

Statistical Modeling:

An Excursion Through 14 Topics

Reinhard Furrer
and the Applied Statistics Group

Version September 11, 2023
Git b91ab56

Contents

Preface

v

1 Resampling and Model Validation	1
1.1 What is a Statistical Model?	1
1.2 Sampling	5
1.3 Cross-Validation	6
1.4 Bootstrap	13
1.5 Example: Description of Antarctic Ice Shelves	16
1.6 Bibliographic remarks	20
1.7 Exercises and Problems	20
2 Principal Component Analysis	25
2.1 Finding some Representative Dimensions	25
2.2 PCA as a functional Decomposition: EOF	30
2.3 Theoretical Aspects of PCA	31
2.4 PCA and Related Decompositions and Concepts	35
2.5 Example: Sea-Level Pressure Over the Northern Atlantic	43
2.6 Bibliographic remarks	47
2.7 Exercises and Problems	48
3 Clustering	51
3.1 Hierarchical Clustering	51
3.2 The k -Means Algorithm	59
3.3 Model-Based Clustering	63
3.4 Longitudinal Clustering	65
3.5 Example: A Bioinformatics Data Analysis	69
3.6 Bibliographic remarks	74
3.7 Exercises and Problems	75
4 Discrimination and Classification	77
4.1 Classification in the Gaussian Case	77
4.2 Classification Trees	85
4.3 Bagging, Boosting and More	86

4.4	Machine Learning and Data Mining	93
4.5	Example: Optical Character Recognition	94
4.6	Bibliographic remarks	96
4.7	Exercises and Problems	98
5	Linear Models	99
5.1	Multiple Regression	100
5.2	Analysis of Variance	106
5.3	Extensions of the Linear Model	108
5.4	Bibliographic remarks	112
5.5	Exercises and Problems	113
6	Mixed Models	115
6.1	Motivation	115
6.2	Mixed Effects Models	116
6.3	Linear Mixed Models for Longitudinal Data	118
6.4	Additional Examples	126
6.5	Bibliographic remarks	130
6.6	Exercises and Problems	133
7	Non-Parametric Regression	135
7.1	Motivation	135
7.2	Kernel Approaches	137
7.3	Splines	141
7.4	Locally Adaptive Approaches	145
7.5	Example: Image Compression	148
7.6	Bibliographic remarks	153
7.7	Exercises and Problems	153
8	Generalized Linear Models	155
8.1	Motivation	156
8.2	Logistic Regression	157
8.3	Poisson Regression	163
8.4	Generalized Linear Models	166
8.5	Bibliographic remarks	168
8.6	Exercises and Problems	168
9	Survival Analysis	171
9.1	Censoring	172
9.2	Survivor and Hazard Function	172
9.3	Non-Parametric Methods for Survival Data	175
9.4	Parametric Modeling of Survival Data	180
9.5	Bibliographic remarks	187
9.6	Exercises and Problems	187

10 Time Series	189
10.1 Introduction	189
10.2 ARMA Models	191
10.3 ACF and PACF and Their Estimation	192
10.4 Prediction	196
10.5 Example: Car Drivers Accidents	198
10.6 Bibliographic remarks	200
10.7 Exercises and Problems	201
11 Spatial Statistics	203
11.1 Introduction	203
11.2 Geostatistical Data	204
11.3 Lattice Data	216
11.4 Bibliographic remarks	216
11.5 Exercises and Problems	217
12 Extreme Value Theory	219
12.1 Motivation	219
12.2 Modeling the Maxima	221
12.3 Modeling Peaks over Thresholds	223
12.4* Comparing the Approaches Through Profile Estimates	228
12.5 Bibliographic remarks	229
12.6 Exercises and Problems	229
13 Neural Networks and Deep Learning	231
13.1 Learning as in Machine Learning	232
13.2 Concept of Neural Networks	235
13.3 Deep Learning	241
13.4 Tensorflow and Keras	244
13.5 Bibliographic remarks	250
13.6 Exercises and Problems	251
Postscript	253
A Calculus	255
A.1 Functions	255
A.2 Functions in Higher Dimensions	258
A.3 Approximating Functions	259

B Linear Algebra	261
B.1 Vectors, Matrices and Operations	261
B.2 Linear Spaces and Basis	262
B.3 Projections	263
B.4 Matrix Decompositions	264
B.5 Positive Definite Matrices	265
C Random Variables	267
C.1 Basics of Probability Theory	267
C.2 Discrete Distributions	268
C.3 Continuous Distributions	270
C.4 Expectation, Variance and Moments	272
C.5 Independent Random Variables	273
C.6 Common Discrete Distributions	274
C.7 Common Continuous Distributions	275
C.8 Functions of Random Variables	281
D Multivariate Normal Distribution	285
D.1 Random Vectors	285
D.2 Multivariate Normal Distribution	287
References	293
Glossary	299
Dataset Index	303
Command Index	305
Package Index	307
Video Index	309

Preface

This document accompanies the lecture *STA121 Statistical Modeling* that is read each fall semester since 2013. The lecture is given in the framework of the minor in *Applied Probability and Statistics* (www.math.uzh.ch/aws) and comprises 14 weeks of two hours of lecture and one hour of exercises per week. As the lecture’s topics are structured on a week by week basis, the script contains thirteen chapters, each covering “one” topic.

The lecture builds upon the module *STA120 Introduction to Statistics* and thus we do not start from scratch but may refer to relevant chapters. The script of *STA120 Introduction to Statistics* is available at [www.math.uzh.ch /furrer/download/sta120/](http://www.math.uzh.ch/furrer/download/sta120/). Additionally, the appendix gives a concise summary of relevant concepts, definitions and results in calculus, linear algebra and probability theory. The multivariate Gaussian distribution is used quite extensively and we present its properties in a separate, last appendix.

I have thought long and hard about an optimal structure for this script. Let me quickly summarize my thoughts. It is very important that the document contains a structure that is tailored to the content I cover(ed) in class each week. This inherently leads to 13 “chapters” (last week is a review week). Instead of covering linear models over four weeks, I have cast the material in four *seemingly* different chapters. This structure helped me to better frame the lectures: each week having a start, a set of learning goals and a predetermined end.

So to speak, the script covers not fourteen but essentially only three topics:

1. Background and foundations;
2. Unsupervised models;
3. Supervised models.

Dedicating two hours to each “new topic” does not allow us to discuss the subjects in depth. Notice that there are often entire lectures based on the topics we cover (‘Generalized Regression’, ‘Time Series Analysis’, ‘Survival Analysis’, ‘Modeling Dependent Data’, …). I have therefore augmented this script with short video sequences giving additional – often more technical – insight. These videos are indicated in the margins with a ‘video’ symbol as here.



At the beginning of each chapter, there is a direct link to an R file, that contains all the code to re-create the analysis and figures of the chapter. Moreover, some of these files contain additional very short illustrations, marked with **### Code not shown in script**. To not clutter

the script with plenty of `par()` commands, we do not include them here and in the available R files. Datasets that are not part of regular CRAN packages are available at www.math.uzh.ch/furrer/download/sta121/. In the electronic version of the document there is, whenever possible, a link to the corresponding data set.

The script uses as “classical” notation consistently as possible: upper case for random quantities, lower case for realizations, boldface for vectors (lower case) and matrices (upper case). Inherently, a matrix can not be differentiated from a random vector based on notation only, see also Appendix B. The context and, if necessary, explanations will clarify. We have summarized other notation in a glossary at the end of the script.

Many have contributed to this document: Julia Braun (chapters ‘Survival Analysis’ and ‘Mixed Models’), David Masson (chapter ‘Principal Component Analysis’), Zofia Baranczuk (reading the class and through assisting the exercises), Florian Gerber, Gilles Kratzer, Marta Pittavino and Simone Tiberi (through assisting the exercises), Kelly Reeve (English), Ruben Scherrer (general comments and suggestions), Carolin Strobl and Eva Furrer (through critical suggestions) and Franziska Robmann (for administrative support). Without their help, you would not be reading these lines. No textbook material is perfect. Please let me know of any necessary improvements and I highly appreciate all forms of contributions in form of errata, examples, or text blocks. Contributions can be deposited directly in the following [Google Doc sheet](#).

Major errors that were corrected *after* the lecture of the corresponding semester are listed www.math.uzh.ch/furrer/download/sta121/errata.txt. I try hard that after the lecture, the pagination of the document does not change anymore.

Reinhard Furrer
Summer 2023

Chapter 1

Resampling and Model Validation

This chapter covers two closely related techniques based on sampling. Cross-validation is a resampling technique to tune methods, balancing bias and variance contributions towards a small mean squared error.

Closed form expressions of distributions of estimators are rarely known. Bootstrapping is a resampling technique to approximate distributions based on an empirical sample.

Both tools are illustrated under the umbrella of model validation.

Learning goals for this chapter:

- ◊ Understand the conceptual idea of statistical modeling and workflow
- ◊ Understand how resampling can help us in understanding variability of estimators
- ◊ Understand how prediction error helps us understand how well a prediction model fits the data
- ◊ Explain and apply cross-validation
- ◊ Understand the leave-one-out statistics and leave-one-out cross-validation and its applications
- ◊ Explain and apply bootstrap

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter01.R.

1.1 What is a Statistical Model?

We are often interested in describing a certain phenomena or in predicting a future or unknown state. Typically we do not have a precise understanding of the physical underlying processes or

mechanisms and thus we have no means of a precise (error free) description of the system. In many situations we can observe the system or state in a similar setting, at a different time or location etc. These observations lead to data that are then used with inferential (describing) or predictive statistical tools to get a statistical answer to our questions about the unobserved system. The statistical tools are essentially models, a description of the unknown state where all complex or unknown elements are captured with by stochastic elements.

Example 1.1. Physical laws allow to formally describe how a tossed coin moves and under ideal settings what side should show up. In practical settings, such a physical description is too complex and we simply model the result with using a Bernoulli random variable. ♣

A statistical model tries to mimic reality and hence, “Essentially, all models are wrong, but some are useful” ([Box and Draper, 1987](#) page 424), leading to one of the most cited quotes in statistics.

The question is thus how to measure usefulness of a statistical model. Figure 1.1 illustrates the general, abstract flow of a statistical analysis, starting from a hypothesis and ending at the communication of the scientific conclusion. The “usefulness” of a model determines if we should proceed after ‘Model validation’ or move back to ‘Propose an (alternative) statistical model’ (back loop in Figure 1.1). In this chapter we discuss several ideas of measuring usefulness, whereas many of the remaining chapters will cover different models.

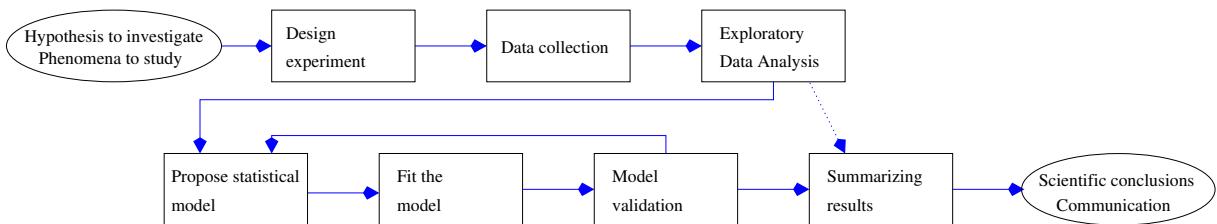


Figure 1.1: Representation of a statistical workflow. In an ideal setting, a statistician should be involved in all the rectangular boxes.

The following example illustrates one of the most important concepts of model fit.

Example 1.2. Figure 1.2 shows the annual bituminous coal production in the USA between 1920 and 1960 ([Makridakis et al., 1998](#), Exercise 7.7). Three different models have been used to describe the data and the resulting fits are added in the panel. The black line is the linear fit of a linear regression, describing the consumption overall with a possible linear change over the years. The other two “follow” the data much better, almost to a degree of interpolation, loosing any predictive capabilities. Predictions based on the green and blue fit outside the observed time span would be very delicate.

Depending on the precise research question, different models are useful. We come back to this example in Chapter 7, where we discuss in details how the fits have been constructed.

As a footnote, all three models are “optimal” in a certain view. ♣

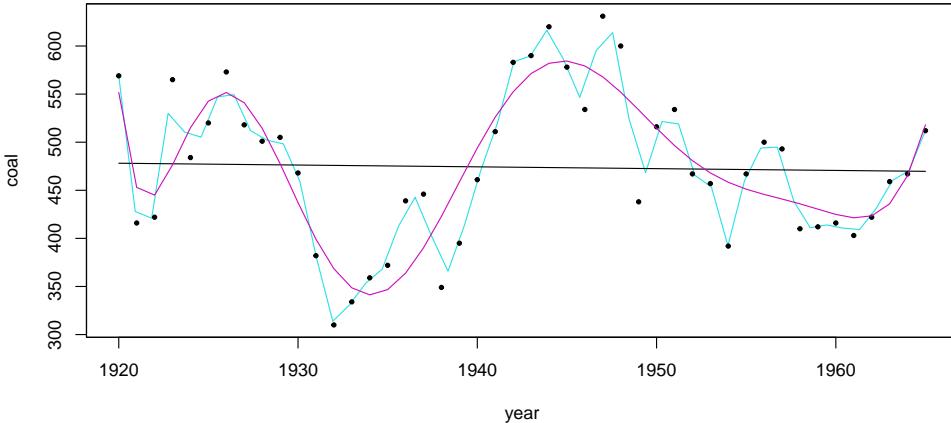


Figure 1.2: Dataset `bicoal` (from the R-package `fma`) with three model fits superimposed. The black line is a simple linear regression fit. The magenta line represents a polynomial model with 11 parameters. Models underlying the cyan fit are so-called non-parametric models and are discussed in Chapter 7.

Suppose that a data collection yields the observations $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$, where y_i is a response variable and \mathbf{x}_i is a covariate with, say, p components. For the sake of illustration, we propose the statistical model

$$Y = g(\mathbf{x}) + \varepsilon, \quad (1.1)$$

where g is some function of (known) covariate \mathbf{x} and ε represents a zero mean white noise (i.e., iid) and with variance σ^2 . For the discussion here, it is not necessary to further discuss properties of g or the distribution of ε . Note, however, that the proposal entails the exploratory data analysis (EDA), as shown in Figure 1.1.

Based on the observations $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ the goal is to predict g from our hypothesized model (1.1), i.e., to find \hat{g} . Of course the technique and approach “find” depends on the details of the precise model assumptions, on the one hand on the g and the distribution of ε . The first few chapters assume almost exclusively an iid setting, moreover $\varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. In later chapters, foremost Chapters 10 to 12, we relax this assumption. The function g is often specified up to some unknown parameter vector $\boldsymbol{\theta}$ and “finding” the function \hat{g} means finding the parameter $\hat{\boldsymbol{\theta}}$. Of course, statisticians or scientists may specify the function g different or may use different approaches to obtain the associated parameter.

In such a parametric model, with iid noise, the estimation of parameters is straightforward (based on a method of moments approach like least squares or a likelihood setting), and we have a good understanding of the theoretical properties of the estimators in many situations. The estimation of “arbitrary” functions g as represented by the cyan line in Figure 1.2 will be discussed in Chapter 7.

Independent of the precise fitting approach and based on weak assumptions only, it is possible to decompose the mean squared error of the prediction $\hat{g}(\mathbf{x})$ into

$$\mathbb{E}((Y - \hat{g}(\mathbf{x}))^2) = (\mathbb{E}(\hat{g}(\mathbf{x})) - g(\mathbf{x}))^2 + \mathbb{E}((\hat{g}(\mathbf{x}) - \mathbb{E}(\hat{g}(\mathbf{x})))^2) + \mathbb{E}((Y - g(\mathbf{x}))^2) \quad (1.2)$$

$$= \text{bias}(\hat{g}(\mathbf{x}))^2 + \text{Var}(\hat{g}(\mathbf{x})) + \sigma^2, \quad (1.3)$$

i.e., a decomposition in positive terms consisting of squared bias, variance and variance of the noise. The latter is inherently present, that is irreducible, and cannot be reduced by a different model. However, different choices of \hat{g} lead to smaller or larger squared bias and variance term but it is not possible to simultaneously reduce both arbitrarily.

In general, complex model capture more of the structure in the data, i.e., smaller residuals. On the other hand residuals of simpler models are larger. Assuming the truth $g(\mathbf{x})$ to be reasonably smooth, only a reasonably complex model is necessary. For example, the black regression line in Figure 1.2 is has low model complexity and high bias as it does not capture g well. The variability of the curve is low because $\text{Var}(\hat{g}(\mathbf{x}))$ is with respect to the prediction error. On the other hand, the magenta line of Figure 1.2 is has high model complexity, captures the trend well (low $\text{bias}(\hat{g}(\mathbf{x}))^2$) but does not have a prediction variance as high as the cyan fit. With such an two component bias-variance interpretation, we do not properly quantify the three terms in (1.3) of course: the irreducible noise term is unknown.

An optimal model is somewhere in between a simple and a complex model. The bias–variance trade-off is illustrated in Figure 1.3. Using a criteria like the mean squared prediction error, we can find “optimal models”.

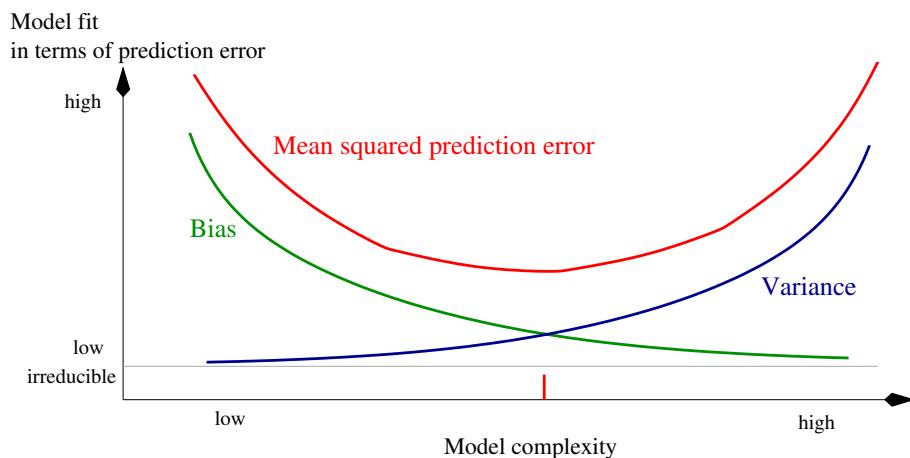


Figure 1.3: Schematic illustration of the bias–variance trade-off in statistical model fitting. The gray line indicates the irreducible error. The optimal model (based on mean squared prediction error) is indicated with a red tick; models on the left thereof underfit, on the right they overfit.

We use the term *overfitting* when our model captures part of the noise in the data. Such a model typically has a low bias and high variance. The model captures the available data (too) well and does not generalize to new and unobserved cases. An example is an interpolation, where connected line segments pass through all points. The opposite is called *underfitting* and represents the case where our model is not sufficiently flexible, is too simplistic. An example is a simple constant line.

Independent of the type of statistical models or the fitting procedure, model validation is crucial. In this chapter we use multiple regression as a model, to illustrate different validation techniques. Subsequent chapters will illustrate further models for which the tools discussed here apply accordingly.

1.2 Sampling

In statistics, the term *sampling* is quite ubiquitous and may have different meanings according to the context. In its basic meaning, sampling is the process of selecting a subset of individuals from a population in order to make inference about the whole population. Of course the discussion of the process of selecting fills entire books. The most important one is simple random sampling, where each individual has the same probability of being chosen. The situation is getting delicate if we have to sample from two different populations to compare a trait or characteristic between the two; especially if the two populations have a different size or if the trait depends on other characteristics of the individual.

In this chapter, we consider sampling as the process of generating “new” data based on observations or other existing data and possibly on some additional statistical assumptions. Here, we do not collect more data from individuals of a population or conduct further experiments, but we assume a hypothetical distribution of the “population”. Later we discuss sampling, when we do not even have such a hypothetical distribution and generate new samples based on the observed one only (like in a permutation test framework).

Many simulation studies in statistics are based on such a sampling principle. Starting from a hypothesized model, we repeatedly draw samples thereof and derive sample properties or statistics from the samples. These statistics represent empirical distributions of often intractable theoretical ones. The following example illustrates how sampling can be used to determine the distribution of a statistic.

In R we draw the samples by generating (pseudo) random numbers with appropriate functions, like `rnorm()`, `rpois()`, `runif()` etc. Conceptually, this is equivalent as having an infinite population and selecting random individuals thereof.

Example 1.3. The variance estimate $\hat{\sigma}^2 = 1/n \sum_{i=1}^n (y_i - \bar{y})^2$ is biased. Therefore, we typically use $s^2 = 1/(n-1) \sum_{i=1}^n (y_i - \bar{y})^2$. But what is the (empirical) variance of $\hat{\sigma}^2$ or even the distribution of associated estimator for a fixed sample size n ?

For large n (with almost no additional assumptions on the original distribution of the random sample), we can invoke the central limit and argue that asymptotically (hence with a finite n approximately), $\hat{\sigma}^2$ is normally distributed, with appropriate means and variances. To calculate these two quantities exactly, we would have to work a bit. In a Gaussian setting, we are able to derive the theoretical distribution (in closed form) for $\hat{\sigma}^2$.

R-Code 1.1 illustrates how to quantify the distribution (including bias, variance) with simulations in the setting of a Gaussian and a Poisson distribution. The results are shown in Figure 1.1. Both histograms resemble somewhat a Gaussian density, but are right skewed. The empirical bias is roughly 0.22 and changing to a different seed, varies the bias. The magnitude of the fluctuations of the empirical bias around the theoretical value of 0.2 is determined by the variable `R`, the number of samples drawn. ♣

R-Code 1.1 Assessing the distribution and bias of $\hat{\sigma}^2$. (See Figure 1.4.)

```

n <- 10          # sample size
R <- 1000        # result weakly depends on R, histograms get smoother
set.seed(16)      # to work reproducible
variance <- 2
sigma2s <- numeric(R)
for (i in 1:R) {
  x <- rnorm(n, mean=0, sd=sqrt(variance)) # sampling from  $X \sim N(0, 2)$ 
  sigma2s[i] <- sum((x-mean(x))^2)/n         # manual calculation of variance
}
hist(sigma2s, prob=TRUE, main="", ylim=c(0, .6), xlim=c(0, 8))
lines(density(sigma2s))
abline(v=c(variance, mean(sigma2s)), col=2:3)
variance - mean(sigma2s) # theoretical value is variance/n, here 0.2
## [1] 0.22384
# similar for Poisson: `x <- rpois(n, variance)`, e.g.,  $X \sim \text{Pois}(2)$ 

```

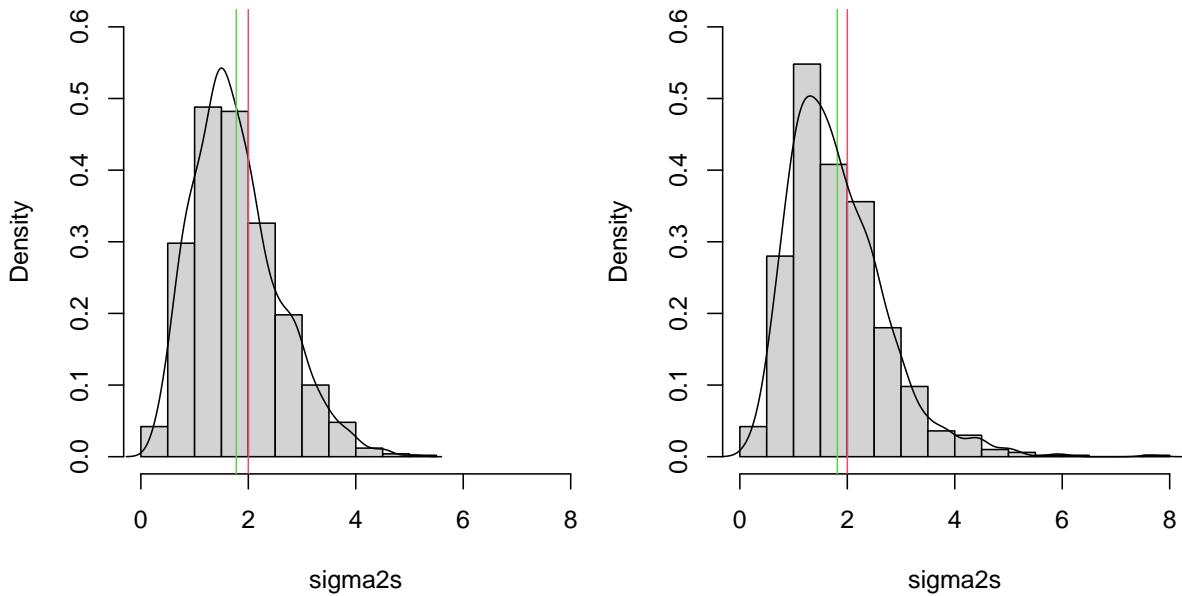


Figure 1.4: Empirical density of $\hat{\sigma}^2$ with underlying Gaussian (left) and Poisson (right) distributions (sample size $n = 10$, $R = 1000$ independent samples). True mean is in red, sample mean in green. (See R-Code 1.1.)

1.3 Cross-Validation

Occam's principle states that we should strive for the simplest model that best explains the data, i.e., one should balance model complexity and model fit. Typically, many predictors lead to a better fit of the observed data but overfitting may be observed when the model is used for prediction, as discussed earlier in the chapter.

To find a suitable model, we often divide the data into a training and a validation set. The training set is used to fit the model, the validation set is used to probe and assess the fitted model (through prediction). To avoid the strong dependency on the training set, we can repeat and sample a new training and validation set we select the optimal model based on validation results. This approach is often termed cross-validation. There is no general rule on how to divide the sample, often 70% to 80% of the data is used in the training step. There may be situations, where it is better to not use a random sample for the validation set (we will discuss this in Chapter 11).

Cross-validation is typically used in nonparametric smoothing, where the fitting paradigm is presented under the a bias-variance tradeoff (we will discuss this in Chapter 7). The tradeoff refers to the problem of simultaneously minimizing different sources of error (bias and variance) that prevent models from generalizing beyond their training data.

Reporting prediction error of a method based on the best training set is overly optimistic: the model with the smallest prediction error has been selected. We can overcome this issue by dividing the data in three sets: a training, a validation set and a test set. The test set is set aside and not used in the fitting and validation procedure. After the best model has been selected, we take the test set and calculate the prediction error.

In machine learning literature it is often recommended to split the observations with proportions 1/2, 1/4, 1/4 to these sets. Of course such a generous handling of data is not always possible, nor the guideline for a training and validation split given above. We now discuss an approach to obtain a training and validation set for small samples.

Leave-one-out cross-validation (LOOCV) is a technique to assess the optimum fit by omitting one observation at a time, fitting the model with the remaining observations and comparing the prediction against the omitted observation. A natural choice for a model selection criterion is to choose the model which minimizes the mean squared prediction error. Note that instead of using the prediction itself, we can construct many other measures as discussed further below.

In settings with large datasets or complex models, it is often impractical or impossible to refit the model for each omitted observation. A “leave- n -out” approach is chosen to balance computing time and assessment quality.

1.3.1 Leave-One-Out Statistics for Linear Models

In the case of a regression model with least squares estimate, several statistics can be constructed based on the leave-one-out procedure and are helpful for model diagnostics.

We see the regression model in more details in Chapter 5 but summarize the necessary equations here. We write the linear regression model with p predictors:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i \quad (1.4)$$

$$= \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, n, \quad n > p, \quad (1.5)$$

where

- Y_i : dependent variable, modeling the observation, data y_i ,

- $\mathbf{x}_i = (1, x_{i1}, \dots, x_{ip})^\top$: free variables, predictors, (known),
- $\boldsymbol{\beta} = (\beta_0, \dots, \beta_p)^\top$: parameter vector (unknown),
- $\varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$: error (unknown).

To write the linear model in matrix notation we stack the equations (1.5) to get

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \varepsilon, \quad (1.6)$$

where \mathbf{X} is the so-called design matrix. Under some weak assumptions (again discussed in Chapter 5, we have the least squares estimate

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \quad (1.7)$$

and the i th fitted value and i th residual are $\hat{y}_i = \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}$ and $e_i = y_i - \hat{y}_i$, respectively.

Natural questions include: what is the effect of the i th observation on the estimate, i.e., what is my estimate (1.7) when I do not consider the i th observation? On the prediction? On the estimated standard errors? These quantities lead to different statistics, including Cook's D statistic and the predicted residual error sum of squares (PRESS) statistic.

The precise meaning of the individual leave-one-out statistics as well as their interpretation are more important than derivation thereof. For the understanding of the statistics, we need to introduce some notation. We start from (1.6) and denote vectors without the i th case as \mathbf{y}_{-i} and a matrix without the i th row as \mathbf{X}_{-i} . Omitting the i th case means that we estimate the parameter $\boldsymbol{\beta}$ of the model $\mathbf{y}_{-i} = \mathbf{X}_{-i}\boldsymbol{\beta} + \varepsilon_{-i}$. Quantities that result from such deletion cases will be denoted with the subscript $\{-i\}$. For example, $\hat{\boldsymbol{\beta}}_{\{-i\}}$ is the estimate obtained from fitting the model without the i th case, i.e., the least squares estimate of $\mathbf{y}_{-i} = \mathbf{X}_{-i}\boldsymbol{\beta}_{\{-i\}} + \varepsilon_{-i}$. Further, we denote with h_{ii} the i th diagonal element of the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$, i.e., $h_{ii} = \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i$. Note that $0 \leq h_{ii} \leq 1$.

Using straightforward linear algebra properties, it can be shown that

$$\hat{\boldsymbol{\beta}}_{\{-i\}} = \boldsymbol{\beta} - \frac{e_i}{1 - h_{ii}} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i. \quad (1.8)$$

The interesting fact is that for $\hat{\boldsymbol{\beta}}_{\{-i\}}$, we do not need to refit the model for each omitted observation i . The estimate can be obtained as a byproduct of the usual regression output.

Summary statistics based on quantities from omitting observations are simply called leave-one-out statistics. They all start with the identity (1.8) and proceed with simplifications. We now enumerate the most important leave-one-out statistics:

- PRESS residuals are based on the fitted model by omitting the i th observations and are given by

$$e_{\{-i\}} = y_i - \hat{y}_{\{-i\}} = e_i + \frac{h_{ii}e_i}{1 - h_{ii}} = \frac{e_i}{1 - h_{ii}}, \quad (1.9)$$

where $\hat{y}_{\{-i\}} = \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}_{\{-i\}}$, i.e., the prediction of the i th observation based on the estimate obtained without the i th one. The predicted residual error sum of squares (PRESS) statistic, sometimes also PREdictive Sum of Squares statistic, is simply $\sum_{i=1}^n e_{\{-i\}}^2$.

- DFFIT (differences in fit) is the difference in prediction

$$\hat{y}_i - \hat{y}_{\{-i\}} = e_{\{-i\}} - e_i = \frac{e_i h_{ii}}{1 - h_{ii}}, \quad (1.10)$$

and quantifies the effect of omitting the i th observation on the fit. This difference is small if h_{ii} is close to zero (small leverage).

- DFFITS (differences in fits) is the standardized difference in prediction, see (1.10):

$$\frac{\hat{y}_i - \hat{y}_{\{-i\}}}{\sqrt{s_{\{-i\}}^2} h_{ii}} = \frac{\sqrt{h_{ii}} e_i}{(1 - h_{ii}) s_{\{-i\}}}. \quad (1.11)$$

Values of DFFITS larger than $2\sqrt{(p+2)/(n-p-2)}$ indicate that the i th observation is influential.

- DFBETAS (differences in betas), naming in analogy to (1.11) and a restating of equation (1.8)

$$\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_{\{-i\}} = \frac{e_i}{1 - h_{ii}} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i. \quad (1.12)$$

- Cook's D_i statistic:

$$D_i = \frac{1}{(p+1)s^2} (\hat{\boldsymbol{\beta}}_{\{-i\}} - \boldsymbol{\beta})^\top (\mathbf{X}^\top \mathbf{X}) (\hat{\boldsymbol{\beta}}_{\{-i\}} - \boldsymbol{\beta}). \quad (1.13)$$

There are several alternative ways to represent Cook's D_i statistic. For example based on residuals we have

$$D_i = \frac{h_{ii}}{(1 - h_{ii})^2} \cdot \frac{e_i^2}{(p+1)s^2}. \quad (1.14)$$

There is no universal consensus of what a large D_i value is. Most use the threshold one, state that values above 0.5 or just very particular values should be scrutinized.

- LOOMSE (my creation) is the unbiased estimator of σ^2 based on all but the i th observation

$$s_{\{-i\}}^2 = \frac{1}{n-p-2} \sum_{j=1, j \neq i}^n (y_j - \mathbf{x}_j^\top \hat{\boldsymbol{\beta}}_{\{-i\}})^2 = \frac{n-p-1}{n-p-2} s^2 - \frac{1}{n-p-2} \cdot \frac{e_i^2}{1 - h_{ii}}. \quad (1.15)$$

- COVRATIO measures the influence of a single observation on the precision of the estimates

$$C_i = \frac{\det(s_{\{-i\}}^2 (\mathbf{X}_{-i}^\top \mathbf{X}_{-i})^{-1})}{\det(s^2 (\mathbf{X}^\top \mathbf{X})^{-1})} = \left(\frac{s_{\{-i\}}^2}{s^2}\right)^{p+1} \frac{1}{1 - h_{ii}}. \quad (1.16)$$

A value of C_i near one indicates that the observation has little effect on the precision of the estimates. Observations outside the interval $1 \pm 3p/n$ need for further investigation.

In R, `help(influence)` and `help(cooks.distance)` give the relevant information about all the implemented functions. We illustrate these leave-one-out statistics in the following example.

Example 1.4. (*LifeCycleSavings* data) Under the life-cycle savings hypothesis developed by Franco Modigliani, the savings ratio (aggregate personal savings divided by disposable income) is explained by per-capita disposable income, the percentage rate of change in per-capita disposable income, and two demographic variables: the percentage of the population less than 15 years old and the percentage of the population over 75 years old.

The dataset *LifeCycleSavings* from the *datasets* package consists of the following five variables observed for 50 countries:

<i>sr</i>	aggregate personal savings ratio,
<i>pop15</i>	% of population under 15,
<i>pop75</i>	% of population over 75,
<i>dpi</i>	real per-capita disposable income,
<i>ddpi</i>	% growth rate of dpi.

The data are averaged over the decade 1960–1970 to remove the business cycle or other short-term fluctuations.

Scatterplots or formally scatterplot matrix, of the data are shown in the top panel of Figure 1.5. R-Code 1.2 illustrates the fitting of a multiple linear regression model using all possible covariates and the bottom panel of Figure 1.5 gives the associated graphical diagnostics panels. Some countries are highlighted therein due to large residuals or large Cook’s distances, also recognizable by the leave-one-out characteristics for a selection of these countries. Libya’s large Cook’s distance is due to its large *ddpi* value of 16.71. Japan and Zambia are flagged because of their high *sr* values. ♣

R-Code 1.2: Leverages of the *LifeCycleSavings* example. (See Figure 1.5.)

```

pairs(LifeCycleSavings, upper.panel=panel.smooth, lower.panel=NULL, gap=0)

summary(lcs <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data=LifeCycleSavings))

##
## Call:
## lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -8.242 -2.686 -0.249  2.428  9.751 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 28.566087   7.354516    3.88  0.00033 ***  
## pop15       -0.461193   0.144642   -3.19  0.00260 **   
## pop75      -1.691498   1.083599   -1.56  0.12553    
## dpi        -0.000337   0.000931   -0.36  0.71917    
## 
```

```

## ddpi          0.409695   0.196197    2.09  0.04247 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.8 on 45 degrees of freedom
## Multiple R-squared:  0.338, Adjusted R-squared:  0.28
## F-statistic: 5.76 on 4 and 45 DF,  p-value: 0.00079
plot( lcs, which=1:4)
rbind(resid=residuals(lcs), standard=rstandard(lcs), PRESSresid=
      rstandard(lcs, type="pred"), hii=hatvalues(lcs),
      CooksDist=cooks.distance(lcs))[,c(40, 2, 7, 23, 46, 49)]
##
##           Switzerland Austria Chile Japan Zambia Libya
## resid      2.4868259 0.6163860 -8.242231 5.28149 9.750914 -2.82953
## standard    0.6794481 0.1728294 -2.209074 1.57595 2.650915 -1.08705
## PRESSresid  2.6843808 0.7007443 -8.561559 6.79999 10.421335 -6.03899
## hii        0.0735942 0.1203839  0.037298 0.22331 0.064332 0.53146
## CooksDist   0.0073347 0.0008176  0.037813 0.14282 0.096633 0.26807
LifeCycleSavings[c("Chile", "Japan", "Zambia", "Libya"),]
##
##           sr pop15 pop75 dpi ddpi
## Chile     0.60 39.74  1.34 662.86  2.67
## Japan    21.10 27.01  1.91 1257.28  8.21
## Zambia   18.56 45.25  0.56 138.33  5.14
## Libya    8.89 43.69  2.07 123.58 16.71

```

1.3.2 Leave-One-Out Cross-Validation for Linear Models

Cross-validation in regression or other models can be used for model fitting or model selection. For example, in a regression model, we choose $\hat{\beta}$ such that the PRESS statistic is minimal.

In brief, cross-validation chooses the model having a smallest error. We typically choose a mean squared error criterion based on, for example, leave-one-out statistics. As shown under (1.9), this minimization is equivalent to minimizing weighted residuals. The weights are given by $1/(1 + h_{ii})$. A minimization is not always straightforward, for example when using Cook's distances, we would have to implement the non-linear optimization manually or iteratively.

In more complex situations than regression models it would be possible to use cross-validation as an estimation procedure. Depending on the model, model assumptions we may find the same estimate as the least squares estimate.

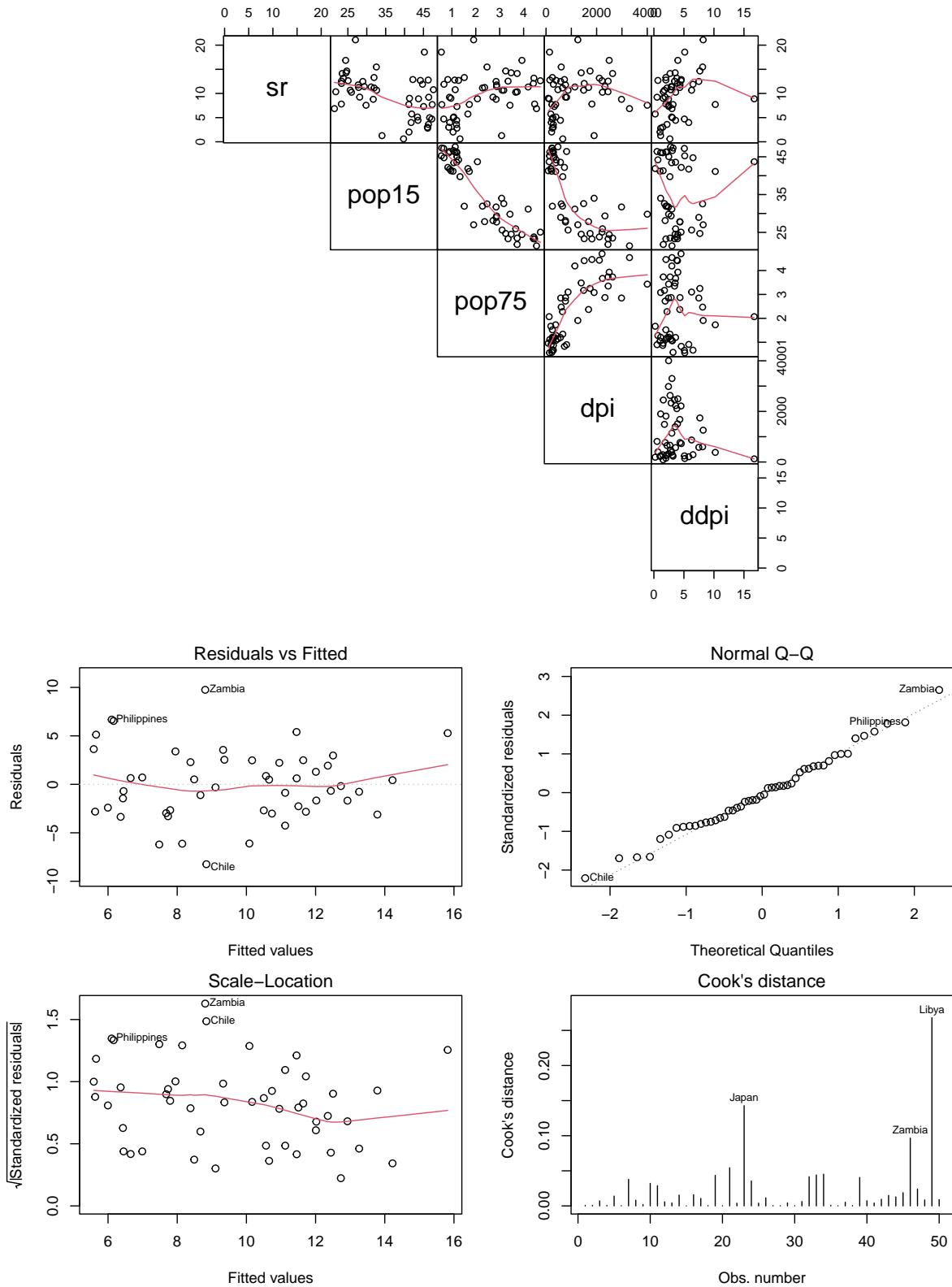


Figure 1.5: *LifeCycleSavings* data. Top: scatterplots of the data. The red line indicates a smooth line. Bottom: model diagnostics. (See R-Code 1.2.)

1.4 Bootstrap

Bootstrapping is a sampling technique to query models or properties of an estimator (such as its variance) by calculating these properties when sampling from an approximating distribution. Typically, the computational cost of sampling is (relatively) cheap and thus we draw a large number of samples.

One standard choice for an approximating distribution is the empirical distribution of the observed data. In other words we sample the observations *with replacement*, keeping sample size fixed. Another choice is a parametric distribution, where we assume an underlying parametric model and where the parameters have been estimated from the data.

1.4.1 Non-parametric Bootstrap

In the non-parametric bootstrap or sample bootstrap setting, new samples are drawn with replacement from the available sample. Suppose we want to study a quantity of interest or an estimator $\hat{\theta}$, in both situations represented by the statistic $\hat{\theta} = g(Y_1, \dots, Y_n)$ for some function g . The non-parametric bootstrap based on data y_1, \dots, y_n can be schematized as follows:

1. Sample y_1^*, \dots, y_n^* with replacement from y_1, \dots, y_n and calculate $\hat{\theta}^* = g(y_1^*, \dots, y_n^*)$;
2. Repeat Step 1 R times to get $\hat{\theta}_1^*, \dots, \hat{\theta}_R^*$;
3. Summarize the estimator based on the R estimates $\hat{\theta}_i^*$.

The algorithm has very few assumptions and seems somewhat magic, i.e., pulling oneself up of a swamp by one's bootstraps.

We illustrate the concept by calculating properties of a ratio in Example 1.5, with a manual implementation of the bootstrap. For more complex settings, we recommend the R package `boot`.

Example 1.5. We reconsider dataset `LifeCycleSavings` and would like to investigate uncertainty of the ratio estimate of the population over 75 to the population under 15 (i.e., `pop75/pop15`). The proportions of population under 15 and over 75 have no simple distribution and, moreover, are not independent (see the corresponding scatterplot in the top panel of Figure 1.5).

R-Code 1.3 samples with replacement from `pop75/pop15` and calculates its mean. The histogram of the bootstrapped means is given in Figure 1.6).

The histogram is symmetric and suggests that a Gaussian approximation might be sufficient: a smooth density estimate is very close to a Gaussian density. The reason for this good approximation is because the central limit theorem applies with almost arbitrary distributions. Here, we do not know the distribution of the ratio, but if we have a sample thereof, the distribution of the empirical mean can safely be approximated with a Gaussian distribution. The same would not hold if we work with the ratios of expectations!



R-Code 1.3 Bootstrap distribution of the *LifeCycleSavings* ratio of *pop75* to *pop15*. (See Figure 1.6.)

```
set.seed(14)      # here bootstrap results do not depend much on this seed
R <- 9999         # nr of replicates
sam <- LifeCycleSavings$pop75/LifeCycleSavings$pop15    # ratio
summary(sam)                                # summary thereof
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0123 0.0250 0.0710 0.0797 0.1289 0.2118
ratio <- numeric(R)
for (i in 1:R)  # try to vectorize this loop
  ratio[i] <- mean(sample(sam, replace=TRUE))
hist(ratio, prob=TRUE, main="")
lines(density(ratio))
curve(dnorm(x, mean=mean(ratio), sd=sd(ratio)), add=TRUE, col=2)
abline(v=c(mean(sam), mean(ratio)), col=c(4,3), lty=c(1,2))
```

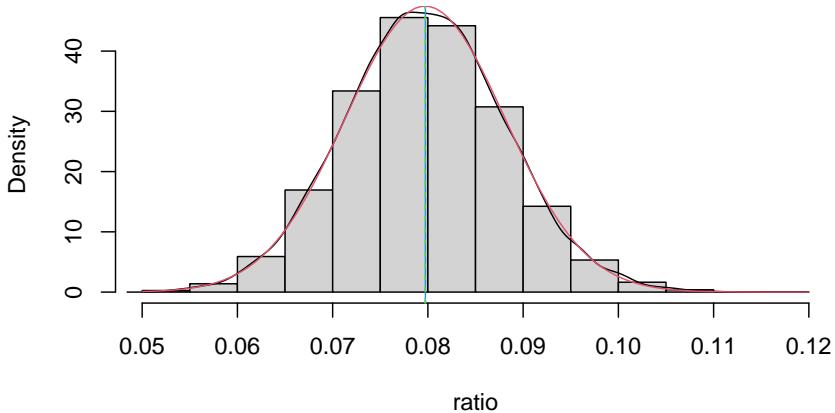


Figure 1.6: Histogram and smooth density estimate of bootstrap ratio of the proportions of population under 15 and over 75. Average of observations is in dashed green, averages of bootstrap means in blue (hardly visible). Gaussian density with matched mean and variance in red. (See R-Code 1.3.)

Especially for very small samples the discreteness of empirical bootstrap distribution is visible. If the sample size is very small, it is possible to consider all possible combinations instead of the sampling the observations (via *sample()*). However, for sample sizes four, five and six, there exist $4^4 = 265$, 3125 and 46656 possibilities based sampling with replacement, respectively. Hence, in all practical situations, bootstrapping is preferable.

1.4.2 Parametric Bootstrap

The parametric bootstrap requires an underlying (parametric) model. The original data is used to estimate parameters of the model and additional samples are drawn from the fitted model using the original estimate.

Hence, parametric bootstrap is similar to the sampling idea introduced at the beginning of the chapter with the difference that the distribution is not entirely known here.

Example 1.6. On a rainy day you get out from the Polybahn and would like to take a tram from the station Central to the main train station because of heavy rain. What is your mean waiting time?

To estimate the frequency of the trams you start to measure the time until you see the first tram from each line departing. Suppose you observe (rounded) 3', 9', 9'45", 14'30". For simplicity, assume that the trams lines depart at the same frequencies and arrive independently of each other, that means, each the departure of each tram after your arrival can be modeled by a uniform random variable $U(0, \theta)$.

We first start by estimating θ and then draw from that density to calculate the mean waiting time and a confidence interval of the waiting time; all illustrated in R-Code 1.4.

In this specific setting, it is possible to derive the density of the waiting time

$$Z = \min(X_1, X_2, X_3, X_4), \quad \text{with} \quad X_i \stackrel{\text{iid}}{\sim} U(0, \theta), \quad (1.17)$$

being $f_Z(z) = n(\theta - z)^{n-1}/\theta^n$. Additionally, the expected waiting time is $E(Z) = \theta/5$. ♣

R-Code 1.4 Parametric bootstrap and resulting mean waiting time. (See Figure 1.7.)

```
set.seed(15) # set the seed to get the same starting data
n <- 4       # trams
R <- 999      # replicates
print(round((sam <- runif(n , max=1/4))*60, 2)) # observations!
## [1] 9.03 2.93 14.50 9.76

# Estimate is max(sam) or even better max(sam)*(n+1)/n
estim <- max(sam)*(n+1)/n

res1 <- apply(matrix(runif(n*R, max=estim), R, n), 1, min)
hist(res1*60, prob=TRUE, main="", xlab="Waiting time [min]")
abline(v=mean(res1*60))
round(mean(res1)*60,2) # mean waiting time in minutes
## [1] 3.76

round(quantile(res1*60, c(.025, .975)), 2) # CI for waiting time
## 2.5% 97.5%
## 0.13 11.15
```

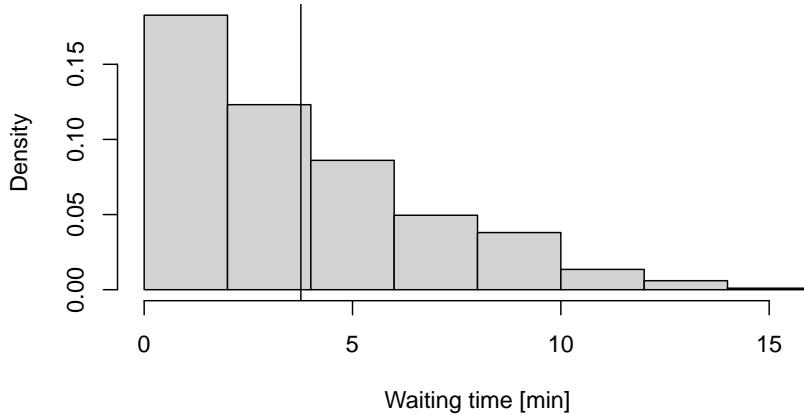


Figure 1.7: Parametric bootstrap distribution of the waiting time. Empirical mean is indicated by the vertical line. (See R-Code 1.4.)

1.5 Example: Description of Antarctic Ice Shelves

Kirchner *et al.* (2013) used characteristics of 17 Antarctic ice shelves and ice tongues to construct simple models of specific characteristics thereof for predictive purposes. For simplicity, we use the term ice shelves only. Here, we use the same data to illustrate various concepts of this chapter: (i) effect of omitting influential observations on the residuals and/or the fitted values (ii) distribution of the median.

R-Code 1.5 starts loading the data consisting of relevant information of the ice sheets and ice tongues. Major ice-shelves are indicated in the left panel of Figure 1.8 and further, detailed information can be found through https://en.wikipedia.org/wiki/List_of_Antarctic_ice_shelves. There are vast differences in sizes of the ice shelves and thus a transformation is necessary. We opt for a log-transformation.

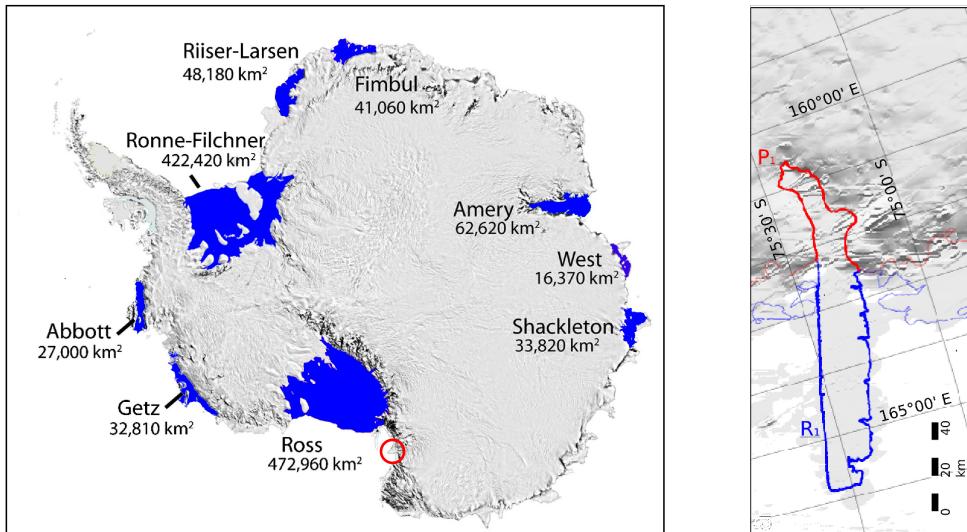


Figure 1.8: Left: major ice shelves in Antarctica (figure adapted and modified from <http://en.wikipedia.org>). Right: Drygalsky ice-tongue (red circle on the left) with grounding line (red) and calving front (blue).

One of the main goals of the original study was to describe ice-shelf area as a function of the grounding line, i.e., the line where the ice stream starts to “flow” on the water and is “detached from the ground”, and possibly other predictors. The right panel of Figure 1.8 illustrates the grounding line and calving front for the Drygalsky ice-tongue. The grounding line remains relatively stable over time as it (essentially) depends on the ice thickness in the vicinity of that line, whereas the calving front and maximal thickness at calving front greatly vary over time (short and long term). We leave the discussion of the model selection process as an exercise and continue with the simplest possible model.

There was considerable concern that individual shelves (e.g., Ross) could have a strong influence on the fit and thus leave-one-out diagnostics were performed. Figure 1.9 illustrates the effect of omitting influential observations on the fit and on the residuals. (Notice how the individual names have been placed. This takes time but an Adobe post-processing of the figure is no alternative!) The four observations with the largest leverages are the ones with the most extreme grounding length (no surprise because we have only one predictor). There is virtually no difference between the studentized and standardized residuals.

R-Code 1.5: Explaining the ice-shelf area as a function of the grounding line. The fits resulting from the estimates without the observations are calculated. (See Figure 1.9.)

```

aa <- read.csv("download/sta121/AAshelves.csv")
str(aa, strict.width="cut")

## 'data.frame': 17 obs. of  8 variables:
##   $ length_of_grounding_line.km.      : int  901 2404 471 501 158 480...
##   $ length_of_calving_front.km.       : int  1032 473 664 61 359 282 ...
##   $ area_ice_shelf_with_ice_rises.km2. : int  47265 62984 30759 4675 2...
##   $ total_area_of_ice_rises.km2.       : int  14132 942 -9999 -9999 -9...
##   $ number_of_ice_rises...             : int  7 5 0 0 0 1 3 6 8 0 ...
##   $ max_iceshelf_draft_at_calving_front.m.: int  190 200 110 150 110 180 ...
##   $ water_temp_at_calving_front.degC.    : num  -1.36 -1.72 -1.75 -1.63 ...
##   $ max_ice_thickness_at_grounding_line.m.: int  1500 3000 1800 2200 4500...

print(icenames <- rownames(aa))

## [1] "Abbot"          "Amery"           "Brunt"
## [4] "Dotson"         "Drygalski"        "Ekstrom"
## [7] "Filchner"       "Fimbul"          "Getz"
## [10] "Mertz Glacier"  "Ninnis Glacier"  "Pine Island Glacier"
## [13] "Riiser-Larsen" "Ronne"           "Ross"
## [16] "Shackleton"     "West"

logarea <- log(aa[[3]])
loglen <- log(aa[[1]])
n <- length(logarea)

cor(aa[[3]], aa[[1]])

```

```

## [1] 0.98302
cor(logarea, loglen)
## [1] 0.95739
lm1 <- lm(logarea~loglen)
hii <- hatvalues(lm1) # equivalent to `influence(lm1)$hat`
names(hii) <- rownames(aa)
sort(hii, decreasing=TRUE)[1:4] # threshold is roughly .235
##          Ross          Ronne Ninnis Glacier      Drygalski
## 0.30918     0.22820     0.22301     0.20837
plot(logarea~loglen, xlim=c(4.5, 9.5))
abline(lm1)
points(loglen,dffits(lm1)+fitted(lm1), col=2)
sel <- c(2, 3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16, 17)
text(loglen[sel]+.1, logarea[sel], icenames[sel], adj=c(0,.5))
sel <- c(10, 8)
text(loglen[sel]-.1, logarea[sel], icenames[sel], adj=c(1,.5))
sel <- c(1, 9)
text(loglen[sel], logarea[sel]+.1, icenames[sel], adj=c(1,.5), srt=-90)
plot(rstandard(lm1), rstudent(lm1), xlim=c(-2,2.8))
abline(c(0,1), col="gray")
sel <- c(2,4,12,1,3)
text(rstandard(lm1)[sel]+.1, rstudent(lm1)[sel], icenames[sel], adj=c(0,.2))

```

As a bootstrapping example, we would like to determine a 90% confidence interval of the median from the log-transformed areas of the 17 ice shelves. As a crude CI one could use the median in conjunction with the standard error of the mean, here $[10.49 \pm z_{95\%} 0.41] = [9.82, 11.16]$. Mixing properties of different estimators is not ideal; we should use such approaches only if there is no alternative available. Based on a bootstrap sample, we find in R-Code 1.6 the interval $[8.87, 10.76]$. Note that the distribution of the median is multimodal due to the small sample size and the discreteness of the median. Both aspects are emphasized by the bootstrap procedure.

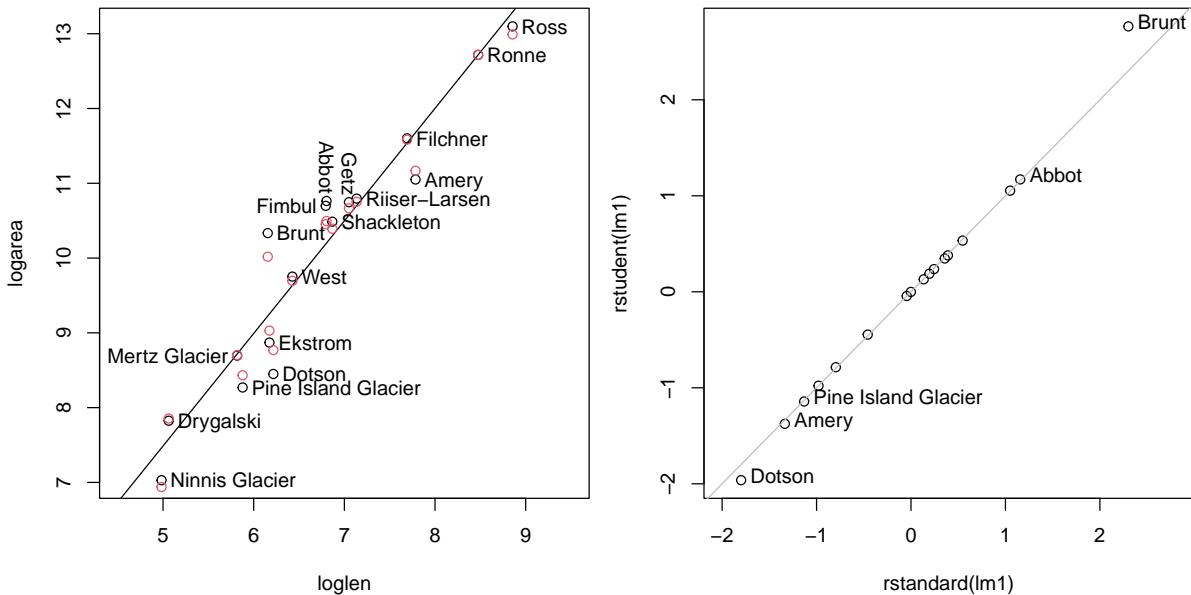


Figure 1.9: Left: Explaining the ice-shelf area as a function of the grounding line (black line). Black dots are observations; red dots are fitted values plus difference in fits. Right: studentized residuals as a function of the standardized residuals, zero-one line in gray. (See R-Code 1.5.)

