
\text{Cov}(\hat{\beta}^{(\lambda)}) &= \text{Cov}\Big(\beta - \lambda(X^\top X + \lambda I)^{-1}X^\top X\beta + (X^\top X + \lambda I)^{-1}X^\top \varepsilon\Big) \\
&= \text{Cov}\Big((X^\top X + \lambda I)^{-1}X^\top \varepsilon\Big) \\
&= (X^\top X + \lambda I)^{-1}X^\top \,\text{Cov}(\varepsilon) X(X^\top X + \lambda I)^{-1} \\
&= \sigma^2(X^\top X + \lambda I)^{-1}X^\top X(X^\top X + \lambda I)^{-1} \\
&= \sigma^2(X^\top X + \lambda I)^{-1} - \lambda\sigma^2(X^\top X + \lambda I)^{-2}.
\end{aligned}$$

While this is an explicit formula, the resulting covariance matrix depends on the unknown error variance σ^2.

If $X^\top X$ is invertible, this covariance will convert to the covariance of the least squares estimator as $\lambda \downarrow 0$. From this we can find the variance of individual components as $\text{Var}(\hat{\beta}_j^{(\lambda)}) = \text{Cov}(\hat{\beta}^{(\lambda)})_{jj}$.

12.2.3 Mean Squared Error

The Mean Squared Error (MSE) of an estimator $\hat{\beta}_j$ for β_j is given by

$$\text{MSE}(\hat{\beta}_j) = \mathbb{E}\Big((\hat{\beta}_j - \beta_j)^2\Big).$$

It is an easy exercise to show that this can equivalently be written as

$$\text{MSE}(\hat{\beta}_j) = \text{Var}(\hat{\beta}_j) + \text{bias}(\hat{\beta}_j)^2.$$

Using the formulas for the variance and bias from above, we can find an formula for $\text{MSE}(\hat{\beta}_j^{(\lambda)})$, but the result is hard to interpret. Instead of considering these analytical expressions, we illustrate the MSE using a numerical example.

Example 12.2. Continuing from example 12.1 above, we can determine the MSE for every value of λ. Here we use the fact that this is simulated data where we know the true values of β and σ^2.

We start with the bias:

```
lambda <- 0.1
Q <- t(X) %*% X + lambda * diag(ncol(X))
lambda * solve(Q, t(X) %*% X %*% beta)
```

```
#                      [,1]
# (Intercept) 0.0009178679
# x1          0.0497789499
# x2          0.0499545553
```

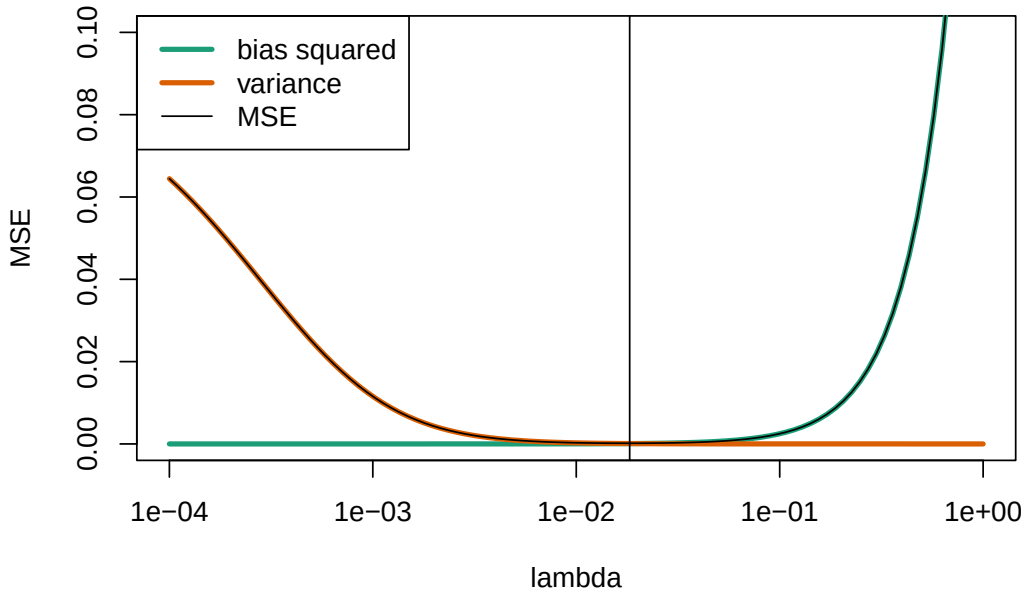
For the covariance matrix we get the following:

```r
C1 <- solve(Q) # Compute the inverse Q^{-1} ...
C2 <- C1 %*% C1 # ... and also Q^{-2}, to get
sigma^2 * (C1 - lambda*C2) # the covariance matrix.
```

```
#                (Intercept)            x1            x2
# (Intercept)  7.291379e-05 -7.580602e-06 -9.226941e-06
# x1          -7.580602e-06  3.831264e-06 -1.540098e-06
# x2          -9.226941e-06 -1.540098e-06  4.221464e-06
```

Now we repeat these calculations in a loop, to plot the MSE as a function of λ:

```r
lambda <- 10^seq(-4, 0, length.out = 100) # range of lambda to try
j.plus.1 <- 1 + 1 # we plot the error of \beta_1

variance <- numeric(length(lambda))
bias <- numeric(length(lambda))
for (i in seq_along(lambda)) {
    lambda.i <- lambda[i]
    Q <- t(X) %*% X + lambda.i * diag(ncol(X))
    C1 <- solve(Q)
    bias[i] <- (lambda.i * solve(Q, t(X) %*% X %*% beta))[j.plus.1]
    C2 <- C1 %*% C1
    C <- C1 - lambda.i*C2
    variance[i] <- sigma^2 * C[j.plus.1, j.plus.1]
}
MSE <- variance + bias^2

plot(lambda, MSE, type = "l", log = "x", ylim = c(0, 0.1))
lines(lambda, bias^2, col="#1b9e77", lw = 3)
lines(lambda, variance, col="#d95f02", lw = 3)
lines(lambda, MSE) # make sure MSE is not obscured by bias/variance
abline(v = lambda[which.min(MSE)]) # mark the minimum

legend("topleft",
       c("bias squared", "variance", "MSE"),
       col = c("#1b9e77", "#d95f02", "black"),
       lw = c(3, 3, 1))
```



The vertical line at $\lambda = 0.0183$ marks the optimal value of λ. We can see that for small λ the MSE is dominated by the variance, whereas for large λ the main contribution is from the bias.

12.3 Standardisation

While ridge regression can be applied directly to the original data, the penalty is only comparable between the components of β, if the components have similar magnitude. Also, we do not want a method which is depends on the units in which the x variables are measured. To address these issues, it is common practice to first standardise each of the x. In this way the penalty will apply equally to all of the coefficients, and the results can be transformed back to obtain $\hat{y}$ on the original scale.

We standardise every column of the input separately: For the input variable x_j, where $j \in \{1, \dots, p\}$, we get

$$w_{ij} := \frac{x_{ij} - \overline{x_j}}{\mathrm{s}_{x_j}},$$

for all $i \in \{1, \dots, n\}$. where $\overline{x_j} = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ and

$$\mathrm{s}_{x_j}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \overline{x_j})^2$$

is the sample variance. We similarly standardise the outputs:

$$z_i := \frac{y_i - \overline{y}}{\mathrm{s}_y},$$

where $\overline{y}$ and s_y are the mean and standard deviation of the y_i. If we denote the regression coefficients for the transformed data by γ, then the residual sum of squares for the transformed data is

$$r_{\mathrm{tfm}}(\gamma) = \sum_{i=1}^{n}\big(z_i - \gamma_0 - \gamma_1 w_{i1} - \cdots - \gamma_p w_{ip}\big)^2.$$

The transformed inputs satisfy

$$\sum_{i=1}^{n} w_{ij} = 0$$

for all $j \in \{1, \dots, p\}$ and a similar relation holds for the output. Using these relations we find

$$\begin{aligned}
r_{\mathrm{tfm}}(\gamma) &= \sum_{i=1}^{n}\Big(\big(z_i - \gamma_1 w_{i1} - \cdots - \gamma_p w_{ip}\big)^2 \\
&\qquad -2\gamma_0\big(z_i - \gamma_1 w_{i1} - \cdots - \gamma_p w_{ip}\big) + \gamma_0^2\Big) \\
&= \sum_{i=1}^{n}\big(z_i - \gamma_1 w_{i1} - \cdots - \gamma_p w_{ip}\big)^2 \\
&\qquad - 2\gamma_0\sum_{i=1}^{n}\big(z_i - \gamma_1 w_{i1} - \cdots - \gamma_p w_{ip}\big) + n\gamma_0^2 \\
&= \sum_{i=1}^{n}\big(z_i - \gamma_1 w_{i1} - \cdots - \gamma_p w_{ip}\big)^2 \\
&\qquad - 2\gamma_0\Big(\sum_{i=1}^{n} z_i - \gamma_1\sum_{i=1}^{n} w_{i1} - \cdots - \gamma_p\sum_{i=1}^{n} w_{ip}\Big) + n\gamma_0^2 \\
&= \sum_{i=1}^{n}\big(z_i - \gamma_1 w_{i1} - \cdots - \gamma_p w_{ip}\big)^2 + n\gamma_0^2.
\end{aligned}$$

Thus, the coefficient for the intercept can be minimised separately and the optimal value is always $\gamma_0 = 0$. For this reason we do not include the intercept in our model for the standardised data. The design matrix is thus

$$W = \begin{pmatrix} w_{1,1} & \cdots & w_{1,p} \\ w_{2,1} & \cdots & w_{2,p} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \cdots & w_{n,p} \end{pmatrix} \in \mathbb{R}^{n \times p},$$

and the reduced coefficient vector is $\gamma \in \mathbb{R}^p$. The least squares estimator for the transformed data is then given by

$$\hat{\gamma} = (W^\top W)^{-1} W^\top z$$

and the ridge regression estimate is given by

$$\hat{\gamma}^{(\lambda)} = (W^\top W + \lambda I)^{-1} W^\top z.$$

To transform back regression estimates obtained for the transformed data, we need to revert the standardisation: we have

$$x_{ij} = \mathrm{s}_{x_j} w_{ij} + \overline{x_j}$$

and

$$y_i = \mathrm{s}_y z_i + \overline{y}.$$

To obtain fitted values for an input $(\tilde{x}_1, \ldots, \tilde{x}_p)$ we have to first transform these inputs: $\tilde{w}_j = (\tilde{x}_j - \overline{x_j})/\mathrm{s}_{x_j}$. Note that the mean and standard deviation come from the data used to fit the model, and are not computed from the $\tilde{x}_i$. We then find the model mean for the transformed data:

$$\begin{aligned} \hat{z} &= \sum_{j=1}^{p} \hat{\gamma}_j \tilde{w}_j \\ &= \sum_{j=1}^{p} \hat{\gamma}_j \frac{\tilde{x}_j - \overline{x_j}}{\mathrm{s}_{x_j}} \\ &= -\sum_{k=1}^{p} \frac{1}{\mathrm{s}_{x_k}} \hat{\gamma}_k \overline{x_k} + \sum_{j=1}^{p} \frac{1}{\mathrm{s}_{x_j}} \hat{\gamma}_j \tilde{x}_j. \end{aligned}$$

Finally, transforming back the response we get

$$\begin{aligned} \hat{y} &= (\overline{y} - \sum_{k=1}^{p} \frac{\mathrm{s}_y}{\mathrm{s}_{x_k}} \hat{\gamma}_k \overline{x_k}) + \sum_{j=1}^{p} \frac{\mathrm{s}_y}{\mathrm{s}_{x_j}} \hat{\gamma}_j \tilde{x}_j \\ &= \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j \tilde{x}_j, \end{aligned}$$

where

$$\hat{\beta}_j = \frac{\mathrm{s}_y}{\mathrm{s}_{x_j}} \hat{\gamma}_j$$

for all $j \in \{1, \ldots, p\}$ and

$$\hat{\beta}_0 = \overline{y} - \sum_{k=1}^{p} \frac{\mathrm{s}_y}{\mathrm{s}_{x_k}} \hat{\gamma}_k \overline{x_k} = \overline{y} - \sum_{k=1}^{p} \hat{\beta}_k \overline{x_k}.$$

This transformation can be used both for least squares regression and ridge regression estimates.

Summary

- Ridge regression is an alternative estimator for the regression coefficients.
- The method is useful when multicollinearity is present.
- Often it is advantageous to standardise the data before performing ridge regression.

13 Model selection

The aim of linear regression is to find a model with the following properties:

1. The model gives good predictions $\hat{y}$.
2. The model is easy to interpret.
3. The modelling assumptions are satisfied.

These three aims can sometimes contradict: models with fewer parameters are often easier to interpret, whereas good predictions sometimes require a large number of parameters. Some trade-off between these constraints must be made. In this section we discuss how to choose the inputs for a model in a systematic way.

13.1 Candidates Models

Here we assume that we are given data with input variables $x_1, \dots, x_p$. The standard model which we have discussed so far adds an intercept to these inputs, so that there are $q = p + 1$ model parameters in total. We can modify this model in various ways:

- When we discussed hypothesis tests, we allowed for the possibility that some inputs do not influence the output. A reasonable approach would be to fit a model with these inputs omitted. Since each input may either be included in or excluded from the model, independent of the others, and similarly the intercept may be included or not, there are 2^{p+1} possible models to consider.
- In the section about Improving the Model Fit we have seen that sometimes it is useful to transform input variables before using them in the model. This can either happen individually, *e.g.* $x_2' = \log x_2$, or collectively, *e.g.* $x^* = x_1 x_2$. The number of models we may obtain in this way is unlimited.

If we want to compare these candidate models in a systematic way, we need to use efficient methods for comparison, since a often a large number of models needs to be considered.

13.2 Misspecified Models

The model is said to be **misspecified**, if the data we are using comes from a different model than the one we use to estimate the coefficients.

13.2.1 Missing Variables

Assume that the data comes from the model

$$Y = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon,$$

(we can include the intercept as $x_1 = 1$ if needed), but we are only using the first q variables $x_1, \dots, x_q$, where $q < p$, to estimate the coefficients. Then the least squares estimate of $\beta = (\beta_1, \dots, \beta_q)$ is given by

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

where $X \in \mathbb{R}^{n \times q}$, and our estimate for the error variance σ^2 is

$$\hat{\sigma}^2 = \frac{1}{n-q} y^\top (I - H) y,$$

where $H = X(X^\top X)^{-1} X^\top$ is the hat matrix computed without using $x_{q+1}, \dots, x_p$.

If we write $\tilde{X} \in \mathbb{R}^{n \times (p-q)}$ for the matrix with columns $x_{q+1}, \dots, x_p$ and $\tilde{\beta} = (\beta_{q+1}, \dots, \beta_p) \in \mathbb{R}^{p-q}$, then the full model can be written as

$$Y = X\beta + \tilde{X}\tilde{\beta} + \varepsilon$$

and we get

$$\begin{aligned}\hat{\beta} &= (X^\top X)^{-1} X^\top Y \\ &= (X^\top X)^{-1} X^\top (X\beta + \tilde{X}\tilde{\beta} + \varepsilon) \\ &= (X^\top X)^{-1} X^\top X\beta + (X^\top X)^{-1} X^\top \tilde{X}\tilde{\beta} + (X^\top X)^{-1} X^\top \varepsilon \\ &= \beta + (X^\top X)^{-1} X^\top \tilde{X}\tilde{\beta} + (X^\top X)^{-1} X^\top \varepsilon.\end{aligned}$$

Thus, we have

$$\mathbb{E}(\hat{\beta}) = \beta + (X^\top X)^{-1} X^\top \tilde{X}\tilde{\beta}$$

and

$$\text{bias}(\hat{\beta}) = (X^\top X)^{-1} X^\top \tilde{X}\tilde{\beta}.$$

This shows that now the estimate is biased in general. There are two special cases when omitting variables does not introduce a bias: The first is when all of the omitted coefficients equal zero, *i.e.* when we have $\tilde{\beta} = 0$. The second case is when the omitted inputs are orthogonal to the included inputs, since then we have $X^\top \tilde{X} = 0$.

Using the formula for $\hat{\beta}$ we find

$$\begin{aligned}\text{Cov}(\hat{\beta}) &= \text{Cov}\Big(\beta + (X^\top X)^{-1} X^\top \tilde{X}\tilde{\beta} + (X^\top X)^{-1} X^\top \varepsilon\Big) \\ &= \text{Cov}\Big((X^\top X)^{-1} X^\top \varepsilon\Big) \\ &= \sigma^2 (X^\top X)^{-1} X^\top I X (X^\top X)^{-1} \\ &= \sigma^2 (X^\top X)^{-1}.\end{aligned}$$

Thus, the variance of $\hat{\beta}$ is not affected by the omitted variables. Similar to our derivation in the section Estimating the Error Variance one can show that

$$\mathbb{E}(\hat{\sigma}^2) = \sigma^2 + \frac{\tilde{\beta}^\top \tilde{X}^\top (I - H)\tilde{X}\tilde{\beta}}{n - q}.$$

Thus, the estimate of the error variance is in general also biased.

This shows that our estimates can become biased if some of the inputs are missing from our model. As in the previous section, it may happen that the MSE of the parameter estimates in the reduced model is smaller than the MSE in the correct model; this is the case if the variance of the estimates decreases enough to compensate for the introduced bias.

13.2.2 Unnecessary Variables

If the data comes from a model with fewer inputs than the ones we are using, the model is not "misspecified" in the strict sense. It is just the case that some of the true β_j exactly equal zero. In this case, our previous results show that the least squares estimate is unbiased.

Including additional variables into a model still can cause problems: One can show that each unnecessary variable added increases the variance of the estimates $\hat{\beta}_j$. Instead of giving a proof of this fact, we illustrate the effect using a numerical example.

Example 13.1. Here we consider the `stackloss` dataset with an added column of noise. We have seen that

$$\text{Var}(\hat{\beta}_j) = \sigma^2 (X^\top X)^{-1}_{jj}.$$

While we don't know the true value of σ^2, we can determine the relative change in variance when adding a column, since this process does not change σ^2 and thus σ^2 will cancel when we determine the relative change in variance.

We first compute the diagonal elements of $(X^\top X)^{-1}$ for the original dataset:

```
X1 <- model.matrix(stack.loss ~ ., data = stackloss)
d1 <- diag(solve(t(X1) %*% X1))
d1
```

```
#  (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
# 13.452726695  0.001728874  0.012875424  0.002322167
```

Now we add a column of noise and re-compute the values:

```
set.seed(20211115)
n <- nrow(X1)
X2 <- cbind(X1, rnorm(n))
d2 <- diag(solve(t(X2) %*% X2))
d2
```

```
#  (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
# 14.397515774  0.001730570  0.015195467  0.002796487  0.064828744
```

Finally, we determine the change in variance in percent:

```
round(100 * (d2[-5] - d1) / d1, 3)
```

```
# (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
#       7.023       0.098      18.019      20.426
```

We see that the variances of the $\hat{\beta}_j$ increased by up to 20% when we added the unnecessary input.

The example illustrates that it is important to keep the model as small as possible.

13.3 Assessing Models

A variety of criteria is used in practice to select variables to include in a regression model.

1. The coefficient of multiple determination, R^2, can be used to compare models. "Good" models have large R^2. If comparisons are made between models with different numbers of inputs then the adjusted R^2-value, R^2_{adj}, must be used.

2. The estimate for the error variance,

$$\hat{\sigma}^2 = \frac{1}{n-q}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

   can be used, where q is the number of columns of the design matrix. (For the full model with an intercept we have $q = p + 1$.) "Good" models have small $\hat{\sigma}^2$. Since

$$\begin{aligned} R^2_{\text{adj}} &= 1 - \frac{\frac{1}{n-q}\mathrm{s}^2_{\hat{\varepsilon}}}{\frac{1}{n-1}\mathrm{s}^2_y} \\ &= 1 - \frac{\frac{n-1}{n-q}\mathrm{s}^2_{\hat{\varepsilon}}}{\mathrm{s}^2_y} \\ &= 1 - \frac{\frac{1}{n-q}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\mathrm{s}^2_y} \\ &= 1 - \frac{\hat{\sigma}^2}{\mathrm{s}^2_y}, \end{aligned}$$

   where $\mathrm{s}^2_{\hat{\varepsilon}}$ and s^2_y are the sample variances as before, minimising $\hat{\sigma}^2$ is equivalent to maximising R^2_{adj}. Thus, this criterion is equivalent to the first one.