R-Code 1.6 Bootstrap distribution of the median. (See Figure 1.10.)

```

set.seed(12)
n <- 17
R <- 5000
sam <- array(sample(logarea, R*n, replace=TRUE), c(R, n))
meds <- apply(sam, 1, median)
means <- apply(sam, 1, mean)
hist(logarea, prob=TRUE, col="gray", ylim=c(0,2.), main="", border=NA)
rug(logarea, ticksize=.1)
hist(meds, add=T, prob=T, col=rgb(1,0,0,.2), border=NA)
lines(density(meds, adjust=2), col=2)
hist(means, add=T, prob=T, col=rgb(0,0,1,.2), border=NA)
curve(dnorm(x, mean(logarea), sd(logarea)/sqrt(n)), add=T, col=4)
abline(v=c(mean(logarea), median(logarea)), col=c(4,2), lwd=3)
median(logarea)+qnorm(c(.05,.95))*sd(logarea)/sqrt(n)
## [1] 9.8163 11.1568
quantile(meds, c(.05,.95)) # better
##      5%      95%
## 8.8701 10.7635

```

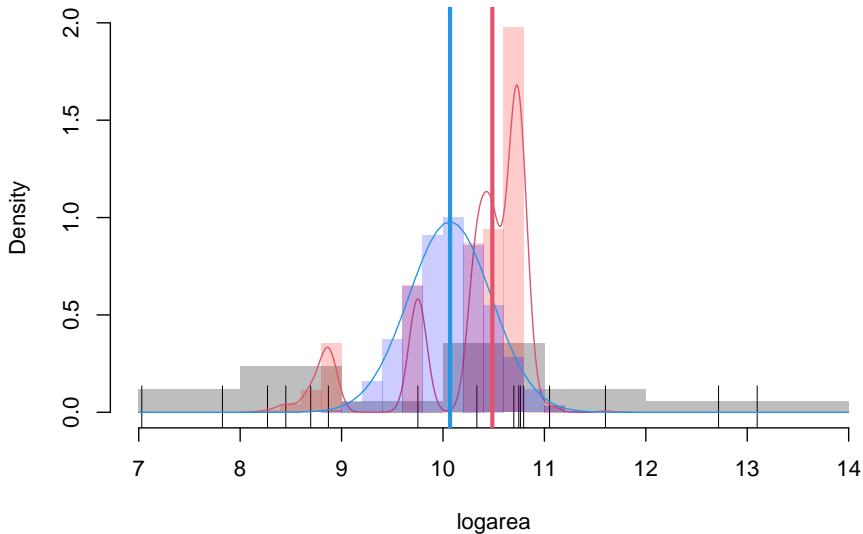


Figure 1.10: Bootstrap distribution of the median and mean. Data is given by the red tick marks and the gray histogram. Bootstrap medians in the red histogram with a smooth density estimate, bootstrap means with the blue histogram. The blue density is the theoretical density of the bootstrap means. Vertical lines indicate sample median (red) and mean (blue). (See R-Code 1.6.)

1.6 Bibliographic remarks

The derivation of the leave-one-out statistics for linear models is comprehensively given in [Abraham and Ledolter \(2006\)](#), for example.

Chapter 7 of [Hastie et al. \(2009\)](#) is a good reference for cross-validation. Note that the first author also provides a pdf copy of the book on the webpage <https://web.stanford.edu/~hastie/ElemStatLearn/>.

The bootstrap has been introduced by [Efron \(1979\)](#) and is an extension of the Jackknife. The term is understood as a large class of resampling techniques. For a general overview, see [Davison and Hinkley \(1997\)](#).

1.7 Exercises and Problems

Problem 1.1 (Sampling uncertainty) In this problem, we want to quantify the uncertainty of the bias estimate in Example 1.1.

- Visualize and summarize the sampling variability of the bias estimate (`variance - mean(sigma2s)` in R Code 1.1).
- The variance of the empirical bias decreases with increasing number of samples (`R` in R Code 1.1). Illustrate with a simulation study that the reduction is proportional to $1/R$.

Problem 1.2 (Mean squared error and model selection) In this problem, we want to select the '*best*' model in terms of the mean squared error using different techniques. For this, we are going to fit two models using the `cars` dataset: `dist~speed`, and `dist~speed+I(speed^2)`.

- a) Take a look to the `cars` dataset using the functions `help()`, `head()` and `plot()`.
- b) (*Using all the sample*) Fit the models using all the sample and compute the mean squared errors.
- c) (*Training and validation set*) Split the data into training and validation sets (80% and 20%), and for each of the given models, compute the mean squared error.
- d) (*Training, validation, and test set*) Split the data into training, validation, and test sets (70%, 20%, and 10%, respectively). For each of the proposed models, compute the mean squared error using the training, and validation sets. Select the '*best*' model and compute the mean squared error using the test set.
- e) (*Leave-one-out cross-validation*). Take a look to the following `r` function to compute the mean squared error using leave-one-out cross-validation. Explain the function.

```
LOOCV <- function(df,formula){
  mse_values <- numeric(nrow(df))
  for(i in 1:nrow(df)){
    validation_data_point <- df[i,]
    train_set <- df[-i,]
    model_to_fit <- lm(formula,data=train_set)
    mse_values[i] <- (predict(model_to_fit,validation_data_point) -
      validation_data_point[,as.character(formula[2])])^2
  }
  return(mean(mse_values))
}
```

- f) Now, for each of the models, compute the mean squared error using leave-one-out cross-validation.

```
loocv_fit1 <- LOOCV(cars, dist ~ speed)
loocv_fit2 <- LOOCV(cars, dist ~ speed + I(speed^2))
loocv_fit1
## [1] 246.4054
loocv_fit2
## [1] 243.0292
```

- g) Compare the results obtained on each of the previous items.

Problem 1.3 (Bootstrap) Consider the simple linear regression $Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$, $i = 1, \dots, n$ and with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. Since we have an artifical situation, we do not know x_1, \dots, x_n and thus we propose to draw these from $X_i \sim \mathcal{N}(\mu, \sigma_x^2)$, X_i and ε_i independent. Note that our model is not quite $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$ as we fit the model conditionally on having observed x_i .

- a) Simulate $n = 15$ data points from that model with $\beta_0 = 1$, $\beta_1 = 2$, $\mu = 4$, $\sigma_x = 4$ and $\sigma = 2$. Then estimate the regression coefficients β_0 and β_1 . Plot the data in a scatter plot and add the true and the estimated regression lines.
- b) For the same values of x_i , repeat the simulation of y_i and the estimation of β_0 and β_1 many ($R = 1000$) times. Look at numerical and graphical summaries of the distribution of $\hat{\beta}_0$ and $\hat{\beta}_1$. Are $\hat{\beta}_0$ and $\hat{\beta}_1$ good estimates of β_0 and β_1 ?
- c) Similar as in b) but now also draw x_i for every simulation. Compare the two simulation settings and comment.

Problem 1.4 (*Optional:* training, validation and test sets) We use the `bicoal` data set provided in the package `fma` and fit a polynomial model, i.e., a linear regression with predictor `year`, `year^2`, `year^3` and so forth up to degree eight. In R this can be achieved with `poly(year, 8)`.

- a) Subset the data in training and validation (80% and 20%) and fit a polynomial model. Repeat the subsetting 100 times and visualize the fits. Which is the best model according to the residual sums of squares?
- b) We now further reserve a test set, consisting of the first and last five years. The intermediate years are split in training and validation (26 and 10 data points). Build an “optimal” model as above and use the `predict` function for the test years. Comment.

Problem 1.5 (*Optional:* Error in variables) Consider the simple linear regression $Y_i = \beta_0 + \beta_1 \tilde{X}_i + \varepsilon_i$ with $\tilde{X}_i = X_i + \varepsilon_\omega$, where $X_i \sim \mathcal{U}(-1, 3)$ and $\varepsilon_\omega \sim \mathcal{N}(0, \sigma_\omega^2)$, and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. \tilde{X}_i and ε_i are independent and $i = 1, \dots, n$. In this problem we investigate the effect of observing X_i with an error based on sampling (\tilde{X}_i).

- a) Simulate $n = 15$ data points from that model with $\beta_0 = 1$, $\beta_1 = 2$, $\sigma = 2$, and $\sigma_\epsilon = 1$. Then estimate the regression coefficients β_0 and β_1 . Repeat the simulation many times and look at numerical and graphical summaries of the distribution of $\hat{\beta}_0$ and $\hat{\beta}_1$. Are $\hat{\beta}_0$ and $\hat{\beta}_1$ good estimates of β_0 and β_1 ? What is the effect on the estimated parameters when σ_ω increases?
- b) For two random variables Y_1 and Y_2 and scalars a , b , c and d , show that have $\text{Cov}(a + b Y_1, c + d Y_2) = b d \text{Cov}(Y_1, Y_2)$.
- c) Let X_1 and X_2 be independent standard normal random variables. Define the random variables Y_1 and Y_2 by

$$Y_1 = 2X_1 + X_2, \quad Y_2 = X_1 - X_2.$$

Find $E(Y_1)$, $E(Y_2)$, $\text{Cov}(Y_1, Y_2)$, and the joint probability density function $f_{Y_1, Y_2}(y_1, y_2)$.

Problem 1.6 (*Optional:* Model selection) Consider the `powerplant.txt` data available on the web page <https://www.math.uzh.ch/furrer/download/sta121/powerplant.txt>. Contains information about 32 light water reactor (LWR) power plants. The variables are:

C: Costs in dollars $\times 10^{-6}$ adjusted to 1967 base

D: Date construction permit issued

T1: Time between application for permit and issue of permit

T2: Time between issue of operating license and construction permit

S: Power plant net capacity (MWe)

PR: Prior existence of a LWR at same site

NE: Powerplant constructed in the north-east area of the USA

CT: Use of cooling tower

BW: Nuclear steam supply system manufactured by Babcock–Wilcox

N: Cumulative number of power plants constructed by each architect

PT: Partial turnkey plant; spatial feature in the contract

Find a good linear model that explains the costs of the power plants. Discuss the leave-one-out statistics.

Chapter 2

Principal Component Analysis

The idea of principal component analysis is to construct a low dimensional representation of the data that explains as much structure as possible.

Learning goals for this chapter:

- ◊ Carry out a principal components analysis using R
- ◊ Explain PCA (including eigendecomposition)
- ◊ Assess how many principal components are needed
- ◊ Interpret principal component scores and loadings

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter02.R.

2.1 Finding some Representative Dimensions

As motivation, consider the scatterplots of 500 data points in \mathbb{R}^3 on the left panel of Figure 2.1. Nothing striking is visible, the correlations between the variables are 0.85, 0.6 and 0.1. Such a `pairs()` plot is only visualizing the marginal scatterplots. When using a three-dimensional visualization, it becomes quickly clear, that the data lives in a two dimensional subspace. As shown in the top panel on the right of Figure 2.1. That means, we can represent the data with two coordinate variables and not three. In other words, we realign the coordinate system and use the corresponding two dimensions only (lower panel on the right of Figure 2.1) without loosing information.

In practice, we often have often many dimensions, such that we need a formal algorithmic approach to determine the optimal realignment of the coordinate system. One such approach is principal components analysis (PCA) that we discuss in this chapter. For an intuitive understanding of PCA, we consider noisy data. Typically, noise is present in all dimensions and the data matrix lives in all dimensions. In the context of the previous dataset, the data points in

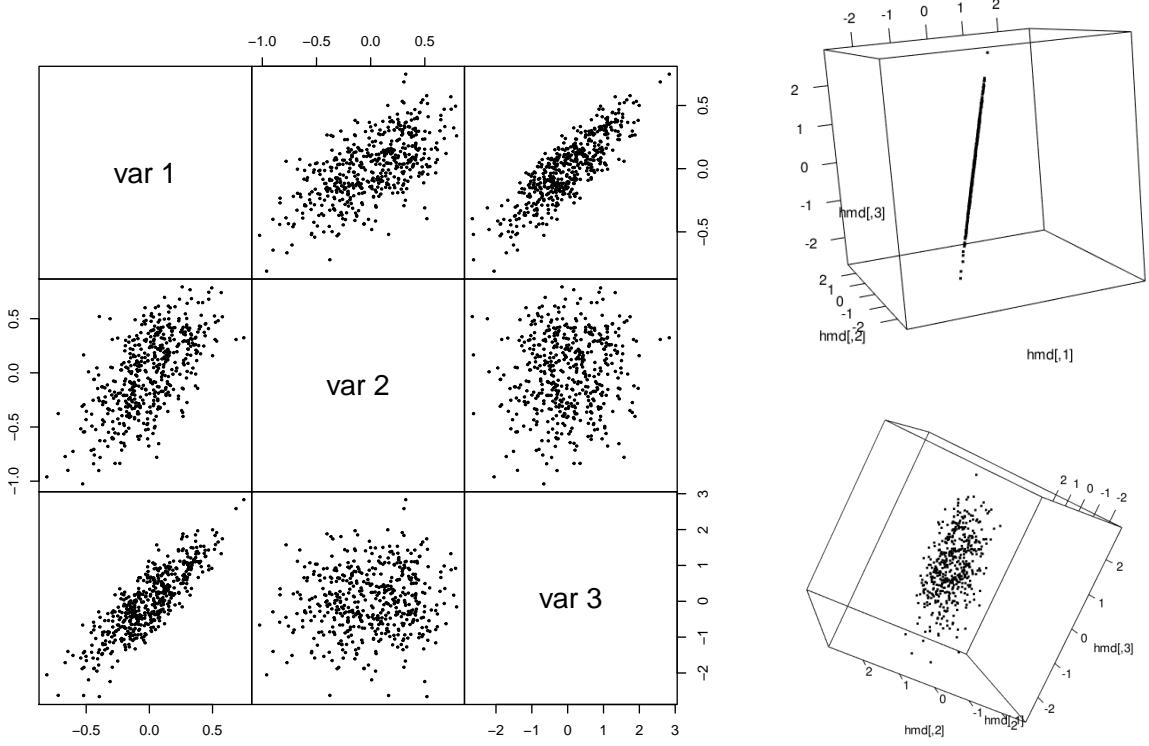


Figure 2.1: Scatterplot of the artificial data `hmd` (left) and snapshots of particular three dimensional visualization with `rgl::plot3d()` (right).

top right representation of Figure 2.1 are not completely aligned but jittered. Directions with low variability likely contain only noise and thus can be omitted. Or, we would like to keep the dimensions with highest variability, preserving as much structure as possible.

More formally, let \mathbf{y} be p -vector. PCA finds an orthogonal basis in \mathbb{R}^p , $\mathbf{e}_1, \dots, \mathbf{e}_p$, and represents the vector \mathbf{y} as

$$\mathbf{y} = \sum_{j=1}^p u_j \cdot \mathbf{e}_j. \quad (2.1)$$

In PCA terminology, the vectors \mathbf{e}_j are known as *loadings* and the coefficients u_j as (*principal component*) *scores*. For a different vector, the same loadings are used and new scores are determined. Of course, we consider \mathbf{y} as the realization of a random vector.

Dimension reduction is achieved when the reconstruction of \mathbf{y} is build on fewer loadings and the vector is approximated by

$$\mathbf{y} \approx \sum_{j=1}^k u_j \cdot \mathbf{e}_j, \quad k < p. \quad (2.2)$$

Hence, instead of using p vectors \mathbf{e}_j , the number of loadings is limited to k . In other words, the dimension is reduced from p to k . This dimension reduction can be interpreted as data compression. Good compression is (naturally) achieved when $k \ll p$. This is only possible when the data (i.e., the system) is dominated by few directions (i.e., modes) of variability $\mathbf{e}_1, \dots, \mathbf{e}_k$, that explain most of the total variance.

We now put forth a formal statistical framework, by assuming a random vector \mathbf{Y} with realization \mathbf{y} , such that the scores U_j are random as well. In the framework of PCA, the decomposition

$$\mathbf{Y} = \sum_{j=1}^p U_j \cdot \mathbf{e}_j, \quad (2.3)$$

fulfills the following criteria:

- Orthogonality of the basis: $\mathbf{e}_j^\top \mathbf{e}_\ell = \delta_{j,\ell}$, where the Kronecker symbol $\delta_{j,\ell} = 1$ if $j = \ell$ and 0 if $j \neq \ell$. Hence, we should rather talk about orthonormality. This assumption was also used in (2.1).
- The principal component scores are mutually uncorrelated: $\text{Cov}(U_j, U_\ell) = 0$, if $j \neq \ell$. This reflects the independence of the new variables U_j in the new basis.
- The variance of the scores U_j has been successively maximized: $\lambda_j = \text{Var}(U_j)$, with $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$.

This decomposition creates a new random vector $\mathbf{U} = (U_1, \dots, U_p)^\top$, which is the *projection* of \mathbf{Y} onto the new basis $\{\mathbf{e}_1, \dots, \mathbf{e}_p\}$. Note that the above criteria define the scores only up to a sign, as directly seen by (2.3) (or by (2.1)).

From a data compression point of view, we require an integer k such that $\sum_{j=1}^k \text{Var}(U_j) = \sum_{j=1}^k \lambda_j$ is close to the total variation, namely $\sum_{j=1}^p \text{Var}(Y_j)$.

We will proceed with a simple example, where we observe several realizations, i.e., vectors $\mathbf{y}_1, \dots, \mathbf{y}_n$. Typically, we subscript the vectors with i , the components of the vector with j and we arrange the vectors in a so-called data matrix \mathbf{Y} , with rows the (transposed) observations. The context will determine if \mathbf{Y} is a $n \times p$ data matrix or a p dimensional random vector. PCA can help with interpretation of datasets, as illustrated in the next example.

Example 2.1. (Socioeconomic data) The ‘European Jobs’ table reports the population percentage employed in different sectors of the economy in 1979. It includes $n = 26$ nations and $p = 9$ variables that give the percentages of employment in: agriculture (*Agr*), mining (*Min*), manufacturing (*Man*), power supply industries (*PS*), construction (*Con*), service industries (*SI*), finance (*Fin*), social and personal services (*SPS*), and transport and communications (*TC*), source: [Euromonitor \(1979\)](#). See R-Code 2.1.

How can one efficiently interpret the data? Intuitively, one could compute all pairwise correlations between two countries to find which national economies are more related. Fortunately, PCA provides a more efficient and comprehensive solution. Let $\mathbf{Y} \in \mathbb{R}^{26 \times 9}$ be the data matrix with country dimension $n = 26$ and economic sector dimension $p = 9$. Let us apply PCA to the corresponding data matrix. The structure of the output is given in R-Code 2.1. The matrix *rotation* are the loadings and matrix *x* the scores. The vector *sdev* contains the *square roots* of the variances λ_j .

The left panel of Figure 2.2 shows the explained variability by each PC. This plot is called the ‘scree plot’. It appears that the first two PCs already explain more than 62% of the total variability and that the data essentially lives in a seven dimensional subspace. Remember that

the PC loadings form the new basis $e_j, j = 1, \dots, 9$ on which to represent the data. Because the first two modes $e_j, j = 1, 2$ explain 62% of the variability, they can be used to efficiently *visualize* the table. Instead of using all 9 dimensions (i.e., the 9 economic sectors), the data matrix \mathbf{Y} is projected onto a bi-dimensional space. The right panel of Figure 2.2 shows the first two PCs as a scatter plot using country names as labels. Different groups are easily identified, for example, eastern European countries cluster in the top part of the panel.



R-Code 2.1 PCA of *jobs* dataset. (See Figure 2.2.)

```

jobs <- read.csv(header=TRUE, "download/sta121/europejobs.csv")
head(jobs, n=4)

##          Agr Min Man PS Con   SI Fin SPS TC
## Belgium    3.3 0.9 27.6 0.9 8.2 19.1 6.2 26.6 7.2
## Denmark    9.2 0.1 21.8 0.6 8.3 14.6 6.5 32.2 7.1
## France     10.8 0.8 27.5 0.9 8.9 16.8 6.0 22.6 5.7
## W.Germany   6.7 1.3 35.8 0.9 7.3 14.4 5.0 22.3 6.1
rownames(jobs)[5:20]

## [1] "Ireland"        "Italy"           "Luxembourg"      "Netherlands"
## [5] "UnitedKingdom"  "Austria"         "Finland"         "Greece"
## [9] "Norway"          "Portugal"        "Spain"          "Sweden"
## [13] "Switzerland"     "Turkey"          "Bulgaria"        "Czechoslovakia"

pca <- prcomp(jobs, scale=TRUE)
str(pca, give.attr=FALSE)

## List of 5
## $ sdev    : num [1:9] 1.867 1.46 1.048 0.997 0.737 ...
## $ rotation: num [1:9, 1:9] 0.52379 0.00132 -0.3475 -0.25572 -0.32518 ...
## $ center   : Named num [1:9] 19.131 1.254 27.008 0.908 8.165 ...
## $ scale    : Named num [1:9] 15.547 0.97 7.008 0.376 1.646 ...
## $ x        : num [1:26, 1:9] -1.677 -0.934 -0.74 -0.836 0.101 ...
round(cumsum(pca$sdev^2)/sum(pca$sdev^2), 3)

## [1] 0.387 0.624 0.746 0.857 0.917 0.960 0.985 1.000 1.000
plot(pca) # equivalent to `screeplot(pca)`
plot(pca$x[,1:2], type="n", ylim=c(-2.7, 3.7), xlim=c(-3, 7))
text(pca$x[,1:2], rownames(jobs))

```

Typically, the data is centered before performing a PCA. Moreover, we typically scale the variables `scale=TRUE` in `prcomp()`. This addresses the issue that variables are unit independent and comparable. Otherwise, one of the variables may dominate all others due to its huge variance and thus PCA may not achieve its goals.

As shown in the last example, the PCs allow groups to be identified. The actual individual scales are not relevant, but how they relate to one another is. Through the transformation

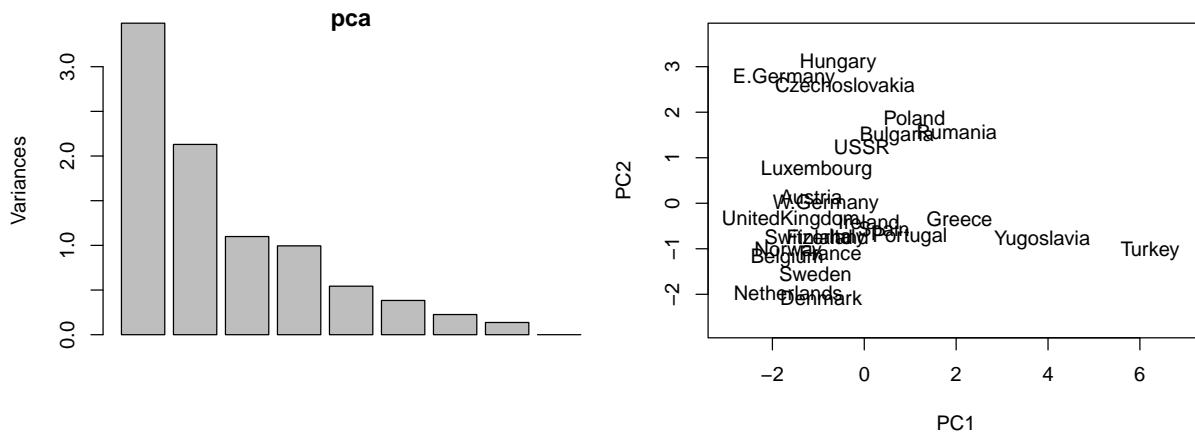


Figure 2.2: Scree plot of the PCA on `jobs` data (right) and socioeconomic data from 26 European countries projected on the subspace spanned by the first two PCs (right). (See R-Code 2.1.)

(scaling with `scale=TRUE` and rotation) the PCs do not necessary possess interpretative units. The relation of the scales are linked to the corresponding λ_{js} .

For a better interpretation, we also project the “variables”, i.e., the original basis, on the PCs. The variables can be seen as p additional observations with values $(1, 0, \dots, 0)^\top, (0, 1, 0, \dots)^\top$ and so forth. To differentiate between the individuals and the “variables” we use arrows for the latter. The observations and variables are typically on a different “scale” and need to be scaled for to improve visualization and ease interpretation. However, there is no unique choice for how to best scale the variables. This plot is called a *biplot* and illustrates which variables contribute to the PCs. Hence, a biplot is a particularly convenient way to understand the essence of the data in a glance, as discussed in the continued example below.

Example 2.2. We retake the PCA decomposition of the `jobs` dataset. Figure 2.3 shows a classical biplot, constructed with the command `biplot(pca)`. Compared to Figure 2.2, arrows have been added and scores have been scaled. The first PC has a high positive loading on the agriculture variable and negative or small loadings on all other variables. This PC may be interpreted as distinguishing between countries with agricultural and industrial economies. Turkey and Yugoslavia have the highest values compared to the other countries, suggesting that their economies were more agricultural than the economies of the other nations. The second PC has negative loadings on service industries, finance, and social and personal services, but has positive loadings on all others. This PC may be interpreted as distinguishing between nations with large and small service sectors. The “capitalist” Western nations have lower scores than the “communist” Eastern nations, suggesting that economies in the West have larger service sectors.

Such an interpretation makes sense only if the two first PCs explain a sufficiently large portion of the variability. Here the first two PCs explain 62.4% of the total variability. ♣

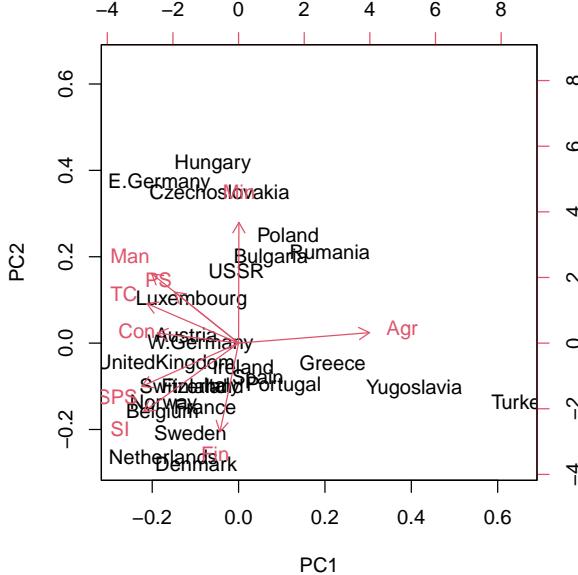


Figure 2.3: Biplot of the socioeconomic data, created with `biplot(pca)`.

2.2 PCA as a functional Decomposition: EOF

2.2.1 Motivation

We now consider an alternative approach to introduce PCA. To illustrate, consider a (sufficiently) smooth function $f : [0, 2\pi] \rightarrow \mathbb{R}$. Based on Fourier analysis theory, it is possible to re-express the function as a sum of sine and cosine functions, i.e.,

$$f(x) = a_0 + \sum_{j=1}^{\infty} a_j \cdot \cos(j \cdot x) + \sum_{j=1}^{\infty} b_j \cdot \sin(j \cdot x), \quad 0 < x < 2\pi. \quad (2.4)$$

We have equality because of the infinite sums. Quite often, a limited amount of basis functions are required for a sufficiently good approximation. Moreover, we often observe the function f at finite number of locations, further justifying a limited number of basis functions. In later chapters we encounter further particular basis functions, which can be written generically by ϕ_j . Hence, consider the approximation

$$f(x) \approx \sum_{j=0}^p a_j \phi_j(x). \quad (2.5)$$

In the case of PCA, the basis functions ϕ_j are not pre-existing, but are entirely characterized by the *data* itself. However, PCA constructs an orthogonal basis. This is the reason PCA is also known as *empirical orthogonal function* (EOF) analysis in certain sciences. More precisely, assume that we observe the function f at n locations x_1, \dots, x_n . We write $\mathbf{y} = (f(x_1), \dots, f(x_n))^{\top}$ and along equation (2.2), $\mathbf{e}_j = (\phi_j(x_1), \dots, \phi_j(x_n))^{\top}$.

2.2.2 Empirical Orthogonal Functions

We can apply a similar approach to space-time datasets such as those found in climate science. Climate data often contains grids of hundreds of locations, and each location corresponds to

a time series of a physical variable (e.g. temperature, precipitation, sea-level pressure, etc.). Neither a visual examination of the time evolution of the field, nor simple statistical summaries (mean, or variance) are much help in analyzing the overall dynamics. When the size of the phase space is a serious difficulty for the interpretation of the data, PCA is a popular tool to discriminate between fewer, but significant components, and a myriad of second-order processes with less relevance. In such contexts of gridded space-time data, we often talk about an empirical orthogonal function (EOF) decomposition instead of PCA.

The main difference between PCA and EOF is notational in nature. Thus, we introduce some notation. The state of the system at L locations at any time t can be represented by vector $\mathbf{y}(t) = (y_1(t), \dots, y_L(t))^\top$. We consider this vector a realization of a random vector $\mathbf{Y}(t)$. Thus, the decomposition (2.3) into principal components reads:

$$\mathbf{Y}(t) = \boldsymbol{\mu} + \sum_{i=1}^L U_i(t) \mathbf{e}_i. \quad (2.6)$$

In this case we have spatial index $i = 1, \dots, L$, (discrete) time index $t = 1, \dots, T$, a mean (or climatic) field $\boldsymbol{\mu}$, and \mathbf{e}_i , $i = 1, \dots, L$, the typical fields representing the main modes of variability of the system. Of course these fields are orthogonal. The basis functions $\{\mathbf{e}_1, \dots, \mathbf{e}_L\}$ are time-independent but are modulated by time-dependent principal component scores $U_i(t)$.

We now consider the *data matrix* \mathbf{Y} containing all observations,

$$\mathbf{Y} = \begin{pmatrix} y_1(t_1) & \dots & y_L(t_1) \\ y_1(t_2) & \dots & y_L(t_2) \\ \vdots & & \vdots \\ y_1(t_T) & \dots & y_L(t_T) \end{pmatrix} = \begin{pmatrix} \mathbf{y}(t_1)^\top \\ \mathbf{y}(t_2)^\top \\ \vdots \\ \mathbf{y}(t_T)^\top \end{pmatrix} \in \mathbb{R}^{T \times L}. \quad (2.7)$$

We work with centered data and define the *centered data matrix* as

$$\mathbf{Y}^c = \begin{pmatrix} (\mathbf{y}(t_1) - \hat{\boldsymbol{\mu}})^\top \\ \vdots \\ (\mathbf{y}(t_T) - \hat{\boldsymbol{\mu}})^\top \end{pmatrix}, \quad (2.8)$$

where $\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, \hat{\mu}_2, \dots, \hat{\mu}_L)^\top$ is the empirical mean of \mathbf{Y} . The l th component thereof is $\hat{\mu}_l = 1/T \sum_{t=1}^T y_l(t)$. The empirical *variance-covariance matrix* is simply $\mathbf{S} = 1/(T-1) \cdot \mathbf{Y}^{c\top} \mathbf{Y}^c$. EOF analysis now applies “PCA” to the centered data matrix. EOFs are now the counterpart to PCs and are — in the case of gridded data — often visualized as image plots. Individual scores receive less attention. Similarly, the classical biplot is not used. We will further illustrate this approach with an extensive example in Section 2.5.

2.3 Theoretical Aspects of PCA

2.3.1 PCA in Terms of Projections

In this section, we describe a procedure on how to find a new basis system that satisfies the criteria presented in Section 2.1. We will skip over some technical details. The first step in PCA

is to find the leading mode of variability \mathbf{e}_1 (subject to $\|\mathbf{e}_1\| = 1$) that minimizes the “error”, say ϵ_1 , made by ignoring the other dimension $2, \dots, p$. More specifically, this error is the difference between the original data and its projection on the sub-space spanned by \mathbf{e}_1 , i.e.:

$$\epsilon_1 = \|\mathbf{y} - (\mathbf{y}^\top \mathbf{e}_1) \cdot \mathbf{e}_1\|^2,$$

where $\mathbf{y}^\top \mathbf{e}_1$ is a scalar representing the length of the projection on the unit vector \mathbf{e}_1 (see Appendix B.3). After some calculations, one can show that ϵ_1 is minimized if and only if \mathbf{e}_1 is the eigenvector of the variance/covariance matrix $\Sigma = \text{Var}(\mathbf{y})$ associated with its largest eigenvalue λ_1 .

By iterating this procedure for the other modes of variability, one finds that the eigenvectors of the covariance matrix Σ build a basis system $\mathbf{e}_j, j = 1, \dots, p$, corresponding to the desired principal modes of variability. In other words, one (successively) finds normed vectors \mathbf{e}_j such that the projections of the data onto \mathbf{e}_j are maximized while ensuring each successive vector \mathbf{e}_{j+1} is orthogonal to all other vectors $\mathbf{e}_\ell, \ell = 1, \dots, j$.

2.3.2 PCA in Terms of Random Vectors

We consider (continuous) random vectors \mathbf{Y} having expectation $E(\mathbf{Y}) = \boldsymbol{\mu}$ and variance Σ , denoted $\mathbf{Y} \sim (\boldsymbol{\mu}, \Sigma)$, without specifying a distribution. Denote the eigenvalue-eigenvector decomposition of Σ as $\Sigma = \boldsymbol{\Gamma} \boldsymbol{\Lambda} \boldsymbol{\Gamma}^\top$. That means that $\boldsymbol{\Gamma}$ is the matrix containing the eigenvectors as columns and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues. We order the eigenvalues in descending order, i.e., $\boldsymbol{\Lambda} = \text{diag}(\lambda_1 \geq \dots \geq \lambda_p > 0)$. Then PCA can be formally defined as follows.

Definition 2.1. Let $\mathbf{Y} \sim (\boldsymbol{\mu}, \Sigma)$. The transformation $\mathbf{C} = \boldsymbol{\Gamma}^\top(\mathbf{Y} - \boldsymbol{\mu})$, where $\Sigma = \boldsymbol{\Gamma} \boldsymbol{\Lambda} \boldsymbol{\Gamma}^\top$, is the principal component transformation.

The principal components have the following (intuitive) property.

Property 2.1. $\mathbf{C} \sim (\mathbf{0}, \boldsymbol{\Lambda})$.

If we do not know the moments of \mathbf{Y} we use estimates and obtain (strictly speaking) the empirical principal component transformation. Under weak assumptions, the sample principal components and eigenvalues are the corresponding maximum likelihood estimates.

Hence, PCA can be seen as an eigen decomposition (also called spectral decomposition) of the empirical covariance matrix. The loadings represent the eigenvectors and `sdev` gives the standard deviations, i.e., the square root of the eigenvalues.

It is important to reiterate that in practice the variables have vastly different variances and thus should be rescaled (with the argument `scale=TRUE` of the function `prcomp()`, for example). If not, the PCs essentially retrieve the individual variables. With rescaling, PCA then translates to an eigenvalue decomposition of the empirical *correlation* matrix.

It is possible to manually perform PCA and create a biplot. As already mentioned, the scales for the latter are somewhat arbitrary. There might also be differences in the sign of the PC. This results from the undetermined sign of the eigen decomposition.



Example 2.3. The dataset `swiss` contains standardized fertility measure and socio-economic indicators for each of 47 French-speaking provinces of Switzerland at about 1888. R-Code 2.2 performs a PCA constructs manually a biplot, which is compared in Figure 2.4 with the standard version.

R-Code 2.2 Applying PCA to `swiss` dataset. Automatic and manual PCA and biplot. (See Figure 2.4.)

```

apply(swiss, 2, sd)
##          Fertility      Agriculture      Examination      Education
##          12.4917        22.7112        7.9779        9.6154
##          Catholic Infant.Mortality
##          41.7049        2.9127

pca <- prcomp(swiss, scale=TRUE)
print(pca, digits=3)

## Standard deviations (1, ..., p=6):
## [1] 1.789 1.090 0.921 0.663 0.452 0.348
##
## Rotation (n x k) = (6 x 6):
##          PC1     PC2     PC3     PC4     PC5     PC6
## Fertility -0.457  0.322 -0.1738  0.5356  0.3831 -0.4730
## Agriculture -0.424 -0.412  0.0383 -0.6429  0.3750 -0.3087
## Examination  0.510  0.125 -0.0912 -0.0545  0.8143  0.2240
## Education    0.454  0.179  0.5324 -0.0974 -0.0714 -0.6808
## Catholic     -0.350  0.146  0.8068  0.0995  0.1832  0.4022
## Infant.Mortality -0.150  0.811 -0.1601 -0.5268 -0.1045  0.0746
print(cumsum(pca$sdev^2)/6, digits=3) # see also screeplot(pca)
## [1] 0.533 0.731 0.873 0.946 0.980 1.000
biplot(pca, scale=0)      # no other scaling here!
## This argument is not well documented (?biplot.princomp).
# Now the manual decomposition:
manu <- eigen(cor(swiss))      # manually performing PCA
manu$sdev <- sqrt(manu$values)
cswiss <- scale(swiss)
plot(cswiss %*% manu$vect[,1:2], xlab="PC1", ylab="PC2", ylim=c(-3, 5))
## transposes cancel
text(cswiss %*% manu$vect[,1:2], rownames(swiss), col=3)
arrows(0, 0, manu$vect[,1]*5, manu$vect[,2]*5, col=2)
## or some other scaling of the vectors

```

An eigen decomposition is not the only approach for extracting the relevant quantities (load-

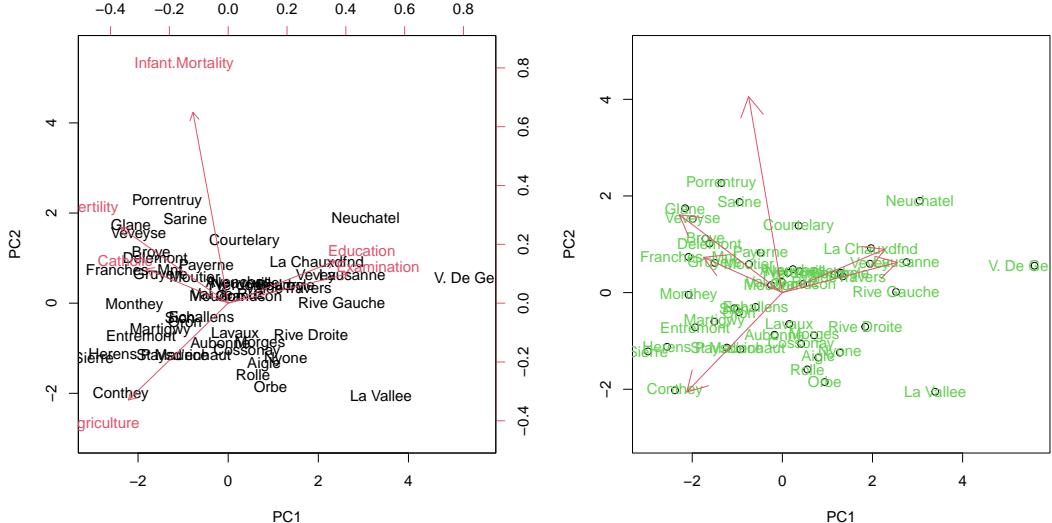


Figure 2.4: Automatic (left) and manual (right) biplot of `swiss` dataset. (See R-Code 2.2.)

ings and scores) from the data. As a matter of fact, it is advisable to use the singular value decomposition (SVD) method found in `prcomp()`, instead of the eigen decomposition-based `princomp()` because of its computational efficiency and flexibility. SVD avoids explicitly computing the covariance matrix and instead computes singular vectors and singular values based on the data.

2.3.3 Truncation

In order to use PCA for dimension reduction, a value k at which to truncate the decomposition $\mathbf{y} \approx \sum_{j=1}^k u_j e_j$ must be chosen. Unfortunately, no unambiguous rule exists to find k . One method is to use the k PCs that account for an arbitrary threshold for the fraction of explained variance $R^2 = \sum_{\ell=1}^k \lambda_\ell / \sum_{j=1}^p \lambda_j$, say 0.75. Other approaches less prone to subjectivity exist.

The first technique relies on the fact that the variance λ_j of Gaussian noise decays in an exponential manner. In Figure 2.5, the spectrum starts to decay as random noise around $k = 10$. The second technique is called *North's rule of thumb*. The uncertainty of the eigenvalues due to measurement errors can be approximated by $\Delta \lambda_j \approx \sqrt{2/n} \lambda_j$. The idea of the North's rule is to truncate the expansion when the sampling error becomes larger than the distance between two eigenvalues, i.e., when $\lambda_j - \lambda_{j+1} < \Delta \lambda_j$ (North *et al.*, 1982).

Example 2.4. For the `europejobs` and `swiss` dataset example, we would need three and two PCs respectively to explain roughly 75% or the total variation. North's rule does not work well here, likely because we have a very small n and p .



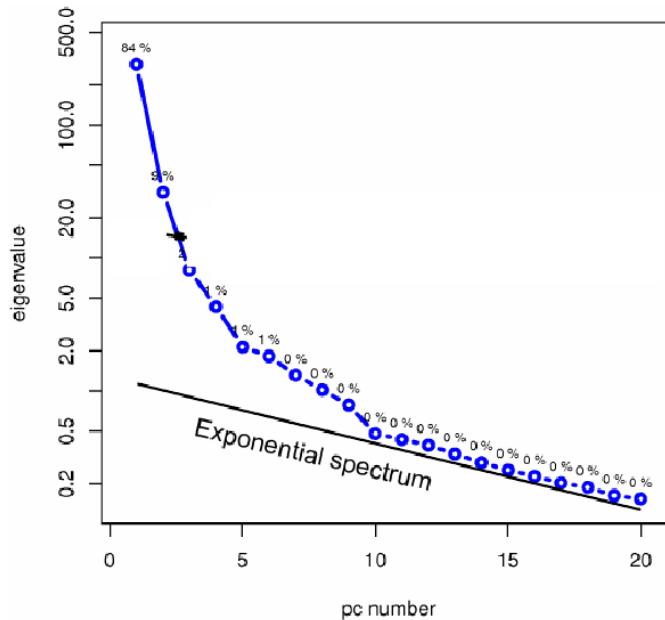


Figure 2.5: PCA variances plotted on a logarithmic scale. A Gaussian noise would decay exponentially, as indicated by the black line.

2.4 PCA and Related Decompositions and Concepts

In this section we discuss two alternative decompositions or projection approaches that are quite similar to PCA.

2.4.1 Canonical Correlation Analysis

PCA can be seen as finding a vector \mathbf{b} such that $\text{Var}(\mathbf{b}\mathbf{Y})$ is maximized, constrained to unitary vectors $\|\mathbf{b}\| = 1$. We now consider two random vectors \mathbf{X} and \mathbf{Y} (or a decomposition of \mathbf{Y} in two components). In canonical correlation analysis (CCA), we want to determine vectors \mathbf{a} and \mathbf{b} such that $\text{Corr}(\mathbf{a}^\top \mathbf{X}, \mathbf{b}^\top \mathbf{Y})$ is maximized. Naturally, we need to constrain the vectors \mathbf{a} and \mathbf{b} by choosing vectors that have a predefined norm.

Example 2.5. CCA for `LifeCycleSavings` data might seek a linear combination of the economic (`sr`, `dpi` and `ddpi`) variables that is highly correlated with different age strata of the population (`pop15` and `pop75`).

The output of the function `cancor` (see R-Code 2.3) shows the normalized vectors in the first column of the components `xcoef` and `ycoef`. The successive columns represent linear combinations that are orthogonal to the previous one and successively maximize the correlation. They are rarely used in practice and thus not considered here. The correlation induced by the (first) canonical vectors \mathbf{a} and \mathbf{b} is $\text{Corr}(\mathbf{a}^\top \mathbf{X}, \mathbf{b}^\top \mathbf{Y}) = 82.5\%$.

The values of the components of the canonical vectors are not a surprise. Both `pop15` and `pop75` are highly negatively correlated and the respective correlations with all other variables have the same sign. Hence, it suffices to pick just one of the two.

The components of the canonical vectors are normed, as shown at the end of the code. This normalization is now further elaborated.



R-Code 2.3 Sensible CCA for *LifeCycleSavings* data.

```

pop <- LifeCycleSavings[, 2:3]      # data.frames here...
eco <- LifeCycleSavings[, -(2:3)]
(out <- cancor(pop, eco) )           # in package:stats

## $cor
## [1] 0.82480 0.36528
##
## $xcoef
## [,1]      [,2]
## pop15 -0.0091109 -0.036222
## pop75  0.0486475 -0.260312
##
## $ycoef
## [,1]      [,2]      [,3]
## sr    0.00847102 3.3379e-02 -5.1571e-03
## dpi   0.00013074 -7.5882e-05  4.5437e-06
## ddpi  0.00417060 -1.2268e-02  5.1883e-02
##
## $xcenter
## pop15  pop75
## 35.090  2.293
##
## $ycenter
##       sr      dpi      ddpi
## 9.6710 1106.7584  3.7576

cor(c(as.matrix(pop) %*% out$xcoef[,1] ),  # we need vectors!
    c(as.matrix(eco) %*% out$ycoef[,1] )     )

## [1] 0.8248

cor(LifeCycleSavings[,c(2,3,1,4,5)] )      # reorder for better interpretability

##      pop15      pop75      sr      dpi      ddpi
## pop15 1.000000 -0.908479 -0.45554 -0.75619 -0.047826
## pop75 -0.908479  1.000000  0.31652  0.78700  0.025321
## sr    -0.455538  0.316521  1.00000  0.22036  0.304787
## dpi   -0.756188  0.787000  0.22036  1.00000 -0.129486
## ddpi  -0.047826  0.025321  0.30479 -0.12949  1.000000

cov(as.matrix(pop) %*% out$xcoef)*(nrow(pop)-1)  # normalization!

##      [,1]      [,2]
## [1,] 1.000e+00 4.221e-16
## [2,] 4.221e-16 1.000e+00

```

The vectors \mathbf{a} and \mathbf{b} are calculated based on eigenvectors of products of the underlying covariance matrices. More specifically, let $\boldsymbol{\Sigma}_{\mathbf{XX}} = \text{Var}(\mathbf{X})$, $\boldsymbol{\Sigma}_{\mathbf{YY}} = \text{Var}(\mathbf{Y})$, and $\boldsymbol{\Sigma}_{\mathbf{XY}} = \text{Cov}(\mathbf{X}, \mathbf{Y})$. The parameter to maximize is

$$\rho = \frac{\mathbf{a}^\top \boldsymbol{\Sigma}_{\mathbf{XY}} \mathbf{b}}{\sqrt{\mathbf{a}^\top \boldsymbol{\Sigma}_{\mathbf{XX}} \mathbf{a}} \sqrt{\mathbf{b}^\top \boldsymbol{\Sigma}_{\mathbf{YY}} \mathbf{b}}} = \frac{\mathbf{c}^\top \boldsymbol{\Sigma}_{\mathbf{XX}}^{-1/2} \boldsymbol{\Sigma}_{\mathbf{XY}} \boldsymbol{\Sigma}_{\mathbf{YY}}^{-1/2} \mathbf{d}}{\sqrt{\mathbf{c}^\top \mathbf{c}} \sqrt{\mathbf{d}^\top \mathbf{d}}}, \quad (2.9)$$

where $\mathbf{c} = \boldsymbol{\Sigma}_{\mathbf{XX}}^{1/2} \mathbf{a}$ and $\mathbf{d} = \boldsymbol{\Sigma}_{\mathbf{YY}}^{1/2} \mathbf{b}$. The next steps require the use of the Cauchy-Schwarz inequality and the solution is based on the eigenvectors from the Rayleigh quotient. Further, $\mathbf{a}^\top \boldsymbol{\Sigma}_{\mathbf{XX}} \mathbf{a} = \mathbf{b}^\top \boldsymbol{\Sigma}_{\mathbf{YY}} \mathbf{b} = 1$. Some other constant works as well; R uses $n - 1$, see last line of code in R-Code 2.3.

Naturally, the correlation is larger than the largest entry of “Corr(\mathbf{X}, \mathbf{Y})”, otherwise we would pick the corresponding variables having largest correlation.

Example 2.6. If one of the dimension of \mathbf{X} or \mathbf{Y} is one, canonical correlation is equivalent to linear least-squares regression, illustrated in R-Code 2.4. We get more details in Chapter 5 but the idea is as follows. Minimizing the squared residuals is equivalent to minimizing the angle between \mathbf{y} and $\hat{\mathbf{y}}$ and to maximizing the correlation between \mathbf{y} and $\hat{\mathbf{y}}$. ♣

R-Code 2.4 Linking CCA to multiple linear regression.

```
rcc <- cancor(LifeCycleSavings[,1], LifeCycleSavings[,-1]) # CCA
str(rcc, give.attr=FALSE, strict.width="cut")

## List of 5
## $ cor      : num 0.582
## $ xcoef    : num [1, 1] 0.0319
## $ ycoef    : num [1:4, 1:4] -2.53e-02 -9.27e-02 -1.85e-05 2.25e-02 -2.48e...
## $ xcenter: num 9.67
## $ ycenter: Named num [1:4] 35.09 2.29 1106.76 3.76
# print(rcc)
rlm <- lm(sr ~ ., data=LifeCycleSavings)    # regression with all variables
# summary(rlm)
coef(rlm)[-1]/rcc$ycoef[,1]                  # betas equal to weights, up to constant
## pop15  pop75    dpi   ddpi
## 18.246 18.246 18.246 18.246
rcc$cor/sqrt(summary(rlm)$r.squared) # correlation equal to square root of R^2
## [1] 1
```

2.4.2 Factor Analysis

Factor analysis is useful for investigating variable relationships for complex concepts. It tries to explain the variability among observed, correlated variables in terms of a few underlying factors. For example, socioeconomic status may be of interest but this is hard to measure directly. There

are many variables we can measure, however, that we believe relate to socioeconomic status, such as occupation, education, and home ownership. The underlying factors (also called latent variables) are hard/impossible to measure directly but we believe some observed variables (also called manifest variables) to be indicators of the factors.

Before we describe the exact model, we look at a very classical example.

Example 2.7. The dataset `scor` consists of the scores earned by 88 students on five exams, some of which were open book exams and others closed book exams (Mardia et al., 1979). The variables are:

- `mec`: mechanics, closed book
- `vec`: vectors, closed book
- `alg`: algebra, open book
- `ana`: analysis, open book
- `sta`: statistics, open book

A short illustration of the data is given at the beginning of R-Code 2.5. All variables seem to be quite related. However, a single explanatory variable would not be sufficient to explain the performance of the student in all of the variables.

A factor analysis tries to explain the underlying correlation in the scores in terms of “unknown” and “unobservable” factors, like “intelligence” (overall component), “recitation ability” (for closed book exams), etc. The factor analysis is performed in R-Code 2.5. The first factor explains an overall effect and the second discriminates between closed book and not (`loadings`). ♣

R-Code 2.5: EDA and factor analysis using the `scor` dataset.. (See Figure 2.6.)

```

data(scor, package="bootstrap")
apply(scor, 2, summary)
##          mec     vec     alg     ana     sta
## Min.    0.000  9.000 15.000  9.000  9.000
## 1st Qu. 30.000 42.000 45.000 35.750 31.000
## Median  41.500 51.000 50.000 49.000 40.000
## Mean    38.955 50.591 50.602 46.682 42.307
## 3rd Qu. 49.250 60.000 57.250 57.000 51.500
## Max.    77.000 82.000 80.000 70.000 81.000

apply(scor, 2, sd)
##      mec     vec     alg     ana     sta
## 17.486 13.147 10.625 14.845 17.256

pairs(scor, gap=0, pch=20)
fa <- factanal(scor, factors=2, rotation="none") # no rotation
fa

```

```

## Call:
## factanal(x = scor, factors = 2, rotation = "none")
##
## Uniquenesses:
##   mec   vec   alg   ana   sta
## 0.466 0.419 0.189 0.352 0.431
##
## Loadings:
##          Factor1 Factor2
## mec      0.628   0.373
## vec      0.695   0.312
## alg      0.899
## ana      0.780  -0.201
## sta      0.727  -0.200
##
##          Factor1 Factor2
## SS loadings     2.824   0.319
## Proportion Var  0.565   0.064
## Cumulative Var 0.565   0.629
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 0.07 on 1 degree of freedom.
## The p-value is 0.785

```

Similar to PCA, factor analysis is a dimension reduction tool. In contrast, to PCA we group variables into fewer sets of new variables (called factors). These factors can be interpreted and typically help to gain insights the underlying mechanism driving the observed process.

For a factor analysis we need to assume a distribution for our observations. As typically done in this context, we assume $\mathbf{Y} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, i.e., p -variate Gaussian distribution. Similar to PCA, the mean $\boldsymbol{\mu}$ is not of interest and focus is on a “decomposition” of $\boldsymbol{\Sigma}$. The underlying model of factor analysis is as follows:

$$\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{F} + \mathbf{U}, \quad (2.10)$$



where $\boldsymbol{\mu}$ is a p -vector of means, $\boldsymbol{\Lambda}$ a $p \times k$ -matrix ($k < p$) of unknown constants. Further, \mathbf{F} is k -variate Gaussian and \mathbf{U} is p -variate Gaussian with

$$\mathbb{E}(\mathbf{F}) = \mathbf{0}, \quad \mathbb{E}(\mathbf{U}) = \mathbf{0}, \quad (2.11)$$

$$\text{Var}(\mathbf{F}) = \mathbf{I}, \quad \text{Var}(\mathbf{U}) = \boldsymbol{\Psi} = \text{diag}(\psi_1, \dots, \psi_p), \quad \text{Cov}(\mathbf{F}, \mathbf{U}) = \mathbf{0} \quad (2.12)$$

Hence,

$$\text{Var}(\mathbf{Y}) = \boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi} \quad (2.13)$$

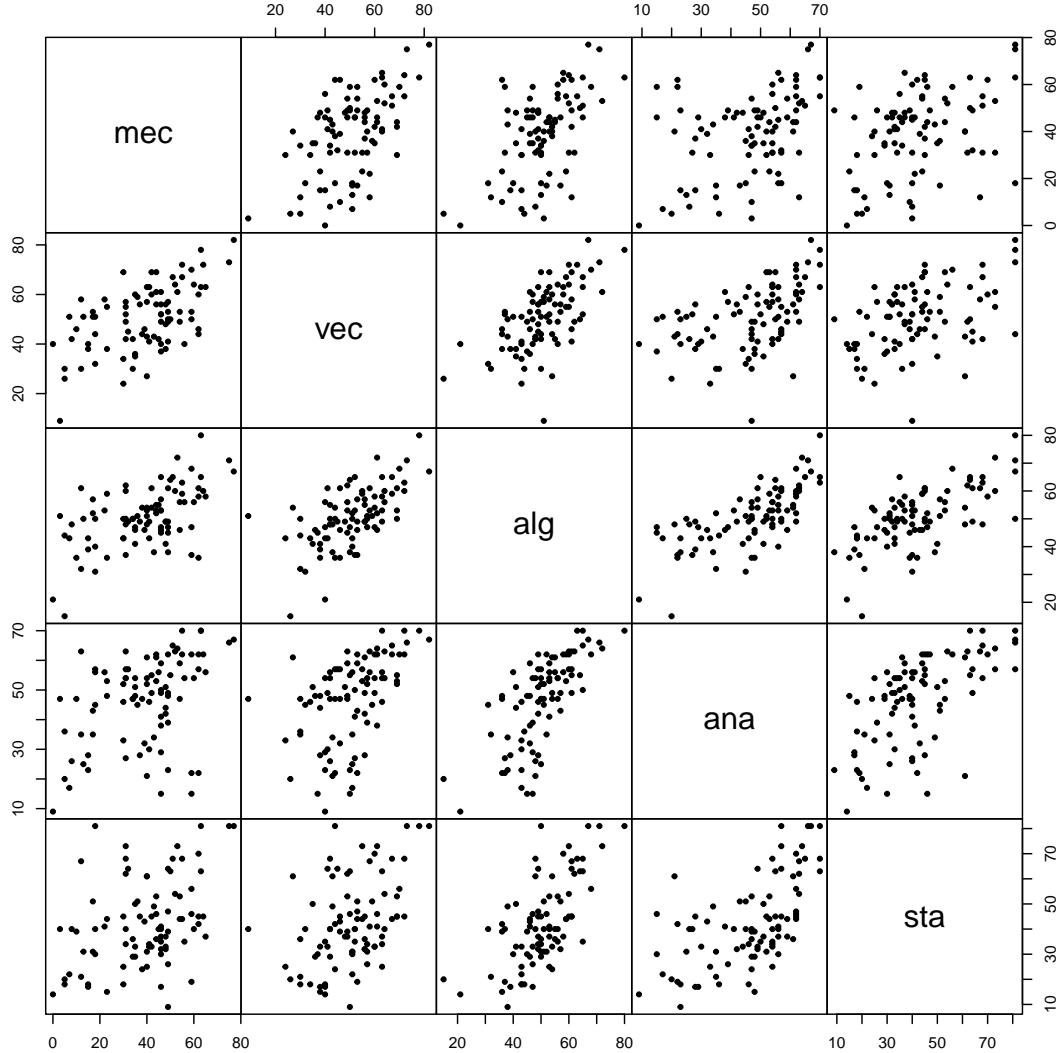


Figure 2.6: Scatter plot of the `scor` dataset. (See R-Code 2.5.)

and a factor analysis consists of decomposing Σ into Λ and Ψ . Similar to PCA, it is possible to decompose the covariance matrix or the associated correlation matrix. The second is more common and done here and the function `factanal()`.

The elements of Λ are called loadings, the diagonal elements of Ψ determine the so-called *uniquenesses* and the elements \mathbf{F} are called *factors* or scores. Estimation of the parameters is done using maximum likelihood, the number of factors is fixed before the decomposition. But because of the distributional assumption on the data, tests can be constructed as indicated at the end of R Code 2.5.

Notice that the factors Λ and Ψ are not uniquely determined. For example, let \mathbf{C} be an orthogonal $k \times k$ -matrix ($\mathbf{CC}^\top = \mathbf{I}$) and set $\mathbf{F} = \mathbf{CF}$. The resulting variance of \mathbf{Y} remains the same. This can be resolved by imposing an extra condition, such as $\Lambda^\top \Psi^{-1} \Lambda$ is diagonal with its elements in decreasing order. Instead of imposing more conditions, one could also use the non-uniqueness of the solution to aid in interpretation. Results of a factor analysis may be difficult to interpret, but rotating the results may simplify them. After the number of factors has

been chosen we can rotate the factors such that loadings on various factors tend to be very high ($-1, 1$) or very low (0). The default rotation in `factanal()` is varimax, an orthogonal rotation in which rotated factors are restricted to being uncorrelated.