3. The **Prediction Error Sum of Squares (PRESS)** compares each output y_i to the fitted value $\hat{y}_i^{(i)}$ for observation i, computed without using sample i:

$$\text{PRESS} := \sum_{i=1}^{n} (y_i - \hat{y}_i^{(i)})^2.$$

In the comparison, $\hat{y}_i^{(i)}$ is used instead of $\hat{y}_i$ so that in models with too many parameters, overfitting does not result in overoptimistic estimates. "Good" models have small PRESS. The following lemma helps to compute PRESS without having to fit n separate models.

Lemma 13.1. *We have*

$$\text{PRESS} = \sum_{i=1}^{n} \Big(\frac{\hat{\varepsilon}_i}{1 - h_{ii}} \Big)^2,$$

where h_{ii} denotes the diagonal elements of the hat matrix H.

Proof. From equation (30) we know how the estimated regression coefficients change, when observation i is deleted from the model:

$$\hat{\beta}^{(i)} - \hat{\beta} = -(X^\top X)^{-1} x_i \frac{\hat{\varepsilon}_i}{1 - h_{ii}}.$$

Thus we have

$$\begin{aligned}
\hat{y}_i^{(i)} - \hat{y}_i &= \big(X\hat{\beta}^{(i)} - X\hat{\beta}\big)_i \\
&= -\big(X(X^\top X)^{-1} x_i \frac{\hat{\varepsilon}_i}{1 - h_{ii}}\big)_i \\
&= -x_i^\top (X^\top X)^{-1} x_i \frac{\hat{\varepsilon}_i}{1 - h_{ii}} \\
&= -\frac{h_{ii}}{1 - h_{ii}} \hat{\varepsilon}_i.
\end{aligned}$$

From this we get

$$\begin{aligned}
y_i - \hat{y}_i^{(i)} &= y_i - \hat{y}_i + \hat{y}_i - \hat{y}_i^{(i)} \\
&= \hat{\varepsilon}_i + \frac{h_{ii}}{1 - h_{ii}} \hat{\varepsilon}_i \\
&= \frac{1}{1 - h_{ii}} \hat{\varepsilon}_i.
\end{aligned}$$

Substituting this expression into the definition of PRESS completes the proof. □

4. **Akaike's Information Criterion (AIC)** is defined as

$$\text{AIC} = 2q - 2\log(\hat{L}),$$

where q is the number of parameters in the model, and $\hat{L}$ is the maximum of the likelihood function when these q parameters are chosen optimally. "Good" models have small AIC.

Since we assume $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, i.i.d. and we have $\varepsilon_i = y_i - x_i^\top \beta$, the likelihood function for our model is

$$\begin{aligned}
L(\beta, \sigma^2; X, y) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(y_i - (X\beta)_i)^2}{2\sigma^2} \Big) \\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\Big(-\sum_{i=1}^{n} \frac{(y_i - (X\beta)_i)^2}{2\sigma^2} \Big).
\end{aligned}$$

Taking logarithms we get

$$\begin{aligned}\log L(\beta, \sigma^2; X, y) &= -\frac{n}{2}\log(2\pi\sigma^2) - \sum_{i=1}^{n}\frac{(y_i - (X\beta)_i)^2}{2\sigma^2} \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}r(\beta),\end{aligned}$$

where $r(\beta)$ is the residual sum of squares, as introduced in the section about Least Squares Estimates. To compute the AIC we need to maximise this expression. The values of β and σ^2 where the maximum is taken are called the **maximum likelihood estimate (MLE)** for β and σ^2. From lemma 2.1 we know that for fixed σ^2, L takes its maximum when β equals the least squares estimate $\hat{\beta}$. Taking derivatives we can then find the optimal value of σ^2. The result is

$$\hat{\sigma}^2_{\text{MLE}} = \frac{1}{n}r(\hat{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (X\hat{\beta})_i)^2 = \frac{n-q}{n}\hat{\sigma}^2.$$

Thus we get

$$\begin{aligned}\text{AIC} &= 2q + n\log(2\pi\hat{\sigma}^2_{\text{MLE}}) + \frac{1}{\hat{\sigma}^2_{\text{MLE}}}r(\hat{\beta}) \\ &= 2q + n\log(2\pi r(\hat{\beta})/n) + n \\ &= 2q + n\log(\|\hat{\varepsilon}\|^2) + n + n\log(2\pi/n).\end{aligned}$$

Summary

- We have discussed how different models can be considered by omitting existing variables or by adding non-linear functions of the inputs as new variables.
- We have seen that removing too many variables can introduce a bias.
- We have seen that adding too many variables can increase the variance of the estimate.
- We have considered different criteria for choosing the "best" model.

14 Automatic Model Selection

For a model with p input variables, there are 2^p different choices for which variables to include in the model (or 2^{p+1}, if we also have to decide whether to include the intercept). This value increases quickly with p: for $p = 10$ we have 1024 models to consider, for $p = 20$ there are 1048576 possible models, and for large p it becomes infeasible to simply try all possible models to find the best one. In this section we consider algorithms for automatically finding a good selection inputs when p is large.

14.1 Exhaustive Search

Efficient algorithms are available to find the best model out of the 2^p possible models for small to moderate p. Here we present a method which can often find the model with the largest R^2_{adj} without having to fit all 2^p models. We consider the least squares estimate throughout.

We can characterise each possible model by the set $J \subseteq \{1, \dots p\}$ of variables included in the model. The model includes variable x_j, if and only if $j \in J$. Here we assume that the intercept is always included, so that $J = \emptyset$ corresponds to the model $Y = \beta_0 + \varepsilon$.

The method described in this section is based on the following three observations:

- In order to *maximise* R^2_{adj}, we can equivalently also find the J which *minimises*

$$\hat{\sigma}^2(J) := \frac{1}{n - |J| - 1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

  where $|J|$ is the number of variables in J and the $\hat{y}_i$ are the fitted values for the model with inputs $j \in J$. We have seen that these criteria are equivalent in section 13.3.

- For $J \subseteq \{1, \dots p\}$, define

$$r(J) := \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

  where $\hat{y}_i$ are the fitted values for the model corresponding to J. This gives the residual sum of squares for each model. The we have

$$\begin{aligned} \min_J \hat{\sigma}^2(J) &= \min_J \frac{1}{n - |J| - 1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \min_{q \in \{0,1,\dots,p\}} \min_{J, |J|=q} \frac{1}{n - q - 1} r(J) \\ &= \min_{q \in \{0,1,\dots,p\}} \frac{1}{n - q - 1} \min_{J, |J|=q} r(J). \end{aligned}$$

  This means that we can first minimise the residual sum of squares for each fixed number q of inputs, and then find the q which gives the best $\hat{\sigma}^2$ in a second step.

- Adding a variables never increases the residual sum of squares. Thus we have $r(J) \leq r(K)$ whenever $J \supseteq K$. We can use this result to exclude certain models without having to fit them.

14.2 Search Algorithm

To find the model with optimal adjusted R-squared value, we perform the following steps. The algorithm is based on the ideas explained in the previous section.

A. Let φ_j denote the residual sum of squares for the model containing all variables except x_j:

$$\varphi_j := r(\{1, \dots, p\} \quad \{j\}).$$

Suppose that the x_j are ordered so that

$$\varphi_1 \geq \varphi_2 \geq \cdots \geq \varphi_p.$$

Any model J with $j \notin J$ has $r(J) \geq \varphi_j$.

B. Compute $\min_{J,|J|=0} r(J) = r(\emptyset)$. This is the residual sum of squares of the model which consists only of the intercept.

C. For each $q := 1, \ldots, p-2$:

1. Let
$$r := r(\{1, \ldots, q\}).$$
This is the only model with q inputs where the first excluded variable has index $k = q+1$. If $r \leq \varphi_{k-1} = \varphi_q$, we know from step A that $J = \{1, \ldots, q\}$ is the best model with q inputs, since any other model will exclude one of the x_j with $j \leq q$. In this case we have found
$$\min_{J,|J|=q} r(J) = r.$$
and we continue step B by trying the next value of q. If $r > \varphi_{k-1}$, no decision can be taken at this point and we continue with step 2.

2. For $j \in \{q+1, \ldots, p\}$ let
$$r_j := r(\{1, \ldots, q-1\} \cup \{j\}).$$
These are all models with q inputs where the first excluded variable has index $k = q$. If $\min(r, r_1, \ldots, r_q) \leq \varphi_{k-1} = \varphi_{q-1}$, then we know from step A that the J corresponing to the minimum is the best model with q inputs. In this case we continue step B by trying the next value of q. Otherwise we proceed to step 3.

3. Similarly to the previous step, we consider all models with q variables where the first excluded variable has index $k = q-1$. If the best RSS amonst these and the previously considered models is less than or equal to φ_{k-1} we are done and consider the next q. Otherwise we decrease k until we reach $k = 1$. At this point we have considered all models with q variables and have found $\min_{J,|J|=q} r(J)$.

D. Compute $\min_{J,|J|=p-1} r(J) = \min_{j \in \{1,\ldots,p\}} \varphi_j$.

E. Compute $\min_{J,|J|=p} r(J) = r(\{1, \ldots, p\})$.

F. Find the $q \in \{0, \ldots, p\}$ for which $\min_{J,|J|=q} r(J)/(n-q-1)$ is smallest. Output the model which has minimal RSS for this q.

This algorithm finds the model with the maximal R^2_{adj}. Often, large savings are achieved by the early exits in step C. Only in the worst case, when all of the comparisons with φ_j in step C fail, this algorithms needs to fit all 2^p models.

Example 14.1. To demonstrate the steps of the algorithm, we implement the method "by hand". We use the QSAR dataset, which we have already seen in section 7:

```
# data from https://archive.ics.uci.edu/ml/datasets/QSAR+aquatic+toxicity
qsar <- read.csv("data/qsar_aquatic_toxicity.csv",
                 sep = ";", header = FALSE)
fields <- c(
    "TPSA",
    "SAacc",
    "H050",
```

```
    "MLOGP",
    "RDCHI",
    "GATS1p",
    "nN",
    "C040",
    "LC50"
)
names(qsar) <- fields
```

To make it easy to add/remove columns automatically, we first construct the design matrix, remove columns as needed, and then use the resulting matrix in the call to `lm()` (instead of specifying the terms to include by name).

```
X <- model.matrix(LC50 ~ ., data = qsar)
X <- X[, -1] # remove the intercept, since lm() will re-add it later
n <- nrow(X)
p <- ncol(X)

y <- qsar$LC50

m <- lm(y ~ X) # full model
summary(m)
```

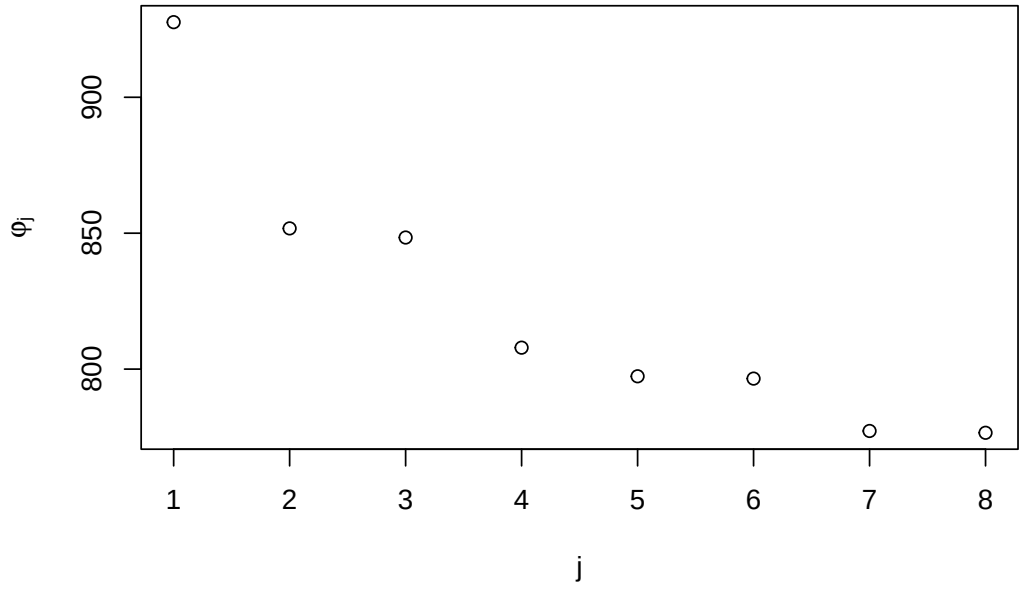
```
#
# Call:
# lm(formula = y ~ X)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -4.4934 -0.7579 -0.1120  0.5829  4.9778
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept)  2.698887   0.244554  11.036  < 2e-16 ***
# XTPSA        0.027201   0.002661  10.220  < 2e-16 ***
# XSAacc      -0.015081   0.002091  -7.211 1.90e-12 ***
# XH050        0.040619   0.059787   0.679 0.497186
# XMLOGP       0.446108   0.063296   7.048 5.60e-12 ***
# XRDCHI       0.513928   0.135565   3.791 0.000167 ***
# XGATS1p     -0.571313   0.153882  -3.713 0.000227 ***
# XnN         -0.224751   0.048301  -4.653 4.12e-06 ***
# XC040        0.003194   0.077972   0.041 0.967340
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 1.203 on 537 degrees of freedom
# Multiple R-squared:  0.4861,  Adjusted R-squared:  0.4785
# F-statistic:  63.5 on 8 and 537 DF,  p-value: < 2.2e-16
```

Comparing the output to what we have seen in section 7 shows that the new method of calling `lm()` gives the same results as before. Now we follow the steps of the algorithm.

A. The values $\varphi_1, \ldots, \varphi_p$ can be computed as follows:

```
phi <- numeric(p) # pre-allocate an empty vector
for (j in 1:p) {
    idx <- (1:p)[-j] # all variables except x[j]
```

```
    m <- lm(y ~ X[,idx])
    phi[j] <- sum(resid(m)^2)
}

# change the order of columns in X, so that the phi_j are decreasing
jj <- order(phi, decreasing = TRUE)
X <- X[, jj]
phi <- phi[jj]
plot(phi, xlab = "j", ylab = expression(varphi[j]))
```



The plot shows the residual sum of squares for the model with input x_j omitted, for j ranging from 1 to 8.

B. Next, we compute the residual sum of squares of the model which consists only of the intercept. This is the case $q = 0$.

```
all.q <- 0:p
best.rss <- numeric(p + 1) # For storing the best RSS for q = 0, ..., p,
best.model <- vector("list", p + 1) # and the corresponding models.

m <- lm(y ~ 1)
best.rss[0 + 1] <- sum(resid(m)^2)
best.model[[0 + 1]] <- integer(0) # a vector of length 0 (no columns)

count <- 1 # number of models fitted so far
```

C. Now we consider $q \in \{1, \ldots, p-2\}$. The algorithm groups these cases by the position k of the first column omitted in the model, starting with $k = q+1$ and then decreasing k in each step. We use the function `combn` from the `sets` package to get all possible choices of $q-k+1$ columns out of $\{k+1, \ldots, p\}$.

```
library(sets) # this defines "combn()"
for (q in 1:(p-2)) {
    best.rss[q + 1] <- Inf

    # Consider all sets of q columns, ...
    for (k in (q+1):1) {
        # ... where the first omitted column is k.
```

```
        # We have to include 1, ..., k-1, and ...
        a <- seq_len(k-1)

        # ... for the remaining q - (k-1) inputs, we try all
        # possible combinations.
        bb <- combn((k+1):p, q - k + 1)
        for (l in seq_len(ncol(bb))) {
            b <- bb[, l]
            included <- c(a, b)

            m <- lm(y ~ X[, included])
            count <- count + 1
            rss <- sum(resid(m)^2)

            if (rss < best.rss[q + 1]) {
                best.rss[q + 1] <- rss
                best.model[[q + 1]] <- included
            }
        }

        if (k > 1 && best.rss[q] <= phi[k-1]) {
            # If we reach this point, we know that we found the best
            # arrangement and we can exit the loop over k early.
            # This is what makes the algorithm efficient.
            break
        }
    }
}
```

D. We already fitted all models with $p-1$ inputs, when we computed `phi`. Since we sorted the models, the best of these models is last in the list. This covers the case $q = p-1$.

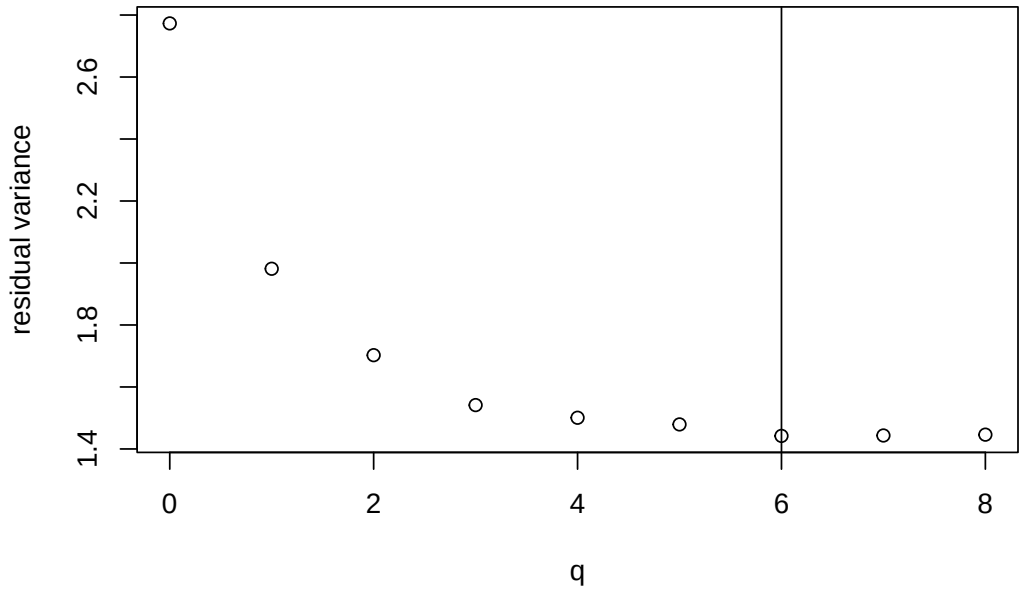
```
best.rss[(p - 1) + 1] <- min(phi)
omitted <- jj[length(jj)]
best.model[[(p - 1) + 1]] <- (1:p)[-omitted]
count <- count + length(phi)
```

E. Finally, for $q = p$ we fit the full model:

```
m <- lm(y ~ X)
best.rss[p + 1] <- sum(resid(m)^2)
best.model[[p + 1]] <- 1:p
count <- count + 1
```

F. Now we can find the model with the best R^2_{adj}:

```
plot(all.q, best.rss / (n - all.q - 1),
     xlab = "q", ylab = "residual variance")
best.q <- all.q[which.min(best.rss / (n - all.q - 1))]
abline(v = best.q)
```



We see that the model with $q = 6$ variables has the the lowest $\hat{\sigma}^2$ and thus the highest R^2_{adj}. The values for $q = 7$ and $q = 8$ are very close to optimal. The best model uses the following variables:

```
colnames(X)[best.model[[6 + 1]]]
```

```
# [1] "TPSA"   "SAacc"  "MLOGP"  "nN"     "RDCHI"  "GATS1p"
```

We can fit the optimal model from the original data:

```
m.best <- lm(LC50 ~ TPSA + SAacc + MLOGP + nN + RDCHI + GATS1p,
             data = qsar)
summary(m.best)
```

```
#
# Call:
# lm(formula = LC50 ~ TPSA + SAacc + MLOGP + nN + RDCHI + GATS1p,
#     data = qsar)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -4.4986 -0.7668 -0.1165  0.5529  4.9758
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  2.758246   0.228483  12.072  < 2e-16 ***
# TPSA         0.026858   0.002608  10.300  < 2e-16 ***
# SAacc       -0.014267   0.001660  -8.596  < 2e-16 ***
# MLOGP        0.434578   0.060611   7.170 2.49e-12 ***
# nN          -0.218445   0.047101  -4.638 4.43e-06 ***
# RDCHI        0.514758   0.133430   3.858 0.000128 ***
# GATS1p      -0.602971   0.146920  -4.104 4.69e-05 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 1.201 on 539 degrees of freedom
# Multiple R-squared:  0.4857,  Adjusted R-squared:  0.4799
# F-statistic: 84.82 on 6 and 539 DF,  p-value: < 2.2e-16
```

This shows that R^2_{adj} has indeed marginally improved, from 0.4785 to 0.4799

To conclude, we check that the complicated algorithm indeed saved some work:

```
total <- 2^p
cat(count,
    " models fitted (", round(100 * count / total, 1), "%)\n",
    sep = "")
```

```
# 88 models fitted (34.4%)
```

This shows that we only needed to fit 88 of the 256 models under consideration.

Example 14.2. We can perform the analysis from the previous example automatically, using the function `regsubsets()` from the `leaps` package:

```
library(leaps)
m <- regsubsets(LC50 ~ ., data = qsar,
                method = "exhaustive")
summary(m)
```

```
# Subset selection object
# Call: regsubsets.formula(LC50 ~ ., data = qsar, method = "exhaustive")
# 8 Variables  (and intercept)
#        Forced in Forced out
# TPSA       FALSE      FALSE
# SAacc      FALSE      FALSE
# H050       FALSE      FALSE
# MLOGP      FALSE      FALSE
# RDCHI      FALSE      FALSE
# GATS1p     FALSE      FALSE
# nN         FALSE      FALSE
# C040       FALSE      FALSE
# 1 subsets of each size up to 8
# Selection Algorithm: exhaustive
#          TPSA SAacc H050 MLOGP RDCHI GATS1p nN  C040
# 1  ( 1 ) " "  " "   " "  "*"   " "  " "    " " " "
# 2  ( 1 ) "*"  " "   " "  "*"   " "  " "    " " " "
# 3  ( 1 ) "*"  "*"   " "  "*"   " "  " "    " " " "
# 4  ( 1 ) "*"  "*"   " "  "*"   " "  " "    "*" " "
# 5  ( 1 ) "*"  "*"   " "  "*"   " "  "*"    "*" " "
# 6  ( 1 ) "*"  "*"   " "  "*"   "*"  "*"    "*" " "
# 7  ( 1 ) "*"  "*"   "*"  "*"   "*"  "*"    "*" " "
# 8  ( 1 ) "*"  "*"   "*"  "*"   "*"  "*"    "*" "*"
```

This shows the best model of each size. The only step left is to decide which q to use. This choice depends on the cost-complexity tradeoff. Here we consider R^2_{adj} again:

```
round(summary(m)$adjr2, 4)
```

```
# [1] 0.2855 0.3860 0.4441 0.4588 0.4666 0.4799 0.4795 0.4785
```

This gives the R^2_{adj} for the optimal model of each size again. At the end of the list we recognise the values `0.4799` and `0.4785` from our previous analysis.

14.3 Other Methods

There are other, approximate methods available which can be used when the number p of model parameters is too large for an exhaustives search.

14.3.1 Stepwise Forward Selection

Here the idea is to start with a minimal model, and then add variables one by one, until no further (large enough) improvements can be achieved.

Start with only intercept term: $y = \beta_0 + \varepsilon$, then consider each of p models:

$$y = \beta_0 + \beta_j x_j + \varepsilon,$$

for $j \in \{1, \dots, p\}$.

Choose the model with the smallest residual sum of squares, provided that the "significance of the fitted model" achieves a specified threshold. The process continues by adding more variables, one at a time, until either

- All variables are in the model.
- The significance level can not be achieved by any variable not in the model.

The "significance of the model" can be examined by considering a t-test as in lemma 5.4.

14.3.2 Stepwise Backward Selection

Here the idea is to start with the full model, and to remove variables one by one until no good enough candidates for removal are left.

In each step we consider the test statistic $|T_j|$ for the tests

$$H_0 : \beta_j = 0 \quad vs. \quad H_1 : \beta_j \neq 0$$

for $j \in \{1, \dots, p\}$, again as in lemma 5.4. The method selects x_j corresponding to the smallest $|T_j|$. If this is below a given threshold, then remove x_j and re-fit the model. Repeat until either:

- No variables are left in the model.
- The smallest $|T_j|$ is above the threshold.

14.3.3 Hybrid Methods

Either start with a full model or just intercept. At each stage consider:

- removing least significant variable already in the model,
- adding most significant variable not currently in the model,

with significance levels set to avoid a cyclical behaviour.

Summary

- We discussed different algorithms to perform model selection in practice.

Problem Sheet 4

You should attempt all these questions and write up your solutions in advance of the workshop in week 8 where the answers will be discussed.

13. Consider the dataset `pressure`, built into R, which describes the vapour pressure P of mercury as a function of temperature T.

a. For given T_0, explain how a model of the form $P = \beta_0 + T\beta_1 + \max(0, T - T_0)\beta_2 + \varepsilon$ can be fitted to the data.

Solution:

We can fit a model with an intercept and two inputs, T and $\max(0, T - T_0)$. In R we can use the following commands to fit such a model:

```
T0 <- the.given.value # substitute the value of T0 here
m <- lm(pressure ~ temperature + I(pmax(0, temperature-T0)),
        data = pressure)
```

b. Why might a model of this form make sense for the given data?

Solution: Such a model may make sense, because the slope of the samples increases faster than linear as the temperature T increases. The term $\max(0, T - T_0)$ is zero for small T, but adds an additional slope once T exceeds the cutoff temperature T_0.

c. What is a good value of T_0 for this dataset?

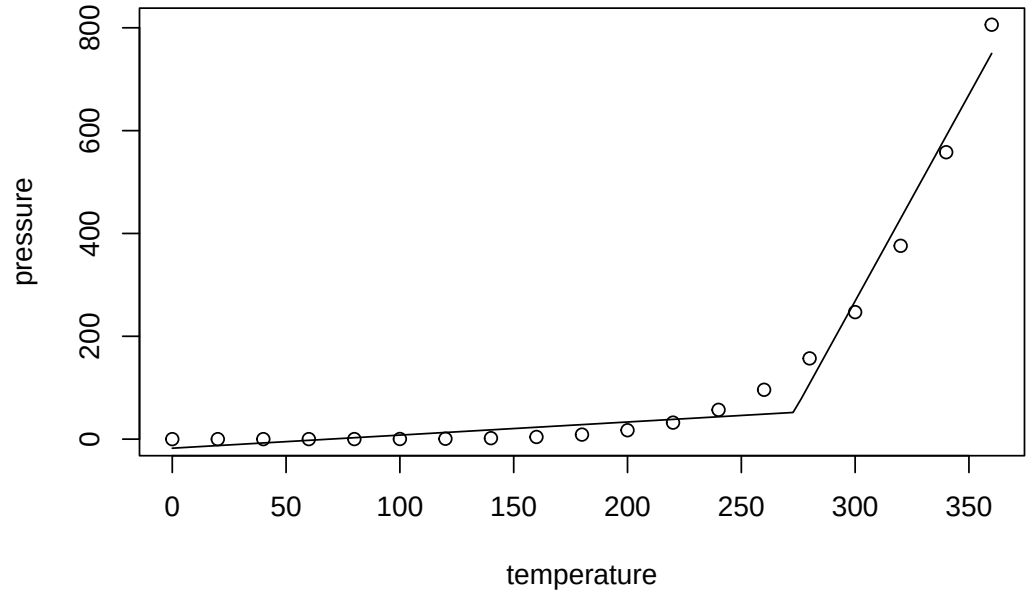
Solution: T_0 should be placed at the end of the "flat" part of the plot, where the pressure starts increasing more quickly. A quick experiment shows that the best R^2 value is achieved for $T_0 \approx 273$:

```
T0 <- 273
m <- lm(pressure ~ temperature + I(pmax(0, temperature - T0)),
        data=pressure)
summary(m)
```

```
#
# Call:
# lm(formula = pressure ~ temperature + I(pmax(0, temperature -
#     T0)), data = pressure)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -53.125 -17.553  -6.308  13.017  55.839
#
# Coefficients:
#                              Estimate Std. Error t value Pr(>|t|)
# (Intercept)                 -17.62933   14.63735  -1.204   0.2459
# temperature                   0.25472    0.08774   2.903   0.0104 *
# I(pmax(0, temperature - T0))  7.77118    0.38024  20.437 6.85e-13 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 29.86 on 16 degrees of freedom
# Multiple R-squared:  0.9843,  Adjusted R-squared:  0.9823
# F-statistic: 501.3 on 2 and 16 DF,  p-value: 3.705e-15
```

To get a better understanding of how the extra term $\max(0, T - T_0)$ improves model fit, we plot the data together with the fitted model:

```
plot(pressure)
T.plot <- seq(0, 360, l=100)
p.plot <- predict(m, data.frame(temperature=T.plot))
lines(T.plot, p.plot)
```



As expected, for $T > 273 = T_0$, the slope of the fitted line increases, leading to greatly improved model fit. (An alternative, and probably superior approach would be to transform the data and try to fit a straight line to the transformed data instead.)

14. Where possible, rewrite the following regression models in linear form.

a. $y = \beta_0 \beta_1^x$

Solution: We can take logarithms:

$$\log(y) = \log(\beta_0) + x \log(\beta_1) =: \beta_0' + x\beta_1'.$$

Notice that writing $\log(y) = \beta_0' + x\beta_1' + \varepsilon$ implies $y = \beta_0 \beta_1^x \exp(\varepsilon)$.

b. $y = \beta_0 + \beta_1 \sin(x_1) + \beta_2 \exp(x_2)$

Solution: We can simply set $x_1' = \sin(x_1)$ and $x_2' = \exp(x_2)$. This transformation does not affect the error term.

c. $y = \frac{x}{\beta_0 + \beta_1 x}$

Solution: We can write

$$\frac{1}{y} = \frac{\beta_0 + \beta_1 x}{x} = \beta_0 \frac{1}{x} + \beta_1.$$

Thus we can use $x' = 1/x$, $y' = 1/y$, $\beta_0' = \beta_1$ and $\beta_1' = \beta_0$. This transformation affects the noise, too, so some care is needed.

15. Consider the dataset from

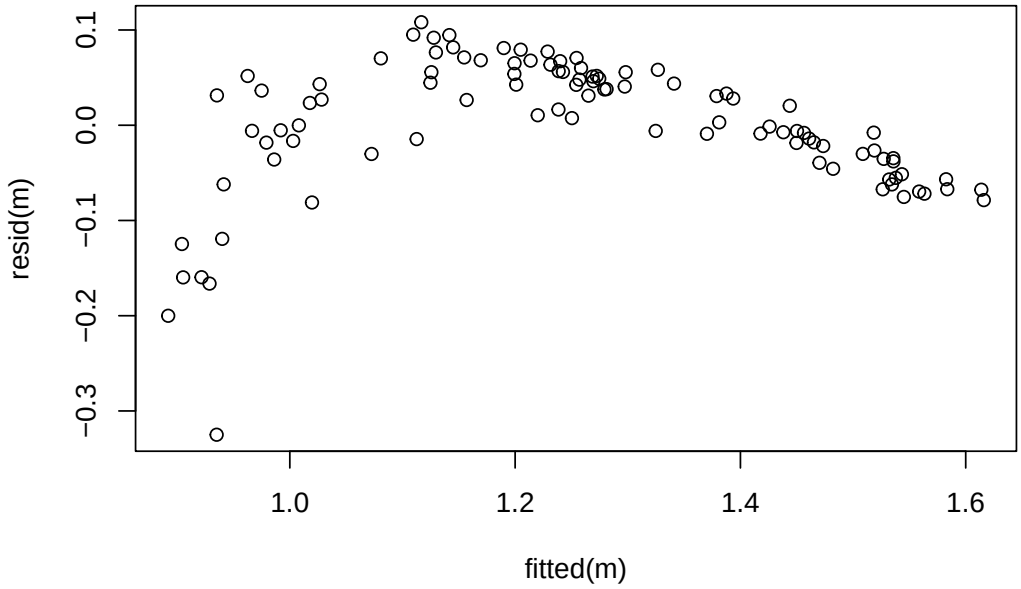
- https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/P04Q14.csv

The dataset contains samples $(x_{i,1}, x_{i,2}, y_i)$. We want to find a model for y as a function of x_1 and x_2.

a. Fit a linear model to the data and produce a residual plot. Based on the plot, discuss how well the model fits the data.

Solution: We can fit the model as follows:

```
# data from https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/P04Q14.csv
d <- read.csv("data/P04Q14.csv")
m <- lm(y ~ x1 + x2, data = d)
plot(fitted(m), resid(m))
```

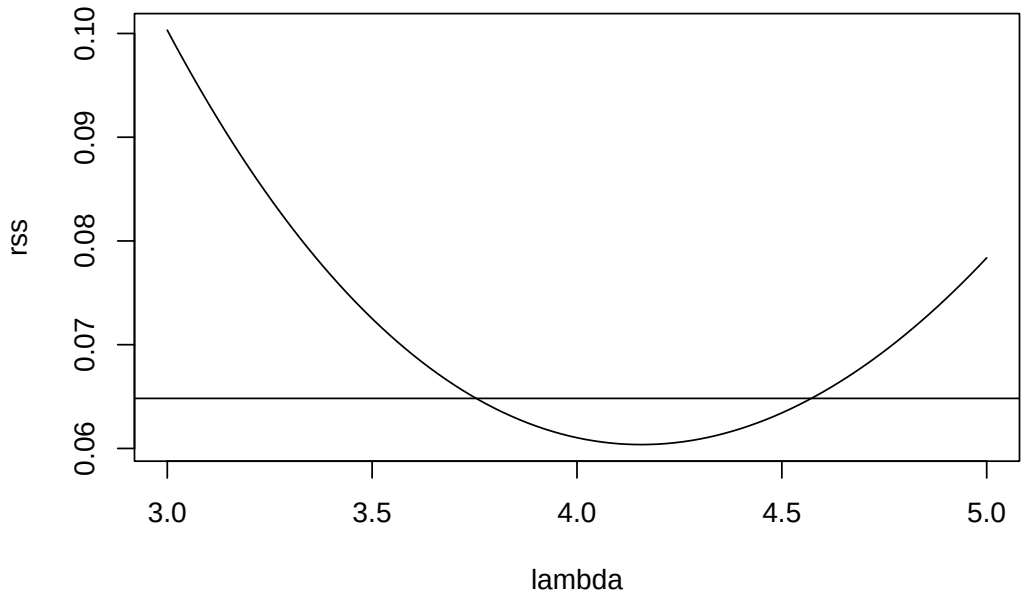


The mean of the residuals does not form a straight line, so the dependency of y on the x_i seems to be nonlinear. Also, the variance of the residuals decreases as $\hat{y}$ increases. Model fit is not good and a transformation of the data may be useful.

b. Apply a Power Transform to the model, as described in section 11.3 of the notes. Explain your choice of λ.

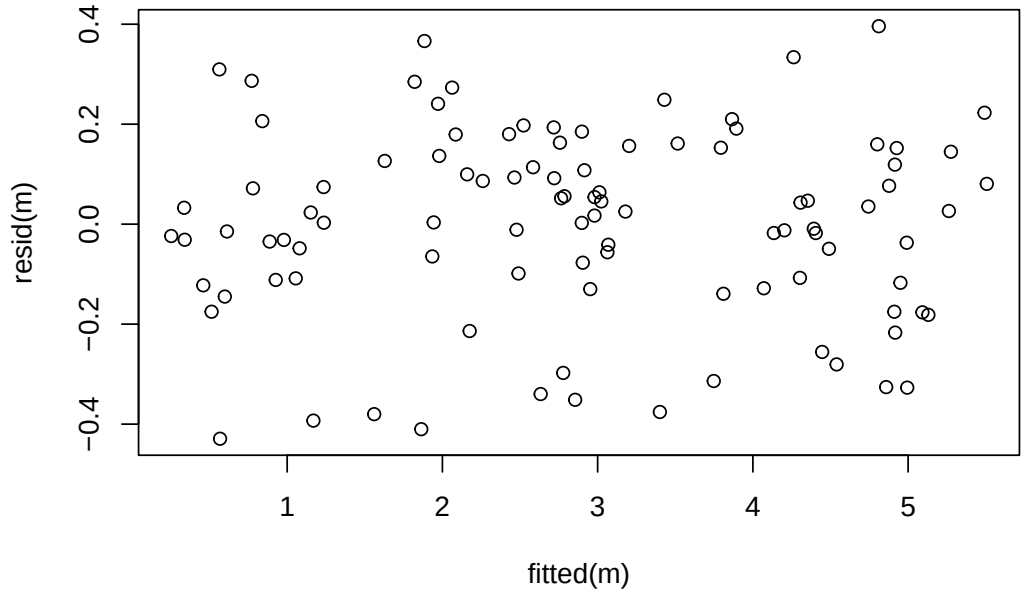
Solution: To choose the exponent λ, we plot the residual sum of squares as a function of λ.

```
gm <- exp(mean(log(d$y)))
lambda <- seq(3, 5, length.out = 101)
rss <- numeric(length(lambda))
for (i in seq_along(lambda)) {
    li <- lambda[i]
    y.prime <- (d$y^li - 1) / (li * gm^(li-1))
    mi <- lm(y.prime ~ x1 + x2, data = d)
    rss[i] <- sum(resid(mi)^2)
}
plot(lambda, rss, type="l")

cutoff <- min(rss) * (1 + qt(0.971, 51)^2 / 51)
abline(h = cutoff)
```



The horizonal line in this plot indicates the cutoff suggested by the rule of thumb in (31). Of the values of λ where `rss` is below the cutoff, $\lambda = 4$ seems the most "simple", so we try this value here.

```
y.prime <- d$y ^ 4
m <- lm(y.prime ~ x1 + x2, data = d)
plot(fitted(m), resid(m))
```



Now the residual plot looks perfect!

16. The Prediction Error Sum of Squares (PRESS) is defined as

$$\text{PRESS} = \sum_{i=1}^{n} (y_i - \hat{y}_i^{(i)})^2.$$

Using lemma 13.1, or otherwise, determine the PRESS value for the `stackloss` dataset built into R (using `stack.loss` as the response variable).

Solution:

We can use the formula

$$\text{PRESS} = \sum_{i=1}^{n} \left(\frac{\widehat{\varepsilon}_i}{1 - h_{ii}} \right)^2$$

from the lemma. Implementation in R is straightforward.

```
m <- lm(stack.loss ~ ., data = stackloss)
X <- model.matrix(m)
H <- X %*% solve(t(X) %*% X, t(X))
hii <- diag(H)
sum((resid(m)/(1 - hii))^2)
```

```
# [1] 291.8689
```

15 Factors

In this section we will learn how to fit linear models when some or all of the inputs are categorical. Such inputs are sometimes called "factors". These can be binary (*e.g.* sex recorded as male/female) or ordered (*e.g.* quality recorded as poor/medium/good) or unordered (*e.g.* marital status recorded as single/married/divorced/widowed).

15.1 Indicator Variables

Our aim here is to include categorical input variables within the usual regression model

$$y = X\beta + \varepsilon.$$

Assume that $x_j \in \{a_1, \dots, a_k\}$ is a factor which has k possible values. The possible values of the factor, $a_1, \dots, a_k$ are also called the **levels** of the factor. The easiest way to include a factor in a regression model is via **indicator variables**. For this we represent x_j using $k-1$ columns in the design matrix, say $\tilde{x}_1, \dots, \tilde{x}_{k-1}$, where

$$\tilde{x}_{\ell i} = \begin{cases} 1 & \text{if } x_{ji} = a_\ell \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

for all $\ell \in \{1, \dots, k-1\}$ and $i \in \{1, \dots, n\}$. Using this convention, $x_{ji} = k$ is represented by setting $\tilde{x}_{1,i} = \cdots = \tilde{x}_{k-1,i} = 0$. The value a_k which is not represented as a separate column is called the **reference level**. The regression coefficients corresponding to the columns $\tilde{x}_1, \dots, \tilde{x}_{k-1}$ can be interpreted as the effect of level a_ℓ, relative to the effect of level a_k. Since the indicator is constant 1 for all samples corresponding to a given factor, regression coefficients representing factors lead to changes of the intercept, rather than of slopes.

We are using only $k-1$ columns to represent a factor with k levels in order to avoid collinearity. If we would use k columns, defined as above, we would get

$$\sum_{\ell=1}^{k} \tilde{x}_{\ell i} = \mathbf{1},$$

and in a model with an intercept we would have exact collinearity of the corresponding columns in the design matrix X.