Example 2.8. In this example, we illustrate the factor analysis decomposition of the correlation matrix of `LifeCycleSavings` data. R-Code 2.6 reconstructs the correlation matrix for two factors, based on the five variables. The decomposition is quite good, as the degrees of freedom of the factor analysis match the degrees of freedom of the correlation matrix. Using a `varimax` rotation results in a coordinate system rotated by 11 degrees; but the rotation the factors does not improve the fit.

Figure 2.7 illustrates that the two factors are quite similar to first two principal components in relative terms.



R-Code 2.6: Comparison of factor analysis results with a PCA biplot. (See Figure 2.7.)

```

fa <- factanal(LifeCycleSavings, factors=2, rotation="none")
fa$loadings

##
## Loadings:
##          Factor1 Factor2
## sr      0.436   0.760
## pop15 -0.959
## pop75  0.953  -0.125
## dpi    0.800  -0.187
## ddpi     0.375
##
##          Factor1 Factor2
## SS loadings   2.661   0.771
## Proportion Var 0.532   0.154
## Cumulative Var 0.532   0.686

diag(fa$uniquenesses) + (fa$loadings)%*%t(fa$loadings) - cor(LifeCycleSavings)

##          sr      pop15      pop75      dpi      ddpi
## sr     -6.9915e-08 1.3603e-03 3.9851e-03 -1.3514e-02 -8.8157e-04
## pop15  1.3603e-03 -7.4334e-08 3.2707e-04 -2.4475e-03 -1.2147e-02
## pop75  3.9851e-03  3.2707e-04 5.3404e-08 -7.6601e-04 -3.0301e-02
## dpi    -1.3514e-02 -2.4475e-03 -7.6601e-04  2.0830e-06  9.4581e-02
## ddpi   -8.8157e-04 -1.2147e-02 -3.0301e-02  9.4581e-02 -5.3645e-06

plot(-fa$loadings, ylim=c(-1,1), xlim=c(-1,1), type="n")
arrows(0,0, -fa$loadings[,1],-fa$loadings[,2], col=2, length=0.1)
text(-fa$loadings*1.1, names(LifeCycleSavings))
abline(h=0,v=0, col=7)           # unrotated coordinate system

```

```

fa2 <- factanal(LifeCycleSavings, factors=2) # varimax rotation
acos(fa2$rotmat[1,1])*180/pi # rotation angle
## [1] 11.802
abline(a=0, b=fa2$rotmat[1,1]/fa2$rotmat[2,1], col=3) # rotated
abline(a=0, b=fa2$rotmat[1,2]/fa2$rotmat[2,2], col=3) # coordinate system
biplot(pc <- prcomp(LifeCycleSavings, scale=TRUE)) # PCA and biplot
pc
## Standard deviations (1, ..., p=5):
## [1] 1.67990 1.12074 0.77751 0.48954 0.27872
##
## Rotation (n x k) = (5 x 5):
##          PC1      PC2      PC3      PC4      PC5
## sr     -0.308462 -0.554246  0.750144 -0.13015 -0.134196
## pop15  0.570653 -0.012673  0.026549 -0.41621 -0.707291
## pop75 -0.560431  0.100994 -0.211830  0.39043 -0.691674
## dpi    -0.513506  0.266302 -0.145016 -0.80134  0.047034
## ddpi   -0.037872 -0.782007 -0.608833 -0.12348  0.033263

```

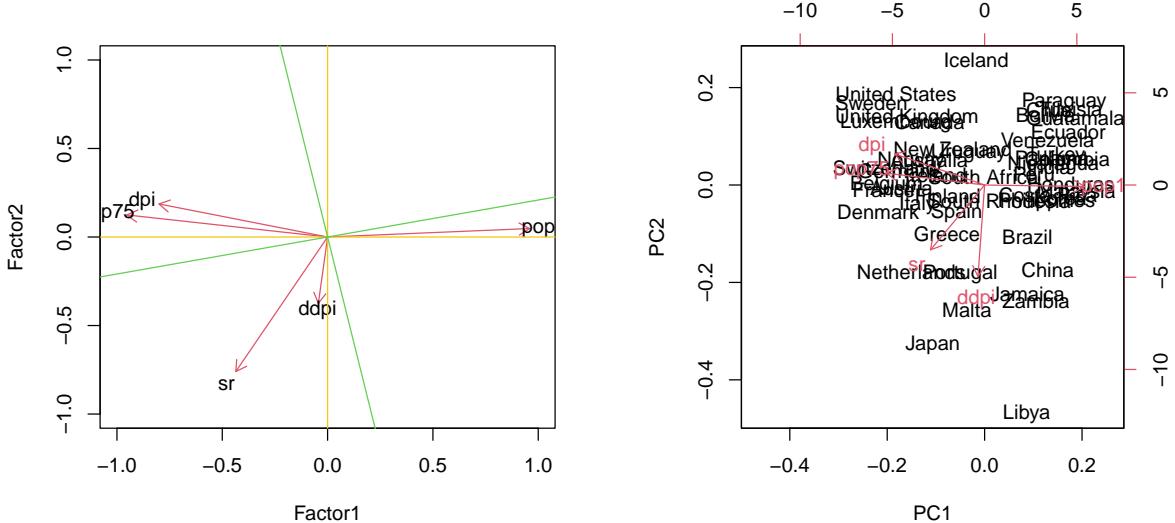


Figure 2.7: Left: factors in unrotated (yellow) and rotated (green) coordinate system of a factor analysis of the `LifeCycleSavings` dataset. Right: biplot of a PCA of the same dataset. (See R-Code 2.6.)

PCA is quite similar to factor analysis. A complete discussion of the similarities and difference is beyond the scope of this course but browsing the web yields many opinions — typically somewhat biased towards the writer's background; see, e.g., Suhr (2009).

2.5 Example: Sea-Level Pressure Over the Northern Atlantic

The spatial distribution of atmospheric pressure over the northern Atlantic Ocean plays an important role in European climate. Here, the *ERA Interim* sea-level pressure dataset provided by the European Centre for Medium-Range Weather Forecasts (ECMWF) is used. A subset of 225 winter monthly averages (November, December, January, February, and March) between 1957–2002 is analyzed. The data is represented on a regular 2.5° grid covering the region of interest and involves $43 \times 19 = 817$ locations. A map of the winter climatic field is shown in Figure 2.8 and shows the typical contrast between the Icelandic low and Canaries high pressure systems. This field corresponds to μ in the nomenclature presented in the previous section. R-Code 2.7 loads the data, indicate how to construct an animation of the pressure and plots the mean climatic field. As the data is stored in a so-called netCDF format, data preparation seems lengthy.

R-Code 2.7: Visualizing the pressure and climatic field of the *ECMWF* data. (See Figure 2.8.)

```

library(ncdf4)
library(fields)      # package maps is available through package fields
nc <- nc_open("download/sta121/ERA40_slp.nc") # Load pressure data
# str(nc)           # uncomment only if you are brave!
tmp <- ncvar_get(nc)
## Geographic selection:
LON <- nc$dim$lon$vals
LAT <- nc$dim$lat$vals
ilon <- (LON>=300) | (LON<=45) # Restrict to 300--45 degrees
ilat <- (LAT>=30) & (LAT<=75) # 30--75 degrees
lon <- LON[ ilon]
lat <- LAT[ ilat]
## Select NDJFM winter months:
time <- as.vector(unlist(read.table(
  file="download/sta121/ERA40_slp_time.dat", sep=",",)))
months <- as.numeric(substr(time,5,6))
tsel <- months %in% c(11,12,1,2,3)
time <- time[ tsel]
dat <- tmp[ ilon, ilat, tsel]
## Some manipulations, so we can simply plot with `image.plot`:
olat <- order(lat)
lat <- lat[ olat]
dat <- dat[ ,olat,]
iilon <- lon>50
lon[iilon] <- lon[ iilon]-360
olon <- order(lon)
lon <- lon[ olon]
```

```

dat <- dat[,olon,]/1000 # work with kPa
dim(dat) # lon, lat, time
## [1] 43 19 225

## Extract the mean and sd:
mu <- apply(dat, MAR=c(1,2), FUN=mean)
image.plot(lon, lat, mu, horizontal=TRUE, main="Mean field")
map("world", add=T)
image.plot(lon, lat, apply(dat, MAR=c(1,2), FUN=sd), horizontal=TRUE,
           main="Standard deviations")
map("world", add=T)

## Visualize time evolution:
for (t in seq(dim(dat)[3])){
  image.plot(lon, lat, dat[,t], main=time[t], zlim=range(dat))
  map("world", add=T)
  Sys.sleep(0.05)
}

```

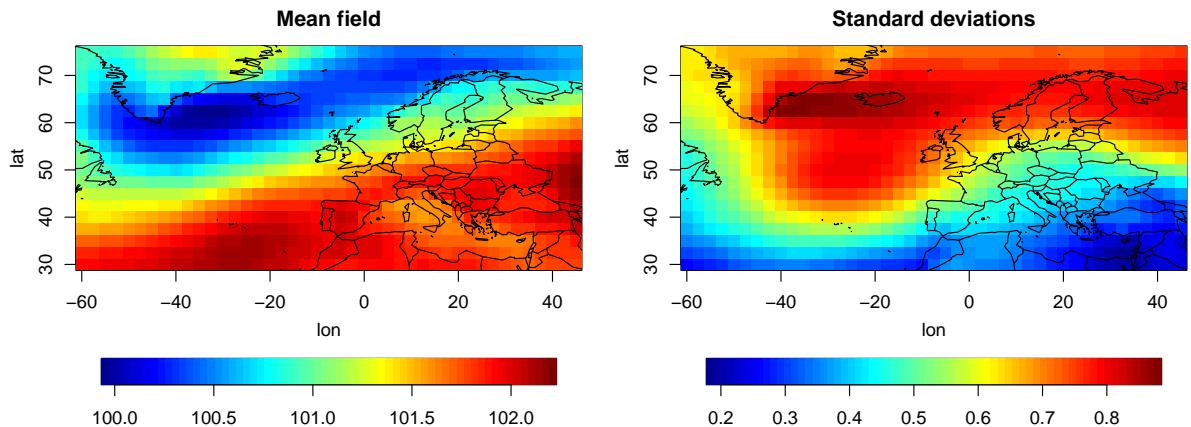


Figure 2.8: Average winter sea-level pressure (in kPa) for the period 1957–2002. Standard deviations in the right panel. (See R-Code 2.7.)

Here, we have a data matrix $\mathbf{Y} \in \mathbb{R}^{T \times L}$ with $T = 225$ months and $L = 817$ locations, based on a 19×43 latitude/longitude grid. Notice that T corresponds to n and L corresponds to p . That means that we have more “dimensions” than observations! A decomposition of the empirical variance-covariance matrix \mathbf{S} has to be based on a singular value decomposition (instead of an eigen-decomposition).

As the relevant quantities are used later on, we manually subtract the mean. The function `prcomp()` is applied to the centered data matrix \mathbf{Y}^c . The first four modes of variability e_j , $j = 1, \dots, 4$, are shown in Figure 2.9, and explain 41%, 20%, 15% and 10% of the total variability

(see Figure 2.9, lower row). The first loading (PC1) corresponds to a north-south contrast, the second loading (PC2) corresponds to an east-west contrast, while the third loading (PC3) corresponds to a pressure dipole between the British Isles and Greenland. The forth loading (PC4) is dipole like with a strong center over the north western part of the north Atlantic and secondary center located over north eastern Siberia and opposite center over western Europe.

R-Code 2.8 Loadings and (normalized) scree plot. (See Figure 2.9.)

```

Y <- dat - array(mu, dim=dim(dat)) # Centering the data
Y <- t(array(Y, dim=c(dim(dat)[1]*dim(dat)[2], dim(dat)[3])))
pca <- prcomp(Y)

for (i in (1:4)){                      ## Plot the first modes of variabilty
  ei <- pca$rotation[,i]
  ei <- array(ei, dim=dim(dat)[c(1,2)])
  image.plot(lon, lat, ei, main=dimnames(pca$x)[[2]][i], horizontal=TRUE)
  map("world",add=T)
}

npcs <- 21
screeplot(pca, type="l", main="", npcs=npcs)
# plot( log(pca$sdev[1:npcs]^2), type="b" ) # Screeplot on log scale
plot(cumsum(pca$sdev[1:npcs]^2)/sum(pca$sdev^2), type="b",
      ylim=c(0,1), axes=FALSE, ylab="% variance explained", xlab="")
abline(h=1, col="gray")
axis(2)
axis(1, at=1:npcs, labels=paste(1:npcs))
print(rbind(explained.var=pca$sdev[1:10]^2/sum(pca$sdev^2),
            cumulative.var=cumsum(pca$sdev[1:10]^2)/sum(pca$sdev^2)), 2)
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## explained.var 0.41 0.20 0.15 0.10 0.03 0.026 0.02 0.016 0.011 0.0066
## cumulative.var 0.41 0.61 0.75 0.86 0.89 0.912 0.93 0.948 0.959 0.9657

```

The fraction of variance explained by the first four principal modes is $\sum_{\ell=1}^4 \lambda_\ell / \sum_{j=1}^L \lambda_j = 0.86$. This result indicates that 86% of the information can be reconstructed using only 4 PCs. In other words, only 4 time series are because the transformed variables $u_j(t)$, $j = 1, \dots, 4$, make for a fairly good approximation of the original data. In contrast, the original contains 817 time series $y_j(t)$, $j = 1, \dots, 817$. Therefore, the compression achieved by PCA decomposition is very large. As an illustration, the left panel of Figure 2.10 shows a reconstructed pressure field for January 1960. The reconstruction is very close to the original field (R-Code 2.9 and right panel of Figure 2.10). This indicates that the PCA helps to filter redundancy and/or noise contained in the original data. Hence, PCA can also be seen as a very rudimentary data compression.

Of course here in this EOF approach, we use our classical PCA decomposition. Here, it is

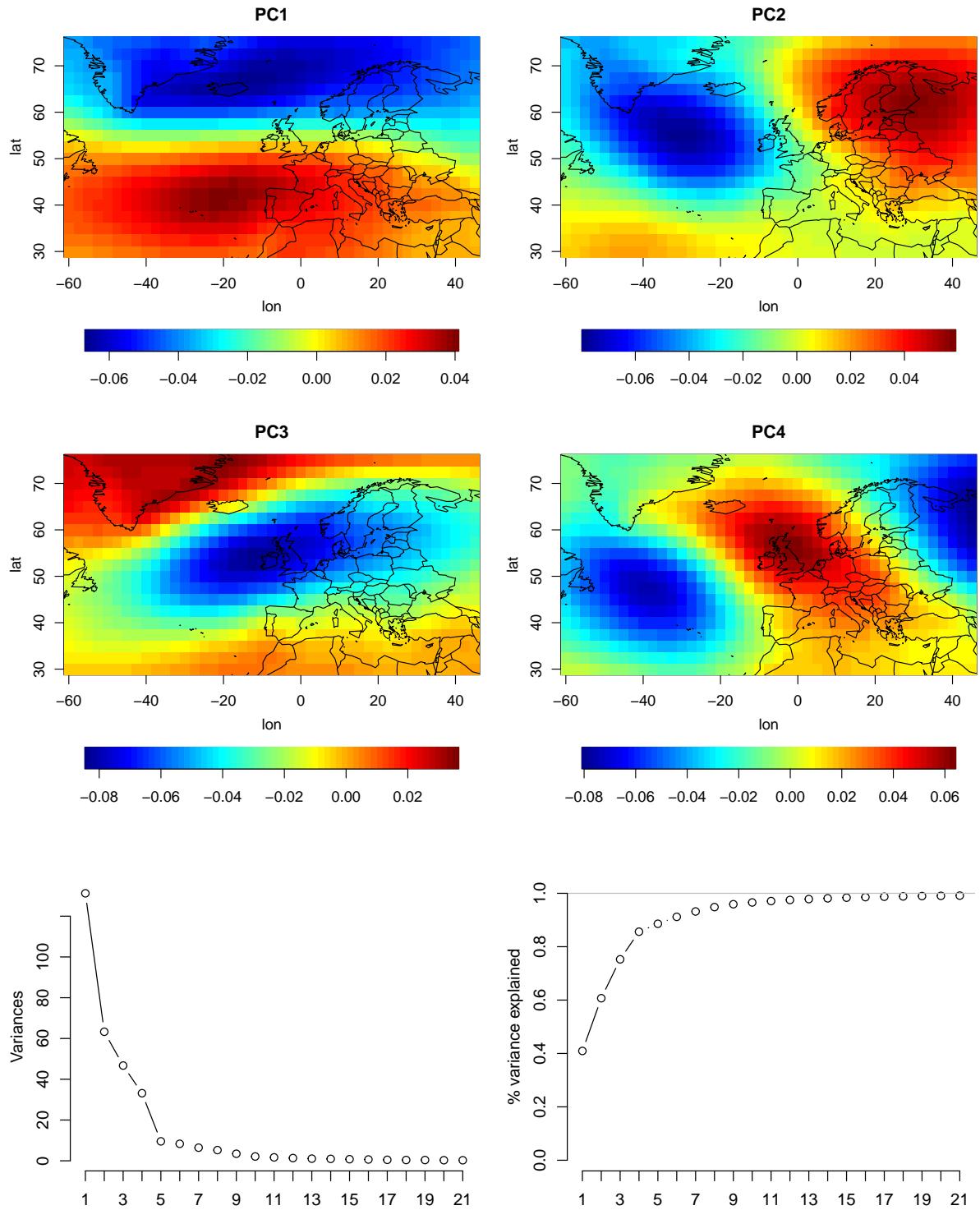


Figure 2.9: First four principal components (loadings). Bottom row shows a scree plot and the explained fraction of total variability. (See R-Code 2.8.)

easier to directly “attach” the dimensions to the corresponding matrix and we write, for example, $\mathbf{Y}_{T \times L}$ instead of $\mathbf{Y} \in \mathbb{R}^{T \times L}$. Using matrix notation, the decomposition (2.6) can be conveniently rewritten as $\mathbf{Y}_{T \times L}^c = \mathbf{U}_{T \times L} \mathbf{E}_{L \times L}^\top$, where matrix \mathbf{U} contains the scores and matrix \mathbf{e} contains the basis vectors arranged in columns, i.e., $\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_L)$. A reconstruction of the data with

a limited set of k PCs is written in matrix notation as $\mathbf{Y}_{T \times L}^c \approx \mathbf{U}_{T \times k} \mathbf{E}_{L \times k}^\top$, where the matrices are truncated after k columns.

R-Code 2.9 Reconstructing a field using the first four principal components. (See Figure 2.10.)

```

trunc <- 4
E <- pca$rotation
U <- pca$x
Y.rec <- U[,1:trunc] %*% t(E[,1:trunc]) # vector
Y.rec <- array(t(Y.rec), dim=dim(dat)) # to visualize we need a grid
t <- 13
time[t] # January 1960
## [1] 196001
orig <- dat[, , t]
rec <- mu + Y.rec[, , t]

# image.plot(lon, lat, orig, main="Original")
# map("world", add=T)
image.plot(lon, lat, rec, main="Reconstructed", horizontal=TRUE)
map("world", add=T)
image.plot(lon, lat, rec-orig, main="Error", horizontal=TRUE)
map("world", add=T)
summary(c(rec-orig)/c(rec)) # look how small the relative error is!
##      Min.    1st Qu.     Median      Mean    3rd Qu.      Max.
## -0.004633 -0.002303 -0.000691 -0.000702  0.000861  0.004156

```

2.6 Bibliographic remarks

We mention Jolliffe (2002) and Mardia *et al.* (1979) for general and accessible accounts of PCA. Many other books treat the subject.

Detailed discussions about EOFs, including examples beyond meteorological and climate data, can be found in von Storch and Zwiers (1999); Wilks (2005). Information about the North Atlantic Oscillation can be found in, e.g., Barnston and Livezey (1987) or Hurrell *et al.* (2013).

Selecting the number of relevant PC is still an unsolved problem. Minka (2000); Gavish and Donoho (2014); Choi *et al.* (2015) are recent (technical) attempts towards some solution, all relying of formal model assumptions. The link <https://stats.stackexchange.com/questions/33917> contains detailed information and some helpful simulations on how to determine k , the number of necessary PC to reconstruct a signal.

Nice blog about CCA is found at <http://gregorygundersen.com/blog/2018/07/17/cca/>.

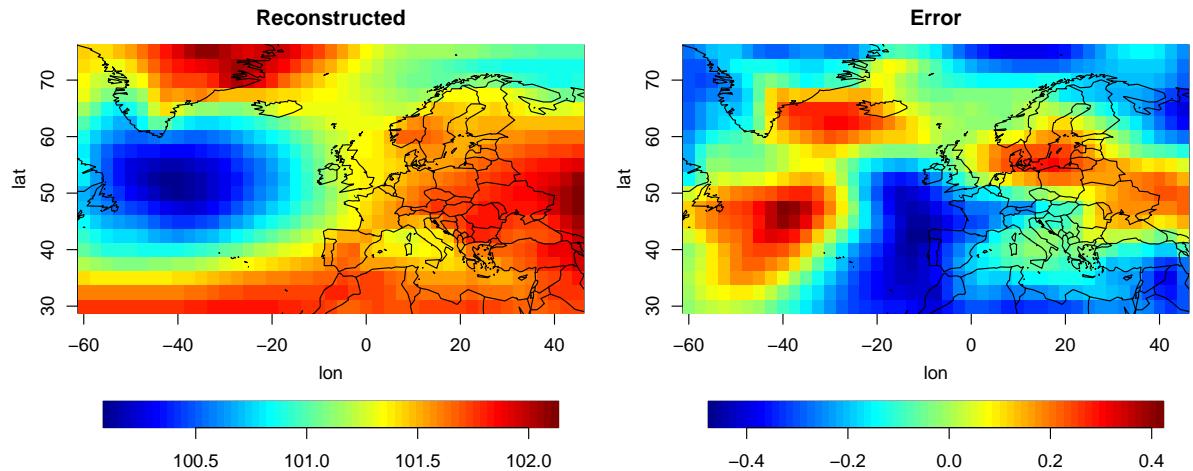


Figure 2.10: Sea-level pressure (in kPa) for January 1960: reconstructed field using the first four principal components (left) and error (right). Relative error ranges is within ± 0.005 . (See R-Code 2.9.)

2.7 Exercises and Problems

Problem 2.1 (PCA with a classical dataframe) The dataset `protein.txt`, available from <https://www.math.uzh.ch/furrer/download/sta121/protein.txt>, gives measured protein consumption in twenty-five European countries for nine food groups. The variables are:

<i>Country:</i>	Country name
<i>RdMeat:</i>	Red meat
<i>WhMeat:</i>	White meat
<i>Eggs:</i>	Eggs
<i>Milk:</i>	Milk
<i>Fish:</i>	Fish
<i>Cereal:</i>	Cereals
<i>Starch:</i>	Starchy foods
<i>Nuts:</i>	Pulses, nuts, and oil-seeds
<i>Fr&Veg:</i>	Fruits and vegetables

(Weber, A. (1973) Agrarpolitik im Spannungsfeld der internationalen Ernaehrungspolitik, Institut fuer Agrarpolitik und Marktlehre, Kiel.)

Use multivariate methods to determine whether there are groupings of countries and whether meat consumption is related to that of other foods.

- a) Perform a exploratory data analysis. Look at the correlations. (hint: use the function `cor()`, or `corrplot()` from the package `corrplot`). Do you think that applying PCA will lead to a important reduction in the number of dimension, while explaining a high percentage of the total variability?

- b) Perform a PCA using the function `princomp()`. Did you perform PCA using the correlation or covariance matrix? Comment.
- c) How many PC dimensions would you keep? Why?
- d) Given this information, subset the number of PC dimensions and write down their estimations. What do the components represent?
- e) Plot a biplot on the first two PC dimensions. Interpret the plot.

Problem 2.2 (EOF) The dataset `prec_jan_feb.RData` is available on the web page https://www.math.uzh.ch/furrer/download/sta121/prec_jan_feb.RData. It contains the precipitable water content of the atmosphere in Europe from the first 60 days (approximately January and February) for 2012 (NCEP Reanalysis 2). Images are recorded every 6 hours. The variables are:

- `pre`: is an array with dimensions `c(lon, lat, time) = c(31, 19, 240)`
- `lon`: is a vector with the corresponding longitudes in degree
- `lat`: is a vector with the corresponding latitudes in degree

- a) Look at the data. Display the four images of the 1st January. Display a map of the mean field and the variance field of the data. Add the contours of the countries.
Hint: `load()`, `image.plot()` from package `fields`, `map()` from package `maps`.
- b) Use PCA and display maps of the pattern that explain most of the variance of the data.
- c) Display the variance (eigenvalues) of the principal components. How many principal components are relevant?
- d) According to the “North et al. rule of thumb”, it is recommended to truncate the numbers of components at i , if

$$\frac{\lambda_i - \lambda_{i+1}}{\lambda_i} < \sqrt{\frac{2}{N}}, \quad i = 1, 2, \dots, N$$

where N is the number of eigenvalues λ_i (i.e., $N = 240$ in our example). At which locations could you truncate? Visualize the findings in a plot.

Problem 2.3 (PCA) Let $\mathbf{X} = (X_1, X_2)^\top$ follow a multivariate normal distribution with $\boldsymbol{\mu} = (2, 1)^\top$ and $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1 \end{pmatrix}$.

- a) What are the eigenvalues of $\boldsymbol{\Sigma}$? What are the eigenvectors?
- b) Draw 100 realizations of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and depict them in a scatter plot. Then add one arrow for each eigenvector of $\boldsymbol{\Sigma}$ to the plot. The lengths of the arrows should be proportional to the corresponding eigenvalues.
- c) Calculate the principal components of the simulated data. Add the vectors based on the PCA to the previous plot. Why are the eigenvalues in the PCA not exactly the same as in b)? Compare the original data with the one based on the rotated points.

Chapter 3

Clustering

The idea of cluster analysis is to group n observations into k homogeneous classes where k is possibly also unknown (but usually assumed to be much smaller than n).

Learning goals for this chapter:

- ◊ Describe different clustering methods
- ◊ Carry out cluster analysis using R
- ◊ Use a dendrogram to partition the data into clusters of known composition
- ◊ Interpret the outcome of clustering

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter03.R.

3.1 Hierarchical Clustering

As a motivation, consider the `penguin` dataset (available from the package `palmerpenguins`) bill length, bill depth, flipper length, (all in millimeters) and body mass (in gram) of three penguin species `Adelie`, `Chinstrap`, `Gentoo`. Suppose that we do not have knowledge of the species and we wonder how many different morphological groups are present, in the sense how many clusters are in the data and might indicate an different species. Figure 3.1 illustrates the raw data that we will use extensively in this chapter. For clustering per se, knowledge of the species is not necessary, i.e., we can work with the top right panels. For comparison and quality assessment, we often use species information, as shown in the lower left panels.

In this particular example, some panels of Figure 3.1 show that for some variables some species cluster well (e.g., Gentoo with flipper length), whereas other variables can be less separated (e.g., Adelie and Chinstrap with bill depth). Of course we will look at methods that go beyond this marginal description.

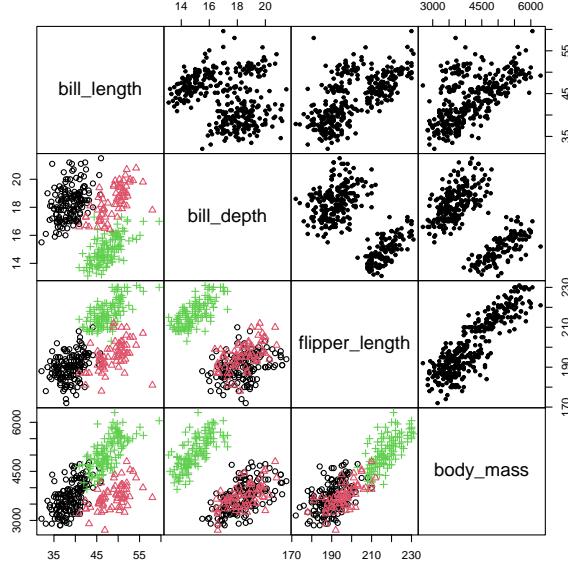


Figure 3.1: Selection of the variables of *Penguin* dataset. Lower part colored according to species. (black \circ : Adelie, red Δ : Chinstrap, green $+$: Gentoo).

More formally, let us consider a dataset consisting of n observations and p variables. Based on such a data matrix, we assign each observation to one of k different clusters or groups or classes. The clustering assignment is based on some measure of similarity, with observations within a group displaying high similarity (or equivalently, a low dissimilarity). In some sense we maximize the similarity within each group (or equivalently, minimize the dissimilarity).

In other words, clustering defines classes of observations that are homogeneous and separate from other classes. For example, looking at the panel of bill depth and body mass in Figure 3.1 we see two groups (Gentoo vs. all others). Clustering is typically an exploratory tool to discover patterns, features or even details in the data. The result of clustering is typically used in further more detailed analysis.

The process of clustering is equivalent to find for all individuals an allocation defining their classes. Formally, an allocation is defined as follows.

Definition 3.1. The function $\gamma : \{1, \dots, n\} \rightarrow \{1, \dots, k\}$ such that $\gamma(i) = \ell$ is called an *allocation*. If $\gamma_i = \ell$, the i th individual belongs to the ℓ th class.

We denote the individual classes with C_ℓ , $\ell = 1, \dots, k$ and the size of the class with $|C_\ell|$. \diamond

That means C_ℓ contains all individuals of class ℓ . Each individual is in one and only one class, which can possibly be empty:

$$C_\ell = \{i : \gamma_i = \ell\}, \quad C_\ell \cap C_j = \emptyset, \quad \bigcup_{\ell=1}^k C_\ell = \{1, \dots, n\}, \quad 0 \leq |C_\ell| \leq n. \quad (3.1)$$

The clustering algorithms should be ideally *optimal* (finding the best possible allocation) according to a predefined loss function or at least *greedy* (finding a very good solution in many cases).

In hierarchical clustering algorithms, we build the clusters from bottom up, that means we construct the clusters from “scratch”. We start with n class, where each individual forms a single cluster and we successively merge two classes. More specifically, we start with the identity allocation $C_i = \{i\}$, for $i = 1, \dots, n$ and at each step we unify two clusters in order to arrive at a simpler classification, containing one class less. Ultimately, all individuals are in the same class ($k = 1$).

Example 3.1. Let $n = 4$. A possible path of hierarchical clustering could be the following one: $\{1\}, \{2\}, \{3\}, \{4\} \hookrightarrow \{1, 3\}, \{2\}, \{4\} \hookrightarrow \{1, 2, 3\}, \{4\} \hookrightarrow \{1, 2, 3, 4\}$. ♣

It is possible to represent the agglomeration path, seen in the Example 3.1, in the form of a classification tree called a *dendrogram*. On the x -axis of a *dendrogram* one finds the n observations that are grouped together into different clusters based on their dissimilarity. The y -axis represents the dissimilarity between clusters.

Example 3.1 illustrated a clustering without any formal rule. In practice we need an agglomeration criteria, or some way to measure the dissimilarity between the clusters, that determine which classes to merge (e.g., why to merge $\{1\}$ and $\{3\}$ and not $\{1\}$ and $\{2\}$). Formally, a dissimilarity function $d(a, b)$ of the objects a and b assigns a value that expresses how dissimilar two observations are, the smaller the value the more alike. A dissimilarity has the following properties:

1. symmetry: $d(a, b) = d(b, a)$,
2. non-negativity: $d(a, b) \geq 0$,
3. identification mark: $d(a, a) = 0$,

We further assume that the dissimilarity increases as the dissimilarity between a and b increases. A distance function satisfies the properties above (and more, like the triangular inequality) and is thus a dissimilarity function as well. By working with dissimilarities instead of distances (like the Euclidean distance), we have the advantage of being able to work with datasets containing interval, ordinal and nominal scales. For example, if a and b are nominal variables, $d(a, b) = I(a = b)$ with I being the indicator function, is a dissimilarity function.

In our setting we will also define the dissimilarity between two clusters, $d(C_i, C_j)$.

For the simplicity we assume for the moment that we have interval scaled observations and that $\|\mathbf{y}_i\|$ denotes the Euclidean norm, i.e., the length of the vector \mathbf{y}_i . The quality of a classification can be measured by the (total) homogeneity within each group:

$$\begin{aligned} & \sum_{\ell=1}^k \{\text{homogeneity measure within group } \ell\} \\ &= \sum_{\ell=1}^k \sum_{i \in C_\ell} \|\mathbf{y}_i - \bar{\mathbf{y}}_\ell\|^2 = \sum_{\ell=1}^k \left(\frac{1}{|C_\ell|} \sum_{i,j \in C_\ell, i < j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \right), \end{aligned} \tag{3.2}$$

where we define the mean of the objects in group ℓ by $\bar{\mathbf{y}}_\ell = \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \mathbf{y}_i$. Having less groups results in larger values of total homogeneity and the quality is reduced.

Assume an allocation for k groups C_1, \dots, C_k . If we agglomerate, say, C_1 and C_2 , we obtain: $\tilde{C}_1 = C_1 \cup C_2$, $\tilde{C}_2 = C_3, \dots, \tilde{C}_{k-1} = C_k$. The difference on the quality is:

$$\sum_{\ell=1}^{k-1} \sum_{i \in \tilde{C}_\ell} \|\mathbf{y}_i - \bar{\mathbf{y}}_\ell\|^2 - \sum_{l=1}^k \sum_{i \in C_\ell} \|\mathbf{y}_i - \bar{\mathbf{y}}_\ell\|^2 = \frac{1}{\frac{1}{|C_1|} + \frac{1}{|C_2|}} \|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2. \quad (3.3)$$

where we used (3.2). A (possible) fusion criteria, based on the previous reasoning, may read: At each step we agglomerate C_m and C_ℓ such that (3.3) is minimized.

We now describe several possible methods of hierarchical clustering.

3.1.1 The Ward Method

We start by defining a dissimilarity that is then used to form the clusters, by minimizing it.

Definition 3.2. The Ward dissimilarity, corresponding to the fusion criteria seen in (3.3), is defined as $d(C_\ell, C_m) = \frac{|C_m||C_\ell|}{|C_m|+|C_\ell|} \|\bar{\mathbf{y}}_m - \bar{\mathbf{y}}_\ell\|^2$.

The hierarchical algorithm that corresponds to the choice of the Ward dissimilarity is called Ward algorithm. \diamond

The Ward dissimilarity also implies that $d(i, j) = 1/2 \|\mathbf{y}_i - \mathbf{y}_j\|^2$.

Once a hierarchical clustering has been performed, we need choose the optimal number of clusters (or groups), i.e., to prune the tree. While there are formal ways of determining this number, a hierarchical clustering approach has as appealing feature the opportunity to explore and to select the number of clusters. Ideally, we want a few, well formed clusters. Hence fusing (or “un-fusing”) groups leads to large changes in dissimilarities. We illustrate that in an example where we use (again) the *Penguin* data.

Example 3.2. We pretend that we do not know the species of the *Penguin* data and apply a hierarchical clustering approach as indicated in R-Code 3.1. The variables are at different scales (variance of the variables are different) thus we do scale them first! Here, scaling means standardizing with `scale()` although centering would not be necessary.

Figure 3.2 shows the resulting dendrogram which suggest that we have two clear groups. The similarity decreases by 222.4 and by 41.6 if we go from 2 to 1 and from 3 to 2 groups, respectively. Two clusters would group (essentially) Gentoo in one cluster and all others in the second cluster. If we use three clusters (still justified by the similarity values), the second and third contain both Adelie and Chinstrap. The function `out <- identify()` might be used to help to identify clusters and individuals therein.

The huge jump in dissimilarity between two and three groups is because Gentoo penguins are straightforward to cluster (see Figure 3.2). Adelie and Chinstrap can be separated well only when we scale the observations first. \clubsuit

R-Code 3.1 Applying Ward method hierarchical clustering to the *Penguin* data. (See Figure 3.2.)

```

load("download/sta121/Penguins.RData") # Essentially from pkg palmerpenguins
hc <- hclust(dist(scale(Penguins[,3:6])), method="ward.D")
dend <- as.dendrogram(hc)
# str(dend)           # a text description of the dendrogram
# plot(dend)          # is virtually similar to plot(hc)
plot(hc, labels=c("-", "--", "---")[Penguins$num_species], hang=-1, main="")
rect.hclust(hc, k=3, border=2:4)           # visualizing 3 clusters
round(rev(hc$height)[1:10], 1)              # height of last ten merges
## [1] 313.8 91.4 49.8 47.0 20.0 15.0 13.6 10.2 9.8 8.6
round(-diff(rev(hc$height))[1:10], 1)        # differences in last ten merges
## [1] 222.4 41.6 2.8 27.0 5.0 1.4 3.4 0.4 1.2 0.2
dd1 <- cutree(hc, 3)                      # getting the three clusters
table(data.frame(cluster=dd1, Penguins$species)) # confusion matrix
##      Penguins.species
## cluster Adelie Chinstrap Gentoo
##    1       149          4       0
##    2        2          64       0
##    3        0          0      123

```

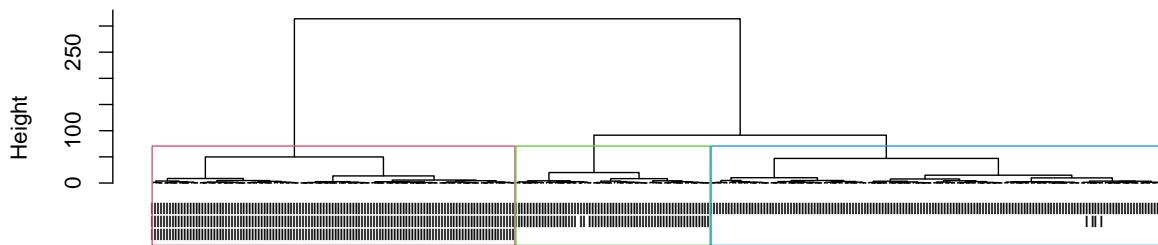


Figure 3.2: Ward method clustering dendrogram for the *Penguin* data. The ticks indicate the (unknown) species and the boxes the three clusters identified. (See R-Code 3.1.)

3.1.2 The Complete Linkage Method

Another possible method to cluster the observations into different groups is complete linkage clustering, also known as the method of the diameter of maximum jump. This clustering method is based on the following dissimilarity criteria:

Definition 3.3. The complete dissimilarity, is defined as: $d(C_\ell, C_m) = \max_{u \in C_\ell, v \in C_m} d(u, v)$. Thus $d(C_\ell \cup C_m, C_k) = \max(d(C_\ell, C_k), d(C_m, C_k))$. The hierarchical algorithm that corresponds to the choice of the complete linkage dissimilarity is called complete algorithm. \diamond

Dissimilarity is based on the maximum possible dissimilarity between clusters. In general, it yields more compact clusters than the Ward linkage method.

Example 3.3. Applying the complete linkage algorithm to the *Penguin* data shows qualitatively a similar dendrogram as the Ward algorithm. In more details, the algorithm suggest two clusters only. But the allocation is quite similar between these two algorithms for $k = 2$ and 3 clusters, as shown in Figure 3.3. The group attributes are arbitrary, thus ♣

R-Code 3.2 Applying complete method hierarchical clustering to the *Penguin* data. (See Figure 3.3.)

```
hc2 <- hclust(dist(scale(Penguins[,3:6])))    # method="complete" is default.
plot(hc2, labels=c("-", "--", "---") [Penguins$num_species], hang=-1, main="")
rect.hclust(hc2, k=3, border=2:4)
round(-diff(rev(hc2$height))[1:10], 1)      # differences in last ten merges
## [1] 2.0 0.7 0.4 0.5 0.4 0.4 0.1 0.2 0.3 0.1
dd2 <- cutree(hc2, 3)
table(data.frame(cluster=dd2, Penguins$species))  # confusion matrix
##          Penguins.species
## cluster Adelie Chinstrap Gentoo
##     1      151        14       0
##     2       0         0      123
##     3       0         54       0
```

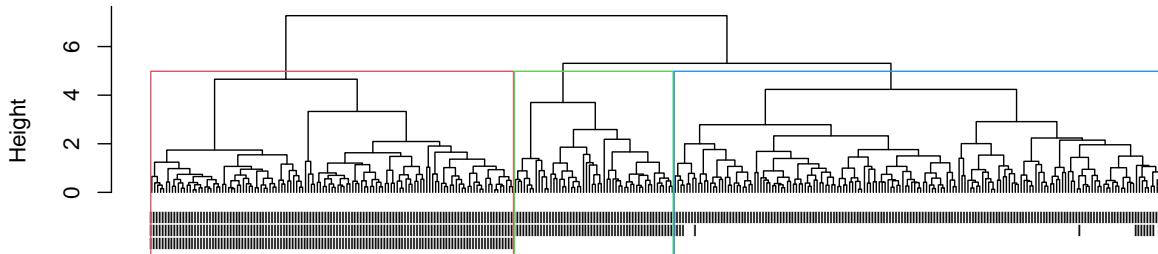


Figure 3.3: Complete linkage method clustering dendrogram for the *Penguin* dataset. The ticks indicate the (unknown) species and the boxes the three clusters identified. (See R-Code 3.2.)

3.1.3 The Single Linkage Method

Another hierarchical method to cluster observations into different groups is the single linkage clustering method, also known as the method of the diameter of minimum jump. This clustering method is based on the following dissimilarity criteria:

Definition 3.4. The single dissimilarity is defined as: $d(C_\ell, C_m) = \min_{u \in C_\ell, v \in C_m} d(u, v)$. Thus $d(C_\ell \cup C_m, C_j) = \min\{d(C_\ell, C_j), d(C_m, C_j)\}$. The hierarchical algorithm that corresponds to the choice of the single linkage dissimilarity is called single algorithm. \diamond

Dissimilarity is based on the minimum dissimilarity between clusters. This method may yield ‘stretched out’ clusters that are less descriptive. The tendency to incorporate observations between clusters into existing clusters instead of initiating a new cluster is known as *chaining*.

Example 3.4. The chaining is also visible when applying the single linkage algorithm to the *Penguin* data. There are two large clusters but pruning would suggest to create three groups, one with single individual (leftmost data point in Figure 3.4). Pruning at lower height continues to creates very tiny clusters (often singletons). \clubsuit

R-Code 3.3 Applying single method hierarchical clustering to the *Penguin* data. (See Figure 3.4.)

```
hc3 <- hclust(dist(scale(Penguins[,3:6])), method="single")
plot(hc3, labels=c("-", "--", "---") [Penguins$num_species], hang=-1, main="")
rect.hclust(hc3, k=3, border=2:4)           # visualizing 3 clusters
dd3 <- cutree(hc3, 3)
```

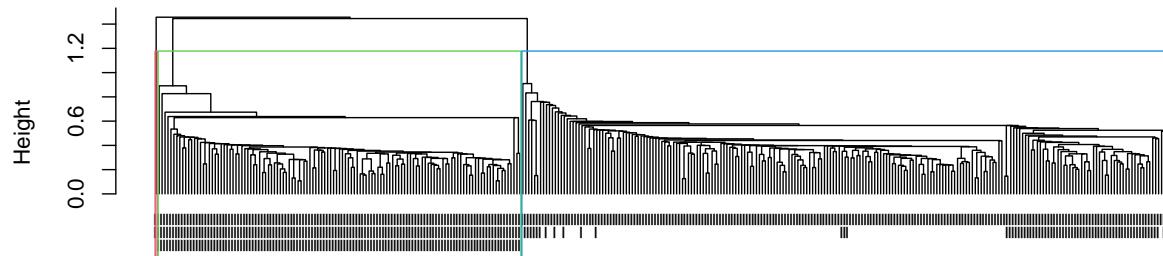


Figure 3.4: Single linkage clustering dendrogram for the *Penguin* data. The ticks indicate the (unknown) species and the boxes the three clusters identified. (See R-Code 3.3.)

We conclude the section with another example.

Example 3.5. Consider again the *jobs* dataset. All three linkage methods seem to reveal very similar structures: Turkey, Greece and Yugoslavia always stand apart. Ward method clearly suggest three groups: (capitalist) Western Europe, communist Eastern Bloc (including Portugal, Spain and Ireland, which have a pronounced agricultural component without any other dominating sector). Note that the data were collected during the Cold War.

The resulting dendrograms of R-Code 3.4 are given in Figure 3.5. Due to the different ordering, it is difficult to quickly compare the clusters. All agglomerations lead to similar results that are coherent with the biplot given in Figure 2.2. \clubsuit

R-Code 3.4 Clustering *jobs* dataset with all three methods. (See Figure 3.5.)

```
jobs <- read.csv(header=TRUE, "download/sta121/europejobs.csv")
# apply(jobs,1,sum) # is 100 up to rounding no scaling necessary

hc1 <- hclust(dist(jobs), method="single")
plot(hc1, hang=-1, main="Single Method")
hc2 <- hclust(dist(jobs))      # complete method is the default
plot(hc2, hang=-1, main="Complete Method")
hc3 <- hclust(dist(jobs), method="ward.D")
plot(hc3, hang=-1, main="Ward Method")
```

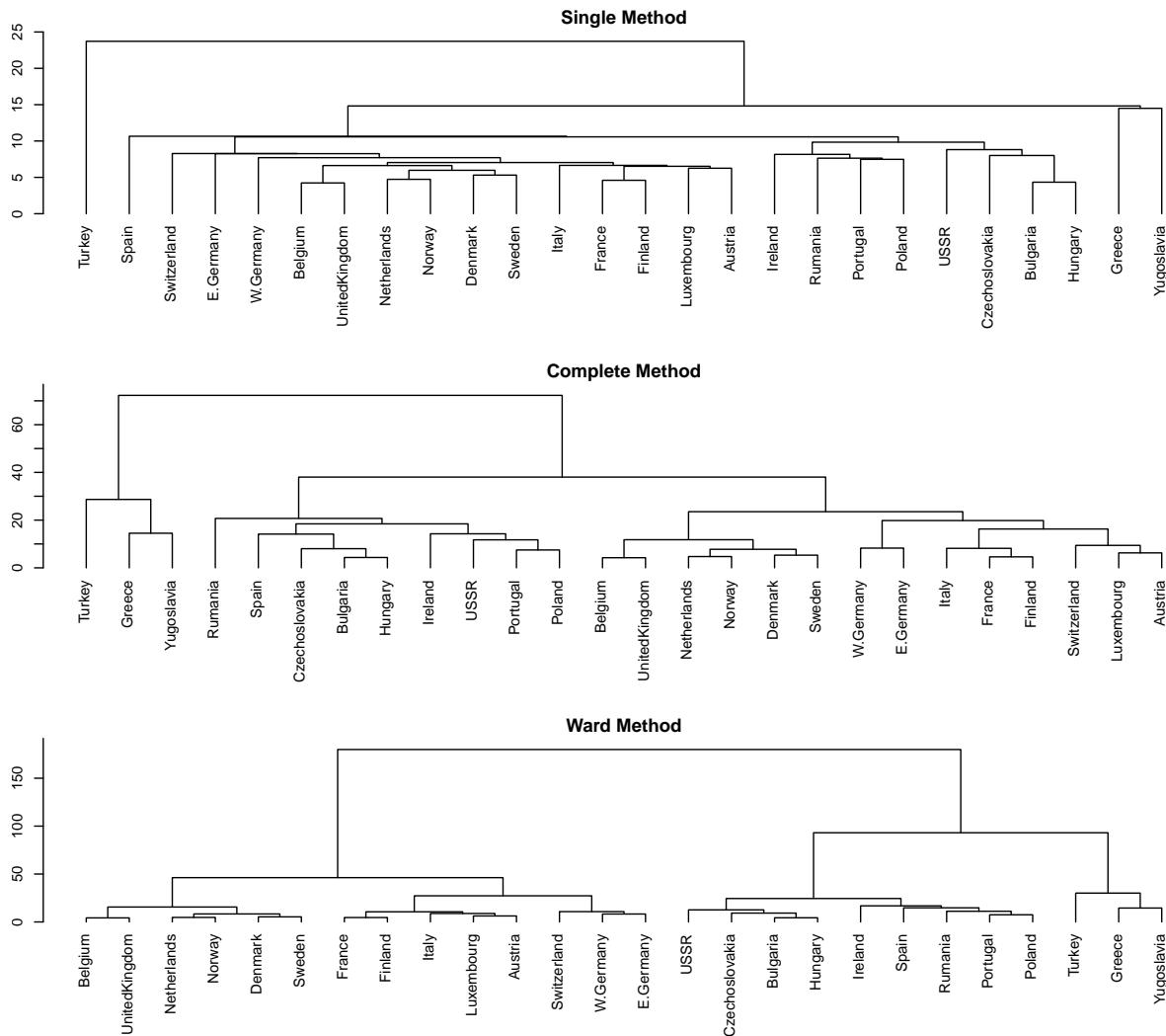


Figure 3.5: Clustering `jobs` dataset with three different hierarchical algorithms. (See R-Code 3.4.)

Alternative dissimilarity measures include `dist()^2`, including different distance measures (`method="euclidean", "manhattan", ...`), `as.dist(1-cor())` and the function `hclust()` supports several additional agglomeration method. Hence, some care is needed that we do not fish with these arguments to obtain a clustering that is appealing or wished for.

3.2 The k -Means Algorithm

In contrast to the hierarchical clustering algorithm, the k -means algorithm is a centroid-based clustering algorithm and has an a priori fixed number of clusters. Here, the clusters are represented by a mean vector (representing the “center” of the cluster). This vector is not necessarily an actual observation. The k -means algorithm is as follows:

1. Start with k cluster centers.
2. Assign observations to the nearest cluster center.
3. Recompute the k cluster centers (mean of observations assigned to each cluster).
4. If centers are the same as before then stop, otherwise go to step 2.

More formally, given an initial set of k means, denoted $\bar{\mathbf{y}}_1^{(1)}, \dots, \bar{\mathbf{y}}_k^{(1)}$, the algorithm proceeds by alternating between the following two steps:

Assignment Step: Each observation is assigned to the cluster whose mean yields the smallest within-cluster sum of squares. Since the sum of squares is the squared Euclidean distance, this is intuitively the “nearest” mean.

$$C_\ell^{(t)} = \{ \mathbf{y}_p : \| \mathbf{y}_p - \bar{\mathbf{y}}_\ell^{(t)} \|^2 \leq \| \mathbf{y}_p - \bar{\mathbf{y}}_j^{(t)} \|^2 \ \forall j, 1 \leq j \leq k \}, \quad (3.4)$$

where each \mathbf{y}_p is assigned to exactly one $C_\ell^{(t)}$. (There is an additional assignment rule if there is no unique cluster.)

Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means.

Update Step: The centroids, or means, of the observations in the new clusters are updated by

$$\bar{\mathbf{y}}_\ell^{(t+1)} = \frac{1}{|C_\ell^{(t)}|} \sum_{\mathbf{y}_j \in C_\ell^{(t)}} \mathbf{y}_j. \quad (3.5)$$

Since the arithmetic mean is a least-squares estimator, this also minimizes the within-cluster sum of squares.

Example 3.6. Consider the variables bill depth and bill length from the `Penguin` dataset. We illustrate in R-Code 3.5 and Figure 3.6 the first few steps of the k -means algorithm. We start with the initial means $(0, -1)$, $(-0.5, 1)$ and $(1.5, 1)$ (magenta dots in the left panel), which are deliberately somewhat arbitrary. The separation is indicated by the polygons. We then calculate the updated means based on all the observations that fall into the corresponding polygons. After three iterations, the algorithm has virtually converged, after step seven the clusters do not change anymore.

R-Code 3.5 Basic k -means algorithm. (See Figure 3.6.)

```

library(deldir)      # for function 'deldir' and 'tile.list'
library(splancs)    # for function 'inout'
x <- scale(Penguins$bill_depth)
y <- scale(Penguins$bill_length)
centx <- c(-0.5, -.5, 1.5)           # starting centroid centers
centy <- c(-1, 1, 1)
for (i in 1:3){                      # Normally looping until convergence, here 7
  # Assignment step:
  z <- deldir(x=centx, y=centy, rw=c(-2.5,2.5,-2.5,2.5), suppressMsge=TRUE)
  w <- tile.list(z)
  g1 <- inout(cbind(x,y), w[[1]][3:4])
  g2 <- inout(cbind(x,y), w[[2]][3:4])
  g3 <- inout(cbind(x,y), w[[3]][3:4])
  # Intermediate plotting
  plot(y ~ x, pch=Penguins$num,
        col=g1+2*g2+3*g3, main=paste("Iteration", i))
  plot(w, add=TRUE)
  points(centx, centy, cex=2, pch=20, col=6)

  # Update step:
  centx <- c(mean(x[g1]), mean(x[g2]), mean(x[g3]))
  centy <- c(mean(y[g1]), mean(y[g2]), mean(y[g3]))
}

```

Note that we have scaled the variables. Without scaling, the groups are somewhat counter-intuitive as one variable dominates the Euclidean distance. ♣

In practice, a dataset may contain many individuals and variables and the actual optimization problem may be very difficult. For example, in complex multi-dimensional problems, the solution may vary according to the initial values. Thus one often runs the method several times with different starting values and compares the resulting solutions. However, many good heuristics exist, especially if the number of clusters is known. Due to the assignment rule (assignment to the “nearest” group), the k -means algorithm tends to yield clusters of similar size (area or volume, not observation number). The clusters are typically convex in shape (no banana shaped clusters), even when the true shape is not.

Example 3.7. R-Code 3.6 illustrates k -means clustering of the four numeric variables of the *Penguin* dataset with $k = 3$ using the built-in `kmeans()` function.

The clustering nicely identifies the three species. The clustering allocation has an “arbitrary” numbering of the clusters, which may not correspond to the natural or intuitive encoding. Hence, the correspondance needs to be manually determined here. The clusters correspond to *Chinstrap*, *Gentoo* and *Adelie*.

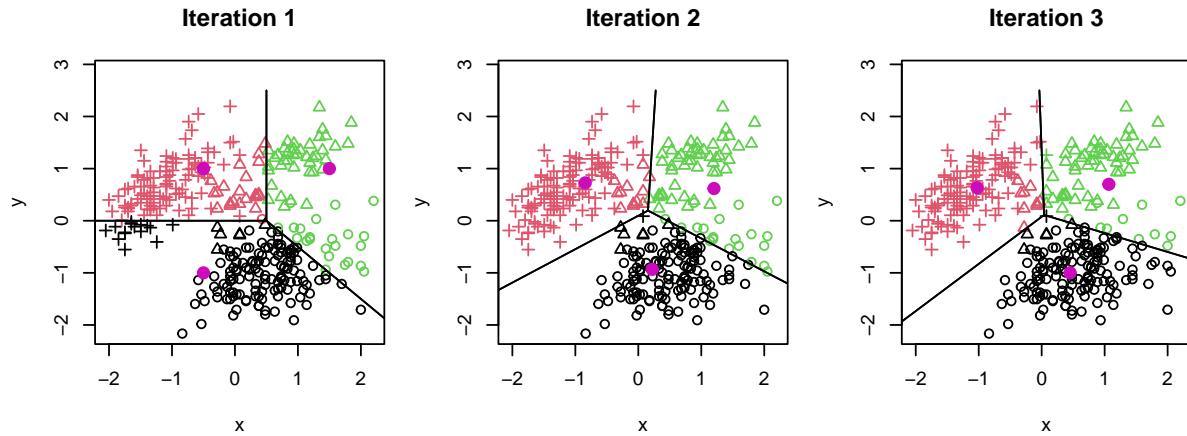


Figure 3.6: Basic illustration of the k -means algorithm. After three iterations, the means and the cluster allocations only slightly change. The magenta points are the group means. Groups are indicated by color and species are indicated by symbol (\circ : Adeline, Δ : Chinstrap, $+$: Gentoo). (See R-Code 3.5.)

Figure 3.7 illustrates the clusters. Because of the marginal projection of the scatterplots, the hyperplane separations as seen in Figure 3.6 are not visible.

The different sum-of-squares gives some insight into how well the clusters are defined. For example, the within-sums-of-squares of the first cluster is much smaller than the other clusters, because the observations of *Chinstrap* are more “tightly” clustered, compared to both other species.

The centroid centers using only two variables (here obtained from our manual algorithm) are very similar to the first two components of the built-in k -means algorithm. ♣

R-Code 3.6: The k -means algorithm applied to the *Penguin* dataset. (See Figure 3.7.)

```

cl <- kmeans(scale(Penguins[,3:4]), 3)
# print(cl) # "Clustering vector" and "Available components" not of interest
table(cluster=cl$cluster, Species=Penguins$species)

##      Species
## cluster Adelie Chinstrap Gentoo
##       1      0       9     116
##       2      4      54      6
##       3    147       5      1

pairs(Penguins[,3:6], col=cl$cluster, pch=Penguins$num_species,
      gap=0, lower.panel=NULL, xaxt="n", yaxt="n")
unlist(cl[grep("ss", names(cl))]) # extract all variables with "ss" in name

##      totss   withinss1   withinss2   withinss3 tot.withinss
##    682.000    59.355    39.034    87.986    186.375

```

```

##      betweenss
##      495.625
cbind(cl$centers, cbind(centy, centx)[c(1,3,2),] )
##    bill_length bill_depth centy    centx
## 1     0.59031   -1.09377 -0.97717  0.51932
## 2     1.10184    0.79854  0.87912  0.93855
## 3    -0.94318    0.55957  0.62073 -1.06262

```

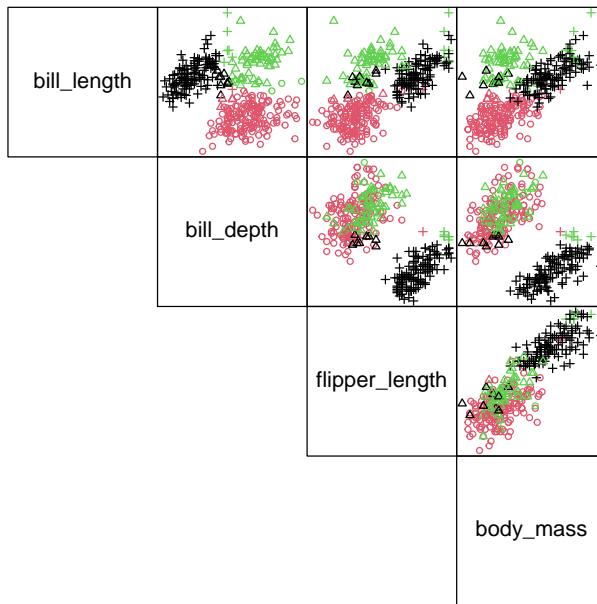


Figure 3.7: Scatter plot of the `penguin` dataset with the k -means groups indicated by color and the species type indicated by symbol (\circ : Adelie, Δ : Chinstrap, $+$: Gentoo). (See R-Code 3.6.)

For hierarchical clustering we can visually assess an good number of cluster by inspecting the dendrogram or more formally looking at the height of clustering heights or their difference. For k -means we do not have a dendrogram and different approaches are required. One graphical tool to assess the quality of clustering are so-called silhouette plots, which help to answer questions like How homogeneous are the clusters? Are there observations that are outlying? etc.

For illustration, suppose that we have assigned to observation i cluster A , let a_i denote the average dissimilarity of i to all other objects in cluster A . Roughly speaking, a_i represents the ‘within’ dissimilarity of cluster A . For all other clusters we define $d_i(C)$ the dissimilarity of i to all other objects in cluster C , $C \neq A$. A silhoutte plot represents for each value the silhouette width s_i , which is defined as

$$s_i = \frac{\min_{C \neq A} d_i(C) - a_i}{\max\{\min_{C \neq A} d_i(C), a_i\}}, \quad (3.6)$$

where the denominator is a normalizing factor, ensuring $-1 \leq s_i \leq 1$.

Suppose that $d_i(C)$ is minimal for cluster B . A value s_i close to one indicates that i has been well clustered. The second best cluster B is not nearly as good as the chosen one. If s_i close to zero, then a_i and $d_i(B)$ are about the same size and observation i could have been assigned to either cluster A or B . If s_i close to minus one, a_i is much larger than $d_i(B)$ and we should have assigned observation i rather to B and we could argue that the observation has been missclassified.

Example 3.8. We classify the `swiss` dataset with k -means and construct the corresponding silhouette plot (R-Code 3.7). It seems that two groups are adequate, each having clearly placed observations. Increasing the number of clusters decreases the quality of individual clusters. Naturally, increasing the number of clusters reduces s_i and the average value thereof in each cluster. The clusters can be visualized by a `pairs()` plot, for example (not shown). ♣

R-Code 3.7 Silhouette plot of `swiss` dataset. (See Figure 3.8.)

```
library(cluster)
set.seed(14)
ks <- kmeans(swiss, centers=2)
si <- silhouette(ks$cluster, dist(swiss))
plot(si, col=2:3, main="")
for (k in 3:4) # compact form for other `k`
  plot(silhouette(kmeans(swiss, centers=k)$cluster, dist(swiss)),
       col=2:(k+1), main="")
# pairs(swiss, col=ks$cluster, pch=19, lower.panel=NULL, gap=0)
```

3.3 Model-Based Clustering

A more formal approach to clustering can be achieved if we assume underlying distributions of the observations. This is also known as finite mixture model clustering. Observations arise from a distribution that is a mixture of two or more components, or clusters. Each cluster is described by a density and has an associated probability or ‘weight’ in the mixture.

We have n p -dimension observations $\mathbf{y}_1, \dots, \mathbf{y}_n$ and assume the joint distribution is a mixture of k components. The density of the mixture is then

$$f(\mathbf{y}) = \prod_{i=1}^n \sum_{\ell=1}^k \pi_\ell f_\ell(\mathbf{y}_i), \quad (3.7)$$

where $f_\ell(\mathbf{y}_i)$ is the density of cluster ℓ , possibly depending on parameters $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}_\ell$, and π_ℓ is the probability that observation \mathbf{y}_i belongs to cluster ℓ . We typically assume multivariate normal densities for $f_\ell(\cdot)$. For example, for $p = 1$, and two clusters, we simplify (3.7) to

$$f(\mathbf{y}) = f(\mathbf{y}; \pi, \mu_1, \mu_2, \sigma_1, \sigma_2) = \prod_{i=1}^n \pi \varphi\left(\frac{y_i - \mu_1}{\sigma_1}\right) \frac{1}{\sigma_1} + (1 - \pi) \varphi\left(\frac{y_i - \mu_2}{\sigma_2}\right) \frac{1}{\sigma_2}. \quad (3.8)$$

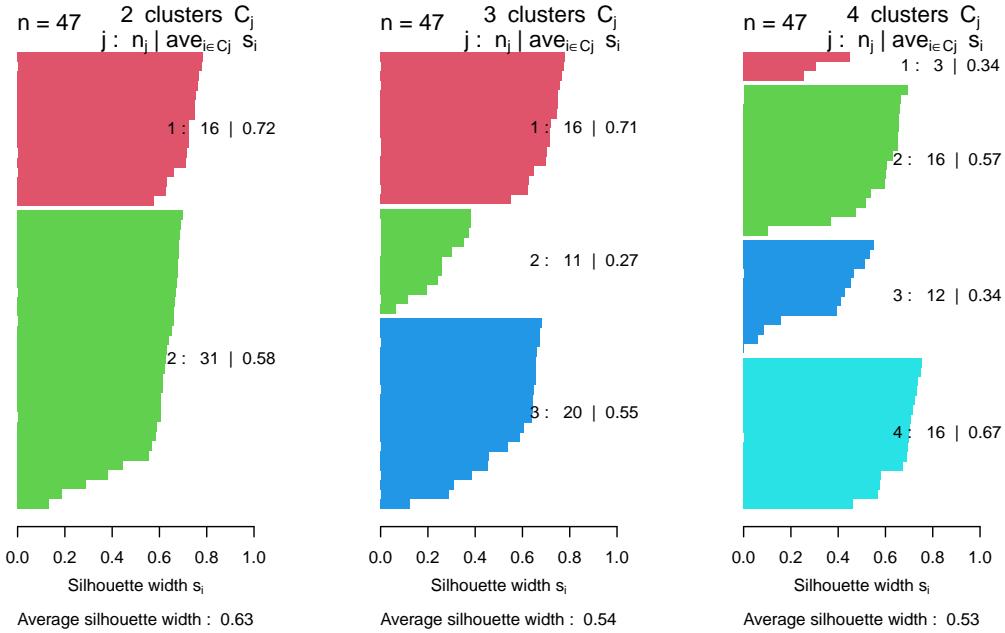


Figure 3.8: Silhouette plots of `swiss` dataset for two, three and four clusters. (See R-Code 3.7.)

Because we do not know cluster allocations, we estimate parameters using a so-called expectation-maximization (EM) algorithm. Starting with initial estimates, the EM algorithm iterates through these steps:

Estimation Step: find the expected value of the full data log-likelihood given the observed data and current parameter estimates.

Maximization Step: Maximize the expression over current parameter values to find the next parameter estimates.

Recall that the covariance matrix Σ_ℓ determines the geometric features of the ℓ th cluster. Assuming different covariance matrix allows for differences in shape, volume, and orientation of the clusters, unlike in the k -means method presented before.

On the other hand, if the observations of group ℓ , $\ell = 1, \dots, k$, are iid realizations from $\mathcal{N}_p(\mu_\ell, \Sigma)$, i.e., $\Sigma_1 = \dots = \Sigma_k$. It can be shown that k -means clustering is equivalent to the maximum likelihood framework for such a Gaussian case.

The formal nature of this method (i.e., assuming densities) allows for the use of different model selection criteria for choosing the best model. The BIC is typically used.

As illustration, the next examples clusters eruptions data from the ‘Old Faithful’ geyser in Yellowstone National Park, Wyoming.

Example 3.9. The ‘Old Faithful’ geyser in Yellowstone National Park is probably one of the worlds most famous geysers. The geyser erupts spectacularly and regularly and is thus quite predictive (nice feature for tourists). In R-Code 3.8 we visualize the duration of eruptions and the corresponding waiting time. The scatterplot suggests two, maybe three clusters. Using the

function `Mclust()` from the package `mclust`, we fit a mixture of bivariate Gaussian densities. The function selects the best model automatically (here based on BIC), resulting in four clusters, each with a bivariate but independent normal density (i.e., “**VVI**” model, resulting in individual but diagonal matrices Σ_k).

The left panel of Figure 3.9 shows the four clusters. An advantage of a model based clustering is the availability of cluster probabilities (each observation is assigned to the cluster with highest probability). The observation (65, 4) is almost as likely to be classified in Cluster 1 or 3.

The right panel of Figure 3.9 visualizes f_ℓ in (3.7) based on four bivariate Gaussian densities. Cluster 2 seems missing but its mixture weight is only about 8% and thus not sufficiently pronounced to be visible in a simple perspective plot.

As a final remark, note that there are many different versions of ‘Old Faithful’ available. Often the data is visualized as waiting time versus the previous event duration (e.g., `faithful` from the package `datasets`). ♣

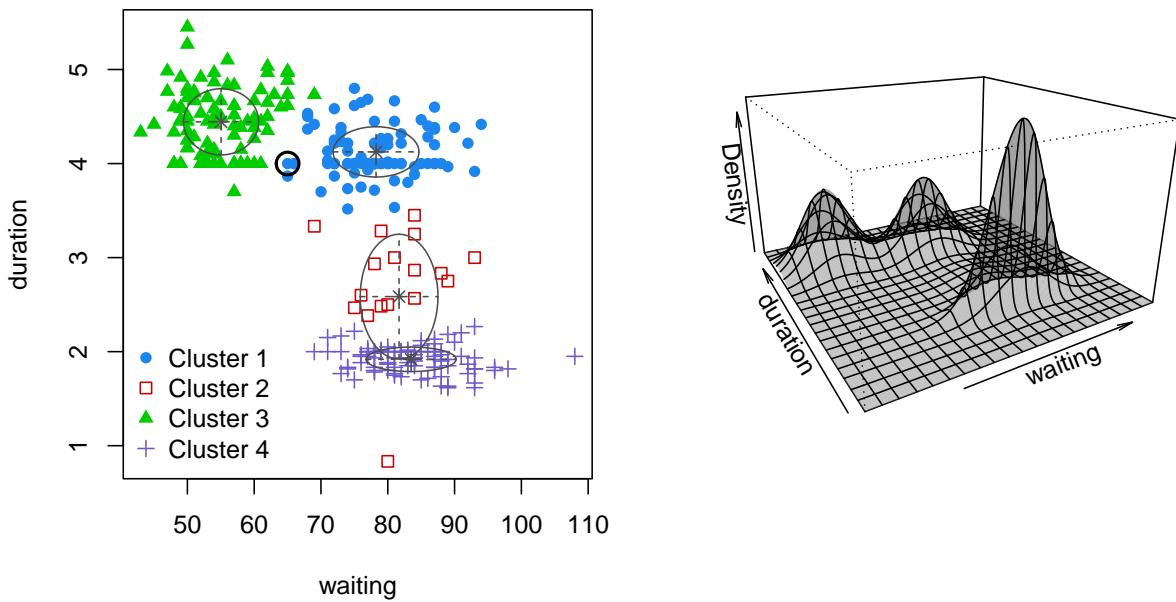


Figure 3.9: Clusters of `geyser` (left) and resulting bivariate density (right). (See R-Code 3.8.)

3.4 Longitudinal Clustering

Quite often we have measurements over time for several individuals or at several measuring locations. Examples for the former are growth curves for piglets, CD4 cell counts in patients, such data is often called longitudinal data. For the latter, we may observe meteorological data in different cities, the current yield of different bonds over time. We will discuss proper statistical models for such time series data in Chapter 10.

In this section, we will discuss clustering approaches for such temporal data, without specifying specific statistical models. The difference to the previous sections is the sequential nature of data, i.e., we do have “curves” (often also referred to as functional data). These curves are

R-Code 3.8 Model based clustering of *geyser* data. (See Figure 3.9.)

```

library(mclust)
data("geyser", package="MASS")
mcl <- Mclust(geyser)
summary(mcl)

## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
## 

## Mclust VVI (diagonal, varying volume and shape) model with 4 components:
## 

## log-likelihood   n df      BIC      ICL
##          -1330.1 299 19 -2768.6 -2798.7
## 

## Clustering table:
##  1  2  3  4
## 90 17 98 94

plot(mcl, what="classification")
legend("bottomleft", legend=paste("Cluster", 1:4), bty="n",
       pch=mclust.options("classPlotSymbols")[1:4], # is 16 0 17 3
       col=mclust.options("classPlotColors")[1:4]) # or some approx color
uncert <- which.max(mcl$uncertainty) # most difficult observation to cluster
points(geyser[uncert,], cex=2, lwd=2)
round(mcl$z[which.max(mcl$uncertainty),], 3) # prob of belonging to cluster
## [1] 0.535 0.001 0.464 0.000

plot(mcl, what="density", type="persp")
mcl$parameters$pro                         # mixture weights  $\pi_k$ 
## [1] 0.292835 0.079684 0.333923 0.293559

```

often quite smooth, i.e., consecutive observations are not independent. A further difficulty is that we often have missing values. Due to the smoothness of the curves, it is often quite easy to interpolate and much more efficient than to eliminate all observations of the corresponding time point.

A clustering algorithm takes into account, that we work with data over time instead of different variables of individuals. The dependence over time should be taken into account when calculating distances between curves. At this point, we do not model such correlations and thus, we do not discuss the technical details of a successful implementation of an algorithm. Many difficulties discussed previously also apply to longitudinal clustering: often, the optimal number of groups is not clear. Minor changes in the algorithmic parameters may lead to different group allocations.

There are several R packages that implement longitudinal clustering. The next example illustrates the clustering of wind data with the package `kml` (standing for *k*-means for longitudinal data). The package provides the functionality to deal with missing values and runs the algorithm several times to help finding appropriate number of clusters.

Example 3.10. The storm Xynthia raged over Europe end of February beginning of March 2010 and did significant damage in France and quite some damage in further European countries (https://en.wikipedia.org/wiki/Cyclone_Xynthia). We analyze wind speed data for 71 stations in Switzerland over a period of 95 hours, starting at 28.02.2010 at 00:00. R-Code 3.9 loads the dataset and selects the mean wind speed as variable of interest. Wind speeds are quite variable (top panel of Figure 3.9) and thus we will use a weighted running average to cluster the data, that is, we represent the wind at time t as $y_t = (4y_t + 3y_{t+1} + 2y_{t+2} + y_{t+3})/16$.

Longitudinal clustering is done with the function `kml()`. Notice that the function `kml()` changes (rather updates) its input argument directly. Therefore no assignment is required.

The right panel of Figure 3.9 shows the smooth wind speeds and superimposed the five cluster means. The clusters represent the stations that experienced the storm in the following matter. (A) no strong wind, hardly affected by the storm; (B) two moderate peaks for both waves of the storm; (C) moderate peak for the first wave, pronounced “early” second peak, (D) permanently high winds with a slightly more pronounced first wave (E) extremely affected by the storm.

The left panel of Figure 3.9 is one way to represent a graphical guidance to choose the number of clusters. For each run of the algorithm `nbRedrawing=12` the specified criterion (here ‘Calinski–Harabatz’) is reported in decreasing order. Here, five groups seem optimal but six or four have quite a similar criterion. In interactive R mode, the plot constructed with the function `choice()` allows to switch between the number of clusters (among the computed ones) and further graphical representations. As a side remark, depending on the criterion, even more groups are chosen (see the output of `plotAllCriterion(clkml)`)

With `table()` it is possible to investigate group sizes and their heritage. As shown in the R output, four clusters are essentially clusters (A), (B), (C) and the union of (D) and (E). Similarly, when constructing six groups we essentially create a high and low version of group (C).

The lower panel of Figure 3.9 shows the positions of the measurement station along with its group. In cluster `E` are two stations, Chasseral and La Dôle, which are placed on the mountains of the same name of the Jura Mountains. Most of the `B` cluster lies in the Swiss Alps, which were not as affected by the storm. It is important to note that we do not work with peak wind gusts which are mainly responsible for storm damage. The highest gusts occur in three different groups, Altdorf in cluster (C) had more than 146 km/h peak gusts. ♣

R-Code 3.9: Longitudinal clustering with `kml`. (See Figure 3.10.)

```
load("download/sta121/Xynthia.RData")
# str(data) # a few variables that we do not use.
Xynthia <- data.frame(Hour=data$Hour, Lat=data$Lat, Lon=data$Lon,
Station=data[,c("WMO.Station.Number")], WS=data[,c("Wind_speed.m.s.")])
```

```

XynthiaWide <- reshape(Xynthia, idvar =c("Station","Lon","Lat"),
                        timevar="Hour", direction="wide")
WindRaw <- t(as.matrix(XynthiaWide[,-c(1:3)]))
matplot(WindRaw, col="gray", type="l", lty=1, ylab="Wind speed [m/s]")

## smooth the values:
ma <- function(x){ filter(x, c(1,2,3,4,3,2,1)/16, sides=2)}
Wind <- apply(WindRaw, 2, ma)      # smooth the wind speeds
Wind <- Wind[ !is.na(Wind[,1]),] # cut start and end due to filter

# Now ready for kml:
library("kml")
set.seed(14)
clkml <- cld(t(Wind))    # construct the object for `kml`
kml(clkml, nbClusters=3:7, nbRedrawing=12) # now cluster!
choice(clkml)             # if working in RStudio, call `x11()` first

# plotAllCriterion(clkml) # Clustering is not very stable but
clkml5 <- getClusters(clkml, 5)        # store cluster assignment
table(clkml5, getClusters(clkml, 4)) # essentially last group is splitted

##
## clkml5  A  B  C  D
##      A 25  0  0  0
##      B  1 23  0  0
##      C  0  0 12  0
##      D  0  2  0  6
##      E  0  0  0  2

# table(clkml5, getClusters(clkml, 6)) # new group out of A, B, C
## Now visualization, lower panel of Figure
library(mapdata)
map("worldHires", "Switzerland")
text(Info$Lon, Info$Lat, as.character(clkml5),
     col=c(2,1,3,4,6)[as.numeric(clkml5)], cex=1.2)
Info[ Info$WMO.Station.Number %in% XynthiaWide[clkml5=="E", "Station"],]
##      WMO.Station.Number Station_name Latitude Longitude Elevation
## 674          6605 CHASSERAL   47.117    7.0500    1595
## 707          6702 LA DOLE    46.417    6.0833    1670

MaxPerStation <- tapply(data$Max_gust, data$WMO.Station, max, na.rm=TRUE)
GustsNr <- order(MaxPerStation, decreasing=TRUE)[ 1:5]
points(Info$Lon[GustsNr], Info$Lat[GustsNr], col=7, cex=3.2, lwd=3)
tapply(MaxPerStation, clkml5, max)*3.6      # Max per Cluster in km/h

```

```
##      A      B      C      D      E
## 85.32 105.84 146.52 135.36 127.80
```

We conclude the section with some comments about missing values. Rubin (1976) classifies missing values in three different categories according to the mechanism that governs the missingness.

Missing completely at random (MCAR) indicates that an observation may be missing with a certain probability but this probability is independent of the observation, measuring process, covariates or suchlike. This setting is the easiest to address statistically but in reality often unreasonable to assume. The package `km1` also assumes this missing data mechanism

Missing at random (MAR) can be understood as missing at random conditionally on other variables at random. For example, they may be MCAR for the particular set of observations but for a new run of the experiment, the mechanism may change and thus MCAR does not hold.

All cases that are not MCAR or MAR are termed missing not at random (MNAR). Unfortunately, these cases are very dangerous, as they introduce a bias in the data. A specific example is the lack of reporting a measurement because of a very low reading (e.g., low inferred ozone levels over the Antarctic in the early 1980) .

There is another intuitive explanation of the differences. Let R be the probability that an observation may be missing, \mathbf{Y}_{obs} and \mathbf{Y}_{mis} the random vectors indicating the observed and missing observations then

$$\begin{aligned} \text{MCAR : } & P(R | \mathbf{Y}_{\text{obs}}, \mathbf{Y}_{\text{mis}}) = P(R), \\ \text{MAR : } & P(R | \mathbf{Y}_{\text{obs}}, \mathbf{Y}_{\text{mis}}) = P(R | \mathbf{Y}_{\text{obs}}), \\ \text{MNAR : } & P(R | \mathbf{Y}_{\text{obs}}, \mathbf{Y}_{\text{mis}}) = P(R | \mathbf{Y}_{\text{obs}}, \mathbf{Y}_{\text{mis}}), \text{ i.e., cannot be simplified.} \end{aligned} \quad (3.9)$$

3.5 Example: A Bioinformatics Data Analysis

This example is taken from Sabates-Bellver *et al.* (2007), whose abstract reads: “Colorectal cancers are believed to arise predominantly from adenomas. Although these precancerous lesions have been subjected to extensive clinical, pathologic, and molecular analyses, little is currently known about the global gene expression changes accompanying their formation. To characterize the molecular processes underlying the transformation of normal colonic epithelium, we compared the transcriptomes of 32 prospectively collected adenomas with those of normal mucosa from the same individuals. Important differences emerged not only between the expression profiles of normal and adenomatous tissues but also between those of small and large adenomas.”

In this subsection we have a rudimentary look at the original data, so-called gene expression data. Naturally, we cannot go into all the detail as this is beyond the scope of this document.

The original dataset as well as the relevant meta-data has been provided to us as raw text files. The data matrix consists of 54675 rows and 64+1 columns and results in $\approx 17\text{MB}$ of data. We only provide a reduced subset of the data.

In the experiment, tissue samples from normal mucosa and adenomas were collected from 28 patients. Four patients contributed two (two plus two) samples, resulting in 32 samples.

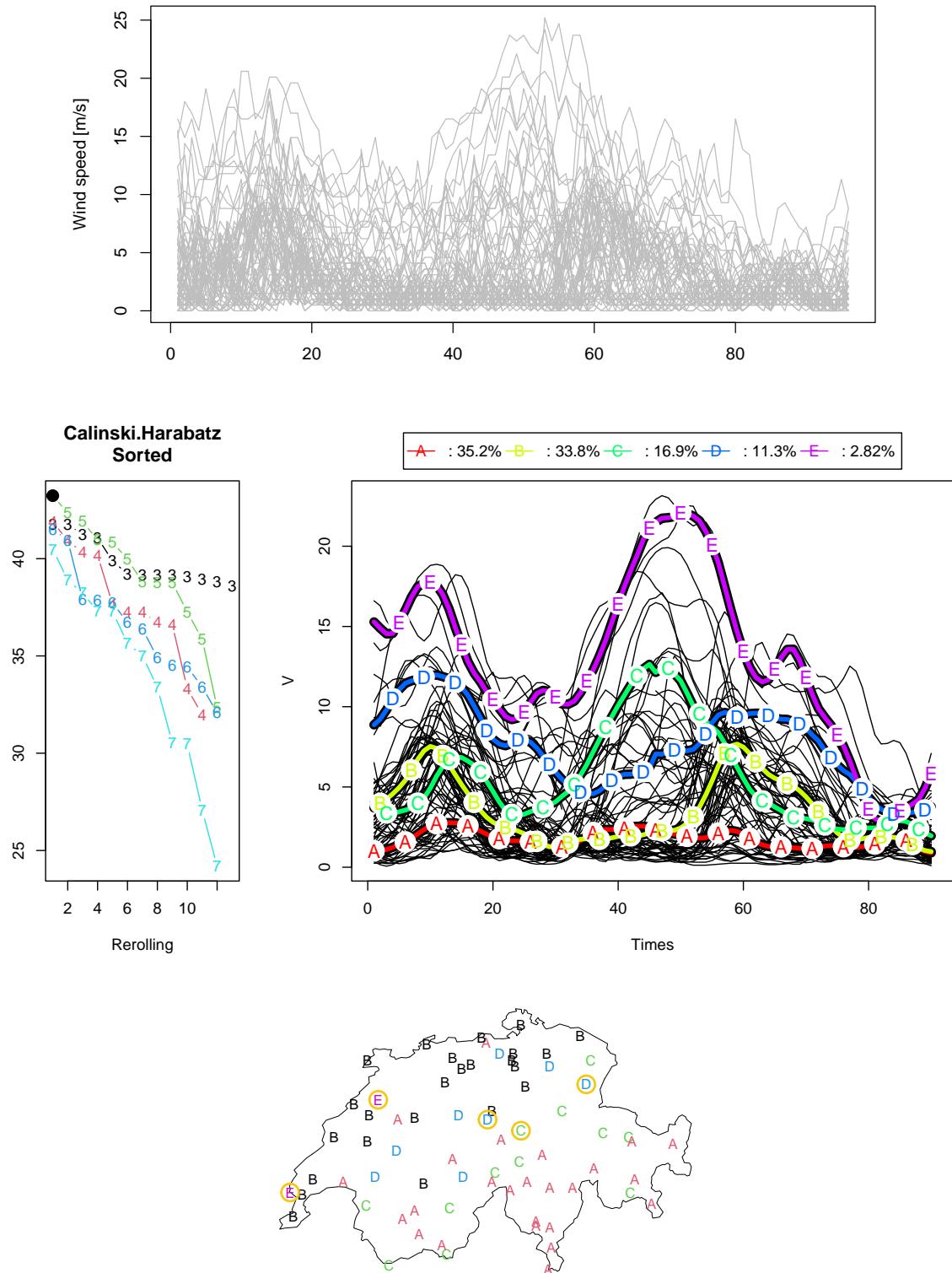


Figure 3.10: Top: wind speed for 71 stations in Switzerland over a period of 95 hours, starting at 28.02.2010 00:00. Middle: output of `choice()` with cluster criterion for each run (left) and cluster means with observations (right). Bottom: stations and their group attributes. The stations with five largest peak gusts have been highlighted with a yellow circle. (See R-Code 3.9.)

We load the dataset `sabates_bellver_U133_all.txt` and \log_2 transform the data. In order to find differences between adonomas and normal tissue, we select the probes (“genes”) that are most prominently different. That means, we determine all the significant probes ($\alpha = 0.01$; Holm correction) and reduce the dataset to these probes.

```
dim(dat <- read.csv("download/sta121/sabates_bellver_U133_all.txt"))[1]
rma <- as.matrix(log(dat[,-1], 2))
p.values <- apply(rma, 1, function(y) t.test(y[1:32], y[33:64], var=T)$p.v )
p.adj      <- p.adjust(p.values, method="holm")

probes <- dat$Probes
rownames(rma) <- probes

inter <- rma[p.adj < 1e-13, ]
# realistically, a slightly larger value should be used
```

We only use probes that are highly significant. Notice that we have adjusted the p -value using a *Holm* correction. Instead of using probes according to a p -value, we can select probes that exhibit a considerable *fold* change, possibly augmented by some genes that we know to be of interest.

```
dim(inter)
## [1] 831  64
type <- factor(rep(c("nm", "po"), each=32))
indn <- colnames(inter)
library(ade4)
pr4 <- dudi.pca(t(inter), scan=F, scale=T)
s.class(pr4$li, type, col=3:4)
s.label(pr4$li, label=indn, add.plot=T)
```

The command `hclust()` performs a hierarchical cluster analysis on a set of dissimilarities. Based on the correlation matrix, we can plot the result of `hclust()` and compare it to the biplot.

```
plot(hc <- hclust(dist(cor(rma))))
cu <- cutree(hc, 4)

covarsraw <- read.csv("download/sta121/sabates_bellver_table1.csv")
str(covarsraw, strict.width="cut")

## 'data.frame': 32 obs. of 12 variables:
##   $ Patient : chr  "GL1" "GL2" "PR" ...
##   $ Age     : int  49 49 74 69 69 57 5...
##   $ Sex    : chr  "M" "M" "F" "M" ...
```

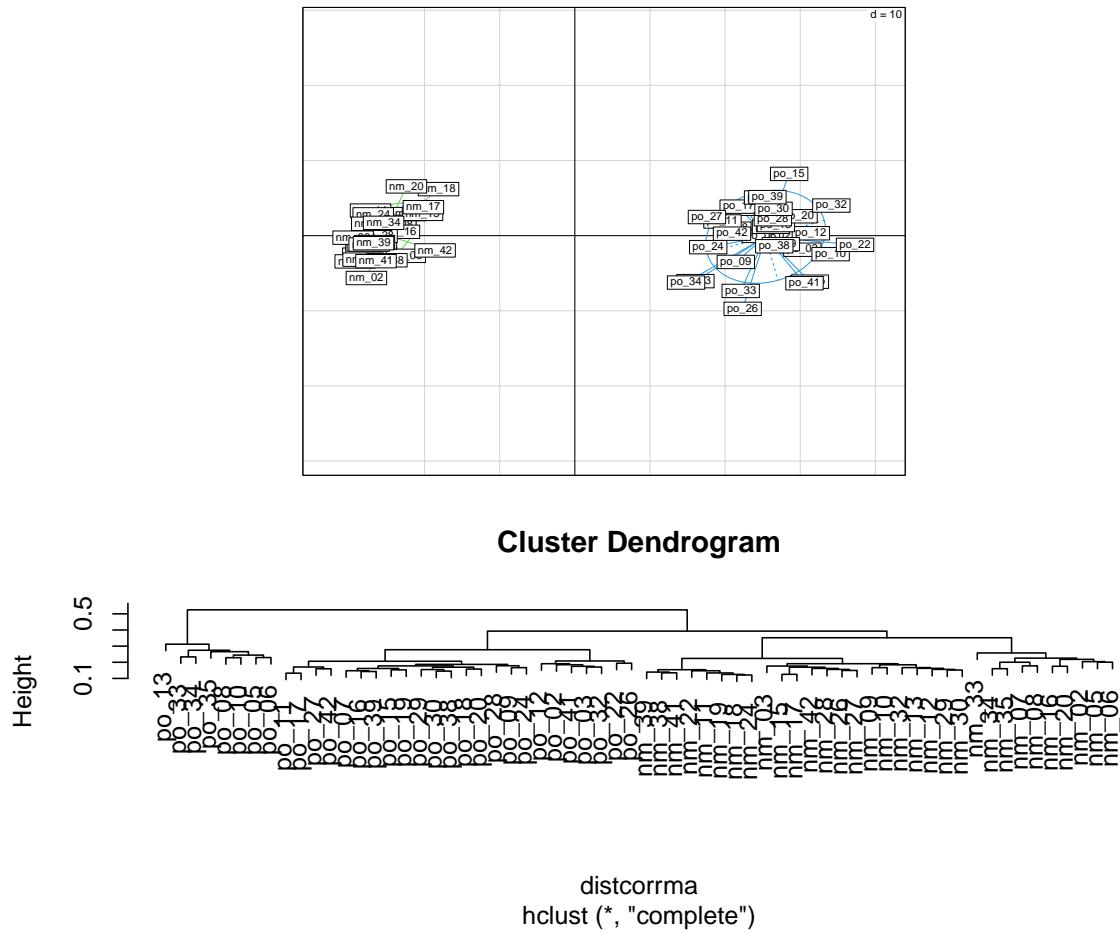


Figure 3.11: PCA (top) and cluster analysis (bottom).

```

## $ Colon.segment.involved : chr  "D"  "S"  "S"  "S" ...
## $ Maximum.adenoma.diameter : int  10 10 20 10 20 15 3..
## $ Microscopic.appearance : chr  "T-V" "T-V" "T-V" "... "
## $ Highest.degree.of.dysplasia : chr  "H"  "L"  "H"  "H" ...
## $ Degree.of.dysplasia : chr  "H"  "L"  "H"  "H" ...
## $ No.adenomas : int  9 9 2 10 10 1 1 2 2 ..
## $ No.previously.excised.adenomas : chr  "4"  "4"  "-"  "-" ...
## $ No.previous.synchronous.hyperplastic.polyps: int  1 1 0 1 1 1 1 2 2 0..
## $ Familiarity.for.colorectal.cancer : chr  "Mother, 70" "Moth"...
dim(covarsraw)
## [1] 32 12
covars <- with(covarsraw,
  data.frame(Age=c(Age,Age), Sex=rep(Sex,2),
  segm=factor(c(Colon.segment.involved,Colon.segment.involved)),
  hist=factor(c(rep(0,32),(Familiarity.for.colorectal.cancer=="No"))))
)

```

```
heatmap(inter, col=topo.colors(100), margin=c(2,2), labRow=NULL,
        ColSideColors=heat.colors(2)[rep(as.numeric(covarsraw$Sex),2)])
# See also:
#   library("gplots")
#   heatmap.2()
```

There is no surprise that in the heatmap (Figure 3.12) we see strong difference between the two groups (NM and PO) as we have selected the probes with the strongest differences. The horizontal stripes are induced by the probe clustering. The red blocks do not have a pattern. That means that there is no structure to the patients by sex.

We improve the PCA by accounting for sex or we can construct a PCA for each group size of the adenomas.

```
library(ade4)
pr4 <- dudi.pca(t(inter), scan=F, scale=T)
s.class(pr4$li, factor(as.numeric(covarsraw$Sex)+rep(c(0,2),each=32)),
        col=1:4, ylim=c(-10,10))

tmp <- factor((covarsraw$Max>20) + (covarsraw$Maximum.adenoma.diameter >=20))
pr5 <- dudi.pca(t(inter[,33:64]), scan=F, scale=T)
s.class(pr5$li, tmp, col=1:4, ylim=c(-10,10))
```

Naturally, depending on the exact selection of probes, we get slightly different results. Hence, the figures here do not exactly correspond with those in the original paper.

It is possible to link probes to gene names, locations (on the chromosomes), etc. The following code snippet gives a glance of how to perform such an analysis.

```
library(hgu133a2.db) # see below on how to install this package
x <- hgu133a2GENENAME
# Get the probe identifiers that are mapped to a gene name
mapped_probes <- mappedkeys(x)
xx <- as.list(x[mapped_probes]) # Convert to a list
xx[1:5]
x <- hgu133a2CHRL0C
# Get the probe identifiers that are mapped to chromosome locations
mapped_probes <- mappedkeys(x)
xx <- as.list(x[mapped_probes]) # Convert to a list
xx$`227475_at` # probe found above... but it is not mapped!
```

```
## Warning in heatmap(inter, col = topo.colors(100), margin = c(2, 2),
labRow = NULL, :  NAs introduced by coercion
```

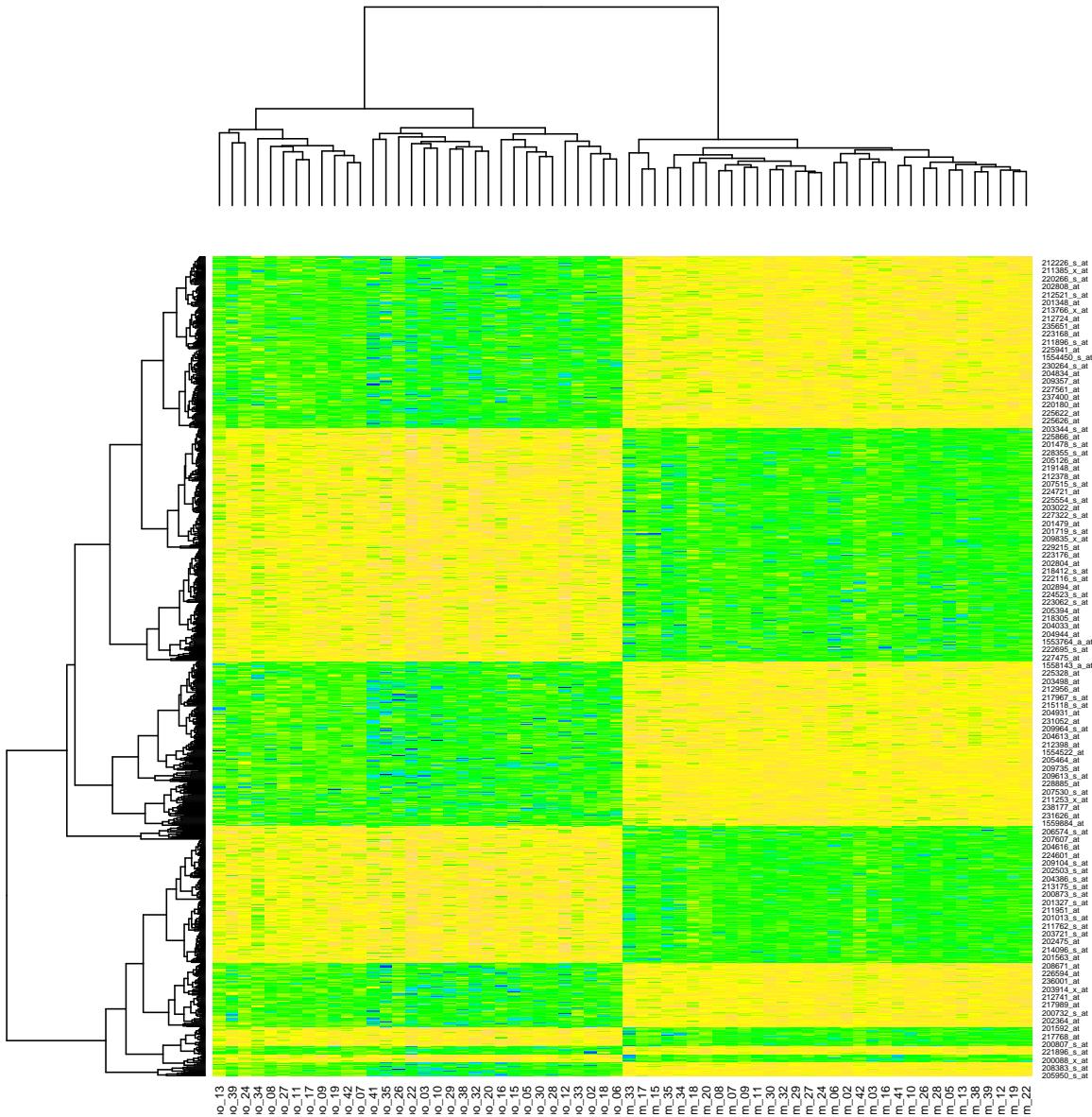


Figure 3.12: Heatmap, see text.

3.6 Bibliographic remarks

The package `cluster` is very established and provides many features. For example, the function `daisy` allows to calculate many different types of distances. The output of a cluster analysis is not an object of class `hclust()` but different methods exist.

The packages `vegan` and `ade4` are very rich and contain many functions for multivariate data analysis. The package `made4` is an extension to `ade4`, tailored for analyses in bioinformatics.

Bioconductor (www.bioconductor.org) is a open source software project for the analysis of genomic data. The vast majority of Bioconductor components are R-packages, which inherently

```
## Warning in factor(as.numeric(covarsraw$Sex) + rep(c(0, 2), each = 32)):
#>   NAs introduced by coercion
## Error in s.class(pr4$li, factor(as.numeric(covarsraw$Sex) + rep(c(0, 2),
#>   : Non equal row numbers 64 0
```

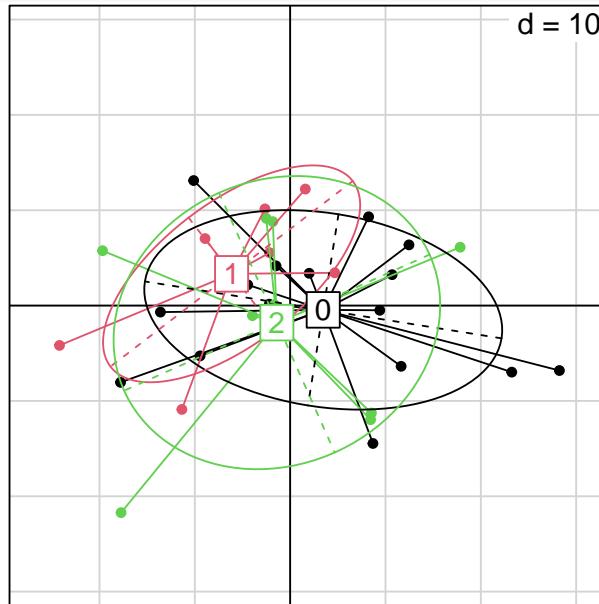


Figure 3.13: PCAs. The spacing of the grid is $d = 10$ units.

run in a classical R environment. To install a Bioconductor package, say `made4`, the following lines are necessary:

```
source("http://bioconductor.org/biocLite.R")
biocLite("made4")
```

3.7 Exercises and Problems

Problem 3.1 (Clustering) We use again the `protein.txt` dataset, available from <https://www.math.uzh.ch/furrer/download/sta121/protein.txt>.

- Conduct a cluster analysis, using all the three agglomeration methods seen during the lesson. Discuss the figures.
- Select one method. Is it possible to cut the tree such that the resulting classification can be interpreted in a meaningful way?
- Compare the results obtained from the application of the PCA analysis and a clustering method. Can you see any similarities or dissimilarities in the analysis performed?

Problem 3.2 (Differences between clustering approaches with artificial data) In this problem, we want to understand how the different agglomerative schemes of hierarchical clustering as well

as a k -means form their clusters. For both situations below, try to look at the scatterplot of the data and cluster mentally with single and complete linkage, Ward and k -means.

- a) Generate artificial data as below and create four clusters. Interpret.

```
set.seed(17)
pts <- rbind(cbind(rnorm(49), rnorm(49)), c(3, 3))
```

- b) Generate artificial data as below and create four clusters. Interpret the result.

```
set.seed(17)
pts <- cbind(rep(c(1, 2.5, 4.5, 7.5), each=10), 1:10) + rnorm(40, sd=.05)
```

Problem 3.3 (Univariate K-means function in R)

Write a function called `my.kmean()`, which takes as an input a vector `x` and the number of clusters `k`, does the k-means clustering in one dimension, returns a data.frame with the original `x` as one column, the assigned cluster numbers as the second column.

For simplicity, you can specify the number of iterations after which the algorithm stops.

Apply your function to `x<-c(1,2,1,3,2,6,5,7,6,12)`, `k=3`.

Chapter 4

Discrimination and Classification

Given data consisting of different, known, groups, we discuss how we can find simple rules (in terms of the variables of the data) that characterize the groups. This characterization can be used as a simple classifier for observations without a known or given group attribute.

Learning goals for this chapter:

- ◊ Understand the basic setup of a classification problem
- ◊ Explain and run in R the linear and quadratic discrimination for Gaussian case
- ◊ Define and apply in R Fisher's linear discrimination rule
- ◊ Explain and apply in R classification trees; interpret the outcome
- ◊ Describe bagging and boosting

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter04.R.

4.1 Classification in the Gaussian Case

Classification is quite similar to clustering and we will refer quite often to the previous chapter. Compared to the case of clustering, we now *know* the group attribute of all observations, i.e., we have data from different groups or populations. The goal is to find a “separation” rule that separates the space of the variables or observations according to the different groups. In practice, these separation rules are hard to describe in closed form and we often reduce the setting to be able to predict group membership for arbitrary values of the space.

We illustrate discrimination first with Gaussian random variables. More specifically, suppose we have observations from $\mathbf{Y}_1, \dots, \mathbf{Y}_{n_1} \stackrel{\text{iid}}{\sim} \mathcal{N}_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ and $\mathbf{Y}_{n_1+1}, \dots, \mathbf{Y}_{n_1+n_2} \stackrel{\text{iid}}{\sim} \mathcal{N}_p(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$ available for discrimination. Then we observe a new value \mathbf{y} . The goal is to determine to which of the two groups the realization \mathbf{y} belongs, i.e., to classify \mathbf{y} . In other words, to find a function $G(\cdot)$, such that $G(\mathbf{y}) = \ell$ with ℓ the group label. In the ideal setting, we not only argue that \mathbf{y} stems from group ℓ but we also attribute a probability that \mathbf{y} is a realization of group ℓ .

4.1.1 Linear Discrimination

As a motivating illustration, consider the simple case where we observe two samples from distributions $\mathcal{N}(\mu_1, \sigma^2)$ and $\mathcal{N}(\mu_2, \sigma^2)$. Intuitively, we partition the real line such that future observations are classified by the partition they fall into. This particular case is illustrated in Figure 4.1, where the real line is separated into the segments $]-\infty, (\mu_1 + \mu_2)/2[$ and $](\mu_1 + \mu_2)/2, \infty[$ because the two densities intersect at $(\mu_1 + \mu_2)/2$. Naturally, if the means are not known, we would use plug-in estimates.

Example 4.1. We consider $Y_1 \sim \mathcal{N}(-1, 1)$ and $Y_2 \sim \mathcal{N}(1.5, 1)$. The densities intersect at $1/4 = (\mu_1 + \mu_2)/2$, yielding equal areas on corresponding sides (see Figure 4.1).

Suppose we do not know the corresponding means and variances but have samples of size 50. The estimates are close to the unknown truth and yield a cutoff of 0.29314, with 7 and 3 missclassifications. ♣

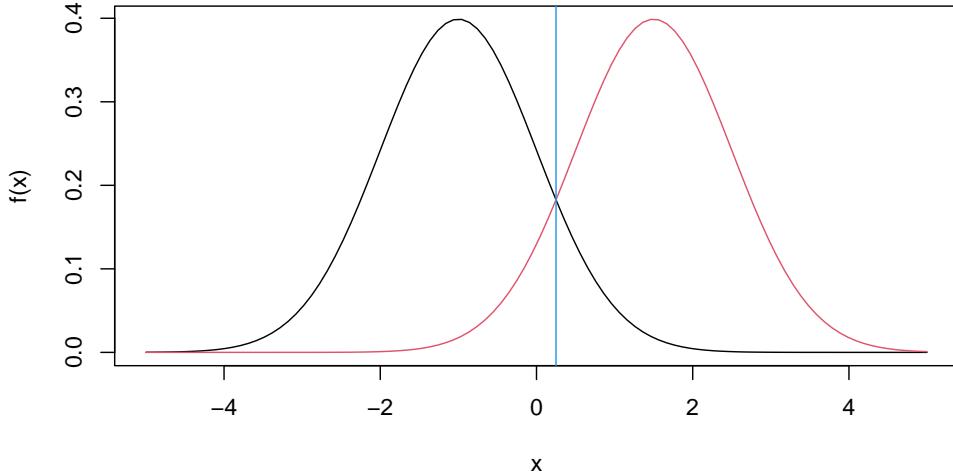


Figure 4.1: Linear discrimination in one dimension: two Gaussian densities. Discrimination has partitioned the observations at $1/2$ because $(\mu_1 + \mu_2)/2 = 1/2$, indicated with the blue vertical line. (See R-Code 4.1.)

The concept of *linear discrimination analysis* (LDA) can be extended to k different groups, each with p observed variables. In two dimensions, we describe the problem as such: find β_0 and β_1 such that $x_2 = \beta_0 + \beta_1 x_1$ partitions the plane. Here the partition is a linear combination, and thus we call the method linear discrimination.

In the case of Gaussian bivariate densities (with equal variances) the parameters β_0 and β_1 can be found by

R-Code 4.1 Linear discrimination of Gaussian densities in one dimension. (See Figure 4.1.)

```

x <- seq(-5, 5, l=100)
plot(x, dnorm(x, -1), type="l", ylab="f(x)")
lines(x, dnorm(x, 1.5), col=2)
abline(v=1/4, col=4)
set.seed(14)
y1 <- rnorm(50, -1) # empirical
y2 <- rnorm(50, 1.5)
c(mean(y1), mean(y2), (mean(y1)+mean(y2))/2)
## [1] -0.79605 1.38234 0.29314
c(sum((mean(y1)+mean(y2))/2<y1), sum((mean(y1)+mean(y2))/2>y2))
## [1] 7 3

```

- maximizing the likelihood of the density, or
- minimizing the error of wrong classification.

The result is illustrated in Example 4.2.

Example 4.2. R-Code 4.2 illustrates linear discrimination in the case of bivariate Gaussian densities with shifted means. The discrimination line has intercept -2 and slope $2/3$. The direction onto we project is given by $\begin{pmatrix} -2 \\ 3 \end{pmatrix} = \Sigma^{-1}(\mu_1 - \mu_2)$. The discrimination line passes through the center of the line between the means $\begin{pmatrix} 3 \\ 0 \end{pmatrix} = (\mu_1 + \mu_2)/2$. Hence, the discrimination line solves $(-2, 3)(\begin{pmatrix} x-3 \\ y-0 \end{pmatrix}) = 0$.

If we sample from the densities and estimate the parameters, the discrimination slightly changes, as illustrated in the right panel of Figure 4.2. Notice that we have not determined the discrimination line directly but “predicted” a fine grid through ♣

R-Code 4.2: Linear classification in the setting of bivariate Gaussian densities with and without known parameters. (See Figure 4.2.)

```

library(mvtnorm)
mu1 <- c(2, 1)
mu2 <- c(4,-1)
Sigma <- matrix(c(6,2,2,2), 2)
x1 <- seq(-3, to=7, length=99)
x2 <- seq(-4, to=4, length=100)
grid <- expand.grid(x=x1, y=x2)
dens1 <- matrix(dmvnorm(grid, mean=mu1, sigma=Sigma), 99, 100)
dens2 <- matrix(dmvnorm(grid, mean=mu2, sigma=Sigma), 99, 100)
contour(x1, x2, dens1, nlevels=7, drawlabels=F, xlab="x1", ylab="x2")

```

```

contour(x1, x2, dens2, nlevels=7, drawlabels=F, col=2, add=T)
(mu <- (mu1+mu2)/2 )
## [1] 3 0
(as <- solve(Sigma, mu1-mu2) )
## [1] -1 2
abline(a=as[1]/as[2]*mu[1]+mu[2], b=-as[1]/as[2], col=4, lwd=2)
points(rbind((mu1+mu2)/2), col=4, pch=19)
## Construct right panel based on empirical quantities
library(MASS)
set.seed(17)
N <- 25
s1 <- rmvnorm(N, mean=mu1, sigma=Sigma)
s2 <- rmvnorm(N, mean=mu2, sigma=Sigma)
obs <- rbind(s1, s2)
colnames(obs) <- c("x", "y")
group <- factor(rep(c("a", "b"), each=N))
tmp <- lda(obs, grouping=group)           # from package MASS
cll <- as.numeric(predict(tmp, grid)$class) # predict the fine grid
image(x1, x2, matrix(cll, 99, 100), col=c(rgb(0,0,0,.2), rgb(1,0,0,.3)))
points(obs, col=group, pch=20)
abline(a=as[1]/as[2]*mu[1]+mu[2], b=-as[1]/as[2], col=4, lwd=2) # as above

```

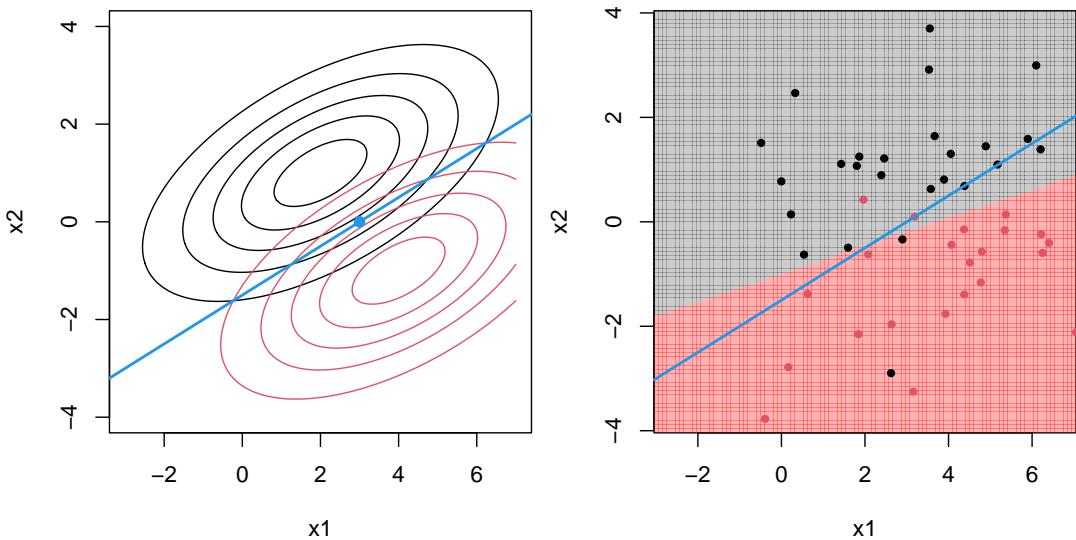


Figure 4.2: Linear discrimination in the case of bivariate densities with known parameters (left) and in the case of samples (right). The separation based on the (known) densities is indicated with the blue line whereas the empirical separation with the gray area. (See R-Code 4.2.)

4.1.2 Quadratic Discrimination

In the case of unequal variances, linear discrimination may not be sufficient. As a motivation, Figure 4.3 shows the log-densities of $\mathcal{N}(-1, 1)$ and $\mathcal{N}(2, 4)$. It is obvious that a single separating line will not classify new observations as accurately as for the simpler case.

Mathematically, when calculating the values for which the (log-)densities intersect, the quadratic terms in the kernel of the Gaussian density do not vanish and thus one has to solve the quadratic equation

$$y^2 \left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2} \right) - 2y \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2} \right) + \left(\frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_2^2}{\sigma_2^2} \right) - 2 \log \frac{\sigma_2}{\sigma_1} = 0. \quad (4.1)$$

Hence the name *quadratic discrimination analysis* (QDA).

Naturally, if $\sigma_2 = \sigma_1$, the unique root of (4.1) is at $(\mu_1 + \mu_2)/2$, as seen in last section. For more than two densities, several roots have to be calculated but not all of these might be relevant or used for the discrimination.

In higher dimensions, the analytic solution does not have a simple closed form and rely again on numeric approaches. The next two examples show the numerical approaches based on the functions `uniroot()` and `MASS::qda()` for one and two dimensions, respectively.

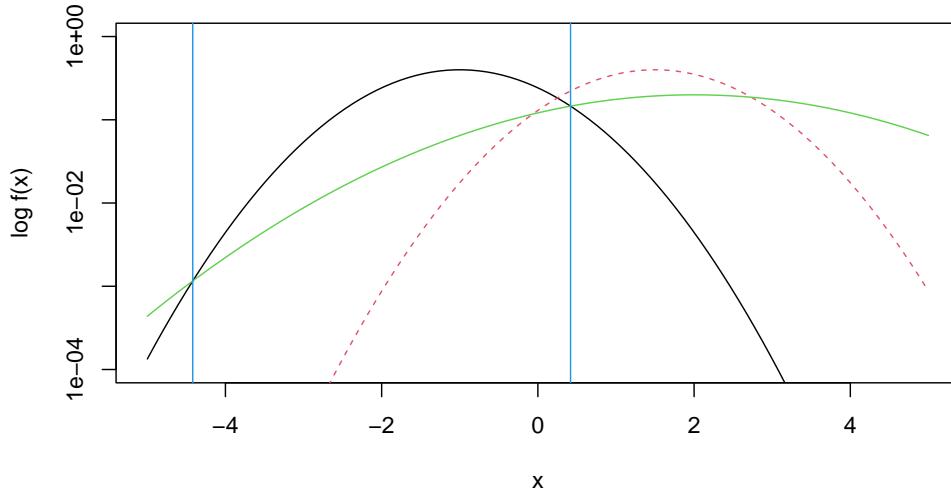


Figure 4.3: Discrimination in one dimension with unequal variances. The blue vertical lines discriminate the densities $\mathcal{N}(-1, 1)$ (black) and $\mathcal{N}(2, 4)$ (green). Red dotted density is as in Figure 4.1. (See R-Code 4.3.)

Example 4.3. Although we would have closed form solutions for the roots of equation (4.1), R-Code 4.3 uses a numerical approach for the densities $\mathcal{N}(-1, 1)$ and $\mathcal{N}(2, 4)$. If we would include $\mathcal{N}(1.5, 1)$ as third density, only small (< 4.42) or large values (< 2.73) would be attributed to $\mathcal{N}(2, 4)$, as seen in Figure 4.3 (for better visibility, with log densities). ♣

R-Code 4.3: Gaussian densities with different variances. (See Figure 4.3.)

```

x <- seq(-5,5, l=100)
plot(x, dnorm(x, -1), type="l", ylab="log f(x)", log="y", ylim=c(1e-4,1))
lines(x, dnorm(x, 1.5), col=2, lty=2)
lines(x, dnorm(x, 2, sd=2), col=3)
# numerically determine intersections:
quadfun <- function(x, m1, m2, s1, s2) x^2*(1/s1^2-1/s2^2) -
  2*x*(m1/s1^2-m2/s2^2) + (m1^2/s1^2-m2^2/s2^2) - 2*log(s2/s1)
(ints <- c(uniroot(quadfun, m1=-1, m2=2, s1=1, s2=2, c(-5, 0))$root,
           uniroot(quadfun, m1=-1, m2=2, s1=1, s2=2, c(0, 1))$root ))
## [1] -4.41834  0.41833
# because there are two zeros, we need two calls to uniroot...
abline(v=ints, col=4)

```

Example 4.4. We now compare linear and quadratic discrimination with three different groups, where two of the three share the same covariance matrix (actually, we use the same distributions as in Example 4.2). We notice that the boundaries between two groups are very similar between LDA and QDA if the covariance matrices are identical. ♣

R-Code 4.4: Linear and quadratic discrimination based on data from three different bivariate Gaussian densities with different covariances. (See Figure 4.4.)

```

mu3 <- c(-1,2)                                # third mean
Sigma3 <- matrix(c(2, -1.2, -1.2, 4), 2, 2)    # and covariance
s3 <- rmvnorm(N, mean=mu3, sigma=Sigma3)        # sample thereof
obs <- rbind(s1, s2, s3)                        # collect all observations
colnames(obs) <- c("x", "y")
group <- factor(rep(c("a","b","c"), each=N)) # define groups
tmp <- lda(obs, grouping=group)
cll <- predict(tmp, grid)$class
image(x1, x2, matrix(as.numeric(cll), 99, 100), # partition of space
      col=c(rgb(0,0,0,.2), rgb(1,0,0,.3), rgb(0,1,0,.3)))
points(obs, col=group, pch=20)
densgrid3 <- dmvnorm(grid, mean=mu3, sigma=Sigma3) # we add contour lines
dens3 <- matrix(densgrid3, 99, 100)
contour(x1, x2, dens1, nlevels=7, col=1, add=T, drawlabels=F)
contour(x1, x2, dens2, nlevels=7, col=2, add=T, drawlabels=F)
contour(x1, x2, dens3, nlevels=7, col=3, add=T, drawlabels=F)

# Now quadratic discrimination:

```

```

tmp <- qda(obs, grouping=group)
clq <- predict(tmp, grid)$class
image(x1,x2, matrix(as.numeric(clq), 99, 100),
      col=c(rgb(0,0,0,.2), rgb(1,0,0,.3), rgb(0,1,0,.3)))
points(obs, col=group, pch=20)

```

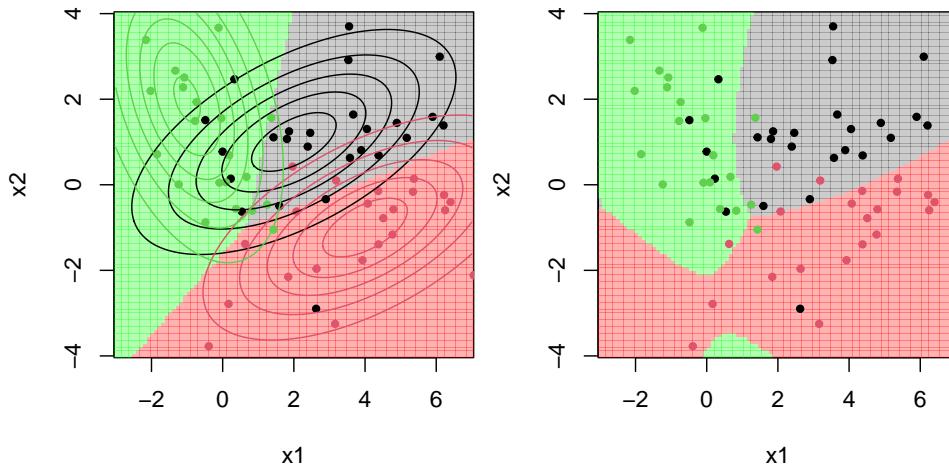


Figure 4.4: Linear (left) and quadratic (right) discrimination in the case of bivariate densities from three samples (based on different covariances). Densities indicated with black and red isolines are as in Figure 4.2. In the right panel contour plots have been omitted for better visibility. (See R-Code 4.4.)