Example 15.1. Consider a dataset consisting of a numerical input variable and a binary factor for gender (`female`, `male`). Assume we are given data $(1, \text{male})$, $(2, \text{female})$, $(3, \text{female})$. Then we can encode this data using the following design matrix:

intercept	x_1	$1_{\{\text{female}\}}(x_2)$
1	1	0
1	2	1
1	3	1

In R there is a dedicated type `factor` for categorical variables. If factors appear in the input of linear regression problem, the function `lm()` will automatically create the indicator variables for us.

Example 15.2. Here we simulated data which could describe the effect of some medical treatment. In the simulated data we have two groups, one which has been "treated" and a control group. There is a "value" which decreases with age, and is lower for the group which has been treated (the intercept is 10 instead of 12).

```
age1 <- runif(25, 18, 40)
group1 <- data.frame(
    value=10 - 0.2 * age1 + rnorm(25, sd=0.5),
    age=age1,
    treated="yes")
age2 <- runif(25, 18, 80)
group2 <- data.frame(
    value=12 - 0.2 * age2 + rnorm(25, sd=0.5),
    age=age2,
    treated="no")
data <- rbind(group1, group2)

data$treated <- factor(data$treated, levels = c("no", "yes"))
```

The last line of the code tells R explicitely that the `treated` column represents a factor with levels "no" and "yes". Internally, this column will now be represented by numbers 1 (for "no") and 2 (for "yes"), but this numeric representation is mostly hidden from the user. Now we fit a regression model:

```
m <- lm(value ~ ., data = data)
summary(m)
```

```
#
# Call:
# lm(formula = value ~ ., data = data)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -1.76041 -0.22188  0.04535  0.23802  1.29250
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept) 11.525306   0.282369   40.82  < 2e-16 ***
# age         -0.189367   0.005557  -34.08  < 2e-16 ***
# treatedyes  -1.845246   0.180890  -10.20 1.68e-13 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 0.5452 on 47 degrees of freedom
# Multiple R-squared:  0.9635,  Adjusted R-squared:  0.962
# F-statistic: 620.5 on 2 and 47 DF,  p-value: < 2.2e-16
```

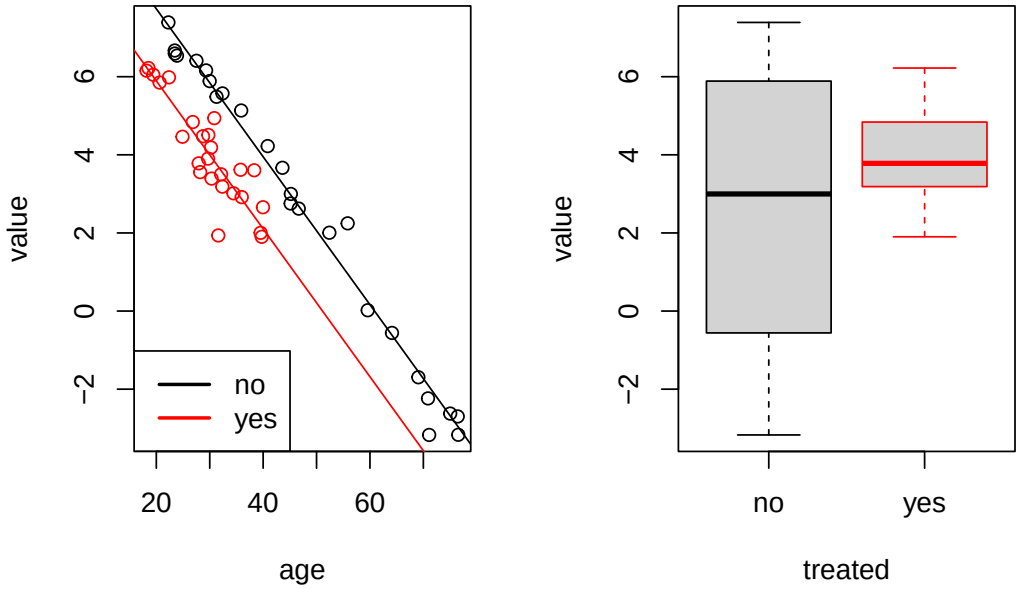
We see that R used "no" as the reference factor, here. The regression coefficient `treatedyes` gives the relative change of the "yes", compared to "no". The true value is the difference of the means: $10 - 12 = -2$, and the estimated value `-1.89` is reasonably close to this. The fitted values for this model are

$$\hat{y} = \begin{cases} \beta_0 + \beta_1 x_1 & \text{if not treated, and} \\ (\beta_0 + \beta_2) + \beta_1 x_1 & \text{if treated.} \end{cases}$$

To get a better idea of the model fit, we plot the data together with separate regression lines for "yes" (red) and "no" (black):

```
par(mfrow = c(1, 2))
plot(value ~ age, data = data,
     col = ifelse(treated == "yes", "red", "black"))
```

```r
# regression line for the reference level `treatment == "no"`
abline(a = coef(m)[1], b = coef(m)[2])

# regression line for the level `treatment = "yes"`:
# here we need to add the coefficient for `treatedyes`
# to the intercept
abline(a = coef(m)[1] + coef(m)[3], b = coef(m)[2], col="red")

legend("bottomleft", c("no", "yes"),
       col = c("black", "red"), lwd = 2)

# also show a boxplot for comparison
boxplot(value ~ treated, data = data,
        border = c("black", "red"))
```



We can see that the two regression lines, corresponding to the two levels are parallel. They have the same slope but different intercepts.

The last command shows a boxplot of `value` for comparison. The boxplot does not allow to conclude that the treatment had an effect, whereas the linear model, which accounts for age, shows the effect of the treatment as a difference in intercepts. The *** in the `treatedyes` row of the `summary(m)` output shows that the difference in intercepts is statistically significant.

15.2 Interactions

In some situations, the strength of the dependency of the response y to an input x_1 might depend on another input, say x_2. The simplest such situation would be, if the coefficient β_1, corresponding on x_1 is itself proportional to x_2, say $\beta_1 = \gamma x_2$. In this case we have $y = \cdots + \beta_1 x_1 = \cdots + \gamma x_1 x_2$. Traditionally, inputs added to a model which are the product of two or more of the original inputs are called **interactions**. As a general rule, an interaction is only included if the "main effects" are also included in the model, so in any variable selection procedure, we would not drop x_1 or x_2 if $x_1 x_2$ is still in the model. An exception to this is when one can directly interpret the interaction term without the main effect.

In R, interactions (*i.e.* products of inputs) are represented by the symbol `:` and `*` can be used to include two variables together with their product. Note also, that if a variable

is known to be a factor, and it is included as an interaction term, then R will make sure to only remove the interaction term in a backward variable selection procedure whilst the main effects are also included.

Example 15.3. Consider the following toy dataset:

```
x1 <- c(2,2,2)
x2 <- 1:3
```

In `lm()` and related functions we can write `x1:x2` as a shorthand for `I(x1*x2)`. The column `x1:x2` in the R output corresponds to the mathematical expression $x_1 x_2$.

```
model.matrix(~ x1 + x1:x2)
```

```
#   (Intercept) x1 x1:x2
# 1           1  2     2
# 2           1  2     4
# 3           1  2     6
# attr(,"assign")
# [1] 0 1 2
```

We can write `x1*x2` as a shorthand for the model with the three terms `x1 + x2 + x1:x2`:

```
model.matrix(~ x1*x2)
```

```
#   (Intercept) x1 x2 x1:x2
# 1           1  2  1     2
# 2           1  2  2     4
# 3           1  2  3     6
# attr(,"assign")
# [1] 0 1 2 3
```

Interactions also work for factors. In this case, products with all of the corresponding indicator variables are added to the model. while the base effect of factor variables is to have different intercepts for the different groups, using interactions with a factor allows to have different slopes for the different groups.

Example 15.4. The classical example of an interaction is to consider how the yield y of a crop is related to temperature x_1 and rainfall x_2. All these variables are continuous, but we might categorize rainfall as "wet" and "dry". Then a simplistic view could be that, for high rainfall, the yield will be positively correlated with temperature, whereas for low rainfall the correlation may be slightly negative, because in hot weather, plants need more water, and if it is very hot and dry, the plants may even die.

We generate a toy dataset in R to represent such a situation:

```
T <- runif(25, 15, 35)
wet <- data.frame(
    yield=40 + 7*T + rnorm(25, sd = 10),
    temperature=T,
    rainfall="high")
T <- runif(25, 15, 35)
dry <- data.frame(
    yield=120 - 2*T + rnorm(25, sd = 10),
    temperature=T,
    rainfall="low")
crops <- rbind(wet, dry)
crops$rainfall <- factor(crops$rainfall, levels = c("low", "high"))
```

Now we fit a model, including `temperature`, `rainfall` and an interaction term:

```
m <- lm(yield ~ temperature*rainfall, data = crops)
summary(m)
```

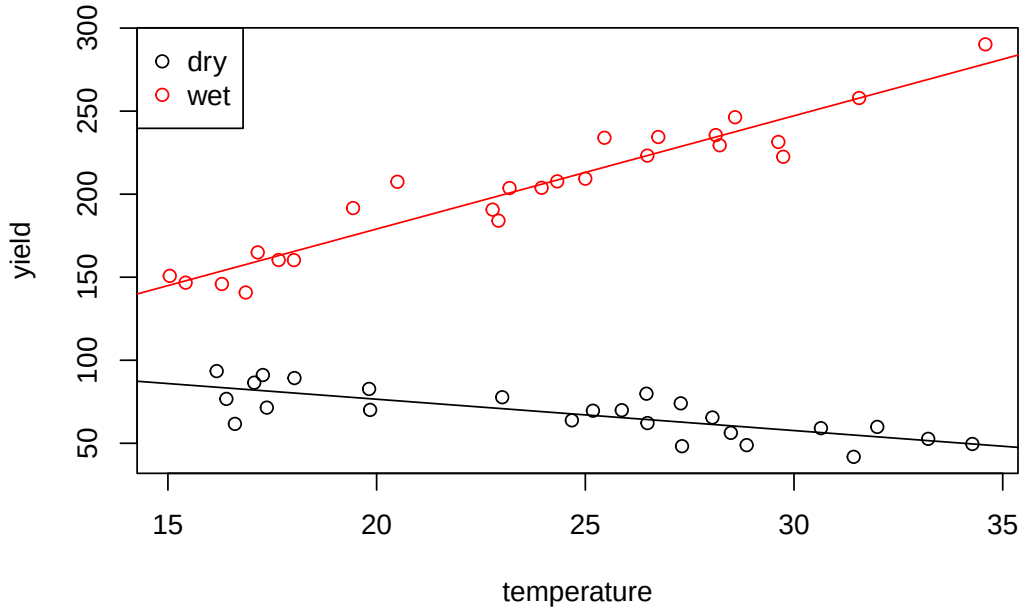
```
#
# Call:
# lm(formula = yield ~ temperature * rainfall, data = crops)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -22.8996  -6.3555   0.0346   5.9355  25.0553
#
# Coefficients:
#                           Estimate Std. Error t value Pr(>|t|)
# (Intercept)               114.2722     9.0929  12.567  < 2e-16 ***
# temperature                -1.8873     0.3617  -5.219 4.21e-06 ***
# rainfallhigh              -71.5892    13.0342  -5.492 1.66e-06 ***
# temperature:rainfallhigh    8.7043     0.5299  16.428  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 10.42 on 46 degrees of freedom
# Multiple R-squared:  0.9814,  Adjusted R-squared:  0.9802
# F-statistic: 810.2 on 3 and 46 DF,  p-value: < 2.2e-16
```

The fitted values for this model are

$$\widehat{y} = \begin{cases} \beta_0 + \beta_1 x_1 & \text{for low rainfall, and} \\ (\beta_0 + \beta_2) + (\beta_1 + \beta_3) x_1 & \text{for high rainfall.} \end{cases}$$

Finally, we can generate a plot of the data together with the two regression lines.

```
plot(yield ~ temperature, data=crops,
     col = ifelse(rainfall == "low", "black", "red"))
abline(a = coef(m)[1], b = coef(m)[2])
abline(a = coef(m)[1] + coef(m)[3], b = coef(m)[2] + coef(m)[4],
       col="red")
legend("topleft", c("dry", "wet"),
       col = c("black", "red"), pch = 1)
```



As expected, the two lines have different intercepts and different slopes.

Summary

- Indicator variables can be used to represent categorical inputs in linear regression models.
- The levels of a factor correspond to different intercepts for the regression line.
- If interaction terms are used, the levels also affect the slope.

Practical

- The MATH3714 and MATH5714M modules are assessed by an examination (80%) and a practical (20%). This is the practical, worth 20% of your final module mark.
- You must hand in your solution (printed on paper) by **Friday, 16th December 2022, 2pm**. The easiest way to hand in is to pass your report to me after one of the lectures. Alternatively, I will also set up a pidgeonhole for you to drop your report into.
- Reports must be typeset (not handwritten) and should be no more than 8 pages in length (but can be shorter). Your report must show your name and student ID on the front page.
- Within reason you may talk to your friends about this piece of work, but you should not send R code (or output) to each other. Your report must be your own work.

Dataset

In this practical we are interested in factors which influence the life expectancy at birth. We consider the following dataset:

- https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/practical.csv

You can read the data into R using the following command:

```
d <- read.csv("https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/practical.csv",
              stringsAsFactors = TRUE)
```

The dataset contain the following variables:

- `country`: country
- `region`: geographic region the country is in
- `sub.region`: geographic sub-region the country is in
- `year`: year the data refers to
- `GDP.per.capita`: Per capita GDP at current prices (USD)
- `health.spending`: Per capita government expenditure on health at average exchange rate (USD, only until 2010)
- `HIV`: Prevalence of HIV among adults aged 15 to 49 (%)
- `alcohol`: Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol)
- `tobacco`: Prevalence of current tobacco use among adults aged ≥ 15 years
- `population.size`: population size, for the given country and year (thousands)
- `population.density`: Population per square kilometre, for the given country and year (thousands)
- `life.expectancy`: Life expectancy at birth (years). This is the variable we are interested in.

Tasks

The aim of this practical is to fit an appropriate model to these data, which predicts the life expectancy `life.expectancy` from the other variables.

This practical is deliberately open-ended, with little guidance on how to proceed.

Task 1. We start by considering `life.expectancy` as a function of `GDP.per.capita` only.

- Using appropriate transformations of the data, find a linear model which can describe the relationship between `life.expectancy` (response) and `GDP.per.capita` (input).

- Using appropriate diagnostics, confirm that your model is acceptable. In your answer, you only need to describe your final model and *one* other model which you have examined, but deemed less appropriate.
- Using your model obtain a 95% confidence interval for the mean (expected) life expectancy at birth, for a country with a per capita GDP of 5000 USD.

Task 2. Now we also consider the remaining variables in the dataset.

- With due consideration to
  - transformations (where, and if, necessary)
  - appropriate choice of variables
  - model selection
  - model checking
  - missing data
  - *etc.*

  obtain a model which is able to predict `life.expectancy` using some or all of these additional variables.
- Note that some variables are missing data. Missing data is indicated by the value `NA` (not available). You should decide how to deal with missing data, *e.g.* by ignoring the corresponding samples. You should explain your choice in your report.
- Justify your choice by comparing at least two "competing" models. The comparison should take note of at least (a) model selection criteria, (b) diagnostics, and (c) interpretability.
- Interpret the parameters in your preferred model.

There is no single right or wrong answer to this practical. The important thing is that you justify your approach.

References

The data were collected from the following sources.

- https://www.who.int/data/gho/data/indicators/indicator-details/GHO/life-expectancy-at-birth-(years)
  `country`, `year`, `life.expectancy`
- https://population.un.org/wpp/Download/Standard/CSV/
  `country`, `year`, `population.size`, `population.density`, `region`, `sub.region`
- https://data.un.org/Data.aspx?d=SNAAMA&f=grID%3A101%3BcurrID%3AUSD%3BpcFlag%3A1
  `country`, `year`, `GDP.per.capita`
- http://data.un.org/Data.aspx?d=WHO&f=MEASURE_CODE%3aWHS7_104
  `country`, `year`, `health.spending`
- https://www.who.int/data/gho/data/indicators/indicator-details/GHO/alcohol-recorded-per-capita-(15-)-consumption-(in-litres-of-pure-alcohol)
  `country`, `year`, `alcohol`
- https://www.who.int/data/gho/data/indicators/indicator-details/GHO/age-standardized-prevalence-of-current-tobacco-smoking-among-persons-aged-15-years-and-older
  `country`, `year`, `tobacco`
- https://www.who.int/data/gho/data/indicators/indicator-details/GHO/prevalence-of-hiv-among-adults-aged-15-to-49-(-)
  `country`, `year`, `HIV`

16 Examples

In this section we illustrate the results of the previous sections with the help of an example.

16.1 Use of Interaction Terms in Modelling

Example 16.1. Researchers in Food Science studied how big people's mouths tend to be. It transpires that mouth volume is related to more easily measured quantities like height and weight. Here we have data which gives the mouth volume (in cubic centimetres), age, sex (female = 0, male = 1), height (in metres) and weight (in kilos) for 61 volunteers. The first few samples are as shown:

```
# data from https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/mouth-volume.txt
dd <- read.table("data/mouth-volume.txt", header = TRUE)
head(dd)
```

```
#   Mouth_Volume Age Sex Height  Weight
# 1       56.659  29   1  1.730  70.455
# 2       47.938  24   0  1.632  60.000
# 3       60.995  45   0  1.727 102.273
# 4       68.917  23   0  1.641  78.636
# 5       60.956  27   0  1.600  57.273
# 6       76.204  23   1  1.746  66.818
```

Normally it is important to check that categorical variables are correctly recognised as factors, when importing data into R. As an exception, the fact that `sex` is stored as an integer variable (instead of a factor) here does not matter, since the default factor coding would be identical to the coding in the present data.