4.1.3 Likelihood and Bayesian Discrimination

Likelihood discrimination partitions the space such that at each location the density of the corresponding class is maximized.

In fact, this is equivalent to the discrimination rule we have applied in the last paragraph. Naturally, likelihood discrimination is not limited to a Gaussian distribution as assumed until here. A new observation is assigned to the group that maximizes the likelihood, i.e., the point is assigned to the group with the highest density at that point. Hence, for likelihood classification we “only” need (parametric form) densities. If exact parameters are unknown, but samples are available, the parameters can first be estimated and then likelihood discrimination applied. For non-Gaussian cases, the optimization problem might be complicated, especially in higher dimensions.

In Bayesian discrimination we additionally assign prior probabilities to the corresponding classes. In practice, these are often chosen proportional to the number of observations. This choice reflects the different uncertainties resulting from the different (group) sample sizes.

In the previous examples, we can add prior belief by setting the `prior` argument in the functions `lda()` and `qda()`.

4.1.4 Fisher's Linear Discrimination Rule

If we do not have parametric densities for the individual groups, a standard likelihood approach cannot be used for classification. An alternative is to search for a hyperplane \mathbf{a} such that the projected observations $\mathbf{a}^\top \mathbf{y}_1, \dots, \mathbf{a}^\top \mathbf{y}_n$ can be easily discriminated. Figure 4.5 illustrates this with three different “hyperplanes” $\mathbf{a} = (0.42, -0.52)^\top$, $\mathbf{b} = (1, 0)^\top$ and $\mathbf{c} = (0, 1)^\top$. The projection on the y -axis ($\mathbf{c} = (0, 1)^\top$) is useless as both groups have a similar range of y values. Projecting on the x -axis yields a much better result and the value 44 seems to separate the two groups quite well with only two individuals wrongly placed. Best is the projection based on \mathbf{a} , the two groups are nearly perfectly separated and within each group, the observations cluster tightly.

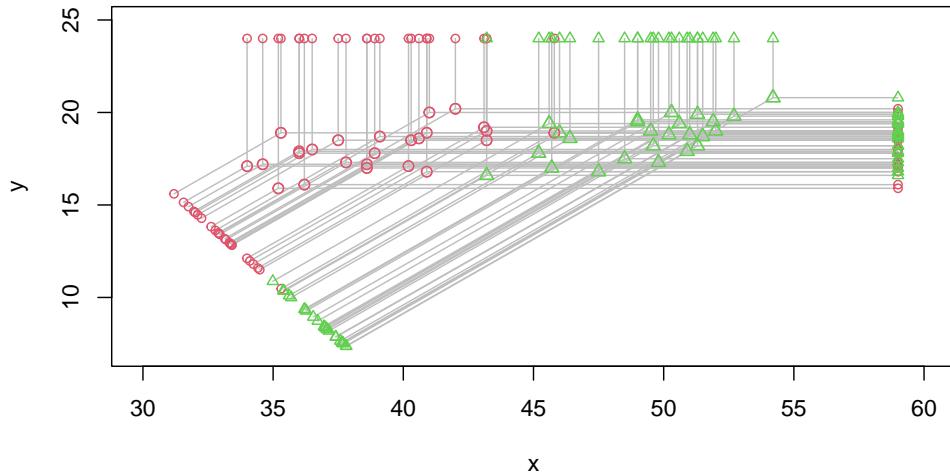


Figure 4.5: Illustration of projections onto two lines (hyperplanes of dimension one) used for discrimination. The line given by $\mathbf{a} = (0.42, -0.52)^\top$ is optimal (Fisher's linear discrimination rule) whereas the ones given by $\mathbf{b} = (1, 0)^\top$ and $\mathbf{c} = (0, 1)^\top$ are not. The code to construct the is given in the online supplement.

The discrimination is based on a sum of squares decomposition:

$$\begin{aligned} & \text{total sums of squares} \\ &= \text{between-groups sum of squares} + \text{within-groups sum of squares} \end{aligned} \tag{4.2}$$

$$= \mathbf{a}^\top \mathbf{B} \mathbf{a} + \mathbf{a}^\top \mathbf{W} \mathbf{a}, \tag{4.3}$$

where the matrices \mathbf{B} and \mathbf{W} are the between- and within-groups (empirical) covariances. The solution is to maximize the first and to minimize the second term. This is equivalent to maximizing the ratio $\mathbf{a}^\top \mathbf{B} \mathbf{a} / \mathbf{a}^\top \mathbf{W} \mathbf{a}$ and thus the solution is the eigenvector associated with the largest eigenvalue of $\mathbf{W}^{-1} \mathbf{B}$. This approach is called Fisher's linear discrimination rule.

In case of two groups with distributions with equal variances $\mathcal{N}_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ and $\mathcal{N}_p(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$, Fisher's linear discriminate rule is identical to linear discrimination.

4.2 Classification Trees

In this section we look at so-called *classification trees* or decision trees as a classifier. This method addresses the problem of group prediction for new observations by breaking down the multivariate data into successive univariate rules that best separate the data.

A classification tree is constructed as follows. At each consecutive step the algorithm separates a subset of all individuals into two groups, augmenting the purity within each group, and thus creating a new node in the tree. The separation is based on a single variable that best discriminates between one group and all others. The assessment of the purity can be accomplished by a criterion like Gini impurity, information gain, variance reduction, etc. Although we do not require distributional assumptions, we can still use the multinomial distribution of the classes within each group and apply entropy type arguments. The assessment rule also determines when to stop separating, i.e., when a terminal node has been reached.

We want to avoid overfitting and overly complex trees that do not generalize well. Different algorithms address this overfitting differently. For example, they may apply pruning after the tree has been constructed. That means we reduce the size of classification trees by removing branches that may improve the overall classification accuracy. The next example illustrates the idea of a classification tree.

Example 4.5. We use the *Penguins* dataset to construct a classification tree as implemented by the function `ctree()` from the package *partykit* (see R-Code 4.5). When using the four numeric variables `bill_length`, `bill_depth`, `flipper_length` and `body_mass` the resulting tree has six inner nodes and seven terminal nodes. Some of the terminal nodes classify the penguins perfectly (e.g., [12]), others not (e.g., [13]).

The visualization of the variable separation is easier when only two variables are used and thus we redo a classification based on the variables `bill_length`, `bill_depth`. Figure 4.6 shows a graphical representation of the classification tree based as a dendrogram and the corresponding separation of the plane. For simplicity we omitted the two terminal nodes that lead to the same prediction in their partitions ([3] [4] and [15] [16]).



R-Code 4.5: Regression tree for the *peng* data.

```
library(partykit)
print(rt4 <- ctree(species ~ bill_length + bill_depth + flipper_length +
                     body_mass,  data=Penguins))

##
## Model formula:
## species ~ bill_length + bill_depth + flipper_length + body_mass
##
## Fitted party:
## [1] root
## |   [2] flipper_length <= 206
## |   |   [3] bill_length <= 43.2
```

```

## |   |   | [4] bill_length <= 42.3: Adelie (n = 139, err = 1%)
## |   |   | [5] bill_length > 42.3: Adelie (n = 11, err = 36%)
## |   |   [6] bill_length > 43.2
## |   |   | [7] bill_depth <= 19.6
## |   |   | [8] bill_depth <= 17: Chinstrap (n = 8, err = 12%)
## |   |   | [9] bill_depth > 17: Chinstrap (n = 43, err = 2%)
## |   |   | [10] bill_depth > 19.6: Chinstrap (n = 12, err = 25%)
## |   |   [11] flipper_length > 206
## |   |   | [12] bill_depth <= 17.3: Gentoo (n = 122, err = 0%)
## |   |   | [13] bill_depth > 17.3: Chinstrap (n = 7, err = 29%)
##
## Number of inner nodes: 6
## Number of terminal nodes: 7
# plot(rt4, margin=.2) # plot is easier to interpret.
# Based on bill variables only:
rt2 <- ctree(species ~ bill_length + bill_depth, data=Penguins)
plot(rt2, margin=.3)

# print(rt2) # uncomment to get proper node numbering
id <- as.numeric(Penguins$species)
plot(bill_length~bill_depth, data=Penguins, col=id, pch=id)
abline(h=42.35)      # [2][5] # to better visualize <= vs >, we add 0.05
lines(c(16.35, 16.35), c(42.35, 62))      # [6][7]
lines(c(16.35, 24), c(44.15, 44.15))      # [8][9]
lines(c(17.35, 17.35), c(44.15, 62))      # [10][13]
lines(c(16.35, 17.35), c(48.15, 48.15))  # [10][13]

```

Figure 4.5 illustrates the biggest disadvantage of classification tree: the resulting subspace is hard to interpret and the discrete separation boundaries are counter-intuitive. However, regression trees are typically very fast to construct. Hence, decision trees are commonly used in situations where we have many predictors and/or complex, unknown relationships between the data and class. Historically, classification trees refer to outcome variables that encode the group and regression trees refer to when the outcome is an arbitrary real number. The term Classification And Regression Trees (CARTs) refers to both of the above procedures. There are many algorithms developed in the machine learning literature.

4.3 Bagging, Boosting and More

Bagging is an abbreviation of bootstrap aggregation. Bagging tries to reduce the prediction variance by artificially increasing the training data. A bootstrap approach resamples from the original (training) dataset and thus creates a set of additional training sets of equal sizes. To each of the samples a simple classification algorithm is applied. While the predictive power of the method is not improved (we use the same type of algorithm for the original and the

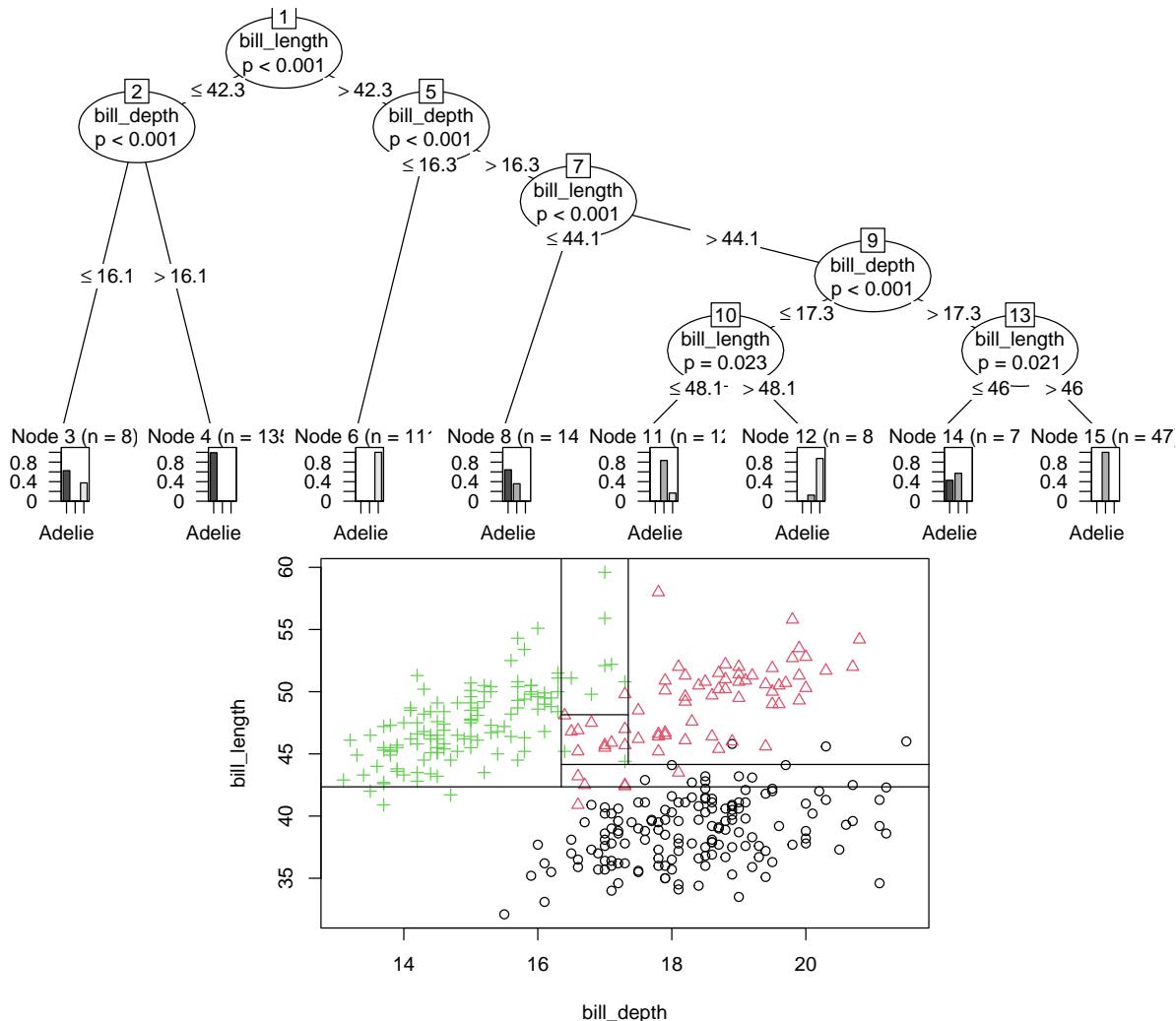


Figure 4.6: Dendrogram of the `peng` regression tree based on variables `bill_depth` and `bill_right` (top) and associated regression tree partition (bottom). (See R-Code 4.5.)

bootstrapped test sets), the variance in the prediction is reduced. Thus bagging can be seen as a meta-algorithm. Through the averaging approach bagging typically prevents overfitting. In fact, bagging is a special case of a so-called *model averaging* approach.

The next example illustrates bagging using the `Penguin` dataset by carefully comparing the predictive capabilities which we first outline.

To compare a set of predicted class labels with observed ones, it is possible to display the so-called *confusion matrix*: a matrix with ij th entry giving the number of predicted class i for observed class j . Assume that the confusion matrix is denoted with `conf.mat`. A good classification yields a almost diagonal matrix confusion matrix. Other measures are as follows. The *accuracy* is the ratio of all correctly identified to all possible cases, i.e., `sum(diag(conf.mat))/sum(conf.mat)`. The *precision* or true positive predictive value for each class is the ratio of correctly identified to all predicted in that particular class, i.e., `diag(conf.mat)/rowSums(conf.mat)`. Similarly, the *sensitivity* or recall for each class is the ratio of correctly identified to all observed in that class, i.e., `diag(conf.mat)/colSums(conf.mat)`.

Example 4.6. We will illustrate bagging using the *Penguin* dataset based on both bill variables. To compare the approaches, we set aside a randomly selected subset (roughly 33%) to use in prediction.

In R-code 4.6, we use the function `bagging()` from the package `adabag`. For bagging, we set the number of iterations to `mfinal=11` as more iterations do not change predictions.

The `predict()` function returns an entire list containing, among others, the class probabilities, predicted class and the confusion matrix. We use the latter to calculate the accuracy, precision and sensitivity. ♣

R-Code 4.6: Bagging based classification of the *Penguins* data.

```
set.seed(14)
sel <- sample(nrow(Penguins), 224)    # define subset for training
train <- Penguins[sel, c(1,3:4)]
test <- Penguins[-sel, c(1,3:4)]

library("adabag")
peng.bagging <- bagging(species~, data=train, mfinal=11)
bagging.pred <- predict(peng.bagging, newdata=test)
print(conf.mat <- bagging.pred$confusion) # confusion matrix

##          Observed Class
## Predicted Class Adelie Chinstrap Gentoo
##      Adelie        48       1       4
##      Chinstrap     1      17       4
##      Gentoo        0       0      43
sum(diag(conf.mat)) / sum(conf.mat)      # accuracy (overall measure)
## [1] 0.91525
diag(conf.mat) / rowSums(conf.mat)        # precision for each class
##      Adelie Chinstrap   Gentoo
## 0.90566 0.77273 1.00000
diag(conf.mat) / colSums(conf.mat)        # sensitivity for each class
##      Adelie Chinstrap   Gentoo
## 0.97959 0.94444 0.84314
```

A random forests algorithm extends the bagging algorithm. Instead of having all predictors at all fitting stages a random subset of the predictors is chosen before fitting each new candidate split. This approach increases variation in the trees grown and prevents strongly correlated trees from occurring if there are very prominent predictors.

Once a forest consists of a (determined) number of trees, the classification for a new observation is given by the majority consensus based on the classification of each tree. We continue to use the *Penguin* dataset to illustrate different random forest algorithms in the next example.

Example 4.7. We use the same data as in Example 4.6. R-code 4.7 illustrates the use of the function `randomForest()` from the same named package and the function `cforest()` from the package `partykit`.

As shown by the confusion matrix, the algorithms give very similar results (we have a quite easy task, even a simple classification tree performs well). Figure 4.7 shows the individual class probabilities, classification is done by taking the largest one.

For the random forests, 50 trees are grown (upper bound) and, although typically not necessary, it is possible to access individual trees and their nodes through `getTree` or `$nodes` (see the last few lines of code).



R-Code 4.7: Random forest based classification of the `Penguin` data.

```

library("randomForest")
peng.rf <- randomForest(species ~ ., data=train, ntree=50, prox=TRUE)
rf.pred <- predict(peng.rf, newdata=test)
table(rf.pred, test$species, dnn=c("Predicted Class", "Observed Class"))

##          Observed Class
## Predicted Class Adelie Chinstrap Gentoo
##      Adelie        48         1         0
##      Chinstrap      1        17         3
##      Gentoo         0         0        48

ct <- cforest(species ~ ., data=train, ntree=50) # with package partykit
ct.pred <- predict(ct, newdata=test)
table(ct.pred, test$species, dnn=c("Predicted Class", "Observed Class"))

##          Observed Class
## Predicted Class Adelie Chinstrap Gentoo
##      Adelie        48         1         2
##      Chinstrap      1        17         5
##      Gentoo         0         0        44

peng.rt <- ctree(species ~ ., data=train) # Compare with simple tree
rt.pred <- predict(peng.rt, newdata=test)
table(rt.pred, test$species, dnn=c("Predicted Class", "Observed Class"))

##          Observed Class
## Predicted Class Adelie Chinstrap Gentoo
##      Adelie        48         1         6
##      Chinstrap      1        14         2
##      Gentoo         0         3        43

x <- seq(13, to=22, length=50)
y <- seq(32, to=60, length=49)
nd <- expand.grid(bill_depth=x, bill_length=y)
p <- predict(ct, newdata=nd, type="prob")

```

```

image(x, y, matrix(p[,1], nrow=length(x)), main="Adelie", zlim=c(0,1),
      xlab="bill_depth", ylab="bill_length", col=tim.colors(10))
points(bill_length ~ bill_depth, data=train, col=id[sel], pch=id[sel])
image(x, y, matrix(p[,2], nrow=length(x)), main="Chinstrap", zlim=c(0,1),
      xlab="Sepal.Length", ylab="Sepal.Width", col=tim.colors(10))
points(bill_length ~ bill_depth, data=train, col=id[sel], pch=id[sel])
image.plot(x, y, matrix(p[,3], nrow=length(x)), main="Gentoo", zlim=c(0,1),
            xlab="Sepal.Length", ylab="Sepal.Width", col=tim.colors(10))
points(bill_length ~ bill_depth, data=train, col=id[sel], pch=id[sel])
length(ct$nodes)           # nr of trees in random forest (== ntree=50)
## [1] 50
# ct$nodes[[11]]           # accessin the 11th tree
# ct$nodes[[11]]$split[1:2] # first split of 11th tree
# ct$nodes[[11]]$kids[1:2] # branches after first split
# ct$nodes[[11]]$kids[[2]]$split[1:2] # second split of second branch, etc

```

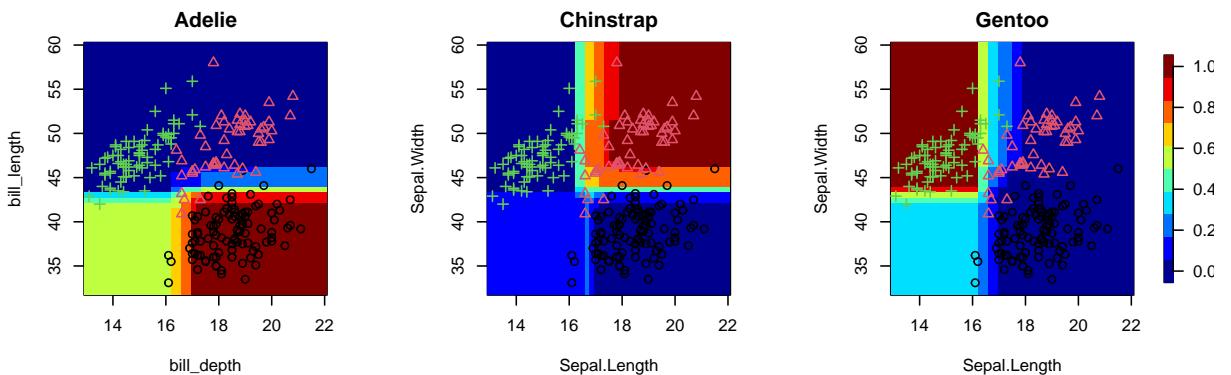


Figure 4.7: Classification probabilities of a random forest. For each pixel, the probabilities sum to one. Training data are visualized (\circ : Adelie, Δ : Chinstrap, $+$: Gentoo). (See R-Code 4.7.)

Boosting is based on the idea that it is possible to construct a strong classifier from many weak classifiers. Basically, boosting calculates the output using several different models and then averages the result using a weighted average approach. The key is to wisely select the the weak classifiers and to appropriately weight their result.

Example 4.8. We use again the same data as in Example 4.6. R-code 4.8 illustrates the use of the function `boosting()` from the package `adaboost`. No surprise, we observe similar classification performance as with random forests.



R-Code 4.8: Boosting *Penguin* data.

```

peng.adaboost <- boosting(species~, data=train, mfinal=11) # in pkg adabag
adaboost.pred <- predict(peng.adaboost, newdata=test)
adaboost.pred$confusion

##           Observed Class
## Predicted Class Adelie Chinstrap Gentoo
##      Adelie        46         1         0
##      Chinstrap      2        17         3
##      Gentoo         1         0        48

```

One of the most famous boosting algorithms is the so-called AdaBoost (Freund and Shapire, 1995) which we outline here. As input we have the triples (\mathbf{y}_i, g_i, w_i) , $i = 1, \dots, n$, where \mathbf{y}_i denotes the variables of the data point i , g_i the associated group label and w_i is the starting weight, set to $1/n$. For $m = 1$ to M algorithm loops over the following steps:

- [1] Fit a classifier $G^{(m)}(\cdot)$ to the training data with weights w_i
- [2] Calculate the error measure $e^{(m)} = \sum_{i=1}^n w_i I(g_i \neq G^{(m)}(\mathbf{y}_i)) / \sum_{i=1}^n w_i$
- [3] Calculate the update weight $a^{(m)} = \log\left(\frac{1 - e^{(m)}}{e^{(m)}}\right)$
- [4] Update the weights $w_i \leftarrow w_i \exp(a^{(m)} I(g_i \neq G^{(m)}(\mathbf{y}_i)))$ and normalize.

The final class is determined by

$$\operatorname{argmax}_k \sum_{m=1}^M a^{(m)} I(k = G^{(m)}(\mathbf{y}_i)). \quad (4.4)$$

In the algorithm above $I(\dots)$ represents the indicator function. Steps [2] to [3] serve to update the weights w_i appropriately in case a classification is incorrect, i.e., $g_i \neq G^{(m)}(\mathbf{y}_i)$.

Of course many variants of this algorithm exist, for K classes, the error measure in Step [2] if further increased by $\log(K - 1)$. We illustrate the boosting algorithm with a manual implementation thereof.

Example 4.9. Suppose that we observe two univariate samples that we want to classify to class 0 and 1. As weak classifier we choose $G(y) = I(y > (\hat{\mu}_1 + \hat{\mu}_2)/2)$, where $\hat{\mu}_k$ are weighted mean estimates of the groups and thus the classifier is in the spirit of LDA, see Example 4.1. Using the indicator function yields directly the class label 0 and 1.

Figure 4.8 shows the evolution of boosting over the iterations; lowest row of points shows the data, each successive steps the selected data points (training set) with size proportional to the error. The top row gives the final classifier. In Step $m = 4$, $a^{(4)} = 0$ and thus the weights are not updated (circle sizes at step 3 and 4 are identical). 

R-Code 4.9: Manual implementation of a boosting algorithm (See Figure 4.8.)

```

set.seed(14)

n1 <- 12      # group sizes
n2 <- 12
M <- 7        # number of iterations
tmp <- rbind(WC=0, err=numeric(M), a=0)
dat <- data.frame(g=rep(0:1, c(n1,n2)),y=c(rnorm(n1, 0), rnorm(n2, 1.5)))
n <- n1 + n2 # total number of observations
plot(dat$y, rep(0, n), col=dat$g+1, ylim=c(0, M+1), pch=19,
      ylab="m (1,...,M)", xlab="x")
col <- c(rgb(0,0,0,.2), rgb(1,0,0,.3))
w <- rep(1/n,n) # weights, will be adapted
WC <- function(dat, w) # weak classifier
  (weighted.mean(dat$y[dat$g==0], w[dat$g==0]) +
   weighted.mean(dat$y[dat$g==1], w[dat$g==1]))/2
for (i in 1:M) {
  sel <- sample(n, .8*n)           # selector for training dataset
  WCout <- WC(dat[sel,], w[sel])   # weak classifier. 0 if x < WCout
  tmp["WC",i] <- WCout
  wrong <- as.numeric(dat$g != (WCout < dat$y)) # wrongly classified
  tmp["err",i] <- sum(w * wrong)/sum(w)           # error measure 2.
  tmp["a",i] <- log((1-tmp["err",i])/tmp["err",i]) # update weight 3.
  w <- w * exp(tmp["a",i]* wrong)                 # update weights
  w <- w/sum(w)                                # normalize weights
  segments(c(-4,WCout), c(i,i), c(WCout, 4), c(i,i), col=col, lwd=10)
  points(dat$y[sel], rep(i, .8*n), cex=exp(w[sel]*n)/3, col=dat$g[sel]+1)
}
tmp                               # show thresholds, errors and update-weights
##      [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## WC  1.40599  0.98652  1.270256  1.2599e+00  1.18347  1.26285  1.17242
## err  0.29167  0.37395  0.496116  5.0000e-01  0.42404  0.55381  0.46144
## a    0.88730  0.51531  0.015535 -4.4409e-16  0.30623 -0.21607  0.15455

x <- seq(0, to=2, l=50)
finaltmp <- colSums(outer(tmp["WC",], x, "<") * tmp["a",]) >
  colSums(outer(tmp["WC",], x, ">") * tmp["a",])
print(final<- x[which.max(finaltmp)])    # final threshold
## [1] 1.1837
segments(c(-4,final), c(M,M)+1, c(final, 4), c(M,M)+1, col=col, lwd=15)

```

Although general statements cannot be made, typically boosting has a very high prediction power, surpassing bagging and random forest one (which are more powerful than simple decision

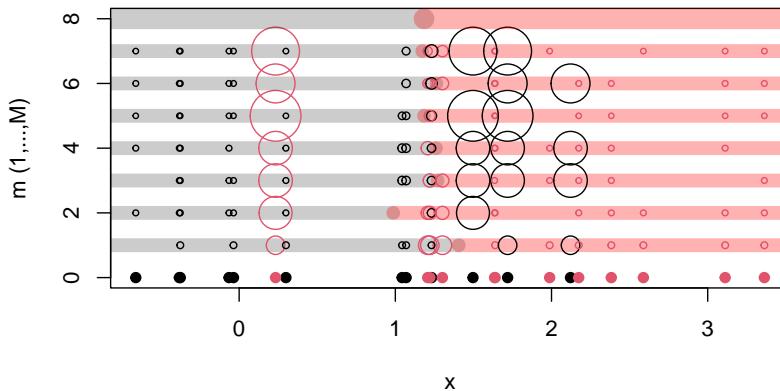


Figure 4.8: Manual implementation of a boosting algorithm. Lowest row of points shows the data, each successive steps the selected data points (training set) with size proportional to the error. The top row gives the final classifier. (See R-Code 4.9.)

trees). Boosting does typically not overfitting but is sensitive to outliers. An outlier may be hard to classify and constantly receives high weight as the classifier tries to overcome the shortcomings of the previous wrongly classified observations. While random forest can be implemented in parallel in a straightforward manner, the sequential nature of boosting prohibits an efficient parallelization.

4.4 Machine Learning and Data Mining

In the last few years machine learning or ML for short seems to be a hot “discipline” and it seems that statistics is getting replaced by ML. Job ads for machine learners seem more ubiquitous than for statistician. So what is ML? As far as I know there is no uniformly accepted and applicable definition of ML. Moreover as a trained statistician, my view here might be slightly biased. One possible approach to differentiate between statistics and ML is with respect to the purpose. ML is mainly concerned with prediction, statistics with inference about the population from a sample. For the former we often have huge datasets containing many observations and many predictors. For the latter we often have very small and limited datasets where we need to be careful about model assumptions, possible dependencies etc. A statistician is very concerned about the interpretability of his method and thus his results. In ML interpretability is of secondary concern, much behind generalizability. For example, to link the content of this chapter, LDA and QDA can be seen as statistical approaches, trees and their generalization more as ML approaches.

To further illustrate, consider as example a linear regression. Regression is probably one of the most classical statistical models relying on very few assumptions. A statistician fits a regression line based on all data and knows (if the assumptions hold) that he found the best solution. A machine learner will separate the data in training, validation and test data and performs several or many regressions. He picks the best and (through the test data) knows something about the performance of the method and, within his measures, knows that the pick is optimal as well.

Of course, as shown in this and the previous chapters, the intersection between both disciplines is not empty. Similarly, a machine learner is a data scientist that has a deep understanding of statistics, algorithms and computer architecture.

Data mining is the discipline (“art”) of discovering patterns for relevant decision processes in vast amounts of data. Hence, data mining can be seen as a subfield of ML.

Table 4.1: Glossary for machine learning terminology. Similar versions of such tables circulate all over the web.

Machine learning	Statistics
network, graphs	model
weights	parameters
features	variables/predictors
learning	fitting
supervised learning	classification or regression
unsupervised learning	clustering or density estimation
generalization	test set performance

The biggest difference between statistics and machine learning/data mining is the terminology. Table 4.1 illustrates the terminology differences of the two fields. While the fields may seem disjoint at first, an understanding of the terminology elucidates the major similarities.

Recently, another hot topic has emerged on the horizon: *big data*. A common description of big data is based on the *x V's* (where *x* increased over time starting with 3):

Volume: the data is large in terms of storage (terabytes and more).

Velocity: the data arrives at high speed (constant flow, streams, . . .). In this context, we refer to the speed at which the data is generated and processed.

Variety: the data is collected by different sources and consists of different types (structured vs. unstructured, measurements, images, . . .).

Veracity: the data is often biased, very noisy and has many features that are not relevant for the purpose of the analysis.

Validity: (sometimes also called volatility, adding a seventh ‘V’) the data has a very limited time window of validity, i.e., the data may be used for making instantaneous decisions, but less useful for understanding the past or making predictions far into the future.

Valuable: the data is highly valuable and ‘big data’ is a big business.

See also https://en.wikipedia.org/wiki/Big_data and <http://insidebigdata.com/2013/09/12/beyond-volume-variety-velocity-issue-big-data-veracity/>.

4.5 Example: Optical Character Recognition

In this section, we illustrate that optical character recognition is straightforward and can be implemented with reasonable precision with a few lines of R-Code.

For simplicity, we try to recognize numbers, based on a small training dataset. For this illustration, we use the png image shown at the top of Figure 4.9. *OCRexample.png* was originally

found at www.oocities.org/santosh_ganti/nn.htm4.gif. We use the top rows as our training set and will predict the very last row. R-Code 4.10 illustrates that OCR is very simple (just a few lines of R-Code!). We first read the image and we aggregate the three RGB values. Most of the code is devoted to transforming the image into an array, where each slice contains one single digit. As a test, we also illustrate a hierarchical clustering approach. The classification is perfect, shown by the middle panel of Figure 4.9. Notice that different aggregation methods tremendously improve the classification. We use classical trees (`rpart()`) and bagging here.

R-Code 4.10: OCR example using `rpart()` and bagging. (See Figure 4.9.)

```

library("png")
datpng <- readPNG("download/sta121/OCRexample.png")
dat <- apply(datpng, c(1,2), sum)
image(1:dim(dat)[1], 1:dim(dat)[2], dat[,], axes=F, xlab="", ylab="")

# specific parameters for this particular example:
nrow <- 10
ncol <- 11
sizer <- dim(dat)[2]/nrow
sizec <- dim(dat)[1]/ncol
abline(v=(1:ncol)*sizec, h=(1:nrow)*sizer)
arr <- array(0, c(sizec, sizer, nrow*ncol))
k <- 1
for (i in 1:ncol) {
  for (j in 1:nrow) {
    arr[,,k] <- dat[ (i-1)*sizec + 1:sizec, ((j-1)*sizer) + 1:sizer]
    k <- k+1
  }
}
# image(arr[,3]) # for example!

arra <- arr
dim(arra) <- c(prod(dim(arr)[1:2]), dim(arr)[3])

plot(hc <- hclust(dist(t(arra))), sub="", main="Complete linkage", xlab="")
# method="ward.D" best, "complete" least
toclass <- cutree(hc, 10)
print(valid <- toclass[ 101:110]-1)      # 101:100 is example specific!!
## [1] 1 4 6 5 3 0 9 7 2 8

# Why does the first a larger dissimilarity?
image(arra[,1]-arra[,3], zlim=c(-3,3), axes=F, xlab="", ylab="")
image(arra[,2]-arra[,3], zlim=c(-3,3), axes=F, xlab="", ylab="")

```

```

# image(1:sizec,1:sizer, apply(arr,c(1,2),sum))
# reduce the pixel size:
arra <- arr[3:(sizec-3), 5:(sizer-3),]
dim(arra) <- c(prod(dim(arra)[1:2]),dim(arra)[3])
arra <- t(arra)
dim(arra)

## [1] 110 240

ar <- data.frame(nr=factor(c(rep(0:9,each=10), valid)),arra)
library("rpart")
rt <- rpart(nr~, data=ar[1:100,] )
rpart.pred <- predict(rt, newdata=ar[101:110,])
rbind(truth=valid, pred=apply(rpart.pred, 1, which.max)-1) # almost perfect
##          101 102 103 104 105 106 107 108 109 110
## truth    1    4    6    5    3    0    9    7    2    8
## pred     1    4    6    5    1    0    9    7    2    8
library("adabag")
ab <- bagging(nr~, data=ar[1:100,], mfinal=40)      # maybe increase mfinal
bagging.pred <- predict(ab, newdata=ar[101:110,])
diag(bagging.pred$confusion)      # perfect
## 0 1 2 3 4 5 6 7 8 9
## 1 1 1 1 1 1 1 1 1 1

```

To avoid alignment issues as illustrated in Figure 4.9, one could eliminate the top and bottom lines with no information (followed by appropriate elimination of columns). With a bilinear interpolator, e.g., `fields::interp.surface()`, all symbols can be brought to one unique size. This works well for “courier” type fonts.

Note that for discrimination, we cannot use `cmtree()` as too much pruning is performed.

4.6 Bibliographic remarks

We can again mention Mardia *et al.* (1979) for a general accessible overview of discrimination.

Decision trees and their extinctions are well explained in Hastie *et al.* (2009). Breiman *et al.* (1984) is the reference for CARTs; Quinlan (1993) for C4.5. (Breiman, 2001) coined the term *Random Forest*, which has since been trademarked. Buehlmann and Hothorn (2007) gives an overview of boosting, algorithmic descriptions thereof can be found in Zhu *et al.* (2009). Strobl *et al.* (2009) gives a very accessible overview of classification (and regression) trees as well as bagging and random forests. There are countless articles about boosting and bagging, as well as many R-packages (`mboost`, `adabag`, ...). The R task view ‘**Machine Learning & Statistical Learning**’ gives a detailed summary of the relevant packages.

The web page <https://statistical-research.com/index.php/2013/04/29/a-brief-tour-of-the-trees-and-forests/> compares recursive partitioning functions of different R-packages.

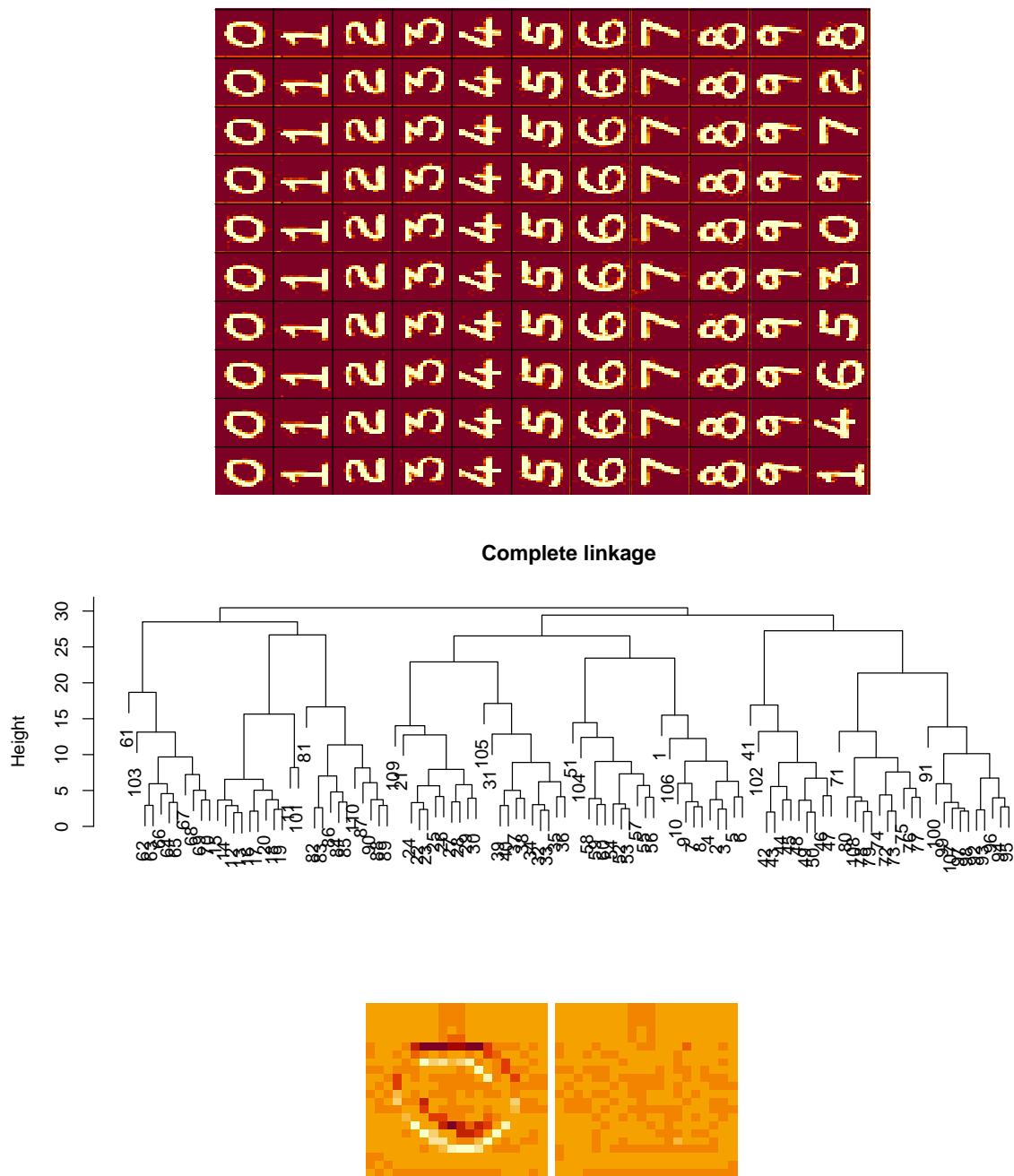


Figure 4.9: Simple OCR example. Top: starting png with delimiting lines for each number. Center: cluster dendrogram using complete linkage. Bottom: illustration of alignment issues. (See R-Code 4.10.)

The R-package `partykit` is a reimplementation of most of the R-package `party`. It has better functionality and more interfaces but is a bit slower because of the R implementation (compared to `party`'s C implementation).

The web page statweb.stanford.edu/~jhf/ftp/dm-stat.pdf gives a more detailed overview of the difference between data mining and statistics.

4.7 Exercises and Problems

Problem 4.1 (LDA/QDA and comparisons with other methods) The dataset `ec.txt` was collected by the Economic Research Department of the Union Bank of Switzerland in 1991. We want to investigate the following question: What was the work and living conditions in different cities around the globe in 1991? (The data are available at <https://www.math.uzh.ch/furrer/download/sta121/ec.txt>.)

The variables are:

<code>City</code> :	City name
<code>Work</code> :	Weighted average of the number of working hours in 12 occupations
<code>Price</code> :	Index of the cost 112 goods and services excluding rent (Zurich=100)
<code>Salary</code> :	Index of hourly earnings in 12 occupations after deductions (Zurich=100)
<code>SalaryCat</code> :	Was generated from Salary with the R code: <code>cut(Salary, breaks=3)</code>

- a) Explore the data and describe them with plots, tables and words. Do you think the collected data of the different countries are comparable?
- b) Which cities are similar? Conduct a cluster analysis. Decide on a “reasonable” size of the grouping. Plot and interpret the results.
- c) Use PCA and display the results in a biplot. Interpret the results.
- d) Use linear and quadratic discriminant analysis to discriminate the three salary levels of `SalaryCat` base on the variables `Work` and `Price`. Display the results graphically. Compare the results of the linear and the quadratic approach.
- e) Compare the findings from the different approaches above. Make five distinct statements and refer to all points a) to d) at least once.

Chapter 5

Linear Models

Linear models are a class of models that relate observations linearly with parameters. They have many appealing properties (statistically and computationally) and thus are widely used.

Learning goals for this chapter:

- ◊ Understand the concept of the least squares criterion
- ◊ Understand the statistical model of linear regression
- ◊ Calculate a predicted value of a dependent variable
- ◊ Conduct linear regression in R, interpret the output
- ◊ Articulate assumptions for multiple linear regression, understand why we need to check the assumptions of the model, check the assumptions using diagnostic plots
- ◊ Know how to calculate a confidence interval for a single slope parameter in the multiple regression setting
- ◊ Translate research questions involving slope parameters into the appropriate hypotheses for testing
- ◊ Explain the statistical model for one-way, two-way, multi-way ANOVA
- ◊ Examine the assumptions for ANOVA and associated diagnostics
- ◊ Use R to run ANOVA and interpret the output
- ◊ Explain the relationship of ANOVA to Regression
- ◊ Know the types of research questions that can be answered using the linear regression and ANOVA

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter05.R.

5.1 Multiple Regression

We have seen regression models in the past chapters already. We are now formalizing the model and further discussing the (most classical) estimation technique in this context: least squares estimation.

The multiple linear regression model with p predictors is given by

$$Y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i \quad (5.1)$$

$$= \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, n, \quad n > p, \quad (5.2)$$

where

- Y_i : dependent variable, modeling the observation, data y_i ,
- $\mathbf{x}_i = (1, x_{i1}, \dots, x_{ip})^\top$: free variables, predictors,
- $\boldsymbol{\beta} = (\beta_0, \dots, \beta_p)^\top$: parameter vector (unknown),
- ε_i : error (unknown), with symmetric distribution and $E(\varepsilon_i) = 0$.

We have a so-called linear model, as (5.2) is linear in the parameter of interest $\boldsymbol{\beta}$. In the basic framework, we assume homoscedastic and independent errors, that means $\text{Var}(\varepsilon_i) = \sigma^2$ and implies $\text{Cov}(\varepsilon_i, \varepsilon_j) = 0, i \neq j$. As we see, least squares estimation does not depend on distributional assumptions and thus formally, no further requirements on the error distribution needed. However, we will derive statistical tests which simplify the discussion if we additionally assume that the errors are Gaussian. Hence we do so.

To derive estimators, estimates, etc., it is convenient to write the n equations of (5.2) using matrix notation

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \varepsilon, \quad (5.3)$$

where \mathbf{X} is an $n \times (p+1)$ matrix with rows \mathbf{x}_i^\top . We assume that the (column) rank of \mathbf{X} is $p+1$ ($\text{rank}(\mathbf{X}) = p+1$).

The ordinary least squares (OLS) principle is based on minimizing the sum of the squared residuals based on the realization \mathbf{y} of (5.3):

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \quad (5.4)$$

$$\Rightarrow \frac{d}{d\boldsymbol{\beta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \frac{d}{d\boldsymbol{\beta}} (\mathbf{y}^\top \mathbf{y} - 2\boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{y} + \boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta}) \\ = -2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} \quad (5.5)$$

$$\Rightarrow \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\top \mathbf{y} \quad (5.6)$$

$$\Rightarrow \hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \quad (5.7)$$

Equation (5.6) is called the *normal equation*, at this very step, we need the rank assumption of \mathbf{X} , such that $(\mathbf{X}^\top \mathbf{X})^{-1}$ is well defined.

Formally, $\hat{\boldsymbol{\beta}}$ as defined in (5.7) is an estimate, i.e., a “classical” vector with $p+1$ scalars. To derive properties from $\hat{\boldsymbol{\beta}}$, we have to consider it as an *estimator* in the sense that we “replace” the actual data vector \mathbf{y} with the random vector $\mathbf{Y} \sim \mathcal{N}_n(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I})$, as defined in (5.3).

Starting from equation (5.2) and (5.7), the following estimates (left column) and distributions (right column) can be derived:

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \quad \hat{\beta} \sim \mathcal{N}_{p+1}(\beta, \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}), \quad (5.8)$$

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{H}\mathbf{y}, \quad \hat{\mathbf{Y}} \sim \mathcal{N}_n(\mathbf{X}\beta, \sigma^2 \mathbf{H}), \quad (5.9)$$

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = (\mathbf{I} - \mathbf{H})\mathbf{y}, \quad \mathbf{e} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 (\mathbf{I} - \mathbf{H})), \quad (5.10)$$



where we term the matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ as the hat matrix.



These results are profound. They state among others that the least squares estimate of β is unbiased and has a multivariate Gaussian distribution. With little additional effort, it would be possible to show the infamous *Gauss–Markov theorem*, that with uncorrelated errors (no Gaussian assumption needed), the OLS estimator (5.7) has a covariance matrix which is “smaller” than that of any other linear estimator. (Here, smaller means that $\text{Var}(\tilde{\beta}) - \text{Var}(\hat{\beta})$ is positive semi-definite matrix for every other linear unbiased estimator $\tilde{\beta} = \mathbf{B}\mathbf{Y}$). The estimator $\hat{\beta}$ is the *best linear unbiased estimator* (BLUE).

Note that the OLS approach does not yield an estimate for σ^2 . To derive test statistics, however, we need an estimate to address the unknown parameter in the distributions of (5.8) to (5.10). We choose the intuitive unbiased estimate

$$\hat{\sigma}^2 = \frac{1}{n-p-1} \mathbf{e}^\top \mathbf{e}, \quad (5.11)$$

The somewhat unusual denominator guarantees that we have an unbiased estimator. Hence, taking an individual element of (5.8) we have the following result

$$\frac{\hat{\beta}_i - \beta_i}{\sqrt{\hat{\sigma}^2 v_{ii}}} \sim t_{n-p-1} \quad \text{with } v_{ii} = ((\mathbf{X}^\top \mathbf{X})^{-1})_{ii}, \quad i = 0, \dots, p. \quad (5.12)$$

We start with an example that highlights the strong dependency of the before mentioned distributions on the covariates. To illustrate, we will sample the covariates in a first step and then continue to work conditionally on it.

Example 5.1. Let's assume a simple regression setting with

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = 1, \dots, n, \quad \varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2). \quad (5.13)$$

The classical least squares solution is

$$\begin{aligned} \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x} \\ \hat{\beta}_1 &= \frac{s_{xy}}{s_x^2} = r \frac{s_y}{s_x} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}. \end{aligned} \quad (5.14)$$

Simpler now, the joint distribution of the estimators of β_0 and β_1 is

$$\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} \sim \mathcal{N}_2 \left(\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}, \sigma^2 ((\mathbf{1}^\top \mathbf{x})^\top (\mathbf{1}^\top \mathbf{x}))^{-1} \right) = \mathcal{N}_2 \left(\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}, \sigma^2 \left(\frac{n}{\sum_i x_i} \frac{\sum_i x_i}{\sum_i x_i^2} \right)^{-1} \right) \quad (5.15)$$

For example, we sample the values of the predictor, say $X_i \stackrel{\text{iid}}{\sim} \mathcal{N}(\mu, \tau^2)$. Hence, on “average”, the variance matrix is

$$\sigma^2((\mathbf{1} \ \mathbf{x})^\top (\mathbf{1} \ \mathbf{x}))^{-1} \approx \sigma^2 \begin{pmatrix} n & \mathrm{E}(\sum_i X_i) \\ \mathrm{E}(\sum_i X_i) & \mathrm{E}(\sum_i X_i^2) \end{pmatrix}^{-1} = \sigma^2 \begin{pmatrix} n & n\mu \\ n\mu & n(\tau^2 + \mu^2) \end{pmatrix}^{-1} \quad (5.16)$$

$$= \frac{\sigma^2}{n} \begin{pmatrix} 1 & \mu \\ \mu & \tau^2 + \mu^2 \end{pmatrix}^{-1} = \begin{pmatrix} 0.10 & -0.03 \\ -0.03 & 0.01 \end{pmatrix}, \quad (5.17)$$

where we use the numerical values $\mu = 3$, $\tau^2 = 1$, $\sigma^2 = 1/4$ and $n = 25$ in the last step. When simulating the predictors, i.e., using $X_i \stackrel{\text{iid}}{\sim} \mathcal{N}(\mu, \tau^2)$, the variances of the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are **0.104** and **0.011**, respectively, which are reasonably close to 0.10 and 0.01 (see Problem 1).

The correlation between the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ is approximately **-0.953** (theoretical is 0.949). To avoid such high correlations, the predictors are often centered before the analysis. Thus, in the previous calculation $\mu = 0$ and thus the correlation is zero as well. Naturally, the slope estimates remains the same and the intercept absorbs the offset:

$$\beta_0 + \beta_1 x_i = \hat{\beta}_0^{\text{new}} + \hat{\beta}_1^{\text{new}}(x_i - \mu) \quad \Rightarrow \quad \hat{\beta}_1 = \hat{\beta}_1^{\text{new}} \quad \hat{\beta}_0 = \hat{\beta}_0^{\text{new}} - \hat{\beta}_1^{\text{new}}\mu. \quad (5.18)$$

In practice, μ is often unknown and we center with the empirical mean \bar{x} . Figure 5.1 compares the (theoretical) quantiles of both approaches. The gray shaded rectangle gives the confidence rectangle based on using the marginal confidence intervals individually (using (5.12)). ♣

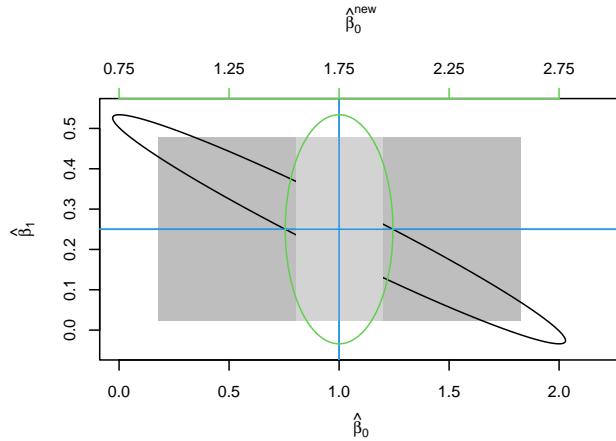


Figure 5.1: “Confidence regions” of $\hat{\beta}$ based on non-centered and centered data. The ellipses define the area for which the probability of $\hat{\beta}$ lying therein is 95%, based on the theoretical distribution (5.8). The rectangles are defined by the corresponding marginal distributions (gray for non-centered and light gray for centered data).

5.1.1 Model validation

As part of assessing the adequacy of the linear model, we should check if the error distribution is adequate and check if there is any evidence against the iid assumption. For the former we can, for example, compare the histogram with the fitted density, plot Q-Q-plots, probability plots etc. The later is a bit more delicate and involves testing if there is serial correlation, e.g.,

$\text{Corr}(Y_i, Y_{i+1}) \neq 0$, if the distribution holds for all i . While serial correlation can be assessed with `cor(y[1:(n-1)], y[2:n])` for a data vector y of length $n <- \text{length}(y)$, it is not possible to check individually if y_i a realization of $Y_i \sim \mathcal{L}$. It is, however, possible to apply a bulk check as outlined further below. We see more about serial correlation in Chapters 10 and 11.

5.1.2 Leverage and Influence

We quantify if an observation is unusual with respect to the predictors (large leverage) and if an observation has large influence on the fit, i.e., if omitting the observation, we would get a substantially different result.

Starting from

$$\hat{y} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top y = \mathbf{H}y, \quad (5.19)$$

$$(5.20)$$

(see equation (5.9)) we rewrite the i th component as

$$\hat{y}_i = h_{ii}y_i + \sum_{j=1, j \neq i}^n h_{ij}y_j. \quad (5.21)$$

Hence, h_{ii} measures how much the i th observation contributes to its prediction and is thus called *leverage*. In general, $1/n \leq h_{ii} \leq 1$; the closer a predictor is to the (hypothetical) center of the predictors, the smaller h_{ii} is. Large values indicate a high leverage, where “large” is typically seen as twice the average of all leverages:

$$h_{ii} > 2 \frac{1}{n} \sum_{i=1}^n h_{ii} = \frac{2(p+1)}{n}. \quad (5.22)$$

An outlying observation, say $\tilde{y}_i = y_i + \Delta$, will not have a large effect, i.e., influence, on the result unless h_{ii} is large. Before we formally quantify this influence the next section, let us have another look at the residuals.

From equation (5.10) we write $e_i = y_i - \hat{y}_i$. Under our model assumptions, we have

$$\text{Var}(e_i) = \sigma^2(1 - h_{ii}) \quad \text{and} \quad \text{Cov}(e_i, e_j) = -\sigma^2 h_{ij}, \quad i \neq j, \quad (5.23)$$

which is somewhat counter intuitive as we have started from $\varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. The reduction in the variance and the correlation are a result of the fitting, more precisely of the projection operation. While the correlation are rarely an issue, the raw residuals are often normalized:

$$\text{standardized residuals:} \quad r_i = \frac{e_i}{s}, \quad (5.24)$$

$$\text{studentized residuals:} \quad t_i = \frac{e_i}{s\sqrt{1 - h_{ii}}}, \quad (5.25)$$

where s^2 is the usual estimate of σ^2 .

R-Code 5.1 Leverages and influence of the `LifeCycleSavings` example. (See Figure 5.2.)

```

lcs <- lm(sr ~ ., data=LifeCycleSavings)
hii <- influence(lcs)$hat
c(2*mean(hii), 2*length(coef(lcs))/length(resid(lcs)))
## [1] 0.2 0.2
highleverage <- hii > 2*mean(hii)
print( hii[ highleverage])
##          Ireland          Japan United States          Libya
## 0.2122363 0.2233099 0.3336880 0.5314568
rbind(residuals=residuals(lcs), standard=rstandard(lcs),
      student=rstudent(lcs))[,1:6]
##          Australia        Austria       Belgium       Bolivia       Brazil       Canada
## residuals 0.8635798 0.6163860 2.2189579 -0.6983191 3.5528094 -0.31689236
## standard  0.2352011 0.1728294 0.6108576 -0.1924503 0.9685881 -0.09083873
## student   0.2327161 0.1709551 0.6065522 -0.1903783 0.9679082 -0.08983197
plot(lcs, which=5)

```

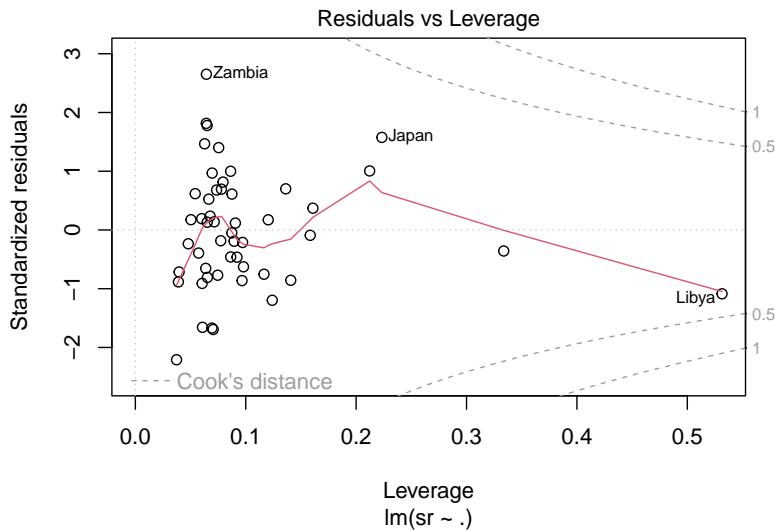


Figure 5.2: Residual vs leverage plot. (See R-Code 5.1.)

Example 5.2. We reconsider the `LifeCycleSavings` data again. R-Code 5.1 looks at the leverages of the model having all variables `lcs`, as seen in R-Code 1.2 of Example 1.4. Of course, the leverage depends on the predictor and change when using different predictors.

Figure 5.2 shows a *Residual vs Leverage* plot, where “isolines” have been plotted based on equation (1.14). ♣

Example 5.3. Back to the `LifeCycleSavings` data. Trimming a model can be done manually

using p -values from the summary output. Of course, there are alternative criteria available, such as the Bayesian information criterion (BIC). Different criteria may not necessarily lead to the same final model, as illustrated at the end of R-Code 5.2, where the variable `pop75` is eliminated.