Our aim is to fit a model which describes mouth volume in terms of the other variables. We start by fitting a basic model:

```
m1 <- lm(Mouth_Volume ~ ., data = dd)
summary(m1)
```

```
#
# Call:
# lm(formula = Mouth_Volume ~ ., data = dd)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -33.242 -10.721  -3.223   8.800  43.218
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)   0.9882    45.2736   0.022   0.9827
# Age           0.3617     0.2607   1.387   0.1709
# Sex           5.5030     5.9385   0.927   0.3581
# Height       15.8980    29.1952   0.545   0.5882
# Weight        0.2640     0.1400   1.885   0.0646 .
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 15.04 on 56 degrees of freedom
# Multiple R-squared:  0.2588,  Adjusted R-squared:  0.2059
# F-statistic: 4.888 on 4 and 56 DF,  p-value: 0.001882
```

Since no coefficient is significantly different from zero (at 5%-level), and since the R^2

value is not close to~1, model fit seems poor. To improve model fit, we try to include all pairwise interaction terms:

```
m2 <- lm(Mouth_Volume ~ .^2, data = dd)
summary(m2)
```

```
#
# Call:
# lm(formula = Mouth_Volume ~ .^2, data = dd)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -35.528  -8.168  -1.995   6.704  44.136
#
# Coefficients:
#                 Estimate Std. Error t value Pr(>|t|)
# (Intercept)    2.875e+02  2.337e+02   1.230    0.225
# Age           -8.370e+00  6.352e+00  -1.318    0.194
# Sex           -9.318e+01  1.325e+02  -0.703    0.485
# Height        -1.540e+02  1.505e+02  -1.023    0.311
# Weight         5.889e-01  2.713e+00   0.217    0.829
# Age:Sex       -9.023e-01  1.035e+00  -0.872    0.388
# Age:Height     5.374e+00  4.062e+00   1.323    0.192
# Age:Weight    -1.754e-03  2.027e-02  -0.087    0.931
# Sex:Height     5.989e+01  7.536e+01   0.795    0.431
# Sex:Weight     3.110e-01  4.177e-01   0.744    0.460
# Height:Weight -2.680e-01  1.755e+00  -0.153    0.879
#
# Residual standard error: 15.22 on 50 degrees of freedom
# Multiple R-squared:  0.3219,  Adjusted R-squared:  0.1863
# F-statistic: 2.374 on 10 and 50 DF,  p-value: 0.0218
```

None of the interaction terms look useful when used in combination. We try to select a subset of variables to get a better fit:

```
library(leaps)
r3 <- regsubsets(Mouth_Volume ~ .^2, data = dd, nvmax = 10)
s <- summary(r3)
s
```

```
# Subset selection object
# Call: regsubsets.formula(Mouth_Volume ~ .^2, data = dd, nvmax = 10)
# 10 Variables  (and intercept)
#               Forced in Forced out
# Age               FALSE      FALSE
# Sex               FALSE      FALSE
# Height            FALSE      FALSE
# Weight            FALSE      FALSE
# Age:Sex           FALSE      FALSE
# Age:Height        FALSE      FALSE
# Age:Weight        FALSE      FALSE
# Sex:Height        FALSE      FALSE
# Sex:Weight        FALSE      FALSE
# Height:Weight     FALSE      FALSE
# 1 subsets of each size up to 10
# Selection Algorithm: exhaustive
#          Age Sex Height Weight Age:Sex Age:Height Age:Weight Sex:Height
# 1  ( 1 )  " " " " " "    " "    " "     " "        " "        " "
```

```
# 2  ( 1 )  " " " " " " "    " "    " "     " "        "*"        " "
# 3  ( 1 )  " " "*" " " "    " "    " "     " "        "*"        "*"
# 4  ( 1 )  " " "*" " " "    " "    " "     " "        "*"        "*"
# 5  ( 1 )  "*" " " "*"    " "    "*"     "*"        " "        " "
# 6  ( 1 )  "*" " " "*"    " "    "*"     "*"        " "        "*"
# 7  ( 1 )  "*" "*" "*"    " "    "*"     "*"        " "        "*"
# 8  ( 1 )  "*" "*" "*"    "*"    "*"     "*"        " "        "*"
# 9  ( 1 )  "*" "*" "*"    "*"    "*"     "*"        " "        "*"
# 10  ( 1 ) "*" "*" "*"    "*"    "*"     "*"        "*"        "*"
#           Sex:Weight Height:Weight
# 1  ( 1 )  " "        "*"
# 2  ( 1 )  "*"        " "
# 3  ( 1 )  " "        " "
# 4  ( 1 )  "*"        " "
# 5  ( 1 )  "*"        " "
# 6  ( 1 )  "*"        " "
# 7  ( 1 )  "*"        " "
# 8  ( 1 )  "*"        " "
# 9  ( 1 )  "*"        "*"
# 10  ( 1 ) "*"        "*"
```

The table shows the optimal model for every number of variables, for p ranging from 1 to 10. To choose the number of variables we can use the adjusted R^2-value:

```
s$which[which.max(s$adjr2),]
```

```
#   (Intercept)           Age           Sex        Height        Weight
#          TRUE          TRUE         FALSE          TRUE         FALSE
#       Age:Sex    Age:Height    Age:Weight    Sex:Height    Sex:Weight
#          TRUE          TRUE         FALSE         FALSE          TRUE
# Height:Weight
#         FALSE
```

The optimal model consists of the five variables listed. For further examination, we fit the corresponding model using `lm()`:

```
m3 <- lm(Mouth_Volume
         ~ Age + Height + Age:Sex + Age:Height + Sex:Weight,
         data = dd)
summary(m3)
```

```
#
# Call:
# lm(formula = Mouth_Volume ~ Age + Height + Age:Sex + Age:Height +
#     Sex:Weight, data = dd)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -36.590  -9.577  -1.545   7.728  42.779
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept)  265.6236   122.3362   2.171   0.0342 *
# Age           -8.4905     4.0559  -2.093   0.0409 *
# Height      -134.0835    72.8589  -1.840   0.0711 .
# Age:Sex       -0.8633     0.4944  -1.746   0.0864 .
# Age:Height     5.3644     2.4047   2.231   0.0298 *
```

```
# Sex:Weight     0.4059     0.1659   2.446   0.0177 *
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 14.65 on 55 degrees of freedom
# Multiple R-squared:  0.3096,  Adjusted R-squared:  0.2468
# F-statistic: 4.932 on 5 and 55 DF,  p-value: 0.0008437
```

This looks much better: now most variables are significantly different from zero and the adjusted R^2 value has (slightly) improved. Nevertheless, the structure of the model seems hard to explain and the model seems difficult to interpret. For comparison, we consider the second-best model, with $p = 2$:

```
m4 <- lm(Mouth_Volume ~ Age:Weight + Sex:Weight, data = dd)
summary(m4)
```

```
#
# Call:
# lm(formula = Mouth_Volume ~ Age:Weight + Sex:Weight, data = dd)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -34.161 -10.206  -3.738   9.087  43.747
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept) 43.678187   4.788206   9.122 8.35e-13 ***
# Age:Weight   0.005554   0.002372   2.342   0.0226 *
# Weight:Sex   0.127360   0.048640   2.618   0.0113 *
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 14.72 on 58 degrees of freedom
# Multiple R-squared:  0.2651,  Adjusted R-squared:  0.2398
# F-statistic: 10.46 on 2 and 58 DF,  p-value: 0.000132
```

The adjusted R^2-value of this model is only slightly reduced, and all regression coefficients are significantly different from zero.

$$\text{mouth volume} = 43.68 + \begin{cases} 0.0056\,\text{age} \times \text{weight} & \text{for females} \\ (0.0056\,\text{age} + 0.1274)\,\text{weight} & \text{for males.} \end{cases}$$

This model is still not trivial to interpret, but it does show that mouth volume increases with weight and age, and that the dependency on weight is stronger for males.

16.2 Alternative Factor Codings

The above coding of factors using indicator variables is the default procedure in R. There are other choices of coding. It transpires that any mapping to vectors which (together with the intercept, if present) span a k dimensional space, will work for a factor with k levels. The choice of coding will not affect fitted values and the goodness of fit, but it will affect the estimates $\hat{\beta}$, so the coding needs to be known before any interpretation of the parameters.

Differnt choices of factor codings are built into R. Here we illustrate some of these choices using a factor which represents week days:

```
days <- factor(c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))
```

- **Indicator variables.** This is the default coding:

  ```
  contr.treatment(days)
  ```

  ```
  #     Tue Wed Thu Fri Sat Sun
  # Mon   0   0   0   0   0   0
  # Tue   1   0   0   0   0   0
  # Wed   0   1   0   0   0   0
  # Thu   0   0   1   0   0   0
  # Fri   0   0   0   1   0   0
  # Sat   0   0   0   0   1   0
  # Sun   0   0   0   0   0   1
  ```

  The seven rows correspond to the weekdays, the six columns show how each weekday is encoded. The regression coefficients can be interpreted as the change of intercept relative to the reference level (in this case Monday).

- **Deviation coding** uses "sum to zero" contrasts:

  ```
  contr.sum(days)
  ```

  ```
  #     [,1] [,2] [,3] [,4] [,5] [,6]
  # Mon    1    0    0    0    0    0
  # Tue    0    1    0    0    0    0
  # Wed    0    0    1    0    0    0
  # Thu    0    0    0    1    0    0
  # Fri    0    0    0    0    1    0
  # Sat    0    0    0    0    0    1
  # Sun   -1   -1   -1   -1   -1   -1
  ```

  This constrasts each level with the final level. For example, $\hat{\beta}_1$ is the amount the intercept is increased for Mondays and decreased for Sundays. This coding can sometimes reduce collinearity in the design matrix.

- **Helmert coding** contrasts the second level with the first, the third with the average of the first two, and so on:

  ```
  contr.helmert(days)
  ```

  ```
  #     [,1] [,2] [,3] [,4] [,5] [,6]
  # Mon   -1   -1   -1   -1   -1   -1
  # Tue    1   -1   -1   -1   -1   -1
  # Wed    0    2   -1   -1   -1   -1
  # Thu    0    0    3   -1   -1   -1
  # Fri    0    0    0    4   -1   -1
  # Sat    0    0    0    0    5   -1
  # Sun    0    0    0    0    0    6
  ```

We can set the coding for each factor separately, using the optional `contrasts.arg` argument to `lm()` and `model.matrix()`:

```
set.seed(20211127)
n <- 20

x1 <- runif(n)

day.names <- c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
x2 <- sample(day.names, size = n, replace = TRUE)
```

```r
# Explicitly specify the levels, in case a day is missing
# from the sample and to fix the order of days:
x2 <- factor(x2, levels = day.names)

dd <- data.frame(x1, x2)

X1 <- model.matrix(~ ., data = dd)
head(X1)
```

```
#   (Intercept)         x1 x2Tue x2Wed x2Thu x2Fri x2Sat x2Sun
# 1           1 0.41745367     1     0     0     0     0     0
# 2           1 0.88671893     0     0     1     0     0     0
# 3           1 0.41924327     0     0     1     0     0     0
# 4           1 0.31986826     0     0     0     0     0     1
# 5           1 0.27863787     0     0     0     0     0     0
# 6           1 0.03346104     0     1     0     0     0     0
```

```r
kappa(X1, exact = TRUE)
```

```
# [1] 10.37243
```

```r
X2 <- model.matrix(~ ., data = dd,
                   contrasts.arg = list(x2 = contr.sum))
head(X2)
```

```
#   (Intercept)         x1 x21 x22 x23 x24 x25 x26
# 1           1 0.41745367   0   1   0   0   0   0
# 2           1 0.88671893   0   0   0   1   0   0
# 3           1 0.41924327   0   0   0   1   0   0
# 4           1 0.31986826  -1  -1  -1  -1  -1  -1
# 5           1 0.27863787   1   0   0   0   0   0
# 6           1 0.03346104   0   0   1   0   0   0
```

```r
kappa(X2, exact = TRUE)
```

```
# [1] 6.677298
```

Summary

- We have seen how interaction terms can be used to improve model fit.
- We have seen how different factor codings can be used in R.

17 Robust Regression

The overall aim of this part of the module is to find regression estimators which work well in the presence of outliers.

17.1 Outliers

As before, we consider the model

$$y_i = x_i^\top \beta + \varepsilon_i,$$

and we try to estimate β from given data (x_i, y_i) for $i \in \{1, \dots, n\}$.

An **outlier** is a sample (x_i, y_i) that differs significantly from the majority of observations. The deviation can be either in the x-direction, or the y-direction, or both. Outliers can either occur by chance, or can be caused by errors in the recording or processing of the data. In this subsection we will discuss different ways to detect outliers.

17.1.1 Leverage

Definition 17.1. The **leverage** of the input x_i is given by

$$h_{ii} = x_i^\top (X^\top X)^{-1} x_i,$$

where X is the design matrix.

Since x_i is the ith row of X, the leverage h_{ii} equals the ith diagonal element of the hat matrix $H = X(X^\top X)^{-1} X$.

Lemma 17.1. *The derivative of the fitted value $\hat{y}_i$ with respect to the response y_i is given by*

$$\frac{\partial}{\partial y_i} \hat{y}_i = h_{ii}$$

for all $i \in \{1, \dots, n\}$.

Proof. We have

$$\hat{y}_i = (Hy)_i = \sum_{j=1}^{n} h_{ij} y_j.$$

Taking derivatives with respect to y_i completes the proof. □

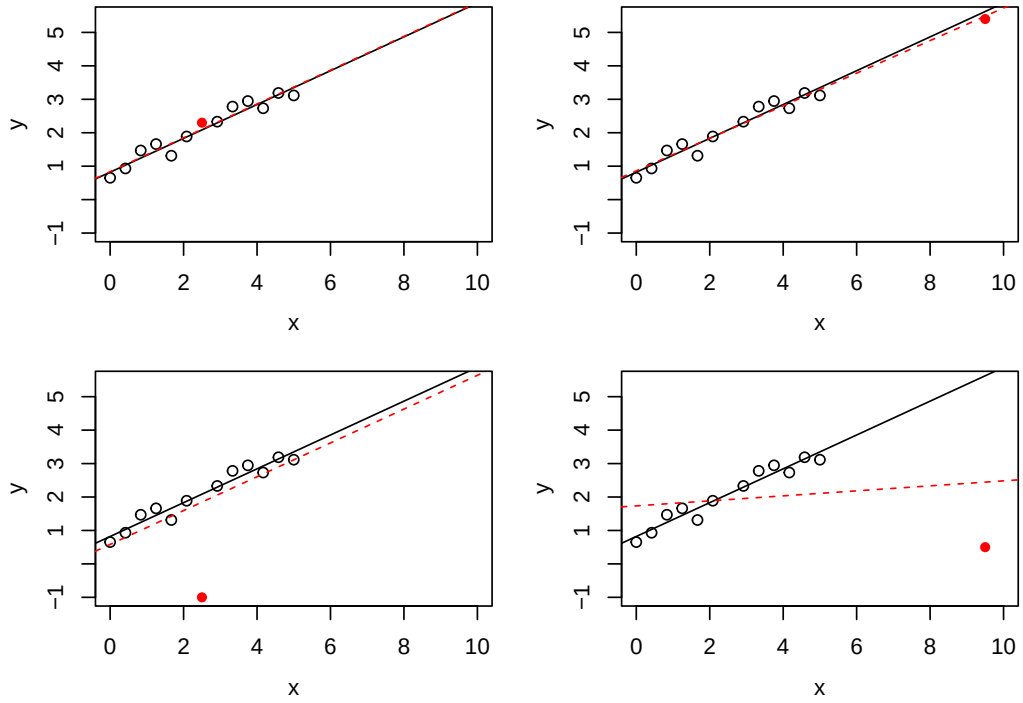
The leverage of x_i describes the *potential* for y_i to affect the fitted line or hyperplane. If h_{ii} is large, changing of y_i has a large effect on the fitted values. This is in contrast to the influence of a sample, which describes *actual* effect on the regression line: in the section about Cook's Distance, we considered a sample to be "influential", if the regression estimates with and without the sample in question were very different.

Since we have $\sum_{i=1}^{n} h_{ii} = p + 1$, the average value of the leverage over all samples is $(p + 1)/n$. Using a more careful argument, one can show that $h_i \in [1/n, 1]$ for every $i \in \{1, \dots, n\}$.

Example 17.1. We use simulated data to illustrate the effect of outliers and the difference between leverage and influence.

```
set.seed(20211130)

n <- 12
x <- seq(0, 5, length.out = n+1)[-(n/2+1)]
y <- 1 + 0.4 * x + rnorm(n, sd = 0.3)
m <- lm(y ~ x)
```

```r
# we try four different extra samples, one after another:
x.extra <- c(2.5, 9.5, 2.5, 9.5)
y.extra <- c(2.3, 5.4, -1, 0.5)

par(mfrow = c(2, 2), # We want a 2x2 grid of plots,
    mai = c(0.7, 0.7, 0.1, 0.1), # using smaller margins, and
    mgp = c(2.5, 1, 0)) # we move the labels closer to the axes.
for (i in 1:4) {
    plot(x, y, xlim = c(0, 10), ylim = c(-1, 5.5))
    abline(m)

    m2 <- lm(c(y, y.extra[i]) ~ c(x, x.extra[i]))
    abline(m2, col="red", lty="dashed")
    points(x.extra[i], y.extra[i], col = "red", pch = 16)
}
```



The black circles in all four panels are the same, and the black, solid line is the regression line fitted to these points. The filled, red circle is different between the panels and the red, dashed line is the regression line fitted after this point has been added to the data. In the top-left panel there is no outlier. In the top-right panel, the red circle is an x-space outlier. The point has high leverage (since the red lines in the top-right and bottom-right panels are very different from each other), but low influence (since the red line is close to the black line). The bottom-left panel shows a y-space outlier. The red circle has low leverage (since the red line is similar to the top-left panel) and low influence (since the red line is close to the black line), but a large residual. Finally, the bottom-right panel shows an x-space outlier with high leverage and high influence.

17.1.2 Studentised Residuals

Sometimes, y-space outliers can be detected by having particularly large residuals. We have

$$\hat{\varepsilon} = (I - H)Y = (I - H)(X\beta + \varepsilon) = (I - H)\varepsilon,$$

where we used equation (19) for the last equality sign. Thus we have

$$\mathrm{Cov}(\hat{\varepsilon}) = \mathrm{Cov}((I-H)\varepsilon) = (I-H)\sigma^2 I(I-H)^\top = \sigma^2(I-H).$$

Considering the diagonal elements of the covariance matrix, we find $\mathrm{Var}(\hat{\varepsilon}_i) = \sigma^2(1-h_{ii})$, where h_{ii} is the ith diagonal element of the hat matrix H. This motivates the following definition.

Definition 17.2. The **studentised residuals** are given by

$$r_i := \frac{\hat{\varepsilon}_i}{\sqrt{\hat{\sigma}^2(1-h_{ii})}}.$$

Lemma 17.2. *We have* $r_i \sim t(n-p-1)$.

Proof. We have

$$\begin{aligned} r_i &= \frac{\hat{\varepsilon}_i}{\sqrt{\hat{\sigma}^2(1-h_{ii})}} \\ &= \frac{\hat{\varepsilon}_i/\sqrt{\sigma^2(1-h_{ii})}}{\sqrt{\frac{(n-p-1)\hat{\sigma}^2}{\sigma^2}/(n-p-1)}}. \end{aligned}$$

The numerator is standard normally distributed. From equation (22) we know that $(n-p-1)\hat{\sigma}^2/\sigma^2 \sim \chi^2(n-p-1)$. Thus we find $r_i \sim t(n-p-1)$, by the definition of the t-distribution. □

Samples where the studentised residuals are large, compared to quantiles of the $t(n-p-1)$-distribution, might be y-space outliers.

With the help of lemma 9.4 we can express Cook's distance as a function of studentised residuals and leverage. We get

$$\begin{aligned} D_i &= \frac{\hat{\varepsilon}_i^2}{(p+1)\hat{\sigma}^2} \cdot \frac{h_{ii}}{(1-h_{ii})^2} \\ &= r_i^2 \frac{h_{ii}}{(p+1)(1-h_{ii})}. \end{aligned}$$

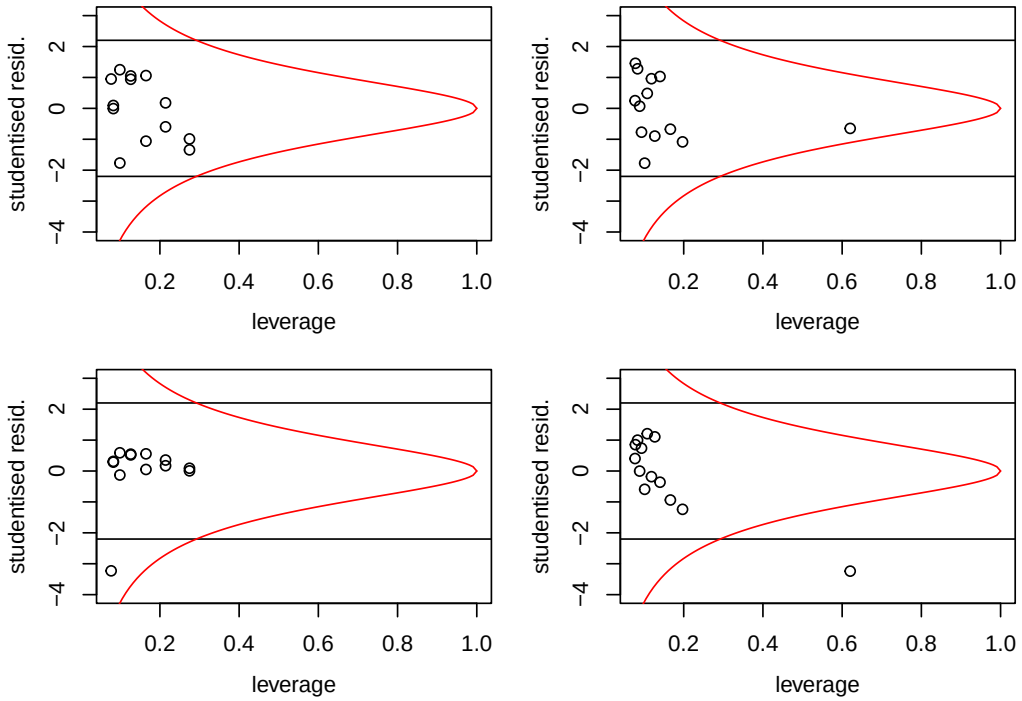
If we denote samples with $D_i \geq 1$ as influential, we can express the condition for influential samples as

$$r_i^2 \geq (p+1)\frac{1-h_{ii}}{h_{ii}}.$$

Example 17.2. Continuing from the previous example, we can plot studentised residuals against leverage. As in section 9.1, we can use the function `influence()` to easily obtain the diagonal elements of the hat matrix.

```
par(mfrow = c(2, 2), # We want a 2x2 grid of plots,
    mai = c(0.7, 0.7, 0.1, 0.1), # using smaller margins, and
    mgp = c(2.5, 1, 0)) # we move the labels closer to the axes.
for (i in 1:4) {
    xx <- c(x, x.extra[i])
    yy <- c(y, y.extra[i])
    n <- length(xx)
    p <- 1

    m <- lm(yy ~ xx)
```

```
    # get the leverage
    hii <- influence(m)$hat

    # get the studentized residuals
    sigma.hat <- summary(m)$sigma
    ri <- resid(m) / sigma.hat / sqrt(1 - hii)

    plot(hii, ri, xlim=c(1/n, 1), ylim=c(-4,3),
         xlab = "leverage", ylab = "studentised resid.")

    # plot a 95% interval for the residuals
    abline(h = +qt(0.975, n - p -1))
    abline(h = -qt(0.975, n - p -1))

    # also plot the line where D_i = 1
    h <- seq(1/n, 1, length.out = 100)
    r.infl <- sqrt((p+1) * (1-h) / h)
    lines(h, r.infl, col="red")
    lines(h, -r.infl, col="red")
}
```



If the model is correct, 95% of the samples should lie in the band formed by the two horizontal lines. In the bottom-left panel we can recognise a y-space outlier by the fact that it is outside the band. In the two right-hand panels we can recognise the x-space outlier by the fact that it has much larger leverage than the other samples. Finally, samples which are to the right of the curved, red line correspond the "influential" observations.