R-Code 5.2: `LifeCycleSavings` data: full model, model simplification and model validation..

```

lcs.all <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data=LifeCycleSavings)
# summary(lcs.all)      # as shown in Example
lcs.aic <- step(lcs.all)          # AIC is default choice
## Start:  AIC=138.3
## sr ~ pop15 + pop75 + dpi + ddpi
##
##          Df Sum of Sq    RSS    AIC
## - dpi     1   1.893 652.61 136.45
## <none>           650.71 138.30
## - pop75   1   35.236 685.95 138.94
## - ddpi    1   63.054 713.77 140.93
## - pop15   1  147.012 797.72 146.49
##
## Step:  AIC=136.45
## sr ~ pop15 + pop75 + ddpi
##
##          Df Sum of Sq    RSS    AIC
## <none>           652.61 136.45
## - pop75   1   47.946 700.55 137.99
## - ddpi    1   73.562 726.17 139.79
## - pop15   1  145.789 798.40 144.53
summary(lcs.aic)$coefficients
##              Estimate Std. Error    t value    Pr(>|t|)
## (Intercept) 28.1246633  7.1837859  3.915020 0.0002969796
## pop15       -0.4517775  0.1409317 -3.205649 0.0024515384
## pop75       -1.8354083  0.9983996 -1.838350 0.0724726983
## ddpi        0.4278317  0.1878856  2.277087 0.0274781754
#
# AIC the smaller the better!
extractAIC(lcs.aic)[2]
## [1] 136.4475
plot(lcs.aic)                  # 4 plots to assess the model
#
# BIC as an alternative:
summary(step(lcs.all, k=log(50), trace=0))$coefficients

```

```

##             Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) 15.5995758 2.33439442 6.682494 2.479591e-08
## pop15       -0.2163762 0.06033473 -3.586263 7.959672e-04
## ddpi        0.4428302 0.19240135  2.301596 2.583739e-02
# => the variable "pop75" drops from the model

```

Remark 5.1. As another note, generalized cross-validation (GCV) is an approximation of (leave-one-out) CV. The idea is to (essentially) replace the sum of functions of the diagonal elements of the hat matrix by an expression in terms of the trace of the hat matrix. In the framework of linear models, for example, $\frac{1}{1-h_{ii}}$ is approximated by $\frac{1}{1-(p+1)/n}$. ♣

5.2 Analysis of Variance

There are many ways to assess model fit, i.e., assessing how well a model fits the actual data. On a general level, a model fit is a partition of variances. The variability in the data is separated into variability components of the model and remaining variability (error). Naturally, we would like the remaining error variance to be as small as possible — compared to the variance explained by the model.

$$\mathbf{y} = \mathbf{y} - \hat{\mathbf{y}} + \hat{\mathbf{y}} = (\mathbf{y} - \hat{\mathbf{y}}) + \hat{\mathbf{y}} = (\mathbf{I} - \mathbf{H})\mathbf{y} + \mathbf{H}\mathbf{y} \quad \Rightarrow \quad (5.26)$$

$$\mathbf{y}^\top \mathbf{y} = ((\mathbf{y} - \hat{\mathbf{y}}) + \hat{\mathbf{y}})^\top ((\mathbf{y} - \hat{\mathbf{y}}) + \hat{\mathbf{y}}) \quad (5.27)$$

$$= \mathbf{y}^\top (\mathbf{I} - \mathbf{H})(\mathbf{I} - \mathbf{H})^\top \mathbf{y} + 2\mathbf{y}^\top (\mathbf{I} - \mathbf{H})\mathbf{H}\mathbf{y} + \mathbf{y}^\top \mathbf{H}\mathbf{H}\mathbf{y} = \mathbf{y}^\top (\mathbf{I} - \mathbf{H})\mathbf{y} + \mathbf{y}^\top \mathbf{H}\mathbf{y} \quad (5.28)$$

by properties of the hat matrix \mathbf{H} . Each of the terms is a *sum of squares*:

$$\begin{aligned} \text{SS}_{\mathbf{y}^\top \mathbf{y}} &= \text{SS}_{\mathbf{y}^\top (\mathbf{I} - \mathbf{H})\mathbf{y}} + \text{SS}_{\mathbf{y}^\top \mathbf{H}\mathbf{y}} \\ \text{SS}^{\text{"total"}} &= \text{SS}^{\text{"error"}} + \text{SS}^{\text{"model"}}. \end{aligned} \quad (5.29)$$

We have used quotes in the last equation as a classical definition differs slightly. To be precise, i.e., to eliminate the quotes, we should start the left-hand side of (5.26) with $\mathbf{y} - \hat{\mu}\mathbf{1}$. In such a case, we would *decouple* the (overall) mean μ from β , or write $\hat{\mathbf{y}} - \hat{\mu}\mathbf{1}$ instead of only $\hat{\mathbf{y}}$. We discuss this decomposition in Problem 2. Results (5.28) and (5.29) can be summarized geometrically, as shown in Figure 5.3.

The sum of squares are linked to variance estimates, e.g., $\text{SS}_{\text{error}} = (n-p-1)\hat{\sigma}^2$. Thus, to properly compare the components, we have to further divide them by their *degrees of freedom*. For SS_{error} this is $n-p-1$, for SS_{model} this is p , because we have p predictors x_{1i}, \dots, x_{pi} . The divided quantities are denoted by MS (*mean squares*). The ratio of mean squares F_{obs} is (under $H_0 : \beta_1 = \dots = \beta_p = 0$) distributed according to an $F_{p,n-p-1}$ distribution. Hence, we reject $H_0 : \beta_1 = \dots = \beta_p = 0$ if $F_{\text{obs}} > F_{p,n-p-1}(1-\alpha)$, the $1-\alpha$ -quantile of the F -distribution with p and $n-p-1$ degrees of freedom. The aforementioned quantities are typically presented in an ANOVA table. Table 5.1 illustrates such a table (where we start with $\mathbf{y} - \mu\mathbf{1}$ in (5.26)). The F -value at the end of a `summary.lm()` call refers to exactly this value F_{obs} .

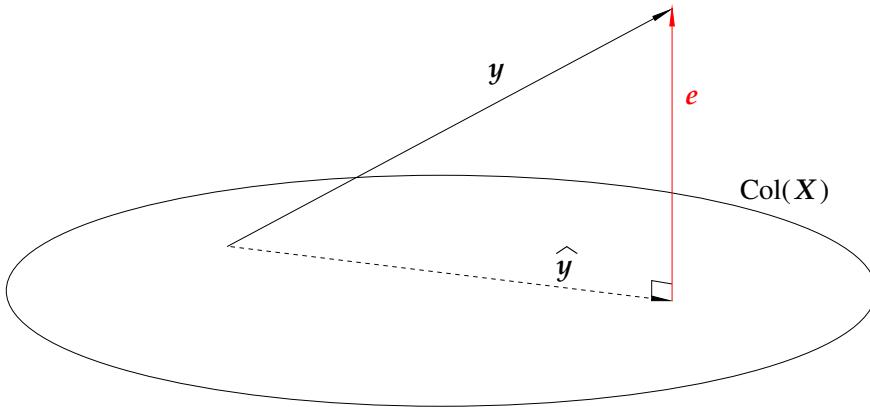


Figure 5.3: Regression considered geometrically. The data vector \mathbf{y} is projected onto the subspace spanned by the columns of X , $\text{Col}(\mathbf{X})$.

Table 5.1: ANOVA table. Note that for the first two columns the quantities add up to the total.

Source	Sums of squares	Degrees of freedom	Mean squares	F -value
Model	$\text{SS}_{\text{model}} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$	$\text{df}_{\text{model}} = p$	$\text{MS}_{\text{model}} = \frac{\text{SS}_{\text{model}}}{\text{df}_{\text{model}}}$	$F_{\text{obs}} = \frac{\text{MS}_{\text{model}}}{\text{MS}_{\text{error}}}$
Error	$\text{SS}_{\text{error}} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$	$\text{df}_{\text{error}} = n - p - 1$	$\text{MS}_{\text{error}} = \frac{\text{SS}_{\text{error}}}{\text{df}_{\text{error}}}$	
Total	$\text{SS}_{\text{total}} = \sum_{i=1}^n (y_i - \bar{y})^2$	$\text{df}_{\text{total}} = n - 1$		

After a model has been fit, we perform model validation/model selection to verify if all the included predictors are necessary or if some can be omitted. We often encounter the situation where we have two nested models, a “smaller” model m and a “larger” model M that contains all predictors of m and some more, formally $m \subset M$. The situation is geometrically illustrated in Figure 5.4, where the residual vector \mathbf{e} is further decomposed. This decomposition is reflected by replacing the corresponding line with two sources in the ANOVA, see Table 5.2.

We reject H_0 : “the smaller model is sufficient” for large values of F_Δ . Formally, we reject the null hypothesis if $F_\Delta > F_{p-q, n-p-1}(1-\alpha)$, the $1-\alpha$ -quantile of the F -distribution with $p-q$ and $n-p-1$ degrees of freedom.

Recall that a squared t_ℓ distribution is identical to an $F_{1,\ell}$ distribution. Therefore, even the classical t -test $H_0: \beta_i = 0$, with the test statistic given by equation (5.12), can be cast into this ANOVA framework: we test if the increase of the error induced by removing a predictor is significant compared to the error (of the reduced model). More specifically, the t -test uses the t_{n-p-1} distribution, which corresponds to $p-q=1$.

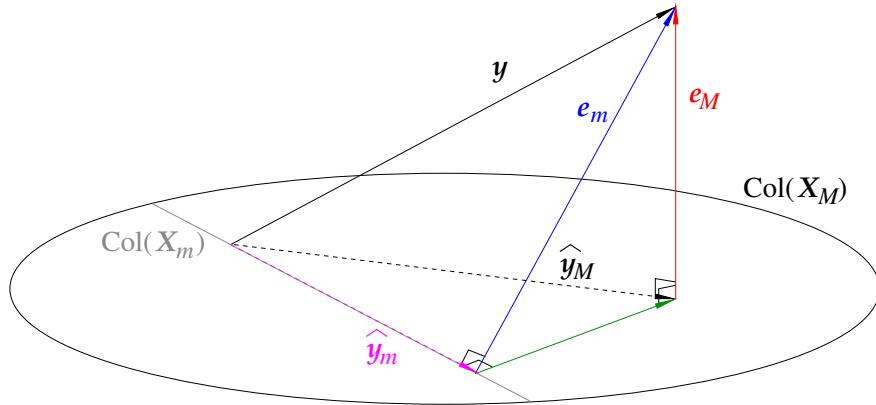


Figure 5.4: Regression considered geometrically. Subscript M refers to a “larger” model and subscript m to a “smaller” model (contained in model M).

Table 5.2: ANOVA table to compare two models M and m with p and q ($p > q$) parameters, respectively (a mean is contained in both and not counted).

Source	Sums of squares	Degr. of free.	Mean squares	F-value
Model m	$SS_{\text{model},m}$	$df_{\text{model},m} = q$	$MS_{\text{model},m} = \frac{SS_{\text{model},m}}{df_{\text{model},m}}$	
Terms from model M only	$SS_{\Delta} = SS_{\text{error},m} - SS_{\text{error},M}$	$df_{\Delta} = p - q$	$MS_{\Delta} = \frac{SS_{\text{error},m} - SS_{\text{error},M}}{p - q}$	$F_{\Delta} = \frac{MS_{\Delta}}{MS_{\text{error},M}}$
Error	$SS_{\text{error},M}$	$df_{\text{error},M} = n - p - 1$	$MS_{\text{error},M} = \frac{SS_{\text{error},M}}{df_{\text{error},M}}$	
Total	SS_{total}	$df_{\text{total}} = n - 1$		

5.3 Extensions of the Linear Model

We discuss a few extensions of the classical linear regression model formulation. The main message is that many models can be cast into a linear model framework.

The case of non-Gaussian observations is discussed in Chapter 8. If the error is just “somewhat” non-Gaussian (similar form as a Gaussian in that we have symmetry and the tails are not too heavy) it is often permissible to proceed by assuming Gaussianity. The t - and F -tests for individual $\hat{\beta}_i$ or sums of squares decompositions are not too sensitive to non-Gaussian errors.

5.3.1 One-Way ANOVA

To investigate if there are at least two mean values $\bar{y}_1, \bar{y}_2, \dots, \bar{y}_I$ which show a significant difference we use classical one-way ANOVA.

More formally, we assume that the I populations are normally distributed with homogeneous variances and that the samples are independent.

Instead of performing $\binom{I}{2}$ two-sample t -tests, we write the problem as a linear regression model with one categorical influential variable.

We have I groups ($i = 1, \dots, I$) and n_i samples per group ($j = 1, \dots, n_i$). This leads to the model

$$Y_{ij} = \mu + \beta_i + \epsilon_{ij} \quad \text{with} \quad \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2), \quad (5.30)$$

where we have the constraint that

$$\sum_{i=1}^I \beta_i = 0. \quad (5.31)$$

Using this model we can test different hypotheses. The most popular one is

$$\beta_1 = \beta_2 = \dots = \beta_I = 0. \quad (5.32)$$

The constraint (5.31) is called the *sum-to-zero* constraint. The default constraint in R is the so-called *treatment* constraint $\beta_1 = 0$. Therefore it is always important to quickly cross-check the output of the `lm()` call.

Example 5.4. (Rehabilitation time after surgery) The original dataset was retrieved from www-stat.stanford.edu/~nzhang/191_web/Data/Rehab.txt and is available at user.math.uzh.ch/furrer/download/sta121/rehab.csv. We want to know whether rehabilitation time after a surgery depends on the fitness level of the respective person. The dataset contains 24 subjects grouped into one of three different fitness levels: below average (8), average (10) and above average (6). Box plots of the three groups are given in Figure 5.5. The analysis is given in R-Code 5.3.

The results from both the ANOVA and the linear model show that there is a significant effect of the fitness level. The estimated group means are 24 days (fitness above average), 32 days (average) and 38 days (below average).

The model has been fitted using a response in the form of a `factor`. Note how the estimated coefficients can be reconstructed. Many other approaches are possible. ♣

R-Code 5.3: Box plots and one-way ANOVA with `rehab` dataset. (See Figure 5.5.)

```
rehab <- read.csv("download/sta121/rehab.csv")
boxplot(Time ~ Fitness, data=rehab, xlab="Level",
         ylab="Rehabilitation time (days)")
options("contrasts"=c("contr.sum", "contr.sum")) # sum-to-zero
model <- lm(Time ~ Fitness, data=rehab)
summary(model)

##
## Call:
## lm(formula = Time ~ Fitness, data = rehab)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.0000 -2.5000  0.0000  2.5000 10.0000
```

```

##   -9.0   -3.0   -0.5    3.0    8.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 31.3333     0.9285 33.747 < 2e-16 ***
## Fitness1    -7.3333     1.4009 -5.235 3.45e-05 ***
## Fitness2     0.6667     1.2339  0.540    0.595
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.451 on 21 degrees of freedom
## Multiple R-squared:  0.6176, Adjusted R-squared:  0.5812
## F-statistic: 16.96 on 2 and 21 DF,  p-value: 4.129e-05
# Estimated group means
# ABOVE: 31.3333 - 7.3333=24
# AVERAGE: 31.3333 + 0.6667=32
# BELOW: 31.3333 + 7.3333 - 0.6667=38
anova(model)
## Analysis of Variance Table
##
## Response: Time
##             Df Sum Sq Mean Sq F value    Pr(>F)
## Fitness      2    672   336.00 16.962 4.129e-05 ***
## Residuals 21    416    19.81
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

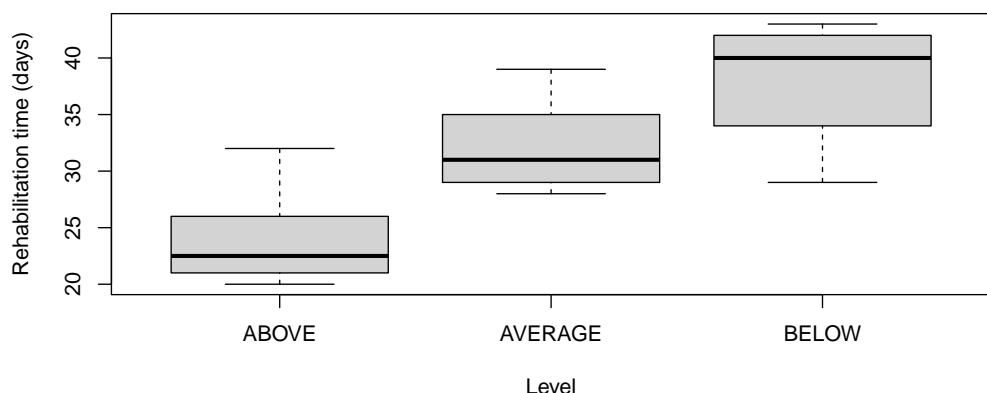


Figure 5.5: Box plots of rehabilitation time for three levels of fitness. (See R-Code 5.3.)

The one-way ANOVA is another simple case in which we can write down the sum of squares decomposition. To simplify the presentation, we use dot notation in order to show that we work with averages, for example,

$$\bar{y}_{i\cdot} = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij} \quad \text{and} \quad \bar{y}_{..} = \frac{1}{N} \sum_{i=1}^I \sum_{j=1}^{n_i} y_{ij}. \quad (5.33)$$

Remember the decomposition of the total sum of squares:

$$\sum_{ij} (y_{ij} - \bar{y}_{..})^2 = \sum_{ij} (y_{ij} - \bar{y}_{i\cdot} + \bar{y}_{i\cdot} - \bar{y}_{..})^2 \quad (5.34)$$



$$= \sum_{ij} (y_{ij} - \bar{y}_{i\cdot})^2 + \sum_{ij} (\bar{y}_{i\cdot} - \bar{y}_{..})^2 + \sum_{ij} 2(y_{ij} - \bar{y}_{i\cdot})(\bar{y}_{i\cdot} - \bar{y}_{..}) \quad (5.35)$$

$$= \underbrace{\sum_{ij} (y_{ij} - \bar{y}_{i\cdot})^2}_{\text{Total}} + \underbrace{\sum_{ij} (\bar{y}_{i\cdot} - \bar{y}_{..})^2}_{\text{Error}} + \underbrace{2 \sum_i \{(\bar{y}_{i\cdot} - \bar{y}_{..}) \sum_j (y_{ij} - \bar{y}_{i\cdot})\}}_0. \quad (5.36)$$

Therefore we can see that

$$\underbrace{\sum_{ij} (y_{ij} - \bar{y}_{..})^2}_{\text{Total}} = \underbrace{\sum_{ij} (y_{ij} - \bar{y}_{i\cdot})^2}_{\text{Error}} + \underbrace{\sum_{ij} (\bar{y}_{i\cdot} - \bar{y}_{..})^2}_{\text{Treatment}}. \quad (5.37)$$

5.3.2 Two-Way and Multi-Way ANOVA

The one-way ANOVA can be extended to several factors. In the case of two factors with, say, I and J levels we have

$$Y_{ijk} = \mu + \beta_i + \gamma_j + \epsilon_{ijk} \quad \text{with} \quad \epsilon_{ijk} \sim \mathcal{N}(0, \sigma^2), \quad (5.38)$$

with n_{ij} samples for each of the corresponding $I \times J$ treatments. For identifiability reasons, as in (5.31), we impose constraints of the form

$$\sum_{i=1}^I \beta_i = 0, \quad \sum_{j=1}^J \gamma_j = 0, . \quad (5.39)$$

Using this model we can test different hypotheses. The popular hypotheses tests are

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_I = 0, \quad (5.40)$$

$$H_0 : \gamma_1 = \gamma_2 = \cdots = \gamma_J = 0. \quad (5.41)$$

The resulting p -values can be directly extracted from an `anova()` output.

For more factors, we “essentially” have to add indices and Greek letters to (5.38).

The decomposition in different sums of squares is straightforward if we have balanced designs, i.e., if $n_{ij} = n$ for all i and j .

If the factors do not exhibit linear effects, interaction terms need to be added. In the two-way setting, a call to `interaction.plot()` may illustrate possible interactions.

5.3.3 Weighted and Generalized Least Squares

In this section we discuss the case in which the errors are not independent or identically distributed.

If the variances vary for the observations, we apply a weighted least squares (WLS) approach. If the variances are not independent, we use a generalized least-squares (GLS) approach. Both approaches work fine if the underlying parameters are known.

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{V}), \quad (5.42)$$

where σ^2 is unknown and \mathbf{V} is an arbitrary (symmetric and positive definite) matrix. Let \mathbf{W} denote a “square root” of \mathbf{V}^{-1} such that $\mathbf{W}^\top \mathbf{W} = \mathbf{V}^{-1}$. The trick is to pre-multiply equation (5.42) with \mathbf{W} such that the transformed errors $\mathbf{W}\boldsymbol{\varepsilon}$ are iid:

$$\mathbf{W}\mathbf{Y} = \mathbf{W}\mathbf{X}\boldsymbol{\beta} + \mathbf{W}\boldsymbol{\varepsilon}, \quad \mathbf{W}\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}). \quad (5.43)$$

In the transformed setting we proceed as in the standard case and thus

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{V}^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{V}^{-1} \mathbf{y}. \quad (5.44)$$

There are several issues with this approach:

1. \mathbf{V} is unknown: iterative approaches or maximum likelihood may be used.
2. If n is large and \mathbf{V} is not a diagonal matrix, the computational cost to calculate (5.44) may be prohibitive.

5.3.4 Multivariate Regression

It is very important to distinguish between multiple and multivariate regression. The former has several predictors and the latter has several variables that are observed, i.e., instead of a vector \mathbf{y} , we observe a data matrix. More precisely, instead of the model $Y_i \sim \mathcal{N}(\mathbf{x}_i^\top \boldsymbol{\beta}, \sigma^2)$, as written at the beginning of this chapter, we have $\mathbf{Y}_i \sim \mathcal{N}_r(\mathbf{x}_i^\top \mathbf{B}, \boldsymbol{\Sigma})$, as discussed in the last chapter. Thus, we need to estimate an entire parameter matrix $\mathbf{B} \subset \mathbb{R}^{p+1 \times r}$.

 6 min There are several ways to solve the multivariate regression problem. One way is to stack the matrices into vectors and to apply a generalized least squares approach, the tool seen in the last subsection.

5.4 Bibliographic remarks

Fahrmeir *et al.* (2013) is a very good, accessible book with both theoretical text blocks and extensive case studies, as well. On a similar level, the same author provides the following pdf: cran.r-project.org/doc/contrib/Faraway-PRA.pdf.

Mardia *et al.* (1979) cover all aspects of this chapter.

Approaching the (statistical) problem from a different angle, Christensen (2011) describes many statistical methods in terms of projections. However, it is somewhat more technical and requires a rigorous linear algebra background.

5.5 Exercises and Problems

Problem 5.1 (Correlation of estimates) Let's assume a simple regression setting with

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = 1, \dots, n, \quad \varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2).$$

As values for the predictor, we take a realization of $X_i \stackrel{\text{iid}}{\sim} \mathcal{N}(\mu, \tau^2)$. We use the numerical values $\mu = 3$, $\beta_0 = 1$, $\beta_1 = 1/4$, $\sigma^2 = 1/4$, $n = 25$ and $\tau^2 = 1$.

- a) Calculate $\text{Corr}(\hat{\beta}_0, \hat{\beta}_1)$.
- b) Simulate $R = 500$ times the regression setting for the non-centered and the centered setting. Plot the $\hat{\beta}_0$ and $\hat{\beta}_1$. Give an interpretation. What changes if the centering is based on the sample average?
- c) If we denote $\hat{\beta}_0^{(c)}$ and $\hat{\beta}_1^{(c)}$, where the index c specifies the centering choice (i.e., non, μ , \bar{x}). What are the empirical values of $\text{Corr}(\hat{\beta}_k^{(c)}, \hat{\beta}_k^{(c)})$, $k = 0, 1$? Give an interpretation.

Problem 5.2 (Centered predictors and sums of squares derivation) In this problem, we consider centered predictors and decouple the intercept β_0 from the remaining parameters. More specifically, we consider the model

$$Y_i = \mu + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i = \mu + \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, n, \quad n > p,$$

and $\sum_{i=1}^n x_{ij} = 0$ for $j = 1, \dots, p$ and with all other assumptions as in our standard setting. In matrix notation we have $\mathbf{Y} = \mathbf{1}\mu + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$.

- a) Show that the OLS estimates of μ and $\boldsymbol{\beta}$ are given by $\hat{\mu} = \bar{y} = 1/n \mathbf{1}^\top \mathbf{y}$ and $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$.
- b) Derive the distribution of $\mathbf{Y} - \hat{\mu}\mathbf{1}$ (where $\hat{\mu}\mathbf{1} = 1/n \mathbf{1}^\top \mathbf{Y}$) and show that the distribution of $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$ is equivalent to $(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{Y} - \hat{\mu}\mathbf{1})$. What is a convenient consequence of these results?

Hint: It is worthwhile to define $\mathbf{J} = 1/n \mathbf{1} \mathbf{1}^\top$ and to show that \mathbf{J} and thus $\mathbf{I} - \mathbf{J}$ are idempotent and that $\mathbf{J}(\mathbf{I} - \mathbf{J}) = \mathbf{0}$ and $\mathbf{J}\mathbf{H} = \mathbf{0}$, the null matrix.

- c) Derive the sums of squares decomposition (5.29) when starting from $\mathbf{y} - \bar{y}\mathbf{1}$ in (5.26).

Problem 5.3 (Multiple linear regression – teacher salaries) The file `salary.txt` is available on the course web page. It contains information about average teacher salaries for 325 school districts in Iowa for the 2000–2001 school year. The variables are

<code>District</code>	name of the district
<code>districtSize</code>	size of the district: 1: <1000 students, 2: 1000–2000 students, 3: >2000 students
<code>salary</code>	average teacher salary (in dollars)
<code>experience</code>	average teacher experience (in years)

- a) Read in the file `salary.txt`. Create a factor variable for the size of the district and add it as additional variable (with columnname `size`) to the data set. *Hint:* `read.table()`, `within()`, `factor()`
- b) Make numerical and graphical summaries of the data. What is the average salary in CHF?
Hint: `summary()`, `by()`, `aggregate()`, `plot()`, `pairs()`, `boxplot()`, `hist()`, ...
- c) Fit the following two models to the data
- A: $\text{salary} \sim 1 + \text{experience} + \text{districtSize}$
B: $\text{salary} \sim 1 + \text{experience} + \text{size}$
- Compare the models. How does the interpretation of the parameters change if we use a factor instead of an integer for a categorical variable?
- d) Consider model B and discuss the fit of the model. Is there an effect of experience and/or district size? Could we drop one variable from the model? How can we interpret the parameter estimates?
- e) How does model B change if we use “experience – 11” instead of experience in the model?
Hint: `I()`
- f) Look at the residuals and discuss the adequacy of the model.
- g) What is the predicted value of a salary according to the model A and B for a teacher with 12 years experience in the district category “large”?

Problem 5.4 (Multiple linear regression – artificial data) When fitting the model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

to a set of $n = 15$ cases, we obtained least squares estimates $\hat{\beta}_0 = 10$, $\hat{\beta}_1 = 12$, $\hat{\beta}_2 = 15$, and $s^2 = 2$. It is also known that

$$(\mathbf{X}^\top \mathbf{X})^{-1} = \begin{pmatrix} 1 & 0.25 & 0.25 \\ 0.25 & 0.5 & -0.25 \\ 0.25 & -0.25 & 2 \end{pmatrix}$$

- a) What is the standard error $\text{SE}(\hat{\beta}_2)$?
Hint: Do NOT calculate \mathbf{X} .
- b) Test the hypothesis that $\beta_2 = 0$.
- c) Estimate the covariance between $\hat{\beta}_1$ and $\hat{\beta}_2$. What is the standard error of $\hat{\beta}_1 - \hat{\beta}_2$?
- d) Test the hypothesis that $\beta_1 = \beta_2$.
- e) The total sum of squares is $\text{SST}=120$. Construct the ANOVA table and test the hypothesis that $\beta_1 = \beta_2 = 0$. Obtain the percentage of variation in y that is explained by the model.

Chapter 6

Mixed Models

If we have several measurements from clusters or if we have several measurements from the same individuals over time the linear model assumptions may be violated. Mixed models extend the linear model framework. They can integrate both fixed effects and random effects.

Learning goals for this chapter:

- ◊ Extend the treatment design to include random effects
- ◊ Explain the difference between random effect models and ANOVA
- ◊ Explain random and fixed effects
- ◊ Use of random effects models
- ◊ Combine fixed and random effects in the mixed model – theory and R

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter06.R.

6.1 Motivation

We now return to extensions of the linear model. To motivate random effects and mixed effects models we start with simple examples illustrating the need to extend the classical linear model.

(1a) In a factory, three different production machines are used. The total production time is recorded. The model,

$$Y_{ij} = \mu + \beta_i + \varepsilon_{ij} \quad \text{with} \quad \varepsilon_{ij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2) \quad (6.1)$$

and some constraint on the β_i s (e.g., sum to zero constraint), is sufficient to investigate if all machines have the same (similar) production time.

(1b) In a factory, three different production machines are each run by one specific but arbitrarily (randomly) selected operators. The total production time is recorded. The model,

$$Y_{ij} = \mu + \beta_i + \alpha_j + \varepsilon_{ij} \quad \text{with} \quad \varepsilon_{ij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2). \quad (6.2)$$

and some constraints on the β_j s (machines) and the α_j s (operators), are sufficient to make statements about the machines but not about the population of operators since there is no element in the model going beyond the selection of operators.

(2) A so-called hierarchical model: A body condition score is an assessment of the proportion of fat that a cow possesses. It is widely accepted that the score is an important factor in (dairy) cattle management (Roche *et al.*, 2009). The cows within one herd are likely to be more similar than cows from different farmers. Cows from nearby farms are likely to be more similar than cows from distant region. Naturally, one should take into account all possible covariates to control for all these effects.

The model that would account for this structure, would be the following:

$$Y_{ijk} = \mathbf{x}_{ijk}^\top \boldsymbol{\beta} + \alpha_i + \gamma_{i(j)} + \varepsilon_{ijk}, \quad (6.3)$$

where α_i represents the effect of the i th farm, $\gamma_{i(j)}$ the effect of the j th herd j *within* the farm i and k for the cow within herd j *within* farm i . The parentheses in the subscript highlight the fact that there is a *nesting* effect due to the sampling of subpopulations. In practice, it is however not always possible to explicitly control these using covariates (here “fixed” effects) because they may not be known or “modelable”.

The last two examples illustrate that we need to extend the classical linear model (here ANOVA framework) because the basic model assumptions are not satisfied (i.e., independent observations, deterministic, unique predictors). The core idea is that the factors α_i and/or β_j are not fixed (in terms of predictors) but random. These models are called random effect models or, more generally, mixed (effects) models. They are used in situations, where we have

- Repeated measures: we have several observations from the same individual (all other parameters kept fixed),
- Hierarchical models: observations belong to subpopulations (animals in stables, stables in farms, farms in regions . . .),
- Longitudinal settings: observations follow the same individual or subject over time (other parameters may change over time as well).

6.2 Mixed Effects Models

The main difference between a model with one random effect and a fixed effects model (6.1) (i.e., a “normal” one-way ANOVA) is the fact that the levels to be compared are now random. This means that the general interest lies on the variable itself, not on the comparison of specific values of this variable (or factor). We are much more interested in modeling variances or variance

components instead of individual levels. For example, (under an appropriate setting), what is the within-subject variability in the response relative to between-subject variability?

The model is used to deal with two main problems that occur if we have repeated measures:

1. Observations of the same (random) individual or level i are more correlated than observations of different individuals.
2. Parameters of each individual or level i might be slightly different.

A random effect model consists of a model with random effects only. A mixed effects model consists of random and fixed effects.

6.2.1 Random Effect Model

Random effect models allow one to account for correlations or a correlation structure in the data. The correlation is implicitly captured in the model. This can be seen if we have a look at the expected value, variance and covariances of the observations. We illustrate this using the simplest random effects model of the form

$$Y_{ij} = \mu + \alpha_i + \varepsilon_{ij} \quad \text{with} \quad \varepsilon_{ij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \quad (6.4)$$

where α_i is now random, or $\alpha_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\alpha^2)$, and α_i and ε_{ij} are independent.

The expected value simply is

$$\mathbb{E}(Y_{ij}) = \mathbb{E}(\mu + \alpha_i + \varepsilon_{ij}) = \mathbb{E}(\mu) + \mathbb{E}(\alpha_i) + \mathbb{E}(\varepsilon_{ij}) = \mu. \quad (6.5)$$

The variance has the form

$$\text{Var}(Y_{ij}) = \text{Var}(\mu + \alpha_i + \varepsilon_{ij}) = \text{Var}(\alpha_i) + \text{Var}(\varepsilon_{ij}) = \sigma_\alpha^2 + \sigma^2. \quad (6.6)$$

The covariance of two observations of the same individual i is obtained via

$$\text{Cov}(Y_{ij}, Y_{ik}) = \text{Cov}(\mu + \alpha_i + \varepsilon_{ij}, \mu + \alpha_i + \varepsilon_{ik}) \quad (6.7)$$

$$= \text{Cov}(\alpha_i, \alpha_i) + \text{Cov}(\alpha_i, \varepsilon_{ij}) + \text{Cov}(\alpha_i, \varepsilon_{ik}) + \text{Cov}(\varepsilon_{ij}, \varepsilon_{ik}) = \text{Var}(\alpha_i) = \sigma_\alpha^2. \quad (6.8)$$

Analogously, we can see that the covariance of observations from two different individuals is zero. Hence, we have

$$\text{Corr}(Y_{ij}, Y_{ik}) = \frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma^2} \quad \text{and} \quad \text{Corr}(Y_{ij}, Y_{lk}) = 0, \quad i \neq l. \quad (6.9)$$

6.2.2 Mixed Models

It is possible to include both fixed and random effects in one model. The general definition of a linear mixed model is

$$Y_{ij} = \underbrace{\mathbf{X}_{ij}\boldsymbol{\beta}}_{\text{fixed effects}} + \underbrace{\mathbf{Z}_{ij}\boldsymbol{\alpha}_i}_{\text{random effects}} + \varepsilon_{ij}, \quad (6.10)$$

where $\alpha_i \stackrel{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \Sigma)$ and where \mathbf{X}_{ij} and \mathbf{Z}_{ij} are known matrices of appropriate size (here row vectors). Thus, we are also capable of discussing the between-subject variability in covariate effects relative to the mean population effect.

Naturally, by (appropriately) stacking the observations and errors into vectors \mathbf{Y} and $\boldsymbol{\varepsilon}$ and placing \mathbf{X}_{ij} and \mathbf{Z}_{ij} in matrices \mathbf{X} and \mathbf{Z} , we can write (6.10) as $\mathbf{Y} = \mathbf{X}\beta + \mathbf{Z}\alpha + \boldsymbol{\varepsilon}$, with appropriate distributional assumptions.

Estimation in a mixed model is not straightforward because unlike in the classical linear model framework, no closed-form solutions exist for the estimates. There are three main approaches to estimation of model parameters:

1. Maximum likelihood (ML) estimation,
2. Restricted maximum likelihood (REML) estimation,
3. Markov chain Monte Carlo (MCMC) approaches.

ML estimation is easy to explain and beneficial in comparing different (nested) models but yields less stable variance component estimates than REML. The last approach is computationally demanding and is not often used in practice. However, even ML estimation is quite elaborate and based on profile likelihood steps.

To fit the models we will use the function `lmer()` available in the package `lme4`. The specification of the model is quite similar to a fixed effects model, but with the addition of a `(. | .)` model term for the random effects.

With respect to (6.10), we can visualize the formula component as `(Z | i)`, where `i` specifies the variable for which a random component is considered. In the previous example, we could consider `i` to be the different operators of the machinery. Table 6.1 gives more practical details.

Table 6.1: Formula syntax for function `lmer()` in package `lme4`.

Formula	Model
<code>(1 group)</code>	random intercept within <code>group</code>
<code>(x group)</code>	random slope for <code>x</code> and intercept within <code>group</code> , with correlation between intercept and slope
<code>(0 + x group)</code>	random slope for <code>x</code> within <code>group</code> , no variation in intercept
<code>(1 group1) + (1 group2)</code>	random intercepts for two (crossed or nested) grouping factors
<code>(x group)</code> or <code>(1 group) + (0 + x group)</code>	uncorrelated random intercept and random slope for <code>x</code> within <code>group</code>

6.3 Linear Mixed Models for Longitudinal Data

Linear mixed models are often applied in the special situation where we have repeated measurements of several individuals over time. The two random effects that are most common in

this situation are a random intercept for each individual and a random slope for time for each individual.

For example, when we model the response by Y_{ij} for i individuals with j repeated measures (over time) each, we have several possible models:

- Global model where all observations are pooled:

$$Y_{ij} = \beta_0 + \beta_1 x_{1,ij} + \varepsilon_{ij}, \quad \varepsilon_{ij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2). \quad (6.11)$$

This model ignores within-subject correlation. As a consequence, the variability of coefficients is underestimated because correlated data contain less information than independent data.

- Subject-specific models, where we fit different models for each individual (no pooling at all):

$$Y_{ij} = \beta_{0,i} + \beta_{1,i} x_{1,ij} + \varepsilon_{ij}, \quad \varepsilon_{ij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma_i^2). \quad (6.12)$$

The fitted model can only be interpreted with regard to individuals in the sample. No statements with respect to a population are possible. The number of parameters to be estimated can grow quite large.

The mixed linear model is somewhere in between these two approaches and represents partial pooling (over the variances).

$$Y_{ij} = \bar{\beta}_0 + (\beta_{0,i} - \bar{\beta}_0) + \bar{\beta}_1 x_{1,ij} + (\beta_{1,i} - \bar{\beta}_1) x_{1,ij} + \varepsilon_{ij}, \quad (6.13)$$

which we rewrite as (using $\beta_0 \leftarrow \bar{\beta}_0$, $\alpha_{0,i} \leftarrow (\beta_{0,i} - \bar{\beta}_0)$, \dots)

$$Y_{ij} = \beta_0 + \beta_1 x_{1,ij} + \alpha_{0,i} + \alpha_{1,i} x_{1,ij} + \varepsilon_{ij}, \quad (6.14)$$

where $\varepsilon_{ij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$ are independent of $\boldsymbol{\alpha}_i = \begin{pmatrix} \alpha_{0,i} \\ \alpha_{1,i} \end{pmatrix} \stackrel{\text{iid}}{\sim} \mathcal{N}_2(\mathbf{0}, \boldsymbol{\Sigma})$. In terms of (6.10), $\mathbf{X}_{ij} = \mathbf{Z}_{ij} = (1 \ x_{1,ij})$.

Example 6.1. (Pigs' weight) We track the weight (in kg) of 48 pigs over time for nine consecutive weeks (Diggle *et al.*, 2002). To emphasize that we track individual pigs, we plot individual lines for each pig.

If we have a look at the plot in Figure 6.1, we can see that all pigs gain considerable weight, but they start at different levels. Animals with larger weights at the beginning seem to have larger weights at the end as well. The variability seems larger towards the end (probably due to variation in the growth rate of the individual animals).

We might be interested in the growth (growth rate) but also in how the individual animals behave with respect to this rate. To start, we would fit a simple linear model (`m1` in R-Code 6.1). The linear model fits the “union” of all data points well, but does not capture the fact that individual pigs deviate almost “linearly” from the fitted line.

Therefore, it is advisable to fit a linear mixed model and include a random intercept in the model (`m2` in R-Code 6.2). This means that for each pig, we would fit an individual line modeling the weight. The regression line `m1` represents the population average.

R-Code 6.1 Pigs' weight over time. (See Figure 6.1.)

```

pigs <- read.csv("download/sta121/pigs.csv",
                  col.names=paste0("weight.", 1:9), header=F)
dim(pigs)
## [1] 48   9

matplotlib(pigs, type="l", ylab="Weight [kg]", xlab="Week", lty=1,
           col="gray")

# Reshape the data "wide" -> "long"
pigslong <- reshape(pigs, direction="long", varying=1:9)
str(pigslong, give.attr=FALSE)

## 'data.frame': 432 obs. of  3 variables:
## $ time  : num  1 1 1 1 1 1 1 1 1 ...
## $ weight: num  24 22.5 22.5 24 24.5 23 22.5 23.5 20 25.5 ...
## $ id    : int  1 2 3 4 5 6 7 8 9 10 ...
m1 <- lm(weight ~ time, data=pigslong)
abline(m1, lwd=2, col=1)
lines(1:9, colMeans(pigs), lwd=2, col=2)

```

Looking at the individual slopes, we can see that they are more variable at later time points than at earlier time points. For this reason we could also include a random slope in the model ([m3](#) in R-Code 6.2). ♣

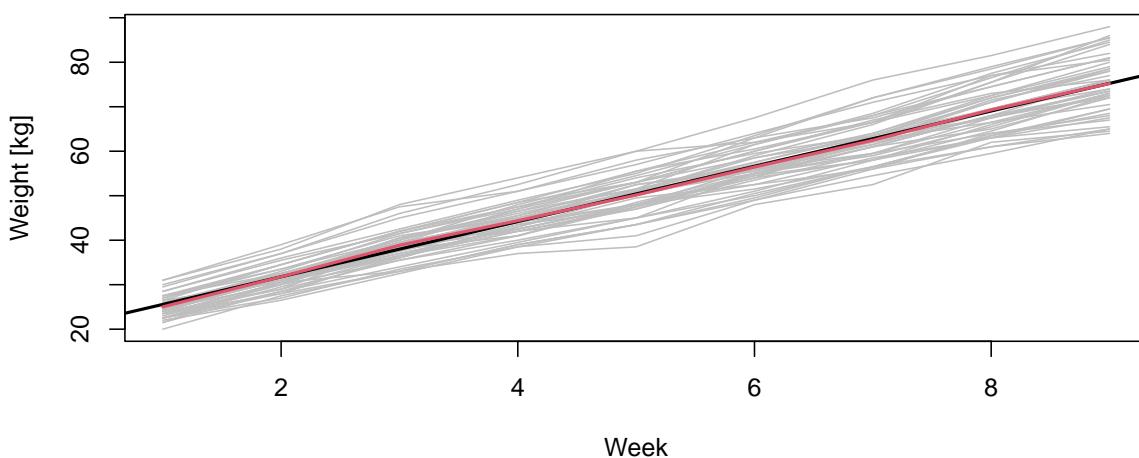


Figure 6.1: Weight of 48 pigs over time. Overall population average is drawn in red and the simple regression fit in black. (See R-Code 6.1.)

R-Code 6.2: Linear mixed models for the `pigs` dataset: model with random intercept (m2) and with random intercept and slope (m3).

```

library(lme4)
m2 <- lmer(weight ~ time + (1 | id), data=pigslong)
summary(m2)

## Linear mixed model fit by REML ['lmerMod']
## Formula: weight ~ time + (1 | id)
##   Data: pigslong
##
## REML criterion at convergence: 2033.8
##
## Scaled residuals:
##       Min      1Q  Median      3Q     Max
## -3.7390 -0.5456  0.0184  0.5122  3.9313
##
## Random effects:
##   Groups   Name        Variance Std.Dev.
##   id       (Intercept) 15.142    3.891
##   Residual             4.395    2.096
##   Number of obs: 432, groups: id, 48
##
## Fixed effects:
##           Estimate Std. Error t value
## (Intercept) 19.35561   0.60314 32.09
## time        6.20990   0.03906 158.97
##
## Correlation of Fixed Effects:
##   (Intr)
## time -0.324

# Now fitting a random intercept and random slope
m3 <- lmer(weight ~ time + (time | id), data=pigslong)
summary(m3)

## Linear mixed model fit by REML ['lmerMod']
## Formula: weight ~ time + (time | id)
##   Data: pigslong
##
## REML criterion at convergence: 1740.9
##
## Scaled residuals:
##       Min      1Q  Median      3Q     Max
## -3.6202 -0.5474  0.0150  0.5486  2.9939

```

```

## 
## Random effects:
##   Groups   Name        Variance Std.Dev. Corr
##   id       (Intercept) 6.986    2.6432
##           time        0.380    0.6164  -0.06
##   Residual             1.597    1.2637
## Number of obs: 432, groups: id, 48
##
## Fixed effects:
##   Estimate Std. Error t value
## (Intercept) 19.35561   0.40387 47.93
## time        6.20990   0.09204 67.47
##
## Correlation of Fixed Effects:
##   (Intr)
## time -0.132

```

It is important to note that in cases where we do not have a balanced design (different number of observations at the same time points for each individual) a mixed model will lead to different estimates than a linear model. In general, by taking into account the correlation structure one observes a shrinkage of the estimates towards the mean.

The structure of the resulting object of an `lmer()` call is more elaborate than that of an `lm()` call. Besides having more “components” due to the more complex nature, it possesses an `S4` structure which can be seen as a structured list. The individual components from the fitted object can be extracted through `fitted()`, `resid()`, `fixef()` (fixed effects), `ranef()` (random effects) and `coef()` (both the fixed and random effects), for example. Many “methods” exist; see the result of the R-call `methods(class="merMod")` for more.

Example 6.2. (Pigs’ weight continued) Let us return to the `pigs` dataset and look at some uncertainties. One can visualize the random effects (extracted using `ranef()`) using a caterpillar plot, as in Figure 6.2. The caterpillar plot does not indicate any structure between the variables (as seen with the low correlation). Similarly, the plot `plot(ranef(m3))$id` shows no structure. This is no surprise since there is only a very weak correlation between the random effects (estimated at roughly -0.06). Thus the call `lmer(weight ~ time + (time || id), data=pigslong)` yields numerically very similar results (`m4` in R-Code 6.3).

Finally, models `m2`, `m3` and `m4` are nested and an analysis of variance (`anova()` call) can be applied. The result is based on the log likelihood (and thus the model is possibly refitted with argument `REML=FALSE`) and the p -values provide the strength of evidence against H_0 : “the smaller model is sufficient”. Hence we would argue that independent random intercepts and slopes should be used to model the weight gain.



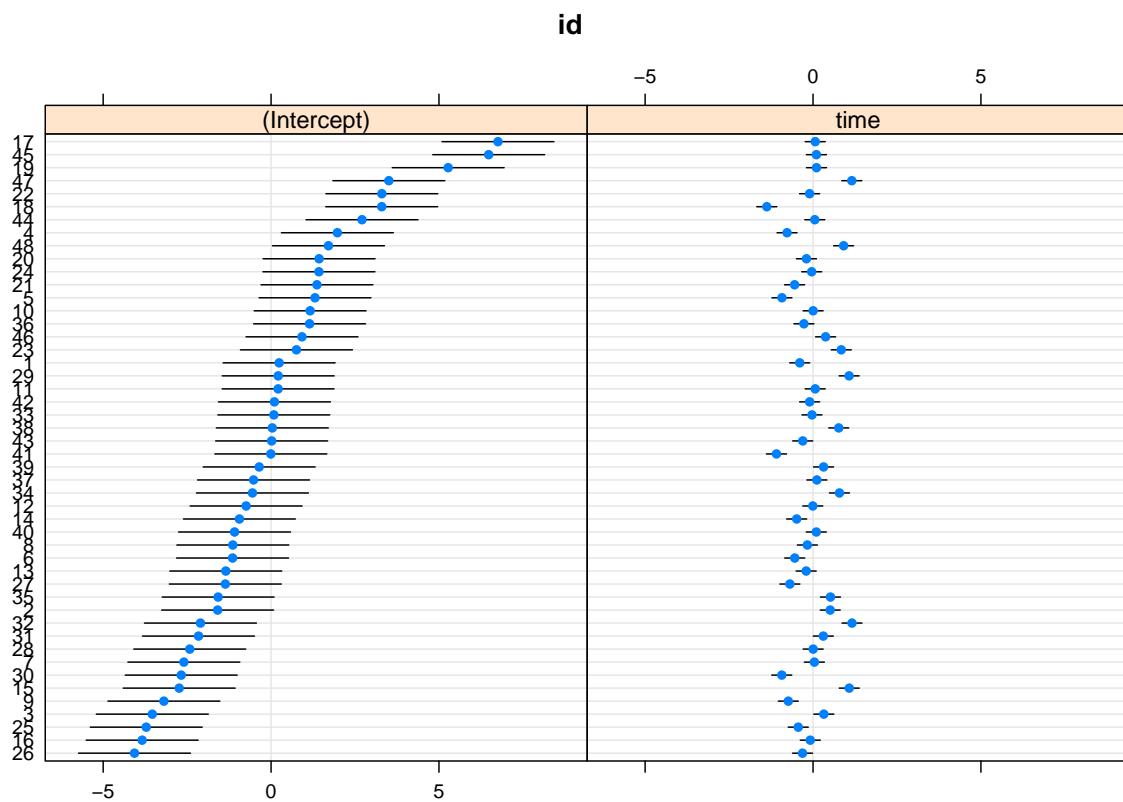


Figure 6.2: Random effects and associated uncertainties (based on conditional covariances) of the `pigs` dataset. (See R-Code 6.3.)

R-Code 6.3: Inferential aspects for the `pigs` dataset.

```

library(lattice)
dotplot(ranef(m3, condVar=TRUE))

## $id

confint(m3, nsim=50, method="boot") # nsim could be maybe slightly larger
## Computing bootstrap confidence intervals ...
##          2.5 %    97.5 %
## .sig01     2.0298911  3.4688652
## .sig02    -0.3638148  0.1305362
## .sig03     0.4680248  0.8511890
## .sigma     1.1808395  1.3618936
## (Intercept) 18.6071582 20.1424181
## time       5.9717635  6.4655477

### the nonsignificant coefficient is now called .sig02
### An alternative way to calculate the CI is through profiling:

```

```
# confint(m3) # but this takes a couple seconds!

### cor_time.(Intercept)/id is not significant: fit independent random effects:
m4 <- lmer(weight ~ time + (time || id), data=pigslong)
anova(m2, m3, m4, REML=FALSE)

## refitting model(s) with ML (instead of REML)
## Data: pigslong
## Models:
## m2: weight ~ time + (1 | id)
## m4: weight ~ time + ((1 | id) + (0 + time | id))
## m3: weight ~ time + (time | id)
##   npar    AIC    BIC  logLik deviance    Chisq Df Pr(>Chisq)
## m2     4 2037.8 2054.1 -1014.93   2029.8
## m4     5 1748.1 1768.4  -869.04   1738.1 291.7771   1      <2e-16 ***
## m3     6 1749.9 1774.3  -868.96   1737.9  0.1528   1      0.6959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The predict `predict()` method works for mixed models as well as for linear models. However, one has to additionally specify if one needs predictions with respect to the fixed effects or random effects (etc). R-Code 6.4 illustrates several different types of predictions and compares the values with the original observations (for pig `id==1`).

R-Code 6.4 Prediction for the `pigs` dataset.

```
timepts <- c(1, 3, 5, 7, 9)
newpig <- data.frame(time=timepts, id=1)
# fixed effects:
p1 <- predict(m3, newdata=newpig, re.form=~0)
# fixed and random intercept effect:
p2 <- predict(m3, newdata=newpig, re.form=~(1|id))
# fixed and random effects
p3 <- predict(m3, newdata=newpig, re.form=~(time|id))
print(round(rbind(pigs[1,timepts], p1, p2, p3), 2))

##   weight.1 weight.3 weight.5 weight.7 weight.9
## 1    24.00    39.00    48.00    61.00    72.00
## 2    25.57    37.99    50.41    62.82    75.24
## 3    25.81    38.23    50.65    63.07    75.48
## 4    25.41    37.04    48.66    60.29    71.92
```

A final remark concerning the interpretation of the time variable: Note that this is dependent on the coding of the time variable. Time point zero can be defined as the time of some baseline

measurement or an otherwise defined starting point. Alternatively, time could be centered around the midpoint. In this case, the random intercept reflects individual properties around the middle of the time axis. Even the last time point or a time point not contained in the data set can be defined as time point zero, in which case the intercept parameter represents an extrapolation of time.

The consequence of this is that if the individual heterogeneity is different at different time points, the estimated variance of the random intercept can be different.

6.3.1 General Model Formulation

Let's have a look at the form of the model one more time, but this time as a summary of the measurements of one individual i :

$$\mathbf{Y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\boldsymbol{\alpha}_i + \boldsymbol{\epsilon}_i \quad \text{with } \boldsymbol{\alpha}_i \stackrel{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad \text{independent of } \boldsymbol{\epsilon}_i \stackrel{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}). \quad (6.15)$$

To simplify the notation, we omit the corresponding dimensions of the vectors. In most cases, the columns of \mathbf{Z}_i are a subset of the columns of \mathbf{X}_i . Note that there are two different perspectives from which we can interpret the model:

- Conditional: The focus of the conditional linear mixed model is on the random effects. Using the estimated individual random effects, we can make individual forecasts for time points that are not yet contained in the model.
- Marginal: Here we focus on the fixed effects, which inform us about the mean behavior of the population. Using this information, we can provide predictions for new individuals.

The expected value of the conditional distribution of \mathbf{Y}_i (i.e. with given random effects) is

$$E(\mathbf{Y}_i | \boldsymbol{\alpha}_i) = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\boldsymbol{\alpha}_i, \quad (6.16)$$

and the respective covariance is

$$\text{Cov}(\mathbf{Y}_i | \boldsymbol{\alpha}_i) = \sigma^2 \mathbf{I}. \quad (6.17)$$

To obtain the expected value of the marginal distribution of \mathbf{y}_i , we use the law of iterated expectations:

$$E(\mathbf{Y}_i) = E(E(\mathbf{Y}_i | \boldsymbol{\alpha}_i)) = E(\mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\boldsymbol{\alpha}_i) = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i E(\boldsymbol{\alpha}_i) = \mathbf{X}_i\boldsymbol{\beta}. \quad (6.18)$$

For the covariance matrix of the marginal distribution, we use the variance decomposition formula:

$$\text{Cov}(\mathbf{Y}_i) = E(\text{Cov}(\mathbf{Y}_i | \boldsymbol{\alpha}_i)) + \text{Cov}(E(\mathbf{Y}_i | \boldsymbol{\alpha}_i)) = E(\sigma^2 \mathbf{I}) + \text{Cov}(\mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\boldsymbol{\alpha}_i). \quad (6.19)$$

We know that

$$\text{Cov}(\mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\boldsymbol{\alpha}_i) = \text{Cov}(\mathbf{Z}_i\boldsymbol{\alpha}_i) = \mathbf{Z}_i \text{Cov}(\boldsymbol{\alpha}_i) \mathbf{Z}_i^T = \mathbf{Z}_i \boldsymbol{\Sigma} \mathbf{Z}_i^T. \quad (6.20)$$

Using this in the equation above, we can finally write

$$\text{Cov}(\mathbf{Y}_i) = \sigma^2 \mathbf{I} + \mathbf{Z}_i \boldsymbol{\Sigma} \mathbf{Z}_i^T. \quad (6.21)$$

Hence, it is important to specify the conditional or marginal setting.

6.4 Additional Examples

6.4.1 Sitka Dataset

We consider measurements on the log-size of 79 Sitka spruce trees, 54 of which were grown in ozone-enriched chambers and 25 which served as controls. The size (height times diameter squared) of each tree was measured eight times, at roughly monthly intervals.

In R-Code 6.5 we compare two models: one with a random intercept and one with a random intercept and a random slope. We rescale the time variable to months after 1989 to avoid convergence issue from the fitting procedure. (A fitting without rescaling issues warnings and suggests rescaling).

We choose to fit the model with the classical maximum likelihood method instead with the default REML method in order to compare the resulting AICs with arbitrary fixed effects fits; see also the technical remark at the end of this section.

R-Code 6.5: *Sitka* data. (See Figure 6.3.)

```

library(lme4)
data(Sitka89, package="MASS")
str(Sitka89)

## 'data.frame': 632 obs. of  4 variables:
##   $ size : num  6.16 6.18 6.48 6.65 6.87 6.95 6.99 7.04 5.2 5.22 ...
##   $ Time : num  469 496 528 556 579 613 639 674 469 496 ...
##   $ tree : int  1 1 1 1 1 1 1 1 2 2 ...
##   $ treat: Factor w/ 2 levels "control","ozone": 2 2 2 2 2 2 2 2 2 2 ...
# help("Sitka89", package="MASS") # "size" is already on a log scale!

with(Sitka89, {
    # potting:
    time <- unique(Time)
    plot(time, size[tree==1], xlab="Time (days since 1988)", ylab="Size",
         ylim=range(Sitka89$size), type="l", col=3-(treat=="ozone"))
    for(i in 2:79)
        lines(time, size[tree==i], col=3-((treat[tree==i])[1]=="ozone"))
    means <- tapply(size, list(treat,Time), mean)
    lines(time, means[2,], lwd=4)
    lines(time, means[2,], col=2, lty=2, lwd=2)
    lines(time, means[1,], lwd=4)
    lines(time, means[1,], col=3, lty=2, lwd=2)
    legend("bottomri", legend=c("Ozone", "Control"), col=2:3, bty="n", lwd=1)
})
Sitka89$Time <- (Sitka89$Time-366)/(356/12) # rescaling

```

```
### Fitting ans two summary outputs
summary(out1 <- lmer(size ~ Time + (1 | tree), data=Sitka89, REML=FALSE))
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: size ~ Time + (1 | tree)
##   Data: Sitka89
##
##      AIC      BIC  logLik deviance df.resid
## -339.5 -321.8  173.8   -347.5     628
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -3.6472 -0.6969  0.0019  0.6768  3.1761
##
## Random effects:
##   Groups   Name        Variance Std.Dev.
##   tree     (Intercept) 0.4192   0.6474
##   Residual           0.0175   0.1323
## Number of obs: 632, groups: tree, 79
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) 5.248551  0.074795 70.17
## Time       0.108325  0.002356 45.98
##
## Correlation of Fixed Effects:
## (Intr)
## Time -0.216
print(out2 <- lmer(size ~ Time + (1 + Time | tree), data=Sitka89, REML=FALSE))
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: size ~ Time + (1 + Time | tree)
##   Data: Sitka89
##      AIC      BIC  logLik deviance df.resid
## -381.4216 -354.7282 196.7108 -393.4216     626
##
## Random effects:
##   Groups   Name        Std.Dev. Corr
##   tree     (Intercept) 0.69780
##           Time         0.02492 -0.40
##   Residual           0.11816
## Number of obs: 632, groups: tree, 79
## Fixed Effects:
## (Intercept)      Time
```

```

##      5.2486      0.1083

### the following is a way to extract relevant info from summary:
lme4:::prt.VC(summary(out2)$varcor,3,comp=c("Variance", "Std.Dev."))

## Random effects:
## Groups   Name        Variance Std.Dev. Corr
## tree     (Intercept) 0.486927 0.6978
##           Time       0.000621 0.0249   -0.40
## Residual            0.013962 0.1182

AIC(lm(size ~ Time, data=Sitka89), out1, out2)

##                     df      AIC
## lm(size ~ Time, data = Sitka89) 3 1275.8940
## out1                      4 -339.5494
## out2                      6 -381.4216

# confint(out2, method="boot", nsim=50) # e.g., leads to similar results.

```

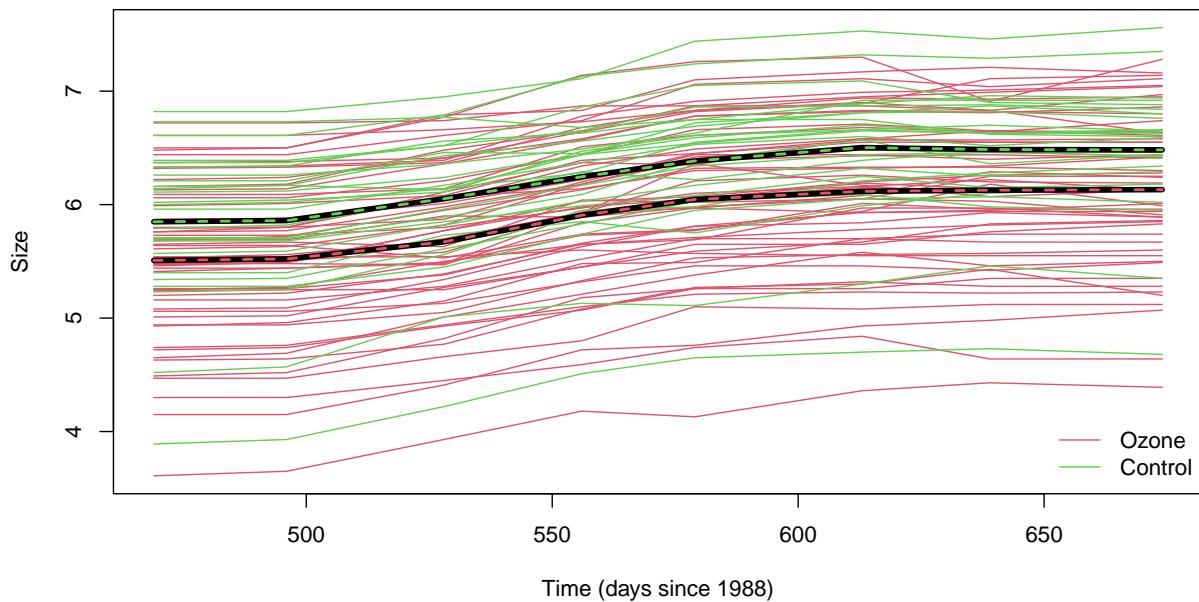


Figure 6.3: *Sitka89* dataset. The means are indicated with bold (black) lines. (See R-Code 6.5.)

Technical note: When comparing arbitrary fixed effects models and mixed effects models, ML method is required. If the fixed effects terms are the same in all models, either REML or ML is fine, but REML usually preferred. Also note that the interpretation of the results is usually difficult when both fixed effects and random effects are changing. Hence, we recommend changing only one or the other at a time.

Using a likelihood ratio test (LRT) is delicate when testing if one variance component is zero because the typical chi-squared approximation does not hold. This is because the null-hypothesis, $H_0 : \sigma_\alpha^2 = 0$ say, has the parameter on the boundary of the parameter space. There

is an entire theory on the distribution of the LRT in these situations (see, e.g., [Self and Liang, 1987](#)). For this special case, where one parameter lies on the boundary, the likelihood ratio has a $1/2\chi_0^2 + 1/2\chi_1^2$ -distribution. In practice, one has to halve the p -value based on χ^2 .

6.4.2 Pigs Dataset Revisited Using ggplot2

We illustrate the `pigs` dataset using the plotting functionality of `ggplot2()`, a package that provides a very powerful mechanism for displaying all sorts of plots. The mechanism is almost like a language on its own and, in practice, one often starts with a suitable example that is then further modified.

R-Code 6.6 superimposes different fits (pooled linear, individual linear and random intercept) for a selection of pigs (specified by `id.sel`). The pooled fit is naturally the worst and the mixed model one does a good job. Subject 18 seems to have a different slope than the majority of other subjects. Notice that the first few lines have been called previously and would not be needed again.

R-Code 6.6 `pigs` dataset revisited. Fancy plotting routines. (See Figure 6.4.)

```
library(lme4)
library(ggplot2)
pigs <- read.csv("download/sta121/pigs.csv",
                  col.names=paste0("weight.", 1:9), header=F)
pigslong <- reshape(pigs, direction="long", varying=1:9)
id.sel <- c(1,2,9,17,18,30)
pigsplot <- pigslong[pigslong$id %in% id.sel,]
m2 <- lmer(weight ~ time + (1 | id), data=pigslong)
mm2 <- data.frame(id=id.sel, intercept=coef(m2)[[1]][id.sel,1],
                   slope=coef(m2)[[1]][id.sel,2])
g <- ggplot(data=pigsplot, aes(x=time, y=weight)) + geom_point() +
  facet_wrap(~id) + theme_bw() +
  geom_smooth(formula=y ~ x, method="lm", aes(color="lm.ind"), se=FALSE) +
  geom_smooth(formula=y ~ x, method="lm", aes(color="lm.pool"),
             data=within(pigslong, id <- NULL), se=FALSE) +
  geom_abline(aes(intercept=intercept, slope=slope, color="lme"),
              data=mm2) +
  scale_colour_manual(name="Model", values=c("lm.ind"="blue2",
                                             "lm.pool"="orange", "lme"="green4"))
g
```

R-Code 6.7 illustrates the shrinkage effect of the mixed model approach. For a better illustration, we have redone the analysis with centered `time`, or such that the design matrix \mathbf{X} has orthogonal columns. The random intercept model has one common slope (identical to the pooled one). The values for the intercepts are slightly smaller than those for the other models. There are virtually no differences between the green and blue dots, confirming that a dependence

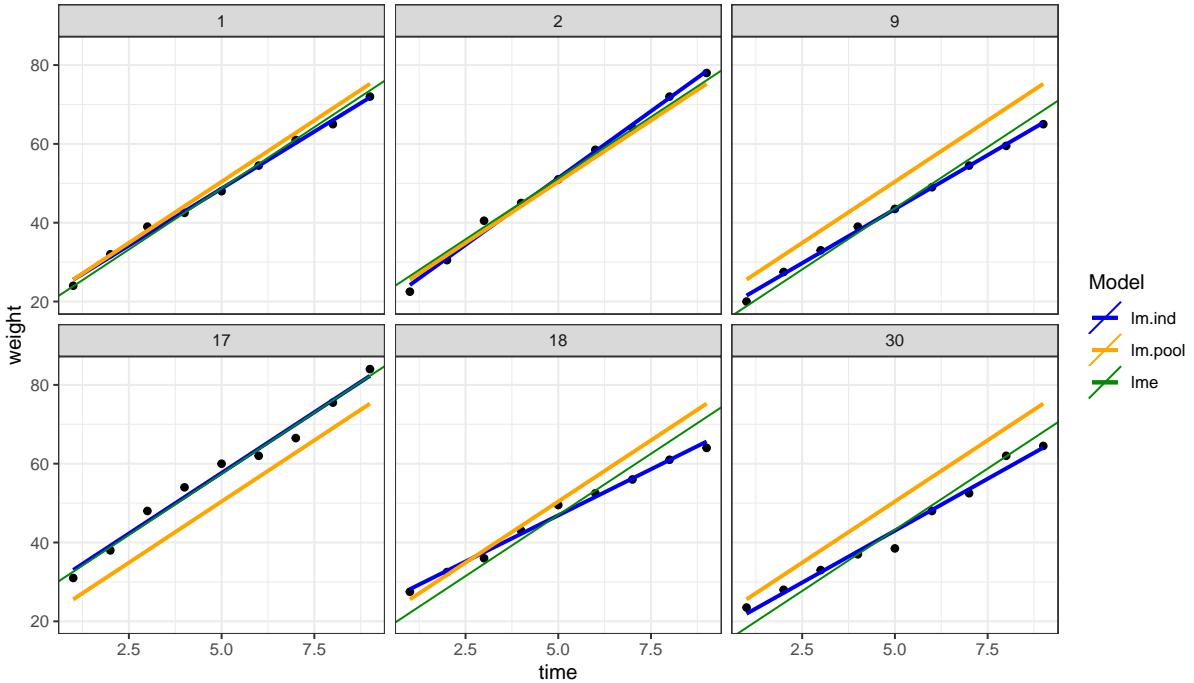


Figure 6.4: `pigs` dataset revisited. We use a plotting approach provided by the package `ggplot2()`. For visibility, we select six individual pigs. (See R-Code 6.6.)

between the effects is not really necessary.

Remark 6.1. Note that in the Bayesian context, there is not such a difference between fixed and random effects as in the case of a classical approach. If we take a linear model $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ and put a prior on $\boldsymbol{\beta}$ this is equivalent to a random effects model. Similarly, if we have a random effects model in the classical sense, say $\mathbf{Y} = \boldsymbol{\alpha}_i + \boldsymbol{\varepsilon}$ then this is as a Bayesian model with a normal prior on $\boldsymbol{\alpha}$. Of course for a proper specification, we need additional, carefully specified, priors. ♣

6.5 Bibliographic remarks

There are many R packages available for fitting mixed models: mainly `lme4` and `n1me`, but also `mgcv` and `MCMCglmm`. The model specification slightly varies between the different packages.

ASReml (www.vsni.co.uk/software/asreml, Butler *et al.*, 2009) is another (standalone) software that fits linear mixed effects models through a particular implementation of REML (Gilmour *et al.*, 1995, see also Gilmour *et al.*, 2004). There exists an R-interface through the package `asreml-R`. The model specification is relatively similar in the sense that one specifies the fixed effects and the random effects through separate arguments (as for `n1me`). However, ASReml is neither open-source nor free.

It is also possible to fit, e.g., logistic regression models by adding random effects to the linear predictor. This results in so-called generalized linear mixed models (GLMMs). We will see generalized linear models (GLMs) in a later chapter.

R-Code 6.7 Shrinkage effect of mixed models. (See Figure 6.5.)

```

library(plyr)
fits <- data.frame(id=rep(unique(pigslong$id), 5),
  model=rep(c("individual", "mixed2", "mixed3", "mixed4", "pool"),
  each=48), intercept=NA, slope=NA)
m1 <- lm(weight ~ time, data=pigslong)
fits[fits$model == "pool", "intercept"] <- coef(m1)[1]
fits[fits$model == "pool", "slope"] <- coef(m1)[2]

fits[fits$model == "individual", c("intercept", "slope")] <-
  ddply(.data=pigslong, .variables="id", .fun=function(x){
    coef(lm(weight ~ time, data=x) ) } )[,2:3]
m2 <- lmer(weight ~ time + (1 | id), data=pigslong)
m3 <- lmer(weight ~ time + (time | id), data=pigslong)
m4 <- lmer(weight ~ time + (time || id), data=pigslong)
fits[fits$model == "mixed2", c("intercept", "slope")] <- coef(m2)$id
fits[fits$model == "mixed3", c("intercept", "slope")] <- coef(m3)$id
fits[fits$model == "mixed4", c("intercept", "slope")] <- coef(m4)$id
g1 <- ggplot(data=fits, aes(x=intercept, y=slope, color=model)) +
  geom_point() + theme_bw()
g1

```

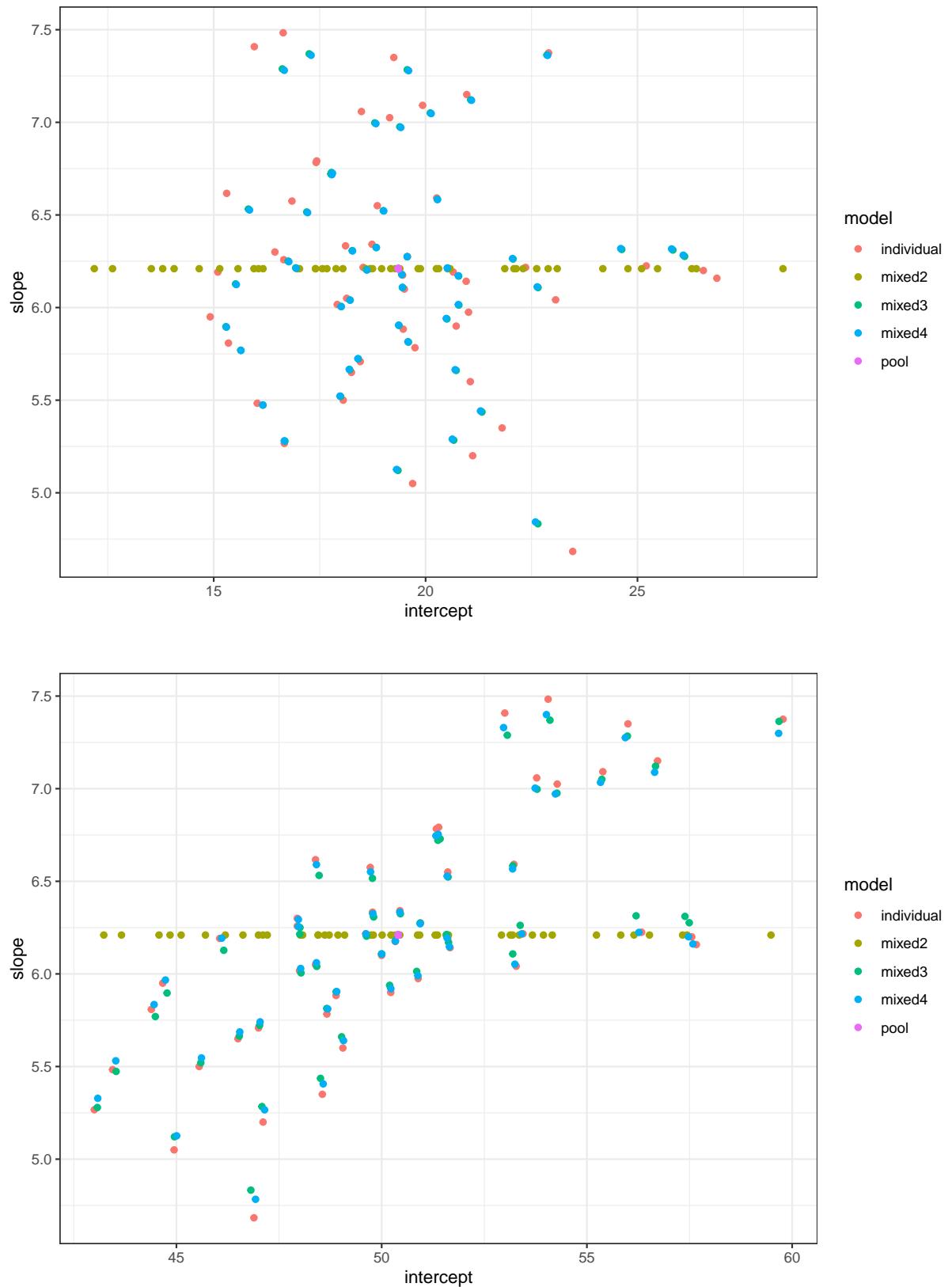


Figure 6.5: Shrinkage effect of mixed models. The bottom panel is based on the centered `time` covariate. `mixed[2,3,4]` correspond to the mixed models with random intercept, correlated random intercept and random slope and uncorrelated random intercept and random slope. (See R-Code 6.7.)

6.6 Exercises and Problems

Problem 6.1 (ANOVA with random effect) The dataset `hearing.txt` is available at <https://www.math.uzh.ch/furrer/download/sta121/hearing.txt>. The data are from a hearing experiment:

24 subjects with normal hearing listened to standard audiology tapes of English words at low volume with a noisy background. They repeated the words and were scored correct or incorrect in their perception of the words.

Each subject was exposed to four word list with 50 words. The order of the list was randomized.

This word lists are standard audiology tools for assessing hearing. They are calibrated to be equally difficult to perceive. However, the original calibration was performed with normal-hearing subjects without noise background. The experimenter wished to determine whether the lists were still equally difficult to understand in the presence of a noisy background.

The three observed variables are:

`SubjectID`: Code for each subject - 24 of them

`ListID`: Code for each of the four lists played

`Hearing`: Score received on hearing test (\% heard correct)

(Loven, Faith. (1981). A Study of the Interlist Equivalency of the CID W-22 Word List Presented in Quiet and in Noise. Unpublished MS Thesis, University of Iowa.)

- a) Make an exploratory analysis of the data.

Hint: Among others, produce a box-plot comparing the hearing scores of the four different word lists.

- b) Which variables explain much of the variation in the variable `Hearing`?

Hint: `lm()`, `anova()`

- c) Model the variability of `Hearing` with a linear mixed model that takes the possibly correlated responses with each subject into account. Do the hearing scores differ for different lists?

Hint: Use `lme4::lmer()` with `SubjectID` as random effect.

- d) Compare the results from b and c.

Problem 6.2 (Longitudinal data) The dataset `termites.txt` is available from the <https://www.math.uzh.ch/furrer/download/sta121/termites.txt>. It contains data collected in an experiment that investigated the effects of certain tree resins on termites.

More precisely, a resin derived from the bark of tropical trees was dissolved and placed on filter paper in different dose levels (5mg and 10mg). For each dosage level, eight dishes are set up with 25 termites in each dish. The termites are fed the dosed filter paper and a daily count is made of the number of termites surviving.

The question was, if the resin harm the termites and if the different dosage levels influence effect the survival.

The observed variables are:

```

dish: dish number
dose: 5 or 10 mg of resin
day1: Number of termites still alive on this day
day2: Number of termites still alive on this day
...
day15: Number of termites still alive on this day

```

- a) Explore the data.
- b) Restructure the data to *long format*. Plot the data with functions from package `lattice`.
Hint: `reshape()` (`stats` package) or `melt()` from package `reshape2`. `xyplot()` and `bwplot()`.

The data in long format could look as follows:

```

head(term)

##   dish dose day n.termites
## 1    1    5   1       25
## 2    2    5   1       25
## 3    3    5   1       25
## 4    4    5   1       25
## 5    5    5   1       25
## 6    6    5   1       25

```

- c) Use a linear model to see whether the survival of the termites depends on `dose`, `day`, `dish`. What is problematic with the variable `dish`? Plot the predicted values of the model you found (x-axis: day, y-axis: predicted values, dishes indicated with colours).
- d) Fit the model again with a random intercept for each level of `dish`. Compare the results with c. Plot the predicted values.
- e) Fit the model again with a random intercept and slope for each level of `dish`. Compare the results with answers from questions c, and d. Plot the predicted values.
- f) Use bootstrap to simulate a 95% confidence interval for the `dose` parameter in model used to answer question d.

Chapter 7

Non-Parametric Regression

We model noisy observations in terms of an arbitrary function of one or more explanatory variables.

Learning goals for this chapter:

- ◊ Describe the motivation for non-parametric regression
- ◊ Describe kernel approaches, penalized approaches, locally adaptive approaches
- ◊ Calculate the various nonparametric regression estimates in R for given data
- ◊ Know the strengths and weaknesses of the various methods introduced

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter08.R.

7.1 Motivation

We start by looking at two classic datasets consisting of pairs of data. In both examples, simple regression models are not adequate because the models are not *flexible* enough to describe the data sufficiently well. In this chapter we discuss different approaches for such noisy types of data.

Example 7.1. Figure 1.2 shows the annual bituminous coal production in the USA between 1920 and 1968 (Makridakis *et al.*, 1998, Exercise 7.7). A polynomial of degree 10 seems to fit the data well (magenta colored line). However, this fit does not provide any predictive capability or socioeconomic interpretation. For example, when we use the fit to predict the coal production for 1915, `predict(lm(coal ~ poly(year, 10)), newdata=data.frame(year=1915))` our prediction is 6187, larger than the total production between 1920 and 1930! ♣

Example 7.2. The `mcycle` data gives a series of head acceleration measurements from a motorcycle accident simulation study used to test crash helmets (Silverman, 1985). The data is shown in the left panel of Figure 7.1.

We superimpose regression fits for models including time in milliseconds and polynomial powers thereof as predictors.

For this particular example, we use the package `ggplot2`, based on the *grammar of graphics* approach. We choose a layout for these figures close to base figures, without fully matching them. The fits are created using the function `stat_smooth()` and are drawn with uncertainty bands. For the used method `method="lm"`, the bands are based on the classical `predict.lm()` function.



R-Code 7.1 Datasets `mcycle` and `bicoal` plotted with `ggplot()`. (See Figure 7.1.)

```

data(mcycle, package="MASS")
str(mcycle)

## 'data.frame': 133 obs. of  2 variables:
## $ times: num  2.4 2.6 3.2 3.6 4 6.2 6.6 6.8 7.8 8.2 ...
## $ accel: num  0 -1.3 -2.7 0 -2.7 -2.7 -2.7 -1.3 -2.7 -2.7 ...
# plot.accel~times, data=mcycle) # would be a simple option.

library(ggplot2)
pl <- ggplot(mcycle, aes(times, accel)) + geom_point()
pl <- pl + stat_smooth(method="lm") + # add some fits and uncertainty bands
      stat_smooth(method="lm", formula=y ~ poly(x, 5), col=3) +
      stat_smooth(method="lm", formula=y ~ poly(x, 10), col=2)
pl # "print()ing" the object is equivalent to plotting

```

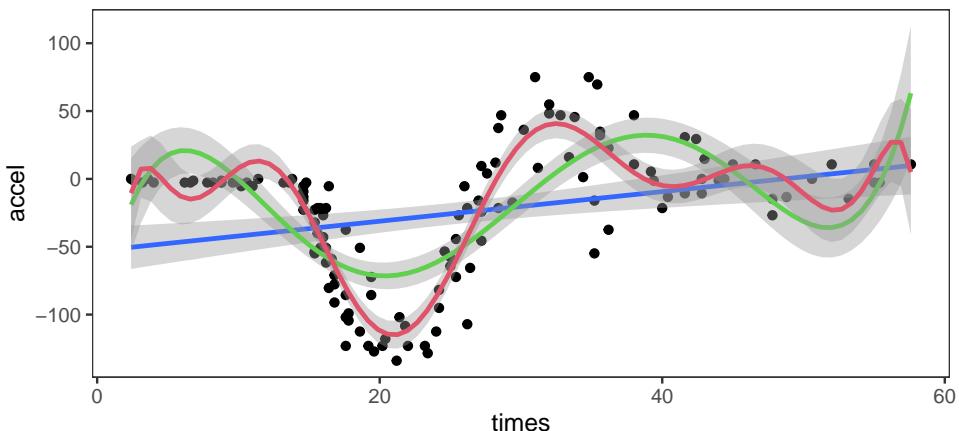


Figure 7.1: Dataset `mcycle` with a few linear model fits superimposed (linear, polynomial with five and ten degrees). (See R-Code 7.1.)

The classical linear model framework is extended and we assume that

$$Y_i = g(x_i) + \varepsilon_i, \quad (7.1)$$

where $\varepsilon_i \stackrel{\text{iid}}{\sim} (0, \sigma^2)$ and g an “arbitrary” function. In fact, it is arbitrary in the sense that it is not represented as a linear function of some parameter vector but, formally speaking, g is a twice differentiable.

In the simple regression case, $g(x) = \beta_0 + \beta_1 x$. If we consider multiple regression based on one covariate, we may have $g(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_p x^p$, as for example in Figure 7.1. We now assume that g is “complicated;” a few parameters are not sufficient to “model” g and thus the goal is to estimate g *non-parametrically*.

Many non-parametric techniques exist and most can be categorized into one of the following three approaches:

- Local estimation approaches (kernel smoothers, local polynomials, see Section 7.2)
- Penalized approaches (splines, see Section 7.3)
- Locally adaptive approaches (wavelets, . . . , see Section 7.4)

The first is motivated by a moving window sliding along x , which is used to predict $\hat{g}(x)$. The second is based on extending the classical least squares minimization by using a more flexible description of $g(x)$. The third takes into account that we need different smoothing strategies depending on x . Here we discuss the first two approaches in more detail and only outline the third one, much like a black box method.

One key property of the approaches discussed here and of many more used in practice is that they are linear smoothers, in the sense that

$$\hat{y}_i = \hat{g}(x_i) = \sum_{j=1}^n h_{ij} y_j \quad (7.2)$$

for some “weights” h_{ij} , resulting in

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}, \quad (7.3)$$

where the hat matrix \mathbf{H} has elements h_{ij} which typically depend on x_1, \dots, x_n . The hat matrix has properties similar to those of the corresponding hat matrix $\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ in the linear model. For example, the rank of the hat matrix determines the maximum “flexibility” of the fitted values. Specifically, a rank one matrix results in a constant value for g . Of course, a large rank does not necessarily imply a “good” fit.

7.2 Kernel Approaches

Kernel approach are moving window type approaches, where the prediction for a particular value depends on the nearby observations. This dependency is determined by weights that are calculated by a so-called kernel.

Definition 7.1. A kernel is a function $k(x)$ that

- is symmetric (around zero) and non-negative, i.e., $k(u) = k(-u)$ and $k(u) \geq 0$;
- is normed, i.e., $\int k(u)du = 1$;
- has a bounded second moment, i.e., $\int u^2k(u)du < \infty$;
- is square integrable, i.e., $\int k(u)^2du < \infty$.

Usually, one assumes that $\int u^2k(u)du = 1$. A kernel is a density but not vice-versa. \diamond

A kernel estimator is defined by a kernel function $k(u)$ and a bandwidth parameter $\lambda > 0$ and weights the observations proportional to $k(x_i - x/\lambda)/\lambda$. Hence, h_{ij} in (7.2)r consists of these normalized weights.

The following (kernel) estimators are popular practical variants:

$$\text{Nadaraya-Watson: } \widehat{g}(x) = \frac{\sum_i \frac{1}{\lambda} k\left(\frac{x_i - x}{\lambda}\right) Y_i}{\sum_i \frac{1}{\lambda} k\left(\frac{x_i - x}{\lambda}\right)} \quad (7.4)$$

$$\text{Gasser-M\"uller: } \widehat{g}(x) = \sum_i \int_{s_{i-1}}^{s_i} \frac{1}{\lambda} k\left(\frac{u - x}{\lambda}\right) du Y_i, \quad (7.5)$$

$$s_i = \frac{1}{2}(x_i + x_{i+1}), s_0 = -\infty, s_n = \infty.$$

Of course there is virtually no limitation on the number of kernels, some are used more often. The most commonly used ones are (right most version is unscaled, meaning $\int u^2k(u)du \neq 1$):

$$\text{uniform (or top-hat): } k(x) = \frac{1}{2\sqrt{3}} I(|x| \leq \sqrt{3}), \quad k(x) = \frac{1}{2} I(|x| \leq 1),$$

$$\text{triangular: } k(x) = \frac{1}{\sqrt{6}} \left(1 - \frac{|x|}{\sqrt{6}}\right) I(|x| \leq \sqrt{6}), \quad k(x) = (1 - |x|) I(|x| \leq 1),$$

$$\text{Epanechnikov: } k(x) = \frac{3}{4\sqrt{5}} \left(1 - \frac{x^2}{5}\right) I(|x| \leq \sqrt{5}), \quad k(x) = \frac{3}{4} (1 - x^2) I(|x| \leq 1),$$

$$\text{Gaussian: } k(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2),$$

where $I(\cdot)$ is the indicator function and restricts the support of the first three kernels. In R, these and more kernels can be visualized with `plot(density(0, kernel="rectangular", bw=1))` and appropriate `kernel` argument.

The key difficulty of kernel estimators is the choice of the bandwidth parameter λ . The bandwidth determines the amount of smoothing and roughly said $\lambda \rightarrow \infty$ results in the mean, i.e., $\widehat{g}(x) = \bar{Y}$ and $\lambda \approx 0$ results in the interpolator.

A sensible and careful choice of the bandwidth parameter balances bias and variance. Small bandwidth values result in a small amount of bias but large variance whereas for a large bandwidth, the relation is inverted.

Choosing the smoothing/bandwidth parameter of non-parametric regressions is the classic example of cross-validation (CV, see Section 1.3), which is illustrated in the next example. For some approaches, CV is computationally intensive and general cross-validation (GCV) procedures are used. These latter ones are essentially approximations to CV that approximate elements of the hat matrix in order to avoid n fitting procedures.

Example 7.3. R-Code 7.2 smooths the motorcycle data `bicoal` with different kernel smoothers: `ksmooth()` and `sm.regression()` (from the package `sm`). Some smoothers have difficulties with replicated data and thus we slightly jitter the times as we have 39 non-unique time stamps. The function `ksmooth()` is used first. A uniform kernel (default) yields very rugged curves. Switching to a Gaussian one (`kernel="normal"`) yields a truly “smooth” curve. The unbounded support of the Gaussian kernel yields a slightly stronger smoothed version. For `sm.regression()` we use a cross-validated bandwidth `h`. The bandwidth for that smoother can also be specified by the value of the trace of the hat matrix, corresponding to the degrees of freedom of the smoother. For our particular setting, we need about 16 degrees of freedom.

Note that `sm.regression()` can be used with two covariates for which the results can be presented with the package `rgl`.



R-Code 7.2 Kernel with `mcycle` data. (See Figure 7.2.)

```

accel <- mcycle$accel                      # for unique values
times <- mcycle$times+rnorm(length(accel), -.001, .001)
plot(times, accel, pch=20)
## If the output of `ksmooth` is not continuous, the default of `x.points`#
## may not be sufficient.
k1 <- ksmooth(times, accel, bandwidth=5, x.points=times)
lines(k1, col=3)
# a normal kernel makes it smooth...
k2 <- ksmooth(times, accel, kernel="normal", bandwidth=5, x.points=times)
lines(k2, col=4)

library(sm)          # hcv and sm.regression
print(cv <- hcv(times, accel, hstart=0.5, hend=3.5, ngrid=31))
## [1] 1.4765

sm.regression(times, accel, h=cv, add=T, col=2)
# Instead of specifying bandwidth, specify the degrees of freedom.
# The cv smooth corresponds to df=16 (found after several trials):
sm.regression(times, accel, df=16, add=T, col=6, lty=4)
legend("bottomright", bty="n", lty=c(1,1,1,4), col=c(3,4,2,6),
       legend=c("ksmooth: uniform, bw=5", "ksmooth: Gauss, bw=5",
               "sm.regression: cv", "sm.regression: df=16"))
cv <- hcv(times, accel, hstart=0.5, hend=3.4, ngrid=31, display="line")

```

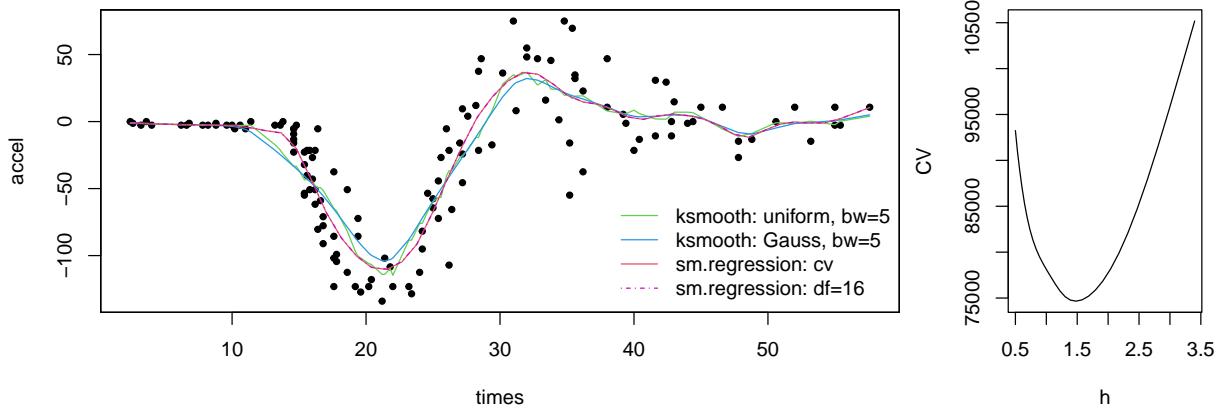


Figure 7.2: Kernel smoothers applied to the `mcycle` data (left) and cross-validation criterion for kernel smoother. (See R-Code 7.2.)

Kernel smoothers are also used when estimating densities. Suppose that we observe x_1, \dots, x_n from an unknown continuous density $f_X(x)$. A histogram is a discrete estimate of the density. An improved version thereof is a running bin, i.e., $\hat{f}_X(x) = \sum_{i=1}^n I(x_i \in [x \pm \delta/2]) / (\delta n)$. Here, based on the indicator function $I(\cdot)$ we estimate is the proportion of observation that fall into the neighborhood of x_i with width 2δ . This estimate is thus equivalent to the uniform kernel with $y_i \equiv 1$. The function `density()` uses a similar approach with a Gaussian kernel as default.

From a theoretical point of view, it can be shown that the bias is $\mathcal{O}(\lambda^2)$ and the variance is $\mathcal{O}(1/(n\lambda))$. Choosing a bandwidth of rate $\lambda = \lambda(n) = \mathcal{O}(n^{-1/5})$, kernel approaches thus have a mean squared error rate of $\mathcal{O}(n^{-4/5})$. Note that this rate is slower than traditional approaches with $\mathcal{O}(1/n)$. This is the price of not assuming a parametric model.

7.2.1 Local Polynomials

Kernel smoothers suffer from boundary bias (at the boundary, we have observations on one side only) and, to a lesser extend, from bias induced by the design points $\{x_i\}$. Local polynomial address both issues and naturally extend kernel smoothers.

For a fixed value x_0 , consider the following minimization problem

$$\min_{a(x_0)} \sum_i w_i(x) (Y_i - a(x_0))^2, \quad (7.6)$$

where $a(x)$ is a constant but depends on the fixed (x) . The solution to this problem is the Nadayara–Watson estimator $\hat{a}(x)$ for which the weights are given by the kernel, i.e., $w_i(x) = k((x - x_i)/\lambda)/\lambda$. The idea of local polynomials is to replace $a(x_0)$ by a q th order polynomial for some small value of q . The idea is that we approximate $g(x_i)$ with, for example, locally with $a_0 + a_1(x_i - x_0)$. The resulting smoother is again a linear smoother and the solution is much like a weighted least squares solution.

A local linear smoother has a smaller bias than Nadayara–Watson but the same variance.

There are two commonly used functions for local polynomials in R: `loess()`, and `lowess()`

(See Example 7.4). The latter is also used as the “red” guide the eye line in scatterplot matrices (e.g., Figure 1.5).

Example 7.4. For completeness, we apply both `loess()` and `lowess()` to the `mcycle` dataset. `lowess()`, which is an older implementation based on a complicated algorithm, and `loess()`, which is newer and more coherent with respect to the arguments. Often, default values for the bandwidth are not useful. Additionally, the bandwidth cannot be shared between different approaches. ♣

R-Code 7.3 Local polynomials and the `mcycle` data. (See Figure 7.3.)

```
plot(times, accel, pch=20)
lobj <- loess(accel ~ times)           # default smoothing value
lines(lobj$x, lobj$fitted, col=5)
lobj <- loess(accel ~ times, span=.1)   # not enough smoothing
lines(lobj$x, lobj$fitted, col=3)
lobj <- loess(accel ~ times, span=0.35) # probably close to optimal
lines(lobj$x, lobj$fitted, col=2, lwd=2)
lines(lowess(times, accel, f=0.35), col=4, lwd=2) # same value
legend("bottomright", lty=1, col=c(5,3,2,4), lwd=c(1,1,2), bty="n", legend=
  c("loess: default","loess: span=0.1","loess: span=0.35","lowess: f=0.35"))
```

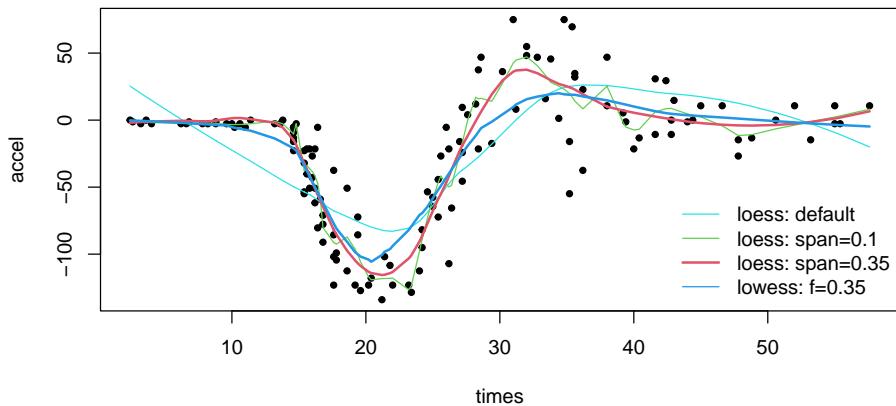


Figure 7.3: Local polynomials with the `mcycle` data. (See R-Code 7.3.)

7.3 Splines

In the classical least squares minimization with $g(x) = \mathbf{x}^\top \boldsymbol{\beta}$, the criterion

$$\operatorname{argmin}_g \sum_{i=1}^n (y_i - g(x_i))^2, \quad g(x_i) = \mathbf{x}_i^\top \boldsymbol{\beta} \quad (7.7)$$

is well defined. The minimization over the g is equivalent a minimization over $\boldsymbol{\beta}$. We now think of a much more flexible description of $g(x)$, where g can be (almost arbitrary). In that case, the

criterion above may not be optimal as we find one function that essentially interpolates, i.e., for all distinct values x_i , $y_i = g(x_i)$. Thus we need extend the criterion to balance residual sums of squares (RSS) and smoothness (in the kernel smoothing sense):

$$\hat{g}(x) = \operatorname{argmin}_g \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda J(g), \quad (7.8)$$

where λ is a smoothing parameter and J is a penalty function that for smoother functions yield smaller penalties.

We aim to find a function g that is not too “wiggly”, i.e., a smoother fit. A common way to quantify the smoothness is through the second derivative of g and we set the penalty function $J(g)$ to

$$J(g) = \int (g''(x))^2 dx, \quad (7.9)$$

and the goal is to find the minimizer of $M(\lambda) = \text{RSS} + \lambda J(g)$ over all possible functions g (strictly speaking, twice differentiable). At first sight, it is surprising that such a seemingly complex problem has a solution which is moreover very simple. The function minimizing $M(\lambda)$ is a natural cubic spline with knots at the data points and is called a natural smoothing spline (see as follows).

To simplify the notation, we assume that we have observations at distinct values (that we order) $x_1 < x_2 < \dots < x_n$. In equation (7.9), we integrate over (a, b) only, often $a = x_1$ and $b = x_n$, but $a = -\infty$ and/or $b = \infty$ or any other value is possible.

Formally, a cubic spline for the knots ξ_1, \dots, ξ_k is a function $g(x)$ that is a cubic polynomial in each interval $[\xi_i, \xi_{i+1}]$, $i = 1, \dots, k-1$, and is twice differentiable ($g'(x)$ and $g''(x)$ are continuous everywhere). A natural cubic spline additionally assumes $g(x)$ to be linear in $(a, \xi_1]$ and $[\xi_k, b)$. The solution to (7.8) is a natural cubic spline with knots at the data points, i.e., $\xi_1 = x_1, \dots, \xi_n = x_n$.

As for kernel smoothers, we can use CV to determine λ or use some a-priori knowledge reflecting the hypothesized smoothness of g .

Example 7.5. R-Code 7.4 fits different smoothing splines to the `mcycle` dataset. In the second part, we manually extract the CV and GCV criterion, also visualized in Figure 7.4. ♣

Numerically, the smoothing spline is found by formulating a basis for the splines and then applying a ridge regression approach. There are several possible spline basis functions, see below. For the moment, assume that the j th basis function is written as $b_j(x)$. The elements of the matrix \mathbf{B} are defined by $b_j(x_i)$. Then the smoothing spline is the solution to

$$(\mathbf{y} - \mathbf{B}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{B}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^\top \boldsymbol{\Omega} \boldsymbol{\beta}, \quad (7.10)$$

where $\boldsymbol{\Omega}$ is the matrix having elements $\int_a^b b_i''(x) b_j''(x) dx$. The solution is simply

$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^\top \mathbf{B} + \lambda \boldsymbol{\Omega})^{-1} \mathbf{B}^\top \mathbf{y}. \quad (7.11)$$

R-Code 7.4 Splines for the *mcycle* data. (See Figure 7.4.)

```
(s1 <- smooth.spline(times, accel))      # generalized CV, default
## Call:
## smooth.spline(x = times, y = accel)
##
## Smoothing Parameter  spar= 1.0655  lambda= 0.00011012 (12 iterations)
## Equivalent Degrees of Freedom (Df): 12.243
## Penalized Criterion (RSS): 61988
## GCV: 565.37

s2 <- smooth.spline(times, accel, cv=T)  # ordinary CV
s3 <- smooth.spline(times, accel, spar=1)
print(c(s1$lambda, s2$lambda, s3$lambda)*1e6)
## [1] 110.125 91.003 36.998

plot(accel~times, pch=20)
lines(s1, col=2)
lines(s2, col=3)
lines(s3, col=4, lty=4)
legend("bottomright", legend=c("default (GCV)", "CV", "spar=1"),
       bty="n", col=c(2,3,4), lty=c(1,1,4))
ln <- 50
output <- array(0, c(ln,4))
sparseq <- seq(0.8, to=1.2, length=ln)
for (i in 1:ln) {
  si <- smooth.spline(times, accel, spar=sparseq[i], cv=TRUE)
  output[i,] <- c(si$lambda, si$cv.crit, smooth.spline(times,
               accel, spar=sparseq[i], cv=FALSE)$cv.crit, si$df)
}
colnames(output) <- c("lambda", "CV", "GCV", "d.f.")
print(opt <- cbind(CV=output[which(min(output[,2])==output[,2]),1],
                     GCV=output[which(min(output[,3])==output[,3]),1]))
##          CV      GCV
## lambda 8.9439e-05 0.00011735
plot(output[,1], output[,2], type="l", xlab=expression(lambda),
      ylab="CV/GCV criterion")
lines(output[,1], output[,3], col=4)
abline(v=opt, col=c(1,4), lty=2)
```

There are several ways to define spline basis functions. One example is the so-called truncated power series basis $\{1, x, x^2, x^3, (x - \xi_1)_+^3, \dots, (x - \xi_k)_+^3\}$, where $x_+ = \max(x, 0)$. Another, computationally more appealing example is the B-spline basis for the natural cubic spline. Figure 7.5

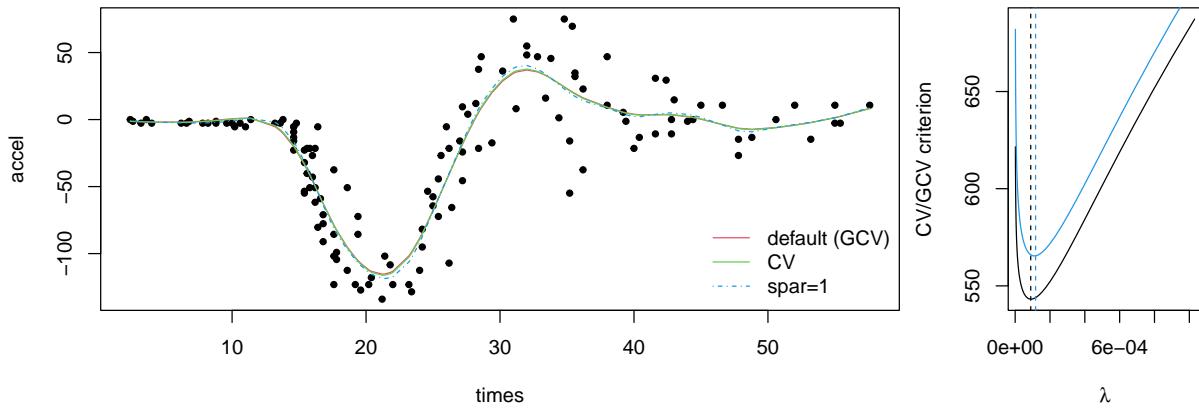


Figure 7.4: Spline smoothers for the `mcycle` data (left) and manually performing CV to find an optimal smoothing parameter value (right). (See R-Code 7.4.)

shows nine B-spline basis elements.

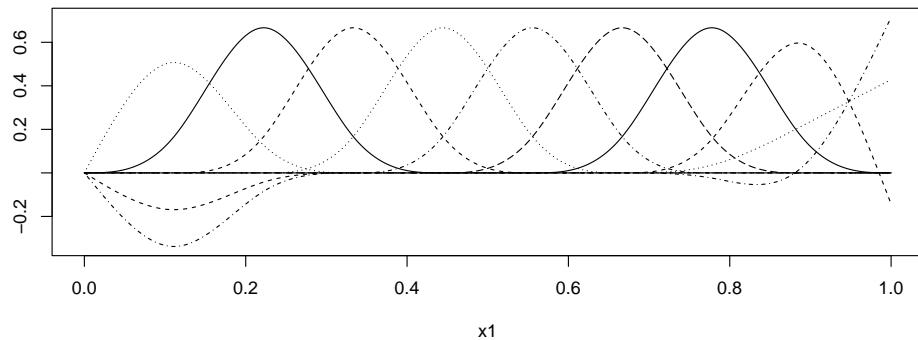


Figure 7.5: Basis functions of natural cubic splines, essentially `matplotlib()` visualization of `splines::ns()`.

Regression splines are an alternative to smoothing splines. We construct spline basis functions and then use these to regress the observations on. However, we restrict the number of *knots*. More precisely, regression splines minimize a residual sum of squares over splines with knots at k determined locations, ξ_1, \dots, ξ_k . The smoothing is then accomplished through the number of knots and the position of the knots, illustrated in R-Code 7.5.

R-Code 7.5 Regression splines. (See Figure 7.6.)

```
plot(accel ~ times, pch=20)
sm1 <- lm(accel ~ bs(times, 6))
lines(times, sm1$fitted)
lines(times, lm(accel ~ bs(times, 12))$fitted, col=3)
knots <- c(1,15,20,25,30,50)
abline(v=knots, col="gray")
lines(times, lm(accel ~ bs(times, knots=knots))$fitted, col=2)
```

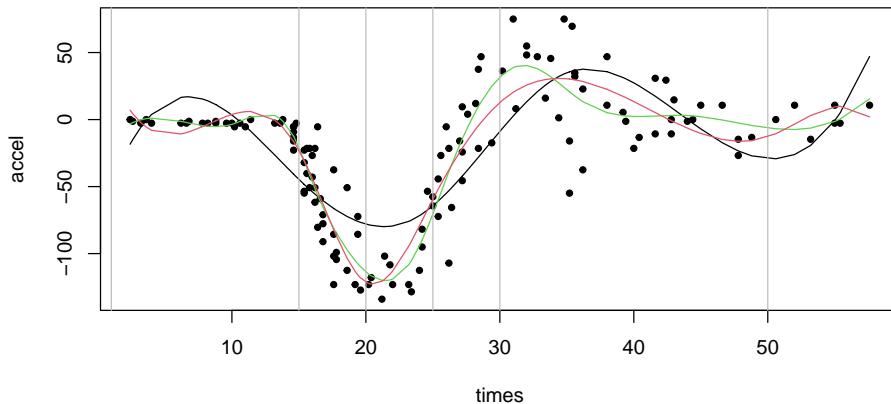


Figure 7.6: Regression splines with `mcycle` data. The vertical lines indicate locations of the knots. (See R-Code 7.5.)

It is possible to cast the splines and regression splines into the kernel framework.

(Interpolating) splines and smoothing splines are also often used in simple drawing programs to parameterize smooth curves. An interpolating spline function can be calculated by the R-function `spline()`.

7.4 Locally Adaptive Approaches

Most approaches discussed thus far determine the smoothing parameter globally, i.e., there is a single parameter for the entire dataset. The exception is regression splines, where the knots are determined manually and thus it is possible to achieve different levels of smoothness in the data. However, all these approaches are not suitable in cases where the curve has extremely different smoothing behaviors in different regions. A classic case is shown in the top panel of Figure 7.7, the so-called ‘doppler’ function. This artificial function serves as a test function for locally adaptive approaches.

The archetype locally adaptive smoother uses wavelets. The theoretical background of wavelets are mathematically quite evolved and thus we need some handwaving here. In a nutshell, a wavelet decomposition is as follows. One defines a scaling function $\phi(x)$ (also called the “father” wavelet function) and a mother wavelet function $\psi(x)$, with descendants $\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k)$. The idea is to write the function $g(x)$ as

$$g(x) = \alpha\phi(x) + \sum_{j=0}^{\infty} \sum_{k=0}^{2^j-1} \beta_{jk}\psi_{jk}(x), \quad (7.12)$$

where α is the scaling coefficient and β_{jk} the detail coefficients. Of course, in order that the last equation holds, the scaling and wavelet functions cannot be arbitrary. In fact, equation 7.12 is a simplified version of a complete and general wavelet decomposition. We shall not bother with the general description here.

A wavelet representation is not much unlike a Fourier representation except that the “basis functions” may represent large and small features. Moreover, the functions are scaled such that important features imply large coefficients. That means that smoothing is equivalent to setting

small coefficients β_{jk} to zero. This procedure is called thresholding and two typical approaches are *hard thresholding* and *soft thresholding*. The former sets all coefficient that exceeds a threshold δ to zero, the latter additionally shifts the coefficients by δ . The thresholding functions can be visualized by `curve(x*(abs(x)>1), -2, 2)` and `curve(sign(x)*pmax(abs(x)-1,0), -2, 2)`.

The concept of a wavelet decomposition is illustrated in the following example.

Example 7.6. The doppler function $f(x) = \sqrt{x(1-x)} \sin(2.1\pi/(x+0.05))$ represents a sinusoidal function with changing amplitude and frequency. We observe the function at $2^8 = 256$ equispaced points with a noise having variance $\sigma^2 = 0.01$, see top panel of Figure 7.7.

We decompose the signal (with default parameters for the wavelet specification, see left panel of second row of Figure 7.7), leading to the different coefficients α and β_{jk} , the latter are represented in the right panel of Figure 7.7). For didactic purposes, we reconstruct the signal with with a few levels only. For example, with the red curve in left panel of the third row is a reconstructed signal based on three coefficients only.

The number of zeros does not change between hard and soft thresholding (at the same threshold) and increases with increasing threshold. The reconstruction illustrates nicely the difference between both thresholding approaches, for `value=.1`, both curves capture the large scale variability well but with `type="soft"`) the residuals are smaller as the coefficients of the last level only essentially represent noise. The stronger the thresholding the more of the original structure of the signal is shrunk. ♣

R-Code 7.6: A wavelet analysis of the doppler data. (See Figure 7.7.)

```

set.seed(14)
doppler <- function(x)  sqrt(x*(1-x))*sin(2.1*pi/(x+.05))
n <- 2^8
xi <- seq(0, 1, length=n)
yi <- doppler(xi)+rnorm(n, 0, sd=0.1)
plot(xi, yi, pch=20)
curve(doppler, add=T, n=1000, col=4)

library(wavethresh)
wd1 <- wd(yi)    # wd=wavelet decomp
summary(wd1)      # printing the object is a bit cryptic

## Levels: 8
## Length of original: 256
## Filter was: Daub cmpct on least asymm N=10
## Boundary handling: periodic
## Transform type: wavelet
## Date: Sun Aug 27 12:55:03 2023
# to access the individual coefficients beta_jk, use accessD(wd, level=j)
round(accessD(wd1, level=3), 4)
## [1] 0.2182 -0.0333 1.1923 1.7530 -0.9345 0.3094 -0.0168 -0.0187

```

```

# Plot the mother basis
draw(wd1)
# Coefficients...
tmp <- plot(wd1, main="", sub="", scaling="by.level")
# NOTE: "wd"=wavelet decomposition and "wr"=wavelet reconstruction.

plot(xi, yi, pch=20, main="Reconstruction with different levels")
curve(doppler, add=T, n=1000, col="gray")
levs <- c(1, 3, 5, 7)
# reconstruction with different levels
for (i in 1:4) {
  tmp <- threshold(wd1, levels=levs[i]:7, policy="manual", value=999)
  lines(wr(tmp)^xi, col=i+1)
}
legend("bottomright", legend=paste(levs, "Levels"), col=2:5, lty=1, bty="n")
# errors:
plot(xi, yi, type="n", main="Error of Reconstruction")
for (i in 1:4) {
  tmp <- threshold(wd1, levels=levs[i]:7, policy="manual", value=999)
  lines(doppler(xi)-wr(tmp)^xi, col=i+1, lwd=1)
}

# With soft and hard thresholding, the number of zeros
# are the same, only the magnitudes of coefficients vary a bit.
t1wdd <- threshold(wd1, policy="manual", value=.1, type="hard")
t2wdd <- threshold(wd1, policy="manual", value=.1, type="soft")
t3wdd <- threshold(wd1, policy="manual", value=.5, type="soft")
t4wdd <- threshold(wd1, policy="manual", value=1, type="soft")
legend("bottomright", , col=2:5, lty=1, bty="n",
       legend=c("hard, 0.1", "soft, 0.1", "soft, 0.5", "soft, 1.0"))
plot(xi, yi, pch=20, main="Reconstruction: different thresholding")
curve(doppler, add=T, n=1000, col="gray")
lines(wr(t1wdd)^xi, col=2, lwd=1)
lines(wr(t2wdd)^xi, col=3, lwd=1)
lines(wr(t3wdd)^xi, col=4, lwd=1)
lines(wr(t4wdd)^xi, col=5, lwd=1)
# We count the number of zeros:
zeros <- array(0,c(4,7+2))
for (i in 1:7) {
  zeros[1,i] <- sum(accessD(t1wdd, level=i)==0)
  zeros[2,i] <- sum(accessD(t2wdd, level=i)==0)
}

```

```

zeros[3,i] <- sum(accessD(t3wdd, level=i)==0)
zeros[4,i] <- sum(accessD(t4wdd, level=i)==0)
}

zeros[,8] <- apply(zeros[,1:7],1,sum) # total number of zeros
zeros[,9] <- 2^8 - zeros[,8]          # nonzero coefficients
# print(zeros) # to be compared with 256 coefficients!
# Notice that the last column of 'zeros' also corresponds to
# 'dof'=degrees of freedom.

plot(xi, yi, type="n", main="Error of Reconstruction")
abline(h=c(0.1, -0.1), col="gray") # one +/-sigma of the added error
lines(doppler(xi)-wr(t1wdd) ~ xi, col=2, lwd=1)
lines(doppler(xi)-wr(t2wdd) ~ xi, col=3, lwd=1)
lines(doppler(xi)-wr(t3wdd) ~ xi, col=4, lwd=1)
lines(doppler(xi)-wr(t4wdd) ~ xi, col=5, lwd=1)
legend("bottomright", legend=paste(zeros[,9]), col=2:5, lwd=1, bty="n")
# Note that the error is smaller than sigma in the artificial data example!

```

7.5 Example: Image Compression

Smoothing splines and wavelets are particularly suitable for extensions in two dimensions. Thin-plate splines are the two-dimensional analog of the cubic spline in one dimension. Here, we only illustrate the wavelet approach with an example. A full treatment of a two-dimensional wavelet would be beyond this lecture.

The top left panel of Figure 7.8 shows a RGB png color image of a chapel in Gspon ($46^{\circ}13'28.6''\text{N}$ $7^{\circ}54'10.5''\text{E}$). The image consists of $512^2 = 262144$ pixels, and we work with the ‘blue’ component, scaled to 0 and 255 (top right). The image has a few regions that are homogeneous (sky, fronts, …), many sharp boundaries and quite a few details (stairs, handrail, stonework, …)

R-Code 7.7 illustrates an image compression approach based on a two-dimensional wavelet approach, similar to that seen in R-Code 7.6. We start with the (uncompressed) decomposition with default arguments to specify the wavelets. The resulting object `imwdG` has $2^{18} = 512^2$ coefficients partitioned in 9 levels (each level has three components: horizontal, vertical, and diagonal). We choose a manual (hard-)thresholding. The parameter `value` of `threshold` has been chosen by trial and error to be `30` (automatic approaches exist, of course). The middle row of Figure 7.8 gives the error of reconstruction (different scale) and the reconstruction itself of the image based on roughly 11000 coefficients (about 96% of the coefficients are zero!). Ideally, smooth areas and fine scale features in the image should be preserved, which is the case here. The reconstruction consists of non-integer values. If we work with proper RGB grayscale images, we would have to round the reconstruction to the nearest integer. We have roughly 13%, 37%

and 56% percent of all pixels that whos reconstruction is zero, ± 1 and ± 2 grayscale units off, respectively.

The compression is straightforward, but the manual extraction of the coefficients is somewhat difficult. The coefficients are stored in many vectors and it seems (almost) cryptic. After a thresholding operation, vectors are stored in a compressed format. The lower left panel of Figure 7.8 shows all non-zero coefficients aligned in a matrix pattern. The lines indicate the horizontal, diagonal and vertical components (counter-clockwise). Most non-zero coefficients are positioned at “feature changes”, i.e., the boundaries and sharp edges are visible. The stairs appear in almost all levels. Note that it is possible to plot the coefficients of each level and component individually via `plot(imwdG, scaling="global", package="R", plot.type="cols")` but the interpretation for large images might not be as easy.

The thresholding also filters iid noise in an image. The last few lines of R code illustrate that the reconstruction remains virtually the same, even if substantial noise is added, e.g., `gspon+rnorm(512^2, sd=10)`. The corresponding reconstructed image is virtually the same as the noise-free one.