17.2 Breakdown Points

In the example we have seen that changing a single observation can have a large effect on the regression line found by least squares regression. Using the formula

$$\hat{\beta} = (X^\top X)^{-1} X^\top y,$$

it is easy to see that it is enough to change a single sample, letting $y_i \to \infty$, to get $\|\hat{\beta}\| \to \infty$.

Definition 17.3. The **finite sample breakdown point** of an estimator $\hat{\theta}$ is the smallest fraction of samples such that $\|\hat{\theta}\| \to \infty$ can be achieved by only changing this fraction of samples. Mathematically, this can be expressed as follows: Let $z = ((x_1, y_1), \dots, (x_n, y_n))$ and consider all z' such that the set $\{i \mid z_i \neq z_i'\}$ has at most m elements. Then the finite sample breakdown point is given by

$$\varepsilon_n(\hat{\theta}) := \frac{1}{n} \min\{m \mid \sup_{z'} \|\hat{\theta}(z')\| = \infty\}.$$

The (asymptotic) **breakdown point** of $\hat{\theta}$ is given by

$$\varepsilon(\hat{\theta}) := \lim_{n \to \infty} \varepsilon_n(\hat{\theta}).$$

Clearly we have $\varepsilon_n(\hat{\theta}) \in [0, 1]$ for all n and thus $\varepsilon(\hat{\theta}) \in [0, 1]$. Robust estimators have $\varepsilon(\hat{\theta}) > 0$. Using the argument above, we see that $\varepsilon_n(\hat{\beta}) = 1/n \to 0$ as $n \to \infty$. Thus, the least squares estimate is not robust.

Example 17.3. For illustration we show a very simple version of a robust estimator for the regression line in simple linear regression. The estimate is called a **resistant line** or **Tukey line**. Assume we are given data (x_i, y_i) for $i \in \{1, \dots, n\}$. We start by sorting the x_i in increasing order: $x_{(1)} \leq \cdots \leq x_{(n)}$. Then we define x_{L}, x_{M} and x_{R} as the median of the lower, middle and upper third of the $x_{(i)}$, and y_{L}, y_{M} and y_{R} as the median of the corresponding y values, respectively. Finally, set

$$\hat{\beta}_1 = \frac{y_{\mathrm{R}} - y_{\mathrm{L}}}{x_{\mathrm{R}} - x_{\mathrm{L}}}$$

as an estimate of the slope, and

$$\hat{\beta}_0 = \frac{y_{\mathrm{L}} + y_{\mathrm{M}} + y_{\mathrm{R}}}{3} - \hat{\beta}_1 \frac{x_{\mathrm{L}} + x_{\mathrm{M}} + x_{\mathrm{R}}}{3}$$

for the intercept.

We can easily implement this method in R. To demonstrate this, we use simulated data with artificial outliers:

```
set.seed(20211129)
n <- 24
x <- runif(n, 0, 10)
y <- -1 + x + rnorm(n)

outlier <- 1:5
y[outlier] <- y[outlier] + 15
```

We start by sorting the data in order of increasing x-values:

```
idx <- order(x)
x <- x[idx]
y <- y[idx]
```

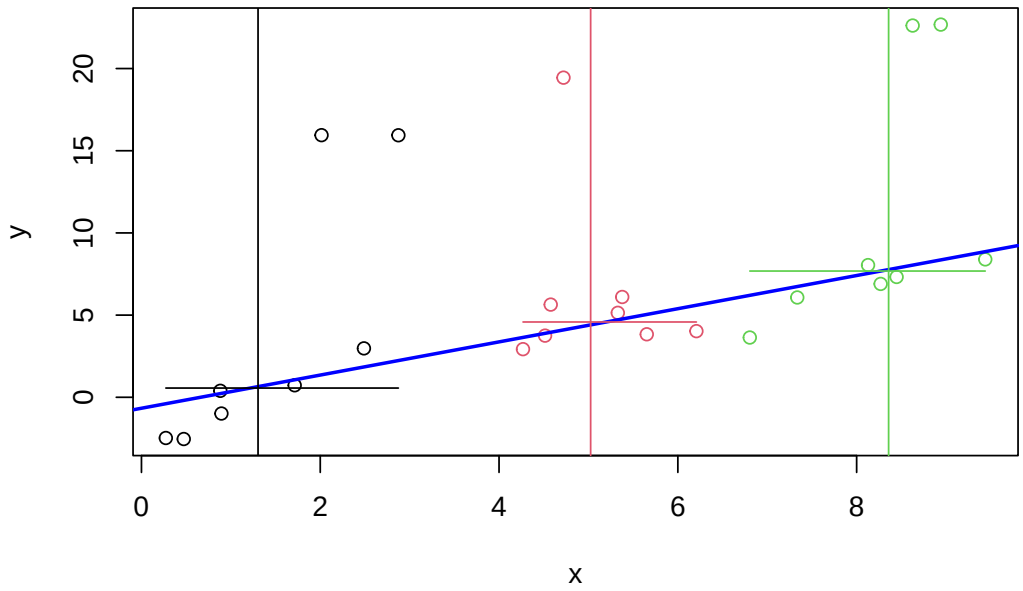
Now we can compute the resistant line:

```
xL <- median(x[1:8])
xM <- median(x[9:16])
xR <- median(x[17:24])
yL <- median(y[1:8])
```

```
yM <- median(y[9:16])
yR <- median(y[17:24])
beta1 <- (yR - yL) / (xR - xL)
beta0 <- (yL + yM + yR) / 3 - beta1 * (xL + xM + xR) / 3
```

Finally, we plot the result.

```
plot(x, y, col = rep(1:3, each=8))
abline(beta0, beta1, col = "blue", lwd = 2)
abline(v = xL, col = 1)
abline(v = xM, col = 2)
abline(v = xR, col = 3)
segments(c(x[1], x[9], x[17]), c(yL, yM, yR), c(x[8], x[16], x[24]),
         col = 1:3)
```



The horizontal and vertical line segments in the plot give the medians used in the computation. The blue, sloped line is the resistant line. One can see that the line closely follows the bulk of the samples and that it is not affected by the five outliers.

A variant of this method is implemented by the R function `line()`.

Summary

- Leverage can be used to detect x-space outliers.
- Studentised residuals can be used to detect y-space outliers.
- The breakdown point of an estimator is a measure for how robust the method is to outliers.

Problem Sheet 5

You should attempt all these questions and write up your solutions in advance of the workshop in week 10 where the answers will be discussed.

17. Consider the dataset at

- https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/P05Q17.csv

which contains samples from variables $x_1, x_2, y \in \mathbb{R}$ and $x_3 \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$.

a. Load the data into R.

b. Fit a simple linear model to describe y as a function of x_1, x_2 and x_3, taking into account that x_3 is a categorical variable. Show that the behaviour of the residuals depends on the levels of x_3.

c. Find an improved model, which includes some of the interaction terms between x_3 and the other variables. For each level of x_3, write the mathematical equation for the fitted model mean of your chosen model.

18. Consider the dataset at

- https://www1.maths.leeds.ac.uk/~voss/2022/MATH3714/P05Q18.csv

Identify the x-space outlier in this dataset.

19. The sample median is a robust estimator of the mean. Determine the breakdown point.

20. Determine the breakdown point for the resistant line from example 17.3.

18 M-Estimators

In this section we introduce a class of estimators which can be robust to y-space outliers.

18.1 Definition

Definition 18.1. An **M-estimator** for the regression coefficients β is a method which computes the estimator $\hat{\beta}$ as

$$\hat{\beta} = \arg\min_{\beta} r(\varepsilon) = \arg\min_{\beta} r(y - X\beta),$$

where r is given by

$$r(\varepsilon) = \sum_{i=1}^{n} \rho(\varepsilon_i)$$

and $\rho\colon \mathbb{R} \to \mathbb{R}$ is a symmetric, continuous function with a unique minimum at 0. The function ρ is called the **objective function** of the M-estimator.

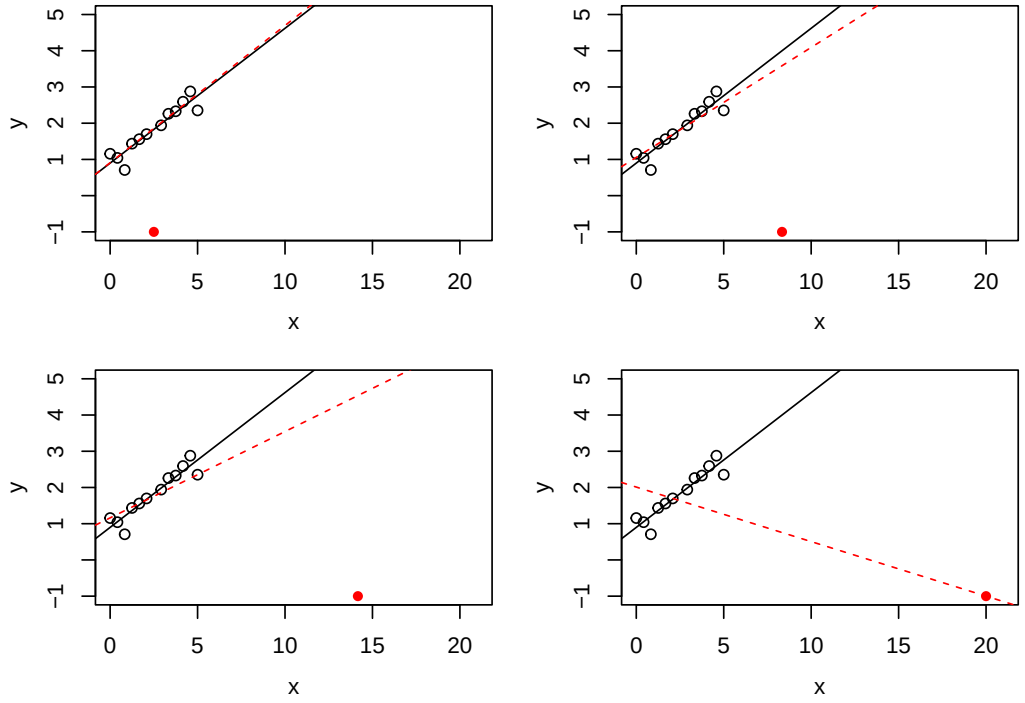
This is a generalisation of the least squares estimator, where the least squares estimator corresponds to the case $\rho(\varepsilon) = \varepsilon^2$.

For most functions ρ there is no closed-form expression for the estimate $\hat{\beta}$, but numerical minimisation can be used to find numerical values for $\hat{\beta}$. In R, the function `optim()` can be used for this purpose.

Example 18.1. The M-estimator with $\rho(u) = |u|$ is called the **least absolute values** (LAV) estimator. Since ρ grows slower than u^2 as $|u| \to \infty$, this estimator is more resistant to y-space outliers than least squares regression is. On the other hand, for x-space outliers which are sufficiently far away from the bulk of the data, the LAV line will pass exactly through the corresponding point (x_i, y_i), so the LAV estimator is more susceptible to x-space outliers than least squares regression is, and the breakdown point is still $1/n$.

```
set.seed(202111202)

n <- 12
x <- seq(0, 5, length.out = n+1)[-(n/2+1)]
y <- 1 + 0.4 * x + rnorm(n, sd = 0.3)
m <- lm(y ~ x)

# we try four different extra samples, one after another:
x.extra <- seq(2.5, 20, length.out = 4)
y.extra <- c(-1, -1, -1, -1)

lav <- function(par, x, y) {
  beta0 <- par[1]
  beta1 <- par[2]
  sum(abs(y - beta0 - beta1*x))
}

par(mfrow = c(2, 2), # We want a 2x2 grid of plots,
    mai = c(0.7, 0.7, 0.1, 0.1), # using smaller margins, and
    mgp = c(2.5, 1, 0)) # we move the labels closer to the axes.
for (i in 1:4) {
  plot(x, y, xlim = c(0, 21), ylim = c(-1, 5))
  abline(m)

  o <- optim(c(0, 0), lav, NULL, c(x, x.extra[i]), c(y, y.extra[i]))
```

```
  abline(o$par[1], o$par[2], col="red", lty="dashed")
  points(x.extra[i], y.extra[i], col = "red", pch = 16)
}
```



Define $s(\beta) = r(y - X\beta)$. Then we have

$$\hat{\beta} = \arg\min_{\beta} s(\beta). \tag{35}$$

Some care is neede since the function s may have more than one local minimum, and even the global minimum may not be uniquely defined.

Lemma 18.1. *Assume that ρ is convex. Then s is convex.*

Proof. Here we use the property that a function $f\colon \mathbb{R}^{p+1} \to \mathbb{R}$ is convex, if and only if

$$f(\lambda\beta^{(1)} + (1-\lambda)\beta^{(2)}) \leq \lambda f(\beta^{(1)}) + (1-\lambda) f(\beta^{(2)})$$

for all $\beta^{(1)}, \beta^{(2)} \in \mathbb{R}^{p+1}$ and all $\lambda \in [0,1]$. Since ρ is convex, we find

$$\begin{aligned}
&s(\lambda\beta^{(1)} + (1-\lambda)\beta^{(2)}) \\
&\quad = r\Big(y - X(\lambda\beta^{(1)} + (1-\lambda)\beta^{(2)})\Big) \\
&\quad = r\Big(\lambda(y - X\beta^{(1)}) + (1-\lambda)(y - X\beta^{(2)})\Big) \\
&\quad = \sum_{i=1}^{n} \rho\Big(\lambda(y - X\beta^{(1)})_i + (1-\lambda)(y - X\beta^{(2)})_i\Big) \\
&\quad \leq \sum_{i=1}^{n} \Big(\lambda\rho(y - X\beta^{(1)})_i + (1-\lambda)\rho(y - X\beta^{(2)})_i\Big) \\
&\quad = \lambda r(y - X\beta^{(1)}) + (1-\lambda) r(y - X\beta^{(2)}) \\
&\quad = \lambda s(\beta^{(1)}) + (1-\lambda) s(\beta^{(2)}).
\end{aligned}$$

Thus s is convex and the proof is complete. □

This shows that the problem of finding $\hat{\beta}$ is better behaved when ρ is convex. For example, in this case, every local minimum of s is also a global minimum and the set of all minima is a convex set.

M-estimates in general are not scale-invariant, *i.e.* if y is measured in different units, the regression line changes. A scale-invariant, robust estimator is given by

$$\hat{\beta} = \arg\min_{\beta} r\Big(\frac{\varepsilon}{\hat{\sigma}}\Big) = \arg\min_{\beta} r\Big(\frac{y - X\beta}{\hat{\sigma}}\Big),$$

where $\hat{\sigma}$ is a robust estimator of the residual standard deviation, for example the **median absolute deviation** (MAD) given by

$$\hat{\sigma} = \text{median}\big|\hat{\varepsilon}_i - \text{median}(\hat{\varepsilon}_i)\big|.$$

18.2 Iterative Methods

If ρ is differentiable, say $\rho' = \psi$, then gradient-based methods can be used to find the minimum in equation (35). The basis of this approach is the following lemma.

Lemma 18.2. *If $\rho' = \psi$, then the M-estimate satisfies*

$$\sum_{i=1}^{n} \psi((y - X\hat{\beta})_i) \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix} = 0 \in \mathbb{R}^{p+1}. \tag{36}$$

Proof. Since $\hat{\beta}$ minimises the function r, we have

$$\begin{aligned}
0 = & \frac{\partial}{\partial \beta_j} r(y - X\beta)\Big|_{\beta=\hat{\beta}} \\
= & \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} \rho((y - X\beta)_i)\Big|_{\beta=\hat{\beta}} \\
= & \sum_{i=1}^{n} \psi((y - X\beta)_i) \frac{\partial}{\partial \beta_j} (y - X\beta)_i\Big|_{\beta=\hat{\beta}} \\
= & \sum_{i=1}^{n} \psi((y - X\beta)_i)(-x_{ij})\Big|_{\beta=\hat{\beta}} \\
= & -\sum_{i=1}^{n} \psi((y - X\hat{\beta})_i) x_{ij}.
\end{aligned}$$

where we use the convention $x_{i0} = 1$. This completes the proof. □

With the result of the lemma, we have turned the problem of finding $\hat{\beta}$ into the problem of finding the roots of a non-linear function. To solve this problem, we can for example use one of the generic root finding algorithms, for example Newton's method.

In the context of M-estimates, a more specialised method is often used in the literature. For this method we define weights

$$w_i := \frac{\psi((y - X\hat{\beta})_i)}{(y - X\hat{\beta})_i} \tag{37}$$

and write W for the diagonal matrix which has $w_1, \ldots, w_n$ on the diagonal. Using this notation, we can rewrite condition (36) as

$$X^\top W(y - X\hat{\beta}) = 0$$

and solving for $\hat{\beta}$ gives

$$\hat{\beta} = (X^\top W X)^{-1} X^\top W y. \tag{38}$$

This equation looks very similar to our formula for the least squares estimate, the only change being the presence of the matrix W. One can check that the value $\hat{\beta}$ computed in this way also represents the solution of the weighted least-squares problem

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} w_i (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2.$$

While equation (38) looks like the solution of a linear problem at first glance, the weight matrix W depends on $\hat{\beta}$ and this equation still represents a non-linear equation for $\hat{\beta}$.

The **iteratively re-weighted least squares** method is an iterative method for solving equation (38). Starting from an initial guess $\hat{\beta}^{(0)}$ for $\hat{\beta}$, *e.g.* the least squares estimate or the zero vector, the method repeats the following steps for $k = 0, 1, 2, ...$:

1. Using equation (37), compute the weight matrix $W^{(k)}$ corresponding to $\hat{\beta}^{(k)}$.
2. Compute $\hat{\beta}^{(k+1)} = (X^\top W^{(k)} X)^{-1} X^\top W^{(k)} y$.

The algorithm finishes either after a fixed number of steps, or when the values of $\hat{\beta}^{(k)}$ don't change much between steps any more, or when the "residual" $X^\top W(y - X\hat{\beta})$ is "small enough".

18.3 Objective Functions

In this section we will discuss different choices of the objective function ρ, together with the derivative ψ and the weight function $w(\varepsilon) = \psi(\varepsilon)/\varepsilon$.

18.3.1 Least Squares Method

Here we can write

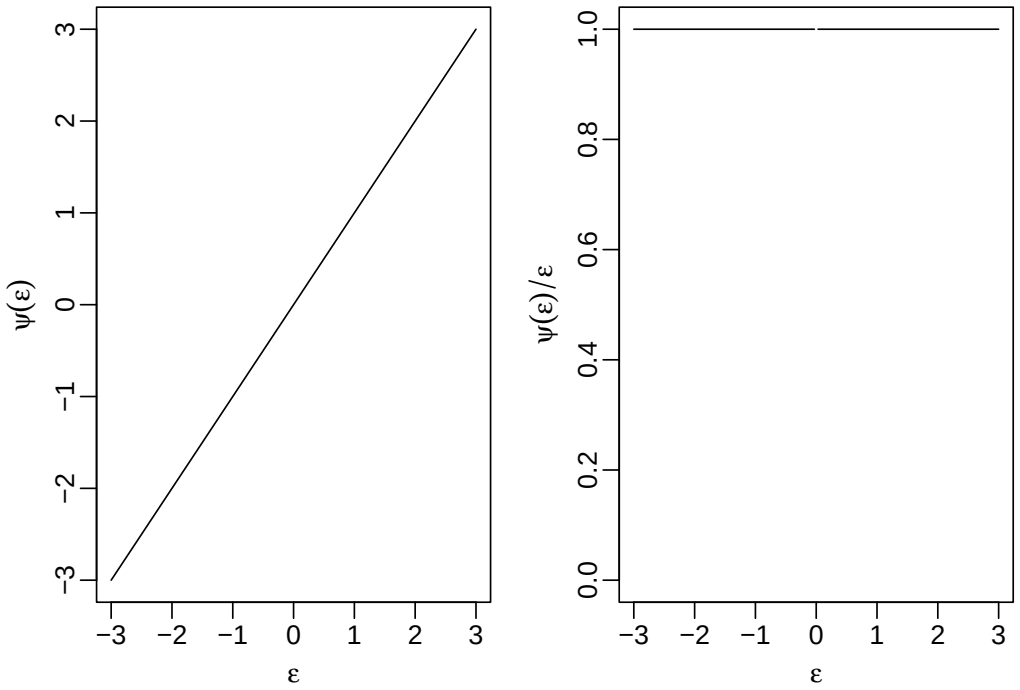
$$\rho(\varepsilon) = \varepsilon^2/2.$$

The factor $1/2$, which we omitted before, does not change the estimate. We include the factor here to give ψ a simpler form and to make comparison to the following examples easier. The derivative of ρ is given by

$$\psi(\varepsilon) = \varepsilon$$

The following R code can be used to plot ψ and the weight function w:

```
eps <- seq(-3, 3, length.out = 201)
psi <- function(eps) eps
par(mfrow = c(1, 2),
    mai = c(0.7, 0.7, 0.1, 0.1),
    mgp = c(1.7, 0.5, 0))
plot(eps, psi(eps), type="l",
     xlab = expression(epsilon), ylab = expression(psi(epsilon)))
plot(eps, psi(eps) / eps, type="l", ylim = c(0, 1),
     xlab = expression(epsilon),
     ylab = expression(psi(epsilon)/epsilon))
```



The right-hand plot shows that weight of observations does not depend on their magnitude. This property is what makes the least squares estimate susceptible to outliers.

18.3.2 Least Absolute Values

Here we have

$$\rho(\varepsilon) = |\varepsilon|.$$

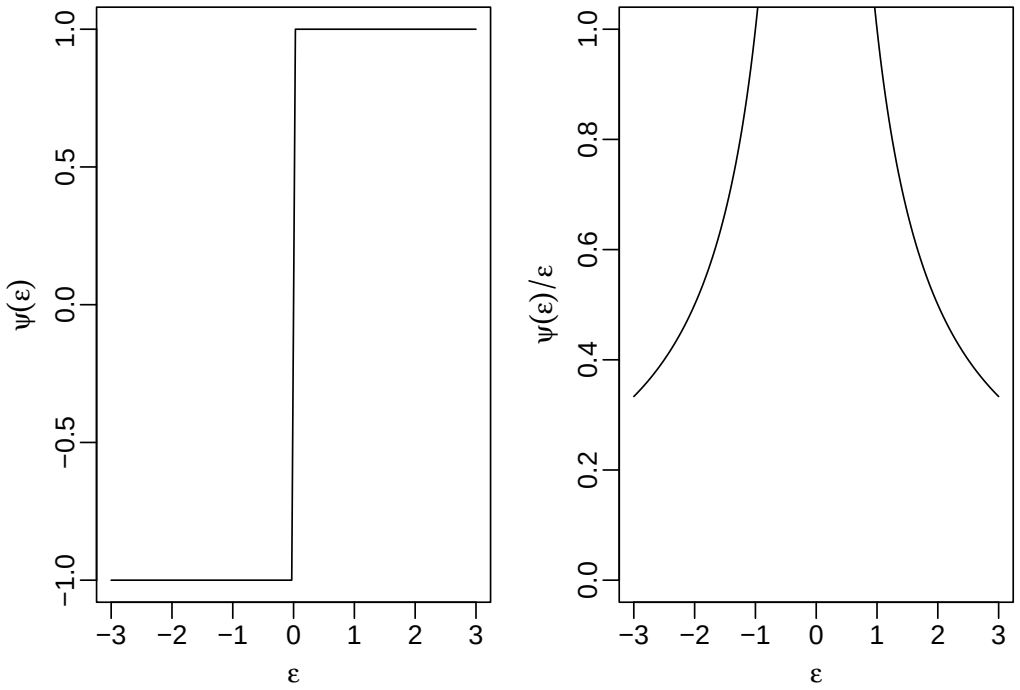
The derivative of this function is

$$\begin{aligned}\psi(\varepsilon) &= \begin{cases} -1 & \text{if } \varepsilon < 0, \\ +1 & \text{if } \varepsilon > 0 \end{cases} \\ &= \operatorname{sign}(\varepsilon)\end{aligned}$$

and the weight function is

$$w(\varepsilon) = \frac{\operatorname{sign}(\varepsilon)}{\varepsilon} = \frac{1}{|\varepsilon|}.$$

```
t <- 1
psi <- function(eps) sign(eps)
par(mfrow = c(1, 2),
    mai = c(0.7, 0.7, 0.1, 0.1),
    mgp = c(1.7, 0.5, 0))
plot(eps, psi(eps), type="l",
     xlab = expression(epsilon), ylab = expression(psi(epsilon)))
plot(eps, psi(eps) / eps, type="l", ylim = c(0, 1),
     xlab = expression(epsilon),
     ylab = expression(psi(epsilon)/epsilon))
```



18.3.3 Huber's t-function

This method uses the following objective function:

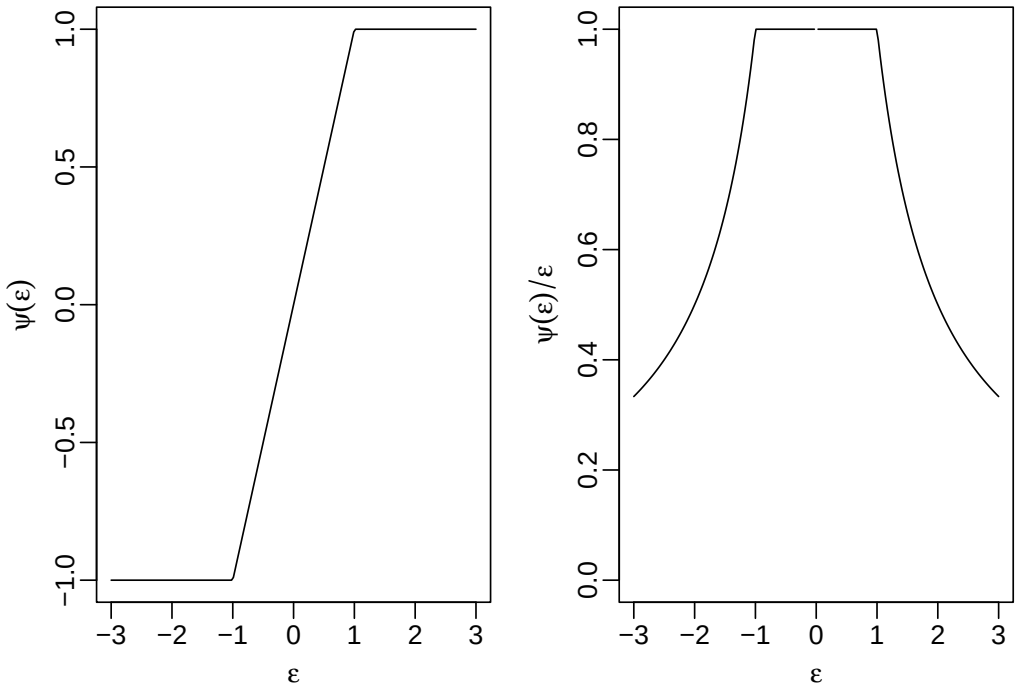
$$\rho(\varepsilon) = \begin{cases} \frac{1}{2}\varepsilon^2 & \text{if } |\varepsilon| \le t \\ t|\varepsilon| - \frac{1}{2}t^2 & \text{if } |\varepsilon| > t \end{cases}$$

The parameter $t > 0$ can be used to tune the method. The influence function ψ is given by

$$\psi(\varepsilon) = \begin{cases} \varepsilon & \text{if } |\varepsilon| \le t \\ t\,\text{sign}(\varepsilon) & \text{if } |\varepsilon| > t. \end{cases}$$

A plot of ψ and $\psi(\varepsilon)/\varepsilon$ shows that for this method the weight of observations where the residuals satisfy $|\varepsilon_i| \le t$ is the same as for least squares, but larger residuals get lower weights, leading to a more robust estimator.

```
t <- 1
psi <- function(eps) ifelse(abs(eps) <= t, eps, t * sign(eps))
par(mfrow = c(1, 2),
    mai = c(0.7, 0.7, 0.1, 0.1),
    mgp = c(1.7, 0.5, 0))
plot(eps, psi(eps), type="l",
     xlab = expression(epsilon), ylab = expression(psi(epsilon)))
plot(eps, psi(eps) / eps, type="l", ylim = c(0, 1),
     xlab = expression(epsilon),
     ylab = expression(psi(epsilon)/epsilon))
```



18.3.4 Hampel's Method

Here we have tuning parameters $0 < a < b < c$, and the objective function is

$$\rho(\varepsilon) = \begin{cases} \frac{1}{2}\varepsilon^2 & \text{if } |\varepsilon| \le a \\ a|\varepsilon| - \frac{1}{2}a^2 & \text{if } a \le |\varepsilon| < b \\ a\frac{(|\varepsilon|-c)^2}{2(b-c)} + \frac{1}{2}a(b+c-a) & \text{if } b \le |\varepsilon| \le c \\ \frac{a(b+c-a)}{2} & \text{otherwise,} \end{cases}$$

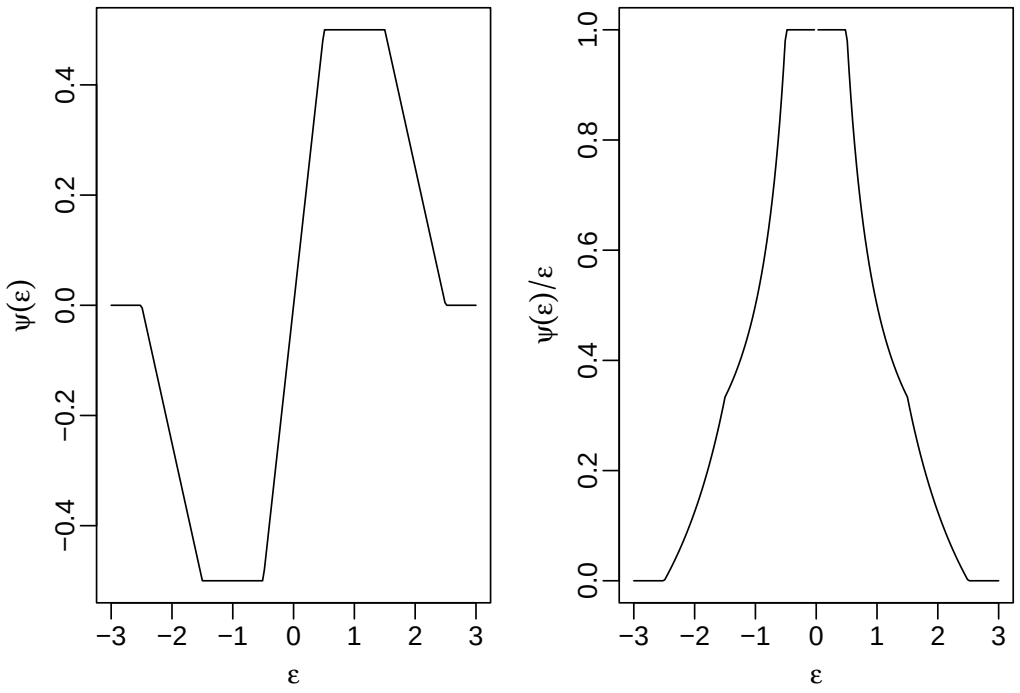
with derivative

$$\psi(\varepsilon) = \begin{cases} \varepsilon & \text{if } |\varepsilon| \le a, \\ a\,\text{sign}(\varepsilon) & \text{if } a < |\varepsilon| \le b, \\ a\frac{\varepsilon - c\,\text{sign}(\varepsilon)}{b-c} & \text{if } b < |\varepsilon| \le c, \\ 0 & \text{if } c < |\varepsilon|. \end{cases}$$

Since these formulas are tedious to understand, we consider plots of ψ and $\psi(\varepsilon)/\varepsilon$ again:

```
a <- 0.5
b <- 1.5
c <- 2.5
psi <- function(eps) {
    ifelse(abs(eps) <= a, eps,
           ifelse(abs(eps) <= b, a * sign(eps),
                  ifelse(abs(eps) <= c,
                         a * (eps - c*sign(eps)) / (b - c),
                         0)))
}
par(mfrow = c(1, 2),
    mai = c(0.7, 0.7, 0.1, 0.1),
    mgp = c(1.7, 0.5, 0))
plot(eps, psi(eps), type="l",
     xlab = expression(epsilon), ylab = expression(psi(epsilon)))
plot(eps, psi(eps) / eps, type="l", ylim = c(0, 1),
```

```
     xlab = expression(epsilon),
     ylab = expression(psi(epsilon)/epsilon))
```



We see that small residuals, with $|\varepsilon_i| \leq a$ get the same weight as for least squares regression, and the weight for larger residuals is gradually reduced. Residuals with $|\varepsilon_i| > c$ have weight 0 and do not affect the location of the regression line at all.

We see that ψ is not increasing and therefore ρ is not convex. A consequence of this fact is that the function s may have several local minima and care must be used when choosing the starting point of iterative minimisation methods.

18.3.5 Tukey's Bisquare Method

Here we have only one tuning parameter, $a > 0$, and the objective function is

$$\rho(\varepsilon) = \begin{cases} \frac{a^2}{6}\left(1 - \left(1 - (\varepsilon/a)^2\right)^3\right) & \text{if } |\varepsilon| \leq a \\ \frac{a^2}{6} & \text{if } |\varepsilon| > a \end{cases}$$

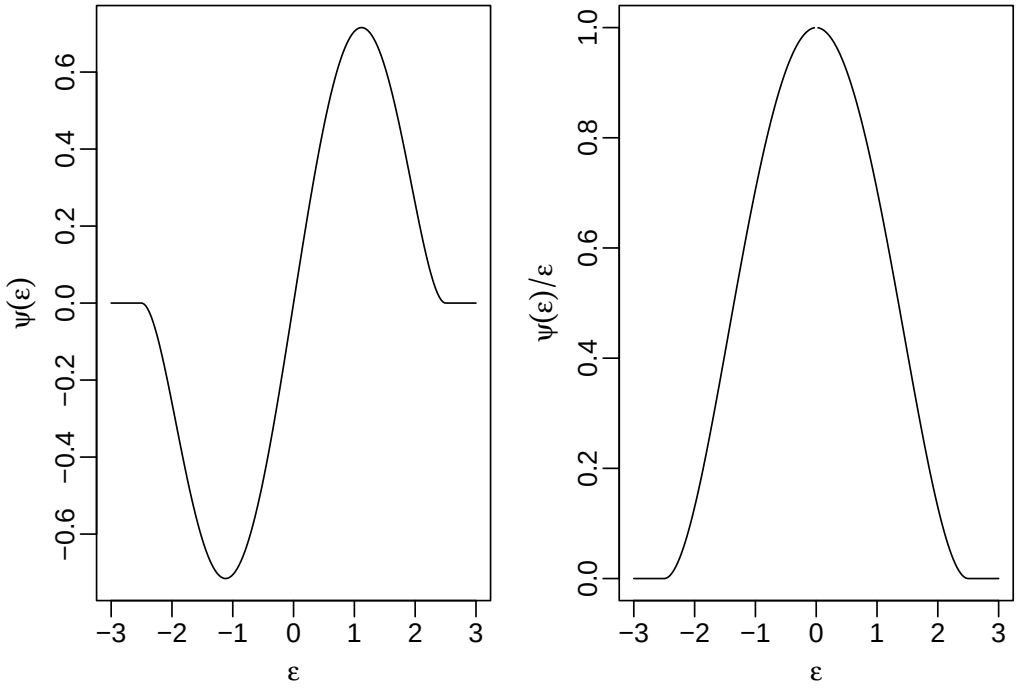
with derivative

$$\psi(\varepsilon) = \begin{cases} \varepsilon\left(1 - (\varepsilon/a)^2\right)^2 & \text{if } |\varepsilon| \leq a \\ 0 & \text{if } |\varepsilon| > a. \end{cases}$$

The plots show that this is a simplified variant of Hampel's estimator:

```
a <- 2.5
eps <- seq(-3, 3, length.out = 201)
psi <- function(eps) {
    ifelse(abs(eps) <= a,
           eps * (1 - (eps / a)^2)^2,
           0)
}
par(mfrow = c(1, 2),
    mai = c(0.7, 0.7, 0.1, 0.1),
    mgp = c(1.7, 0.5, 0))
plot(eps, psi(eps), type="l",
```

```
     xlab = expression(epsilon), ylab = expression(psi(epsilon)))
plot(eps, psi(eps) / eps, type="l", ylim = c(0, 1),
     xlab = expression(epsilon),
     ylab = expression(psi(epsilon)/epsilon))
```



Again, ρ is not convex and care is needed when computing the estimate $\hat{\beta}$.

Summary

- We introduced the M-estimator as a robust method for estimating regression coefficients.
- We discussed the iteratively re-weighted least squares method for computing M-estimates.
- We discussed different choices for the objective function of an M-estimator.

19 Efficiency of Robust Estimators

In theoretical statistics, **efficiency** is a measure of quality of an estimator. A more efficient estimator achieves smaller estimation error for a given number of samples. Here we give an informal introduction to the topic of efficiency and discuss the efficiency of different methods for linear regression.

19.1 Efficiency

In the section about the Mean Squared Error we have seen that

$$\mathrm{MSE}(\hat{\theta}) = \mathrm{Var}(\hat{\theta}) + \mathrm{bias}(\hat{\theta})^2$$

can be used as a measure for the estimation error of an estimator $\hat{\theta}$, and if the estimator is unbiased, this expression simplifies to

$$\mathrm{MSE}(\hat{\theta}) = \mathrm{Var}(\hat{\theta}).$$

Thus, "good" estimators will have small variance. Instead of considering the full definition from theoretical statistics (which requires the concept of Fisher information), here we simply consider

$$\mathrm{efficiency}(\hat{\theta}) = \frac{1}{\mathrm{Var}(\hat{\theta})}$$

as a measure for the efficiency of an estimator. We will use this measure of efficiency to compare different estimators.

Example 19.1. Both the sample mean and the sample median can be used as estimators for the population mean. Here we show that the median has lower efficiency than the mean.

Suppose $X_1, \dots, X_n$ are i.i.d. with CDF $F(x)$, *i.e.* $P(X_i \leq x) = F(x)$. As before, let $X_{(1)} \leq \cdots \leq X_{(n)}$ be the samples arranged in order of increasing value. Then the distribution of $X_{(k)}$ has CDF

$$P(X_{(k)} \leq x) = \sum_{i=k}^{n} \binom{n}{i} F(x)^i (1 - F(x))^{n-i},$$

since $X_{(k)} \leq x$ requires at least k out of n samples to be less than or equal to x. If we differentiate this (using product rule and chain rule) we get the density of $X_{(k)}$:

$$\begin{aligned} f_{(k)}(x) &= \frac{d}{dx} P(X_{(k)} \leq x) \\ &= \cdots \\ &= f(x) \frac{n!}{(n-k)!(k-1)!} F(x)^{k-1} (1 - F(x))^{n-k}. \end{aligned}$$

For simplicity we assume that n is odd, say $n = 2m + 1$. In this case the median is $X_{\text{median}} = X_{(m+1)}$. Thus, the density of the median is

$$\begin{aligned} f_{\text{median}}(x) &= f_{(m+1)}(x) \\ &= f(x) \frac{n!}{(n-m-1)!m!} F(x)^m (1 - F(x))^{n-m-1} \\ &= f(x) \frac{n!}{m!m!} F(x)^m (1 - F(x))^m. \end{aligned}$$

This density can be used to understand the behaviour of the median.

Now assume that $X_1, \ldots, X_n \sim \mathcal{N}(\mu, \sigma^2)$, *i.e.* the density is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big)$$

and the CDF is

$$F(x) = \Phi\Big(\frac{x-\mu}{\sigma}\Big),$$

where Φ is the CDF of the standard normal distribution. In this case we know that

$$\mathrm{Var}(\overline{X}) = \frac{\sigma^2}{n}.$$

For comparison, we can use the density of X_{median} to show (in a series of complicated steps) that

$$\mathrm{Var}(X_{\text{median}}) \sim \frac{\pi}{2}\frac{\sigma^2}{n},$$

as $n \to \infty$. Thus, for large n, the relative efficiency of the median compared to the mean is

$$\begin{aligned}\frac{\text{efficiency}(X_{\text{median}})}{\text{efficiency}(\overline{X})} &= \frac{\mathrm{Var}(\overline{X})}{\mathrm{Var}(X_{\text{median}})} = \frac{\frac{\sigma^2}{n}}{\frac{\pi}{2}\frac{\sigma^2}{n}} \\ &= \frac{2}{\pi} = 0.637.\end{aligned}$$

Thus, the sample median is a less efficient estimator for the mean than the sample mean is.

The situation in the example carries over to linear regression: more robust estimators tend to be less efficient, and there is a trade-off between robustness and efficiency. Without proof, here we list some examples of this principle:

- One can show that usually the variance of the least squares estimator decreases proportionally to $1/n$, and thus the efficiency of the least squares estimator increases proportionally to n.
- The value t in Huber's method can be used to control the balance between robustness and efficiency. As $t \to \infty$, the method converges to least squares regression and becomes less robust, but at the same time efficiency increases. Huber suggests the value $t = 1.345\sigma$ and showed that for this choice the method is 95% efficient for large n, compared to least squares regression.
- Similarly, the parameter a in Tukey's Bisquare Method controls the trade-off between robustness and efficiency, with smaller a leading to a more robust method. In the literature it is suggested that $a = 4.685\sigma$ leads to 95% efficiency for large n, compared to least squares regression.
- The efficiency of the least absolute values estimator can be shown to be 64%.

19.2 Robust estimators

Since the M-estimator only addresses y-space outliers, the breakdown point of M-estimation is still $1/n$. We now consider two estimators which have a higher breakdown point, but lower efficiency.

19.2.1 Least Median of Squares

Rather than minimise the sum of the residual sum of squares (or a weighted version of it), this estimator minimises the median of squared residuals. The estimate is given by

$$\widehat{\beta}^{(\text{LMS})} := \arg\min_{\beta} \operatorname*{median}_{i\in\{1,\ldots,n\}} \big(y_i - x_i^T\beta\big)^2,$$

where $x_1, \ldots, x_n$ are the rows of the design matrix X. This is very robust with respect to outliers, both in x-direction and y-directions: Since the median is used instead of the sum, up to half of the squared residuals can increase to infinity while the estimate stays bounded. Thus the asymptotic breakdown point of the method is $1/2$.

The least median of squares method has poor asymptotic efficiency $n^{2/3}$. In the limit, the relative efficiency compared to ordinary least squares is

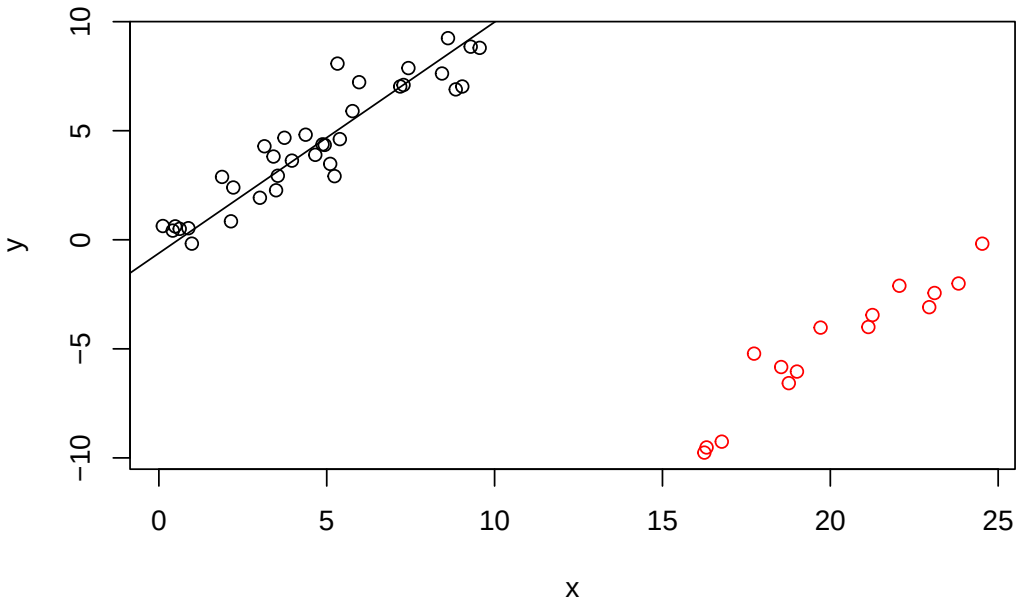
$$\frac{\text{efficiency}(\hat{\beta}^{(\text{LMS})})}{\text{efficiency}(\hat{\beta}^{(\text{LSQ})})} \propto \frac{n^{2/3}}{n} \longrightarrow 0$$

as $n \to \infty$.

In R, the LMS estimate can be computed using the function `lqs(..., method = "lms")` function from the `MASS` library.

Example 19.2. The following code computes an LMS regression estimate. We introduce artifical outliers by shifting 30% of the data to the bottom right. These artificial outliers are represented by the red circles in the plot.

```
library("MASS") # for lqs()

set.seed(20211207)
n <- 50
x <- runif(n, 0, 10)
y <- x + rnorm(n)

# add 30% outliers
n.ol <- floor(0.3*n)
idx <- sample.int(n, n.ol)
x[idx] <- x[idx] + 15
y[idx] <- y[idx] - 10

m <- lqs(y~x, method="lms")
plot(x, y, col = ifelse(1:n %in% idx, "red", "black"))
abline(m)
```



19.2.2 Least Trimmed Squares

This takes as its objective function the sum of h smallest squared residuals and was proposed as a remedy to the low asymptotic efficiency of LMS. The least trimmed squares estimator $\widehat{\beta}^{(\mathrm{LTS})}$ is defined as

$$\widehat{\beta}^{(\mathrm{LTS})} := \arg\min_{\beta} \sum_{i=1}^{k} r^2_{[i]}(\beta),$$

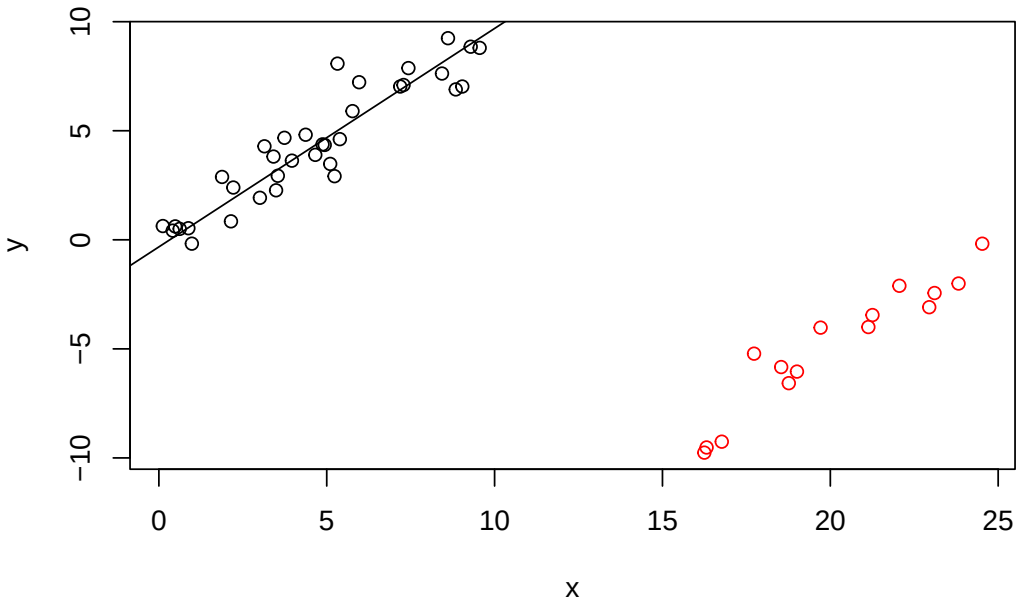
where $r^2_{[i]}(\beta)$ represents the ith smallest value amongst $r_i(\beta)^2 = \left(y_i - x_i^T\beta\right)^2$.

The value k controls the trade-off between robustness and efficiency and the value must satisfy $n/2 < k \leq n$. For a given k the method can tolerate $n-k$ outliers. The bounday case $k = n$ corresponds to the ordinary least squares method. The breakdown point of the method is $(n-k+1)/n$.

Computing the LTS estimate is a non-trivial problem, which involves fitting the least-squares estimate to a carefully chosen subset of the samples. In R, the LTS estimate can be computed using the function `lqs(..., method = "lts")` function from the `MASS` library.

Example 19.3. The following code computes an LTS regression estimate. The data is the same as in the previous example.

```
m <- lqs(y~x, method = "lts", quantile = floor(n/2)+1)
plot(x, y, col = ifelse(1:n %in% idx, "red", "black"))
abline(m)
```



One can show that LTS has an efficiency of approximately 0.08 compared to ordinary least squares regression.

Summary

- In this section we have informally discussed the efficiency of different estimators.
- We have introduced the Least Median of Squares estimator.
- We have introduced the Least Trimmed Squares estimator.

A Linear Algebra Reminders

A.1 Vectors

We write $v \in \mathbb{R}^d$ if $v = (v_1, \dots, v_d)$ for numbers $v_1, \dots, v_d \in \mathbb{R}$. We say that v is a d-dimensional vector, and $\mathbb{R}^d$ is the d-dimensional Euclidean space. Vectors are often graphically represented as "column vectors":

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix}.$$

If $u, v \in \mathbb{R}^d$ are two vectors, the **inner product** of u and v is given by

$$u^\top v = \sum_{i=1}^{d} u_i v_i. \tag{39}$$

Note that the two vectors must have the same length for the inner product to exist.

Using this notation, the **Euclidean length** of a vector v can be written as

$$\|v\| = \sqrt{\sum_{i=1}^{d} v_i^2} = \sqrt{v^\top v}.$$

Vectors $v_1, \dots, v_n$ are said to be **orthogonal**, if $v_i^\top v_j = 0$ for all $i \neq j$. The vectors are said to be **orthonormal**, if they are orthogonal and satisfy $\|v_i\| = 1$ for all i.

A.2 Matrices

We write $A \in \mathbb{R}^{m \times n}$ if

$$A = \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ a_{2,1} & \dots & a_{2,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{pmatrix},$$

where $a_{i,j}$, sometimes also written as a_{ij} are numbers for $i \in \{1, \dots, m\}$ and $j \in \{1, \dots, n\}$.

A.2.1 Transpose

If $A \in \mathbb{R}^{m \times n}$, then the **transpose** of A is the matrix $A^\top \in \mathbb{R}^{n \times m}$, with $(A^\top)_{ij} = a_{ji}$ for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$. Graphically, this can be written as

$$A^\top = \begin{pmatrix} a_{1,1} & a_{2,1} & \dots & a_{m,1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \dots & a_{m,n} \end{pmatrix},$$

Definition A.1. A matrix A is called **symmetric**, if $A^\top = A$.

A.2.2 Matrix-vector Product

If $A \in \mathbb{R}^{m \times n}$ and $v \in \mathbb{R}^n$, then $Av \in \mathbb{R}^m$ is the vector with

$$(Av)_i = \sum_{j=1}^{n} a_{ij} v_j$$

for all $i \in \{1, \dots, m\}$.

If we consider v to be a $(n \times 1)$-matrix instead of a vector, Av can also be interpreted as a matrix-matrix product between an $m \times n$ and an $n \times 1$ matrix. Using this convention, $v^\top$ is then interpreted as an $1 \times n$ matrix and if $u \in \mathbb{R}^m$ we have $u^\top A \in \mathbb{R}^{1\times n} \cong \mathbb{R}^n$ with

$$(u^\top A)_j = \sum_{i=1}^{m} u_i a_{ij}$$

for all $j \in \{1, \dots, n\}$. Going one step further, this notation also motivates the expression $u^\top v$ in equation (39).

A.2.3 Matrix-matrix Product

If $A \in \mathbb{R}^{\ell\times m}$ and $B \in \mathbb{R}^{m\times n}$, then $AB \in \mathbb{R}^{\ell\times n}$ is the matrix with

$$(AB)_{ik} = \sum_{j=1}^{m} a_{ij} b_{jk}$$

for all $i \in \{1, \dots, \ell\}$ and $j \in \{1, \dots, n\}$. This is called the **matrix product** of A and B. Note that A and B must have compatible shapes for the product to exist.

Properties:

- The matrix product is associative: if A, B and C are matrices with shapes such that AB and BC exist, then we have $A(BC) = (AB)C$. It does not matter in which order we perform the matrix products here.
- The matrix product is transitive: if A, B and C have the correct shapes, we have $A(B + C) = AB + AC$.
- The matrix product is *not* commutative: if AB exists, in general A and B don't have the correct shapes for BA to also exist, and even if BA exists, in general we have $AB \neq BA$.
- Taking the transpose swaps the order in a matrix product: we have

$$(AB)^\top = B^\top A^\top \tag{40}$$

A.2.4 Rank

Definition A.2. The **rank** of a matrix $A \in \mathbb{R}^{m\times n}$ is the dimension of the image space of A:

$$\operatorname{rank}(A) = \dim\{Av \mid v \in \mathbb{R}^n\}.$$

The rank can also be characterised as the largest number of linearly independent columns of A, or the largest number of linearly independent rows of A. Thus, for $A \in \mathbb{R}^{m\times n}$ we always have $\operatorname{rank}(A) \leq \min(m, n)$. The matrix A is said to have "full rank" if $\operatorname{rank}(A) = \min(m, n)$.

A.2.5 Trace

Definition A.3. The **trace** of a matrix $A \in \mathbb{R}^{n\times n}$ is given by

$$\operatorname{tr}(A) = \sum_{i=1}^{n} a_i i.$$

Properties:

- $\operatorname{tr}(A + B) = \operatorname{tr}(A) + \operatorname{tr}(B)$
- $\operatorname{tr}(A^\top) = \operatorname{tr}(A)$
- $\operatorname{tr}(ABC) = \operatorname{tr}(BCA) = \operatorname{tr}(CAB)$. The individual matrices A, B, C don't need to be square for this relation to hold, but the relation only holds for cyclic permutations as shown. In general $\operatorname{tr}(ABC) \neq \operatorname{tr}(ACB)$.

A.2.6 Matrix Inverse

If A is a square matrix and if there is a matrix B such that $AB = I$, then A is called **invertible** and the matrix B is called the **inverse** of A, denoted by $A^{-1} := B$. Some important properties of the inverse:

- The inverse, if it exists, is unique.
- Left-inverse and right-inverse for matrices are the same: $A^{-1}A = I$ holds if and only if $AA^{-1} = I$.
- If A is symmetric and invertible, then A^{-1} is also symmetric. This is true because $A(A^{-1})^\top = (A^{-1}A)^\top = I^\top = I$ and thus $(A^{-1})^\top$ is an inverse of A. Since the inverse is unique, $(A^{-1})^\top = A^{-1}$.

Theorem A.1. *Let $A \in \mathbb{R}^{n\times n}$. Then the following statements are equivalent:*

a. A is invertible
b. A has full rank, i.e. $\operatorname{rank}(A) = n$.
c. The equation $Ax = 0$ has $x = 0$ as its only solution.
d. The equation $Ax = b$ has exactly one solution x for every $b \in \mathbb{R}^n$.

A.2.7 Orthogonal Matrices

Definition A.4. A matrix U is called **orthogonal**, if $U^\top U = I = UU^\top$.

Some important properties of orthogonal matrices:

- If U is orthogonal, the inverse and the transpose are the same: $U^\top = U^{-1}$.
- We have $\|Ux\|^2 = x^\top U^\top Ux = x^\top x = \|x\|^2$. Thus, multiplying a vector x by an orthogonal matrix U does not change its length.

A.2.8 Positive Definite Matrices

Definition A.5. A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is called **positive definite**, if

$$x^\top Ax > 0$$

for all $x \in \mathbb{R}^n$ with $x \neq 0$. The matrix is called **positive semi-definite**, if

$$x^\top Ax \geq 0$$

for all $x \in \mathbb{R}^n$.

A.2.9 Idempotent Matrices

Definition A.6. The matrix A is **idempotent**, if $A^2 = A$.

A.3 Eigenvalues

Definition A.7. Let $A \in \mathbb{R}^{n\times n}$ be a square matrix and $\lambda \in R$. The number λ is called an **eigenvalue** of A, if there exists a vector $v \neq 0$ such that $Ax = \lambda x$. Any such vector x is called an **eigenvector** of A with eigenvalue λ.

While there are very many results about eigenvectors and eigenvalues in Linear Algebra, here we will only use a small number of these results. We summarise what we need for this module:

- If A is idempotent and x is an eigenvector with eigenvalue λ, then we have $\lambda x = Ax = A^2x = \lambda Ax = \lambda^2 x$. Thus we have $\lambda^2 = \lambda$. This shows that the only eigenvalues possible for idempotent matrices are 0 and 1.

Theorem A.2. *Let $A \in \mathbb{R}^{n \times n}$ be symmetric. Then there is an orthogonal matrix U such that $D := UAU^\top$ is diagonal. The diagonal elements of D are the eigenvalues of A and the rows of U are corresponding eigenvectors.*

B Probability Reminders

B.1 Independence

Definition B.1. Two random variables X and Y are (statistically) **independent**, if $P(X \in A, Y \in B) = P(X \in A)P(Y \in B)$ for all sets A and B.

We list some properties of independent random variables:

- If X and Y are independent, and if f and g are functions, then $f(X)$ and $g(Y)$ are also independent.

B.2 The Chi-Squared Distribution

Definition B.2. Let $X_1, \dots, X_\nu \sim \mathcal{N}(0,1)$ be i.i.d. Then the distribution of $\sum_{i=1}^{\nu} X_i^2$ is called the χ^2-distribution with ν degrees of freedom. The distribution is denoted by $\chi^2(\nu)$.

Some important properties of the χ^2-distribution are:

- χ^2-distributed random variables are always positive.
- If $Y \sim \chi^2(\nu)$, then $\mathbb{E}(Y) = \nu$ and $\mathrm{Var}(Y) = 2\nu$.
- The R command `pchisq(`x`,`ν`)` gives the value $\Phi_\nu(x)$ of the CDF of the $\chi^2(\nu)$-distribution.
- The R command `qchisq(`α`,`ν`)` can be used to obtain the α-quantile of the $\chi^2(\nu)$-distribution.
- More properties can be found on Wikipedia.

B.3 The t-distribution

Definition B.3. Let $Z \sim \mathcal{N}(0,1)$ and $Y \sim \chi^2(\nu)$ be independent. Then the distribution of

$$T = \frac{Z}{\sqrt{Y/\nu}} \tag{41}$$

is called the t-distribution with ν degrees of freedom. This distribution is denoted by $t(\nu)$.

Some important properties of the t-distribution are:

- The t-distribution is symmetric: if $T \sim t(\nu)$, then $-T \sim t(\nu)$
- If $T \sim t(\nu)$, then $\mathbb{E}(T) = 0$.
- The R command `pt(`x`,`ν`)` gives the value $\Phi_\nu(x)$ of the CDF of the $t(\nu)$-distribution.
- The R command `qt(`α`,`ν`)` can be used to obtain the α-quantile of the $t(\nu)$-distribution.
- More properties can be found on Wikipedia.