R-Code 7.7: Wavelet analysis of an image. (See Figure 7.8.)

```
library(wavethresh)
library(png)      # load a nice image

gsponRGB <- readPNG("download/sta121/Gspon512.png")
plot(0, xlim=c(1,512), ylim=c(1,512), type="n", axes=F, xlab="", ylab="",
     xaxs="i", yaxs="i")
rasterImage(gsponRGB, 1, 1, 512, 512)
graycol <- gray(0:127/128)                      # Colors: dark to white
gspon <- round(t(gsponRGB[512:1,,3])*255)    # Values between 0:255
zl <- c(-60,300)      # Avoid very dark and very light levels
image(gspon, col=graycol, zlim=zl, axes=F)
# actual wavelet analysis:
summary(imwdG <- imwd(gspon)) # default arguments work fine
## Uncompressed image wavelet decomposition structure
## Levels: 9
## Original image was 512 x 512 pixels.
## Filter was: Daub cmptc on least asymm N=10
## Boundary handling: periodic
# to see the used wavelet, one could visualize it through
# draw(imwdG, 2^9, border="gray")
thresh <- 30
timwdG <- threshold(imwdG, pol="manual", value=thresh)

# error of reconstruction (different scale) and reconstruction:
image(imwr(timwdG)-gspon, zlim=c(-100,100), col=graycol, axes=F)
```

```

summary(error <- c(imwr(timwdG)-gspont) ) # should be rounded!
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## -45.912 -2.138  0.005  0.000  2.113 52.384
image(imwr(timwdG), zlim=zl, col=graycol, axes=F)
sapply(0:4, function(x) mean(abs(round(error)) <= x) ) # proportion within
## [1] 0.13224 0.37445 0.56236 0.69272 0.77838
# The object imwdG has  $2^{18}=512^2$  coefficients:
3*(2^16+2^14+2^12+2^10+2^8+2^6+2^4+2^2+2^0)+1 #=512^2
## [1] 262144
# to access these coefficients, we run
tmp <- threshold(imwdG, policy="manual", value=thresh, compression=FALSE)
# and the values can also be accessed through:
c(length(timwdG$w8L1$val), length(timwdG$w8L2$val), length(timwdG$w8L3$val))
## [1] 1529 1380 207
levs <- plot(timwdG, scaling="global", package="S") # all coefs in a matrix
levs[ abs(levs) < 1 ] <- NA # due to thresholding, no small numbers
nnz <- sum(!is.na(levs)) # number of zeros.
print(c(nonzeros=nnz, percent=nnz/512^2 *100)) # significant compression
## nonzeros percent
## 10672.000 4.071
image(levs, axes=FALSE) # show non-zero values
# some lines:
for (i in 1:7) { # number of levels would be `nlevelsWT(timwdG)`
  lines(c(0,2^{-(i+1)}), c(2^{-(i)}, 2^{-(i)}))
  lines( c(2^{-(i)}, 2^{-(i)}), c(0,2^{-(i+1)}))
}
# reconstruct a noisy version of the image:
# image(gspont+rnorm(512^2, sd=10), zlim=zl, col=graycol, axes=FALSE)
imwdG2 <- imwd(gspont+rnorm(512^2, sd=10))
timwdG2 <- threshold(imwdG2, pol="manual", value=thresh)
image(imwr(timwdG2), zlim=zl, col=graycol, axes=FALSE)
summary(c(imwr(timwdG2)-imwr(timwdG)))
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## -55.985 -1.496 -0.008 -0.004  1.484 51.181

```

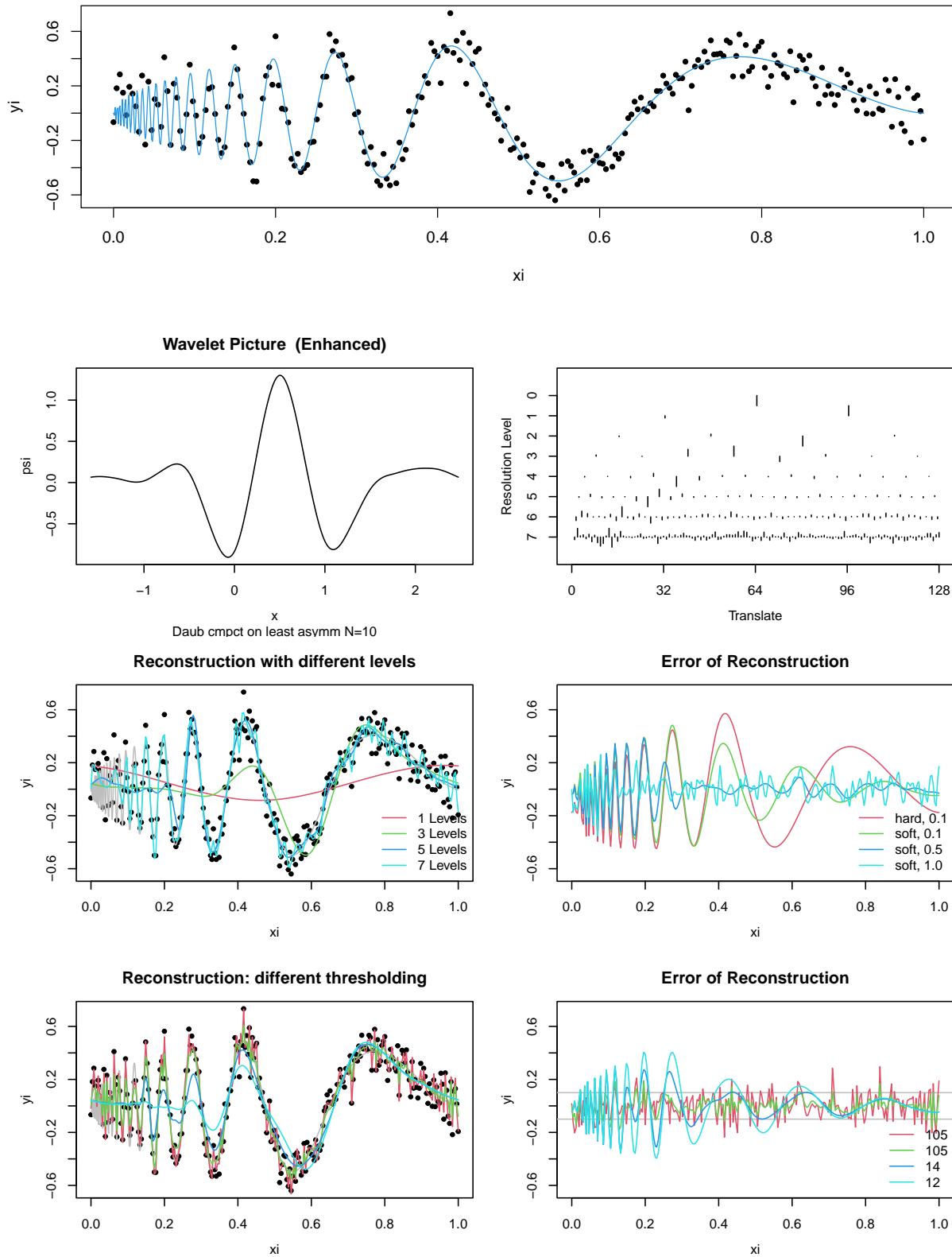


Figure 7.7: A wavelet analysis of the doppler data. Top row illustrates the function and the noisy observations thereof. Second row: wavelet function and coefficients for each level. Third row: reconstruction and error of reconstruction based on a few levels only. Bottom row: reconstruction and error of reconstruction based on a different thresholding type and level.



Figure 7.8: Top row: original images in color and in gray-scale. Middle row: error and reconstructed (compressed) image. Bottom row: wavelet coefficients and reconstructed (threshold) image from a noise version of the original image.

7.6 Bibliographic remarks

Green and Silverman (1994) is an early but accessible reference for non-parametric regression. Wahba (1990) is the classic book on splines. However, Wood (2006) is definitely more relevant here. The book by Fan and Gijbels (1996) is an accessible source for local polynomials. It also briefly outlines wavelets, as well as the other classical smoothing approaches. Similarly, the first few chapters of Hastie and Tibshirani (1990) give an accessible overview of smoothers. Later chapters then link smoothers to (generalized) additive models.

Nason (2010) is the book behind the `wavethresh` package.

Thin plate splines are nicely implemented in R by the function `Tps()` of the package `fields`.

7.7 Exercises and Problems

Problem 7.1 (Non-parametric regression) Gompertz's (1825) theory states that mortality rates of organisms increase at an exponential rate with increasing age. In other words, the probability of dying increases exponentially with the age of an organism. An experiment was setup to test this theory. It considered a start population of 1'203'646 Mediterranean fruit flies and the number of flies found dead each day was counted. The recorded data contains the number of surviving flies for each day.

One variable of interest is "mortality rate" defined as the number of flies dying on a given day divided by the number of flies alive at the beginning of that day.

The dataset `medflies.txt`, available from <https://www.math.uzh.ch/furrer/download/sta121/medflies.txt>, consist of:

<code>day</code> :	Day of the measurement
<code>living</code> :	Number of flies alive at the beginning of the day
<code>mort.rate</code> :	Mortality rate for the flies for each day

- a) Explore the data with tables and plots and describe it with words.
- b) Was the variable `mort.rate` constructed correctly?
- c) Verify Gompertz's theory that mortality rates increase at an exponential rate as age increases. Does it hold for this data? Can you find a subset of the data where the theory holds?
Hint: Use linear regressions to model `log(mort.rate)` and plot the regression together with the data. Verify the model assumptions with diagnostic plots.
- d) Consider `mort.rate` (not on log scale) for the rest of the exercise. Fit a non-parametric regression using a kernel approach. Try different kernels and bandwidths. Plot the data and the resulting fits.
- e) Redo d with (i) the smoothing splines and (ii) the local polynomials method.
- f) Use cross-validation to find an optimal bandwidth for the kernel approach in d. Plot the resulting fit.
Hint: Use functions `hcv()` and `sm.regression()` from package `sm`. See script.

- g) Calculate “manually” a cross-validation based on smoothing splines.
- h) What scientific questions can be answered with a non-parametric model? Find a question that can be answered with a non-parametric model, and another one that can be answered with a linear model.

Problem 7.2 (Kernel smoothing) Revisit the *bicoal* dataset.

- a) Use `ksmooth()` and `loess()` and construct sensible smooths of the data.
- b) The function `hcv()` from the package `sm` is known to produce sometimes unstable results. Compare the results of `sm.regression()` with bandwidth chosen by `hcv()` and `h.select()`. What do you conclude?
- c) Implement an CV approach for your favorite smoother and compare the fit with other smooths.

Chapter 8

Generalized Linear Models

Often the outcome cannot be modeled as a Gaussian realization and extensions of the linear model are required. Logistic regression explains failures and successes (binary outcomes) using one or more explanatory variables. This model is part of an extension of linear models, called generalized linear models, that are for non-Gaussian distributions.

Learning goals for this chapter:

- ◊ Apply logistic regression techniques to datasets with a binary response variable, interpret the output
- ◊ Apply Poisson regression techniques to datasets with a count response variable, interpret the output
- ◊ Understand the basics of fitting and inference for nonlinear regression methods when the regression function acting on the predictors is not linear in the parameters
- ◊ Explain the difference between logistic and linear regression
- ◊ Defines Generalized linear model, exponential family of distributions, checking if a distribution belongs to the exponential family of distributions
- ◊ understand some similarities and differences between LDA and logistic regression

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter10.R.

8.1 Motivation

Within the context of parametric models, we have mainly seen linear models $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ where we often assumed $\varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$ (see, Chapter 5 for details and notation). The following classical and famous dataset illustrates the issue with a least squares approach where we cannot assume a Gaussian response.

Example 8.1. (*orings* data) In January 1986, the space shuttle ‘Challenger’ exploded shortly after take-off from Cape Canaveral, Florida, US. Part of the problem was with the so-called O-ring seals of the solid rocket boosters. In total there are six (primary) O-rings in the two boosters. In several shuttle starts prior to 1986, some of the rings showed failures (the rocket hulls were typically recovered and examined).

The functioning of rubber seals (naturally) depends on temperature. The dataset `data("orings, package="faraway")` gives the number of defects from 23 previous shuttle starts and is shown in Figure 8.1. The question here is if we can predict a failure probability as a function of the temperature. Notice that at launch time of this last Challenger mission the ambient temperature was 31°F (i.e., just below freezing point). ♣

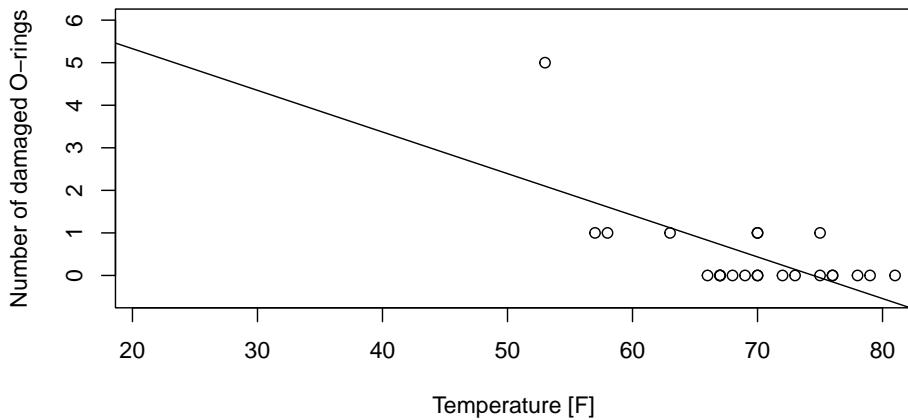


Figure 8.1: *orings* data and a linear fit.

A linear fit has been added to Figure 8.1 and directly illustrates the lack of adequacy of linear models for this data. In the figure we consider the data as count data. The response is then definitely not Gaussian and the linear model should not be used. Alternatively, we can model the failure probability, by normalizing the observed data by the total number of O-rings. When modeling probabilities, that is, $y_i \in [0, 1]$, a linear model cannot guarantee that $\hat{y}_i \in [0, 1]$.

In this chapter we extend the linear model towards data that is non-Gaussian, as - for example - count data or a proportion

8.2 Logistic Regression

To model probabilities as a function of one predictor, for example the probability of a defect as a function of the temperature, we start with

$$p = P\{\text{defect}\} = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)} = \frac{1}{1 + \exp(-\beta_0 - \beta_1 x)}, \quad (8.1)$$

where x is the temperature. The advantage of using the function (8.1) to model a probability is the guarantee that $p \in [0, 1]$ whatever value of x . Through inverting the relation, we have a linear model for the log-odds

$$g(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x. \quad (8.2)$$

In general, the function $g(\cdot)$ is called the *link function*. Here, $g^{-1}(\cdot)$ is the logistic function and thus a regression problem based on this function is called logistic regression.

8.2.1 Estimation, Inference and Prediction

When we apply a modeling approach as outlined, it is important to realize that we do not attempt to fit a non-linear regression, nor transform the data, but use a general methodology. We observe binomial variable $Y_i \sim \text{Bin}(n_i, p_i)$ and estimate the probabilities \hat{p}_i based on the observed data y_i/n_i . We implicitly use the fact that $E(Y_i)/n_i = p_i$. The transformed probabilities are linked to one or several predictors using the so-called logit function

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \mathbf{x}_i^\top \boldsymbol{\beta}. \quad (8.3)$$

As soon as we have an estimate for $\boldsymbol{\beta}$, we also have estimates for p_i and other quantities that can be derived thereof.

Equation (8.3) illustrates the most prominent difference with respect to the classical (linear) model. We do not specify additive errors but only the distribution of the observations (here Binomial) and the predictors are used additively for transformed parameters of this distribution.

As a consequence, we do not minimize the squared residuals. For the logistic regression case — even if we would use a least squares approach — no closed form expressions for the estimates exist and an iterative approach is used. More precisely, the observations are transformed and regressed on \mathbf{x}_i with appropriate weights (IRWLS, iteratively reweighted least squares). Another consequence is that we cannot derive exact distributions of estimators and therefore confidence intervals are typically not exact.

The estimation (or fitting) is done with the R function `glm()`, which takes similar arguments as the function `lm()`. The result of `glm()` bears quite some similarities with results of `lm()`. R Code 8.1 illustrates a fit of the O-ring data using `glm()`.

To construct confidence intervals for the parameters, we can use the function `confint()` from `MASS` or manually construct Wald-type confidence intervals by taking estimate plus/minus quantile times standard error. The former is slightly more precise, as the latter is, by construction, symmetric. Confidence intervals will be further illustrated in Example 8.2.

R-Code 8.1: *orings* data and estimated failure probability as a function of the ambient air temperature at launch time. (See Figure 8.2.)

```

data(orings, package="faraway")
glm1 <- glm(cbind(damage, 6 - damage) ~ temp, family=binomial, data=orings)
summary(glm1)

##
## Call:
## glm(formula = cbind(damage, 6 - damage) ~ temp, family = binomial,
##      data = orings)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.9529 -0.7345 -0.4393 -0.2079  1.9565
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 11.66299   3.29626   3.538 0.000403 ***
## temp        -0.21623   0.05318  -4.066 4.78e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 38.898 on 22 degrees of freedom
## Residual deviance: 16.912 on 21 degrees of freedom
## AIC: 33.675
##
## Number of Fisher Scoring iterations: 6
plot(damage/6 ~ temp, xlim=c(21, 80), ylim=c(0, 1),
      xlab="Temperature [F]", ylab="Probability of damage", data=orings)
points(orings$temp, glm1$fitted, col=2)
#
ct <- seq(20, to=85, length=100)
p.out <- predict(glm1, new=data.frame(temp=ct), type="response")
lines(ct, p.out)
abline(v=31, col="gray", lty=2)
unlist(predict(glm1, new=data.frame(temp=31), type="response", se.fit=T))
##
##          fit.1      se.fit.1 residual.scale
##          0.99303415   0.01153332   1.00000000

```

Prediction can be performed on the “linear model” level (8.3) or on the level of the response.

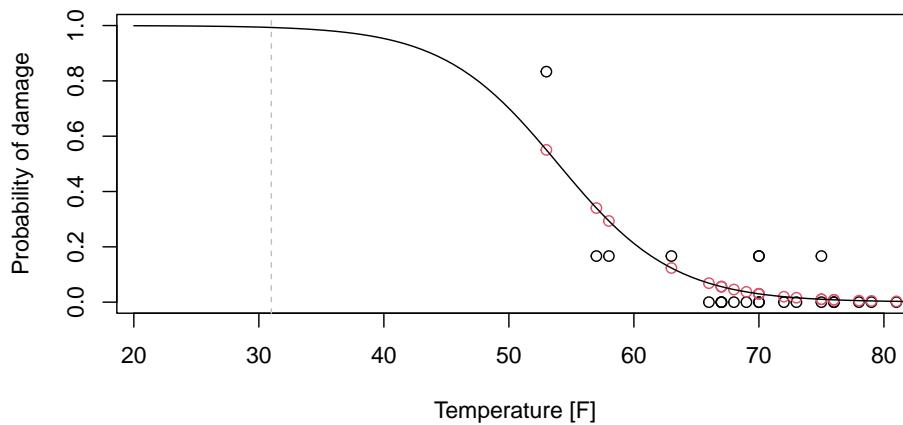


Figure 8.2: *orings* data and estimated failure probability as a function of the ambient air temperature at launch time. (See R-Code 8.1.)

The former is (essentially) $\mathbf{x}_0^\top \hat{\boldsymbol{\beta}}$ and the latter is \hat{p}_0 . Even on the response level, the prediction does not exactly yield \hat{Y} . In the case of logistic regression, the prediction is in $(0, 1)$, whereas the observations are integers. We do not predict the outcomes, but rather the outcome probabilities. In the case of the O-ring example, the probability that *all* rings fail is larger than 95% (`dbinom(6, size=6, prob=.993)`).

Example 8.2. The dataset `aflatoxin` from the package `faraway` contains data on the doses (in ppb) of Aflatoxin B1 fed to lab animals and the number responding with liver cancer. R-Code 8.2 illustrates model estimation, building confidence intervals and illustrates prediction.

From the functional relationship of the logistic function, we can determine the dose for which we expect death with probability 0.5. Often termed, LD50 (*lethal dose*). In our example, it is roughly 33.7 ± 7.5 . ♣

R-Code 8.2: Comparison of calculation of confidence intervals and prediction. (See Figure 8.3.)

```
library(faraway)
data(aflatoxin)
glm2 <- glm(cbind(tumor, total-tumor) ~ dose,
            family=binomial, data=aflatoxin)
print(cmat <- summary(glm2))

##
## Call:
## glm(formula = cbind(tumor, total - tumor) ~ dose, family = binomial,
##      data = aflatoxin)
```

```

## 
## Deviance Residuals:
##      1       2       3       4       5       6 
## -1.2995  0.7959 -0.4814  0.4174 -0.1629  0.3774 
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -3.03604   0.48226 -6.295 3.07e-10 ***
## dose        0.09009   0.01456  6.189 6.04e-10 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 116.524 on 5 degrees of freedom 
## Residual deviance: 2.897 on 4 degrees of freedom 
## AIC: 17.685 
## 
## Number of Fisher Scoring iterations: 5 

library(MASS) # for function confint()
confint(glm2)

## Waiting for profiling to be done...
##           2.5 %     97.5 % 
## (Intercept) -4.10594724 -2.1868361 
## dose        0.06440816  0.1217981 

cmat$coef[,1] + cmat$coef[,2] %*% t(qnorm(c(.025,.975)))
##          [,1]      [,2] 
## [1,] -3.9812477 -2.0908252 
## [2,]  0.0615603  0.1186173 

dose <- 0:100
pr <- predict(glm2, new=data.frame(dose=dose), type="response", se.fit=T)
### instead of pr$fit+qnorm(c(.025,.975))*pr$se.fit) sophisticated version:
plot(tumor/total ~ dose, data=aflatoxin, xlim=c(0,100))
polygon(c(0:100, 100:0), cbind(pmin(1, pr$fit+qnorm(.975)*pr$se.fit),
  rev(pmax(0, pr$fit+qnorm(.025)*pr$se.fit))), col=rgb(.9,.9,.9), border=NA)
polygon(c(0:100, 100:0), cbind(pmin(1, pr$fit+qnorm(.75)*pr$se.fit),
  rev(pmax(0, pr$fit+qnorm(.25)*pr$se.fit))), col=rgb(.7,.7,.7), border=NA)
points(tumor/total ~ dose, data=aflatoxin)
lines(dose, pr$fit)

print(LD50 <- dose.p(glm2, p=0.5)) # lethal dose LD50

```

```
##           Dose      SE
## p = 0.5: 33.7005 3.800093
abline(h=0.5, lty=4)
abline(v=LD50 + qnorm(c(.025,.975))*c(attr(LD50, "SE")) )
```

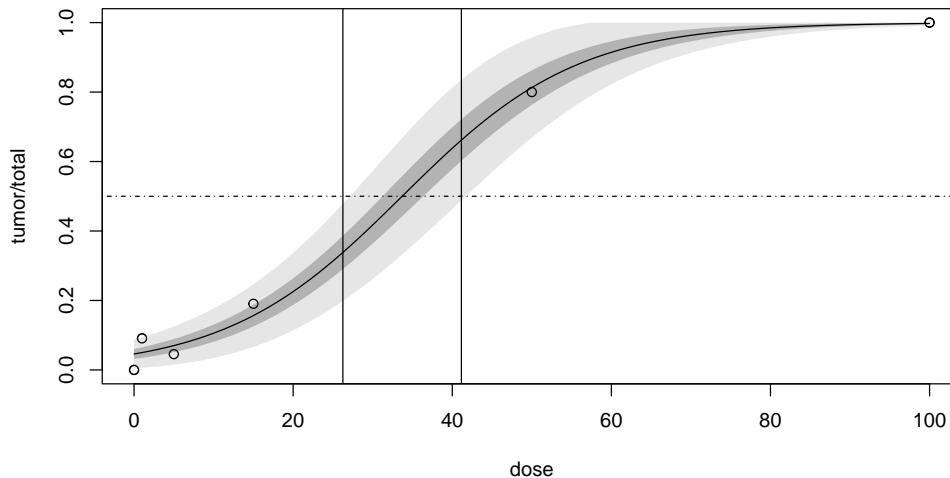


Figure 8.3: Predictive distribution and prediction intervals based on a Wald-type approximation. (See R-Code 8.2.)

In practice, there might be cases where the data exhibits a larger variability compared to the binomial variance. With `family=quasibinomial`, it is possible to model the over-dispersion.

The logistic regression is a special case of a generalized linear model (GLM) which we briefly introduce in Section 8.4.

8.2.2 Deviances

The deviance is a quantity describing the quality of a fit and is a generalization of the classical sum of squared residuals (SSR). More specifically, instead of SSR, the difference between the log-likelihoods of two models is used. In general, the larger the deviance, the poorer the fit to the data.

The benchmark model is the so-called *saturated model* which is essentially a perfect fit. The residual deviance is (essentially) the difference between the *saturated model* and the fitted model. Another benchmark is the so-called *null model* which corresponds to the null hypothesis and for the simplest cases consists of only one parameter (β_0). The null deviance is (essentially) the difference between the *saturated model* and the null model.

Model deviances are not usually interpreted directly, but rather compared with each other. The difference between deviances are approximately distributed according to a χ^2 distribution with degrees of freedom corresponding to the difference in the number of estimated parameters. Large values of the χ^2 statistic provide evidence against the null hypothesis.

R-Code 8.3 A GLM with *handwash* data.

```

handwash <- read.csv("download/sta121/washroom.csv")
print(handwash)

##   Gender BkPk Others WashYes WashNo
## 1   Male   no   alone     27     26
## 2   Male   no   others      5      5
## 3   Male   yes  alone    13      8
## 4   Male   yes others     3      2
## 5 Female  no   alone     4     28
## 6 Female  no   others     0      5
## 7 Female  yes  alone     7      7
## 8 Female  yes others     4      6

summary(glmhw <- glm(cbind(WashYes, WashNo) ~ Gender + BkPk + Others,
                      family=binomial, data=handwash) )

##
## Call:
## glm(formula = cbind(WashYes, WashNo) ~ Gender + BkPk + Others,
##       family = binomial, data = handwash)
##
## Deviance Residuals:
##      1       2       3       4       5       6       7       8
## 0.5007  0.3784 -0.9366 -0.3991 -0.7920 -1.3055  0.9827  0.4027
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.5432    0.3730 -4.138 3.51e-05 ***
## GenderMale   1.4433    0.3880  3.720 0.000199 ***
## BkPkyes     1.0149    0.3914  2.593 0.009521 **
## Othersothers -0.1398    0.4500 -0.311 0.756116
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 25.1480  on 7  degrees of freedom
## Residual deviance: 4.8901  on 4  degrees of freedom
## AIC: 34.739
##
## Number of Fisher Scoring iterations: 4

```

Example 8.3. Abraham and Ledolter (2006) describe the story behind the `handwash` dataset as follows: “During the fall semester of 2001, a group of undergraduate students in the introductory statistics course collected data on hand washing after using the restroom. Students took observations at several locations around campus. They developed ingenious ways of ‘hiding’ in bathrooms so that students being observed were not aware of their presence.” A short analysis is given in R-Code 8.3.

The data does not reveal any (significant) indication that presence of others effects hands are washed after using the restroom. ♣

8.2.3 Logistic Regression and Discrimination/Classification

The logistic regression can be used to classify a set of observations into two (or possibly more) groups. For example, we predict the probabilities of belonging to a certain class given covariates. For two groups, we may classify the observation in one groups if the probability is below 1/2 and vice versa.

For example for the `aflotoxin` example, the value 33.7 (i.e., `LD50`) can be used as a critical value to separate the classes non-deadly and deadly.

In case of perfect discrimination, `glm()` raises warnings as seen with the example:

```
y <- rep(c(0,1), each=5); glm(y rnorm(10, y, sd=.1), family="binomial");
```

In other words, ‘perfect fits’ cause warnings.

8.3 Poisson Regression

When observing Poisson-type count data (counts of rare events) a classical LM model is not adequate and we have to extend our approach analogously. (We cannot rely on the Poisson approximation of a binomial distribution.) We set $Y_i \sim \mathcal{P}(\lambda_i)$, independent, with $\lambda_i > 0$. To ensure this constraint we set

$$\log(\lambda_i) = g(\lambda_i) = \mathbf{x}_i^\top \boldsymbol{\beta}, \quad (8.4)$$

for each predictor \mathbf{x} . Starting from the density of a Poisson random variable it would again be possible to show that $\boldsymbol{\beta}$ can be estimated using an IRWLS approach. The derivation would, however, take several lines. The subsequent steps are similar to the Binomial setting and are illustrated in the following example and associated R-Code.

Example 8.4. The `elephant` data collected by Poole (1989) contains the number of successful matings between the observed elephants over a span of 8 years. Elephant ages were recorded at the beginning of the study.

R Code 8.4 gives a short analysis. We see that $\hat{\beta}_{AGE} = 0.0687$ and interpret this to mean that as age increases by one year, the probability of mating increases by $100 \cdot (\exp(0.0687) - 1) = 7.1\%$ with a 95% confidence interval of [4%, 10%].

In the second part of R Code 8.4, we compare several models, including: the above-fitted model, the “null” model containing only an intercept, and the above-fitted model with an extra covariate. The `anova()` output reassures us that `AGE` is relevant and also sufficient as a linear predictor. ♣

R-Code 8.4: Modeling the `elephant` data. (See Figure 8.4.)

```

elephant <- read.csv("download/sta121/elephant.csv")
head(elephant, 3)
##   Age Matings
## 1 27      0
## 2 28      1
## 3 28      1

eleglm <- glm(Matings ~ Age, family=poisson, data=elephant)
summary(eleglm)

##
## Call:
## glm(formula = Matings ~ Age, family = poisson, data = elephant)
##
## Deviance Residuals:
##       Min        1Q     Median        3Q       Max
## -2.80798 -0.86137 -0.08629  0.60087  2.17777
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.58201   0.54462 -2.905  0.00368 **
## Age         0.06869   0.01375  4.997 5.81e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 75.372 on 40 degrees of freedom
## Residual deviance: 51.012 on 39 degrees of freedom
## AIC: 156.46
##
## Number of Fisher Scoring iterations: 5
100*(exp(confint(eleglm, 2))-1) # "2" for "Age"
## Waiting for profiling to be done...
##      2.5 %    97.5 %
## 4.255848 10.036024

plot(elephant, ylim=c(-.5, 9.5))
newdata <- data.frame(Age=seq(25, to=55, length=100))
elepre <- predict(eleglm, newdata=newdata, type="response", se.fit=TRUE)
lines(newdata$Age, elepre$fit)
lines(newdata$Age, elepre$fit + elepre$se.fit*qnorm(.1), col=3)

```

```

lines(newdata$Age, elepre$fit + elepre$se.fit*qnorm(.9), col=3)
abline(h=0, col=4)

### Now we perform model selection:
eleglm0 <- glm(Matings ~ 1, family=poisson, data=elephant)
eleglm2 <- glm(Matings ~ Age + I(Age*Age), family=poisson, data=elephant)
anova(eleglm0, eleglm, eleglm2, test="Chisq")

## Analysis of Deviance Table
##
## Model 1: Matings ~ 1
## Model 2: Matings ~ Age
## Model 3: Matings ~ Age + I(Age * Age)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       40    75.372
## 2       39    51.012  1   24.3601 7.991e-07 ***
## 3       38    50.826  1    0.1854     0.6667
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

confint(eleglm2)

## Waiting for profiling to be done...
##                   2.5 %      97.5 %
## (Intercept) -9.044944422 2.893178033
## Age         -0.163750206 0.457733552
## I(Age * Age) -0.004967908 0.002948358

summary(eleglm2)$coef
##                   Estimate Std. Error     z value Pr(>|z|)
## (Intercept) -2.857405968 3.03563826 -0.9412867 0.3465579
## Age          0.135954442 0.15800953  0.8604193 0.3895580
## I(Age * Age) -0.000859507 0.00201239 -0.4271075 0.6693010

```

Poisson regression is commonly used for count data. It can be used to answer how the number of some observable depends on some other variable or to find relationships between rates and variables. The outcome then can be a count y or a rate y/t , where t can be a time or space interval, or some other grouping. For modeling rates, Equation (8.5) changes to

$$g(\mu/t) = \log(\mu/t) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \quad (8.5)$$

$$g(\mu) = \log(\mu) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \log(t). \quad (8.6)$$

The mean count is now proportional to t . This additional term is known as an offset and can be used to make groups more comparable.

One assumption of the Poisson model is that the mean and variance are equal. In practice, this, however, is not the case and we need to further inflate the variance (family `quasipoisson`).

8.4 Generalized Linear Models

The last few sections illustrated the extension of linear models to binomial data and Poisson count data. Natural questions are: For which other distributions can we apply a similar framework? Is it possible to put all these distributions under a common framework? These two questions are discussed in this section and run under *generalized linear models*.

In a nutshell, a generalized linear model (GLM) extends the linear model (LM) framework to a large class of distribution families in which the (transformation of the) expected value of the observation is modeled with a linear model.

A GLM consists of three components:

1. Distribution of Y_i (independent conditional on the predictors)
2. A linear function of predictors $\mathbf{x}_i^\top \boldsymbol{\beta}$
3. A function linking $E(Y_i) = \mu_i$ and the predictors: $g(\mu_i) = \mathbf{x}_i^\top \boldsymbol{\beta}$

Example 8.5. For both the logistic and Poisson regression, Y_i were independent and the link functions were the logit function (8.3) and the logarithm. ♣

The distribution of Y_i cannot be taken arbitrarily. The theory of GLM can be derived under a umbrella of so-called exponential family distribution. This distribution is rather a broad class of distributions and includes: binomial, Poisson, inverse-Gaussian, Gamma and Gaussian distributions. The density of these distributions can be written in the form

$$f_Y(y; \theta) = \exp\left(\frac{\theta y - b(\theta)}{\phi} + c(y, \phi)\right), \quad (8.7)$$

where θ is the parameter (of interest), and $b(\cdot)$, $c(\cdot, \cdot)$ two suitable functions. The parameter ϕ is typically modeling dispersion (variance). For example, the pmf of the Poisson distribution

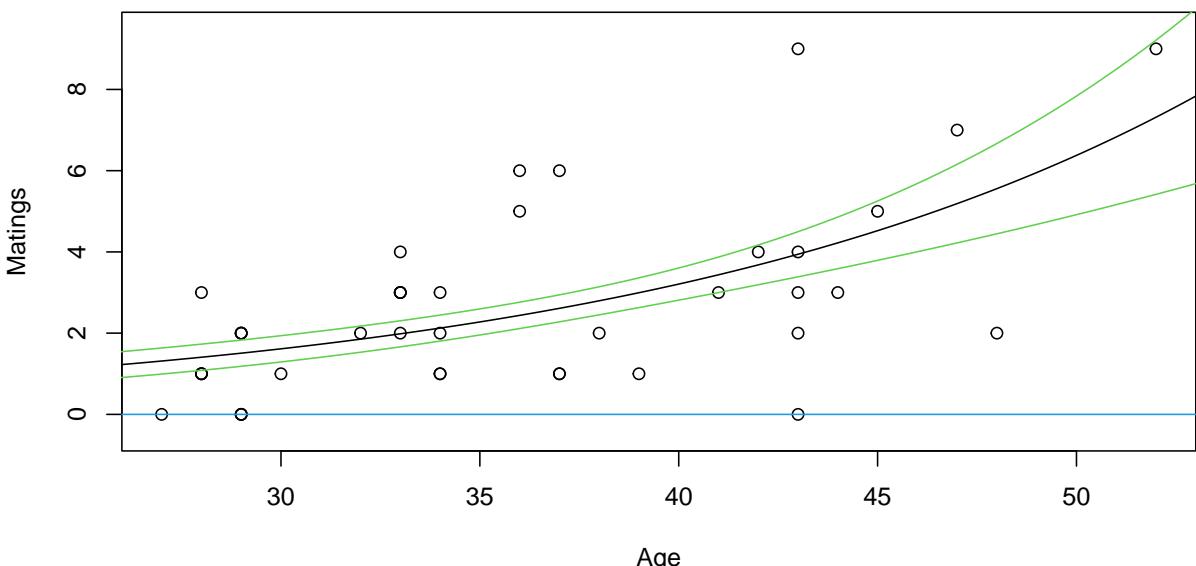


Figure 8.4: Modeling the `elephant` data using Poisson regression. (See R-Code 8.4.)

is written as $f_Y(y; \lambda) = \exp(-\lambda)\lambda^y/y! = \exp((\log(\lambda)y - \lambda)/1 - \log(y!))$, i.e., $\theta = \log(\lambda)$, $b(\theta) = \exp(\theta) = \lambda$, $\phi = 1$ and $c(y, \phi) = -\log(y!)$. Because of the particular form of the density of the exponential family, the estimation process can be unified through the IRWLS algorithm.

Table 8.1 gives a summary of the most important members of the exponential family. For most of the one parameter families, an extension exists ([quasibinomial](#) or [quasipoisson](#)). The advantage of using the framework of the exponential family is its broad use. In the case of the Gaussian distribution, the LM is a particular case of the GLM. See R-Code 8.5.

R-Code 8.5 Comparison of the GLM with the LM. (see R Code 5.2.)

```

lcs.glm <- glm(sr ~ pop15 + pop75 + dpi + ddpi, data=LifeCycleSavings)
summary(lcs.glm)

##
## Call:
## glm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -8.2422 -2.6857 -0.2488  2.4280  9.7509
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.5660865  7.3545161   3.884 0.000334 ***
## pop15       -0.4611931  0.1446422  -3.189 0.002603 **
## pop75       -1.6914977  1.0835989  -1.561 0.125530
## dpi        -0.0003369  0.0009311  -0.362 0.719173
## ddpi        0.4096949  0.1961971   2.088 0.042471 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 14.46029)
##
## Null deviance: 983.63  on 49  degrees of freedom
## Residual deviance: 650.71  on 45  degrees of freedom
## AIC: 282.2
##
## Number of Fisher Scoring iterations: 2

```

Different types of residuals exist. To extract these from a fitted model, the corresponding type needs to be specified. For example, in `residuals()`, the argument `type=c("deviance", "pearson", "working", "response", "partial")` can be used, where default is '`deviance`'. In the linear model setting, `deviance`, `pearson`, `working`, `response` coincide.

Table 8.1: Some members of the exponential family and the natural link function.

Distribution	Link function
Gaussian	Identity: $g(\mu) = \mu$
Binomial	Logit: $g(\mu) = \log \frac{\mu}{1-\mu}$
Poisson	Log: $g(\mu) = \log \mu$
Gamma	Inverse: $g(\mu) = 1/\mu$

Other link functions than those shown in Table 8.1 can be chosen. For the binomial setting, another commonly used one is the so-called *probit()* function, defined as the inverse of the cumulative distribution function of a standard normal distribution.

8.5 Bibliographic remarks

Dalal *et al.* (1989) gives a detailed description of the background of the Challenger accident in a statistical context. The webpage http://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster is very enlightening as well.

Faraway (2006) covers GLMs with a focus on R applications in a very accessible fashion. McCullagh and Nelder (1983) continues to be a valuable reference. Green and Silverman (1994) also contains several chapters about GLMs.

Dobson (2001) is a more theoretical but very accessible text. Many books have at least some chapter about GLMs, notable books include Fahrmeir *et al.* (2013), Hastie and Tibshirani (1990), and Wood (2006).

In case of ordinal data, we have to extend the model to so-called cumulative link models (CLMs). In R, the package `ordinal` handles CLMs in the usual way.

8.6 Exercises and Problems

Problem 8.1 (Logistic regression) Bliss (1935) analysed the numbers of insects dying at different levels of insecticide concentration. The `bliss` dataset is available in the R package `faraway`. See also `?bliss`.

- a) Look at the data. Plot the ratio of dead and total insects for all concentration levels.
- b) Fit a logistic regression using the *logit* link function. Interpret the results.
- c) Calculate the predicted values at the observed concentration levels using the model coefficients (`coef()`). Verify your results with `fitted()`.
- d) Add the line of the predicted probabilities to the plot from a). Add a 95% confidence interval of the predicted values to the plot.
Hint: See `?predict.glm()`.
- e) Redo the last five questions using the *probit* link function (another possible link-function for binomial data).

- f) Compare the *logit* and *probit* fits by plotting the ratio of the predicted values from the *logit* and the *probit* model on the interval $[-1, 5]$. Where do the predicted values agree?
- g) Compute the LD50 for the logit model. LD stands for *lethal dose*, and LD50 is the dose at which 50% of the insects die.

Problem 8.2 (Exponential family) Distributions that can be written as

$$f(y; \theta, \phi) = \exp\left(\frac{y\theta - b(\theta)}{\phi} + c(y, \phi)\right)$$

belong to the exponential family. The methodology of generalised linear models is developed for distributions of the exponential family.

Determine which of the following distributions belong to the exponential family. If so, determine θ , ϕ , $b(\theta)$, and $c(y, \phi)$.

- a) Exponential distribution: $f(y; \lambda) = \lambda e^{-\lambda y}$.
- b) Binomial distribution: $f(y; \pi) = \binom{n}{y} \pi^y (1 - \pi)^{n-y}$, n known.
- c) Uniform distribution $f(y; \theta) = 1/\theta$, $0 < y < \theta$.
- d) Normal distribution with known variance σ^2 .

Chapter 9

Survival Analysis

In the analysis of survival or time-to-event data we model observations representing the time until the occurrence of a certain event. Typically, we do not have full knowledge about all the times and this partial information, called censoring, has to be taken into account.

Learning goals for this chapter:

- ◊ Describe survival data, and the roles played by censoring, and survival and hazard functions
- ◊ Graph survival data, and the Kaplan–Meier curve in R
- ◊ Apply Log-rank test to compare treatment groups, interpret the results
- ◊ Use Cox proportional hazards to model given data set, interpret the results
- ◊ Check the proportional hazard assumption, and compute the hazard ratio in R
- ◊ Apply survival analysis models to the given data, interpret the results

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter09.R.

Survival data (also called time-to-event data) describes the duration from a starting event until an end event. The analysis of such data occurs in many situations and many different fields. Some examples of this kind of data are:

- Time from first diagnosis of HIV until death
- Time between first use of a new computer until it breaks for the first time
- Lifetime of a hippopotamus

- Time until a volcano erupts

Survival data is inherently positive; thus we have to limit possible probability models for survival models to positive random variables. Examples include the exponential distribution, the Gamma distribution, and the Weibull distribution that we will discuss in detail in this chapter. A second main characteristic of survival data is the fact that we may have incomplete information about the subjects or cases. This implies that we need to use models and estimation procedures capable of exploiting partial information.

9.1 Censoring

A typical (biostatistical) study is designed to recruit patients or individuals within a certain time frame and to follow them until the end of the study. In many cases, the end of the study occurs before every individual in the data set has died or experienced the event of interest. Those individuals that are still alive and observed at the end of the observation time are called censored. Sometimes patients move or withdraw their consent, both denoted as *lost to follow-up*, and are considered another instance of censoring. All of these situations are called “right censoring”, the most common censoring mechanism in survival data. Therefore we focus on this censoring type here.

An alternative but similar type is *left censoring*, exemplified by when a disease has been diagnosed but the exact onset of the disease is unknown. A third type is *interval censoring*, which typically occurs when the recurrence of a tumor is diagnosed between two consecutive follow-up visits.

Here and in many practical applications we assume that the censoring is independent (non-informative) of the survival time or any other relevant information of the individual/case.

Example 9.1. Figure 9.1 shows example survival data for six individuals that have been recruited between time $t = 0$ and time $t = 7$. Four individuals have died while under observation, indicated by black squares at their observed times of death (Patients 1, 2, 5 and 6). Patient 3 was lost to follow-up (moved or withdrew the consent) and Patient 4 was still alive at the end of the study ($t = 20$).

The figure shows the survival time in relation to the study time and as absolute survival time. For our analysis, the end of recruitment does not play a role and depending on the setting it may coincide with the start of the study. The age of the individuals (or other covariates) are not indicated in the plot.



9.2 Survivor and Hazard Function

Before we analyze survival data, we have to look at some particular functions “describing” the survival time. Let T be a non-negative random variable which represents the individual lifetimes or the time until event. As in previous chapters, we denote the density function (pdf) of the survival time as $f_T(t)$ and the cdf as $F_T(t) = P(T \leq t)$.

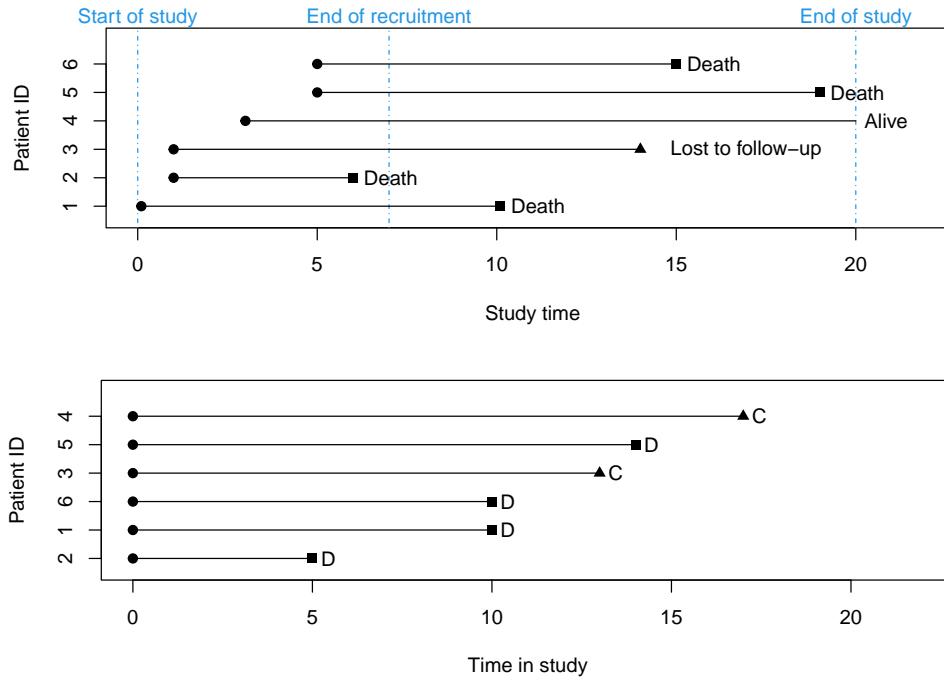


Figure 9.1: Observed survival times with censoring. Top panel shows the individual during the study time. Lower panel shows the absolute survival times, D: death, C: censoring.

Definition 9.1. For a survival time T with pdf $f_T(t)$ and cdf $F_T(t)$ we define the following functions.

1. The survivor function

$$S(t) = \Pr(T \geq t). \quad (9.1)$$

2. The hazard function or hazard rate

$$h(t) = \lim_{\Delta t \searrow 0} \frac{1}{\Delta t} P\{t \leq T < t + \Delta t \mid T \geq t\}. \quad (9.2)$$

3. The cumulative hazard function

$$H(t) = \int_0^t h(u) du. \quad (9.3)$$

The survivor function represents the probability that an individual survives any specified time point. The hazard function can be considered the “infinitesimal rate of death in a very short time” or instantaneous rate of failure or the force of mortality. The cumulative hazard function is the cumulative risk of death up to any specified time given that death has not occurred until then. Note that the cumulative hazard is not a cdf and the function can exceed one.

It is often necessary to switch between these functions and this can be done using several relationships. For the survivor function, we naturally have $S(t) = 1 - F(t)$. Second, starting

from the definition of the hazard function (9.2) and of conditional probabilities:

$$h(t) = \lim_{\Delta t \searrow 0} \frac{1}{\Delta t} \frac{\mathbb{P}(\{t \leq T < t + \Delta t\} \cap \{T \geq t\})}{\mathbb{P}(T \geq t)} \quad (9.4)$$

$$= \lim_{\Delta t \searrow 0} \frac{1}{\Delta t} \frac{\mathbb{P}(t \leq T < t + \Delta t)}{\mathbb{P}(T \geq t)} = \lim_{\Delta t \searrow 0} \frac{(F(t + \Delta t) - F(t))/\Delta t}{\mathbb{P}(T \geq t)} = \frac{f(t)}{S(t)}. \quad (9.5)$$

Finally, one can easily move between the survivor function and cumulative hazard function using the relation:

$$H(t) = \int_0^t h(u) du = \int_0^t \frac{f(u)}{1 - F(u)} du \quad (9.6)$$

$$= -\log(1 - F(u)) \Big|_0^t = -\log(1 - F(t)) = -\log\{S(t)\}. \quad (9.7)$$

Using the definition of conditional probabilities and the concept of the survival function, we also have:

$$\mathbb{P}\{t_1 \leq T \leq t_2 \mid T \geq t_1\} = \frac{S(t_1) - S(t_2)}{S(t_1)}. \quad (9.8)$$

The relationships to move between $H(t)$, $h(t)$ and $S(t)$ (and thus $f_T(t)$ and $F_T(t)$) are quite central and thus captured here as a proposition.

Property 9.1. *For a continuous survival time T , we have for $t \geq 0$:*

$$1. \ h(t) = \frac{f(t)}{S(t)},$$

$$2. \ S(t) = \exp(-H(t)).$$

We now look at two particularly well-suited survival time random variables.

Example 9.2. (Exponential distribution) Let $T \sim \text{Exp}(\lambda)$. Thus $f_T(t) = \lambda \exp(-\lambda t)$ and $F_T(t) = 1 - \exp(-\lambda t)$, $t \geq 0$. Hence the survivor function is $S(t) = \exp(-\lambda t)$ and the hazard rate is $h(t) = f_T(t)/S(t) = \lambda \exp(-\lambda t)/\exp(-\lambda t) = \lambda$. That means that the instantaneous rate of failure does not depend on time. This is not surprising as we know that the exponential distribution is memory-less, i.e., $\mathbb{P}(T > s + t \mid T > s) = \mathbb{P}(T > t)$, for all $s, t \geq 0$. This “no aging” effect is often unrealistic in practice but can be used as a baseline. ♣

Example 9.3. (Weibull distribution) One common (theoretical) distribution of survival times is the Weibull distribution. The density for $t > 0$ is given by

$$f_T(t) = \frac{\alpha}{\beta} \left(\frac{t}{\beta}\right)^{\alpha-1} \exp\left(-(t/\beta)^\alpha\right) \quad (9.9)$$

with shape parameter $\alpha > 0$ and scale parameter $\beta > 0$. Note that for $\alpha = 1$ we have the classical exponential density with rate $\lambda = 1/\beta$. The density, distribution function, survivor function and hazard function take on different forms depending on parameter choice. We illustrate this in Figure 9.2 for a `scale` parameter of 1 and a `shape` parameter of either 3, 1 or 0.5. Notice that the hazard function is monotonic for any choice of the scale parameter. ♣

R-Code 9.1: Plotting density, distribution, survivor and hazard functions using the Weibull distribution with various shape parameter values. (See Figure 9.2.)

```

Time <- seq(0, 2, by=0.01)
legtext <- expression(alpha==3, alpha==1, alpha==0.5)

### Density (pdf):
f3 <- dweibull(Time, shape=3)
plot(Time, f3, type="l", ylab="f(t)", ylim=c(0, 2))
lines(Time, dweibull(Time, shape=1), type="l", col=2)
lines(Time, dweibull(Time, shape=0.5), type="l", col=3)
legend("topright", legtext, lwd=1, col=1:3, bty="n")

### Distribution function (cdf):
F3 <- pweibull(Time, shape=3)
plot(Time, F3, type="l", ylab="F(t)")
lines(Time, pweibull(Time, shape=1), type="l", col=2)
lines(Time, pweibull(Time, shape=0.5), type="l", col=3)
legend("topleft", legtext, lwd=1, col=1:3, bty="n")

### Survivor function:
S3 <- 1 - F3
plot(Time, S3, type="l", ylab="S(t)")
lines(Time, 1 - pweibull(Time, shape=1), type="l", col="red")
lines(Time, 1 - pweibull(Time, shape=0.5), type="l", col="green")
legend("topright", legtext, lwd=1, col=1:3, bty="n")

### Hazard function:
h3 <- f3/S3
plot(Time, h3, type="l", ylab="h(t)", ylim=c(0,8))
h1 <- dweibull(Time, shape=1)/(1 - pweibull(Time, shape=1))
lines(Time, h1, type="l", col="red")
h0.5 <- dweibull(Time, shape=.5)/(1 - pweibull(Time, shape=0.5))
lines(Time, h0.5, type="l", col="green")
legend("topleft", legtext, lwd=1, col=1:3, bty="n")

```

9.3 Non-Parametric Methods for Survival Data

We first show a very common non-parametric method for estimating the survivor function and then discuss testing for differences between two survivor curves.

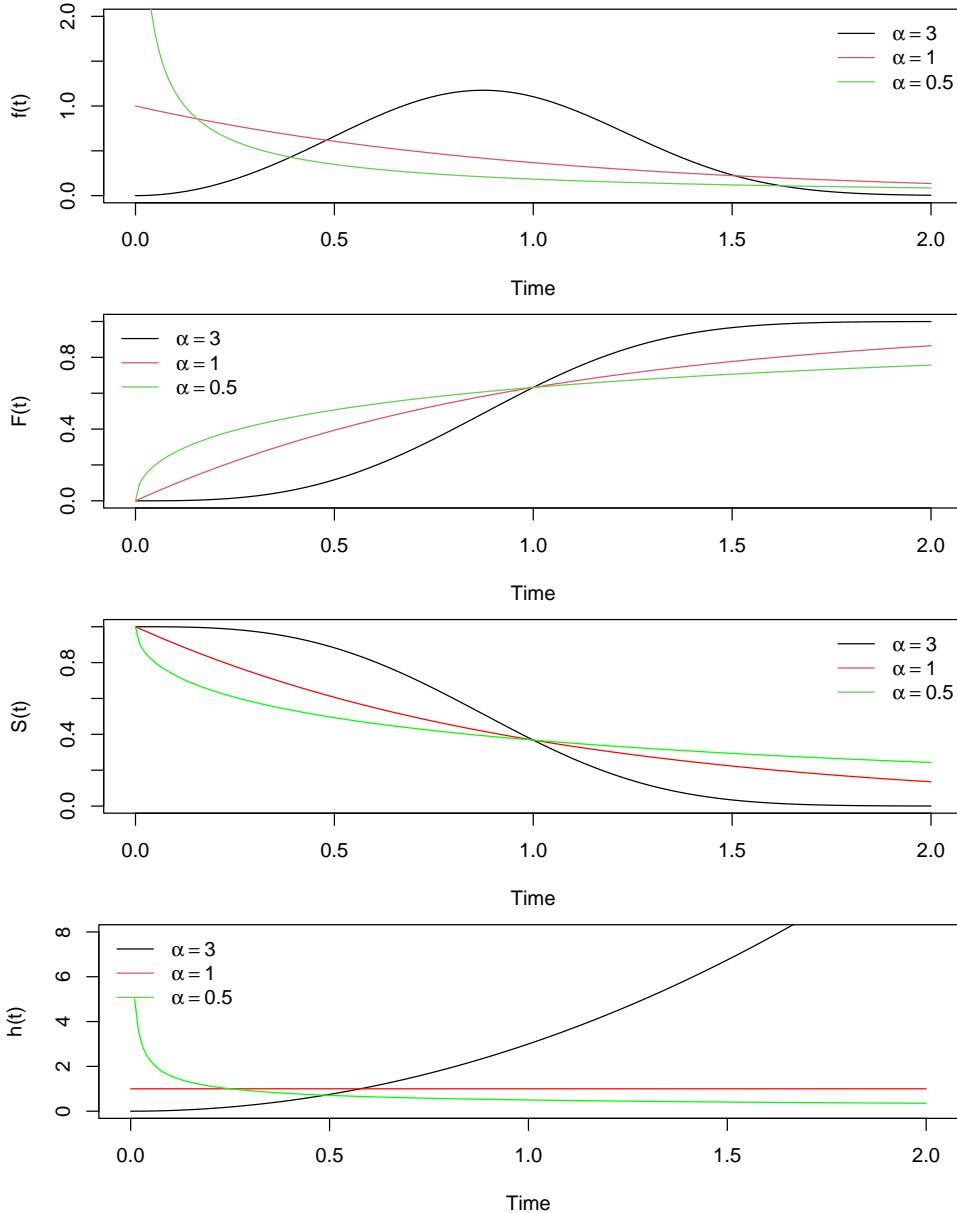


Figure 9.2: Plotting density, distribution, survivor and hazard functions using the Weibull distribution with various shape parameter values. (See R-Code 9.1.)

9.3.1 Kaplan-Meier Estimator for the Survivor Function

The Kaplan-Meier estimator, also known as the product-limit estimator, is a well-known statistic estimating the survivor function $S(t)$. To develop this estimator, we look at the ordered event times (ignoring all censored time points for the moment) $0 < t_{(1)} < t_{(2)} < \dots < t_{(m-1)} < t_{(m)}$. We start with the discretized hazard function, which is in the interval $[t_{(k-1)}, t_{(k)})$

$$h_k = P(T \in [t_{(k-1)}, t_{(k)}) \mid T \geq t_{(k-1)}), \quad (9.10)$$

i.e., the conditional probability of death in interval k , given that interval k has been reached. But we are interested in the opposite, i.e., the conditional probability p_k to survive interval k ,

given that interval k has been reached, given by

$$p_k = 1 - h_k = P(T \geq t_{(k)} \mid T \geq t_{(k-1)}). \quad (9.11)$$

In the end, we would like to look at the unconditional probability to survive the interval k , which we can calculate by multiplying all conditional survival probabilities from the beginning until k :

$$P_k = P(T \geq t_{(k)}) = p_k \cdot p_{k-1} \cdots p_1 = (1 - h_k) \cdots (1 - h_1). \quad (9.12)$$

To get an estimator for h_k , we define d_k as the number of events at $t_{(k)}$, and n_k as the number of individuals at risk at (or shortly before) $t_{(k)}$. This allows us to use

$$\hat{h}_k = \frac{d_k}{n_k} \quad (9.13)$$

in the expression for P_k . The Kaplan-Meier estimator for the survivor function then has the form

$$\widehat{S}(t) = \begin{cases} 1, & \text{if } t < t_{(1)}, \\ \prod_{k:t_{(k)} \leq t} \left(1 - \frac{d_k}{n_k}\right), & \text{if } t \geq t_{(1)}. \end{cases} \quad (9.14)$$

Example 9.4. (Ovarian cancer) We analyse a dataset of 26 patients with ovarian cancer. After surgery, the patients were treated with one of two distinct chemotherapies. The dataset is found in the `survival` package.

The Kaplan-Meier plot shows estimated survivor functions for each chemotherapy group. There seems to be a difference between the two groups, but this should be tested. ♣

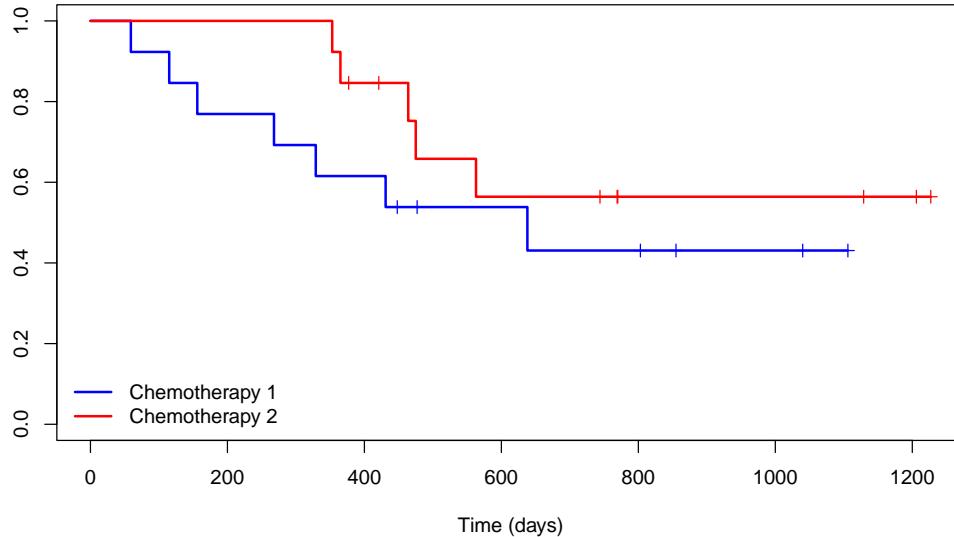


Figure 9.3: Kaplan-Meier curve comparing two different chemotherapies in the treatment of ovarian cancer. Ticks on the curves represent censored times. (See R-Code 9.2.)

It is possible to approximate the uncertainty of $\widehat{S}(t)$ by calculating a standard error for

$$\log(\widehat{S}(t)) = \sum_{k:t_{(k)} \leq t} \log(\widehat{p}_k). \quad (9.15)$$

R-Code 9.2 *ovarian* dataset: Kaplan-Meier curve comparing two different chemotherapies in the treatment of ovarian cancer. (See Figure 9.3.)

```

library(survival)    # load necessary library
head(ovarian, 3)    # data in package:survival, inspect first entries

##   futime fustat      age resid.ds rx ecog.ps
## 1     59       1 72.332       2  1       1
## 2    115       1 74.493       2  1       1
## 3    156       1 66.466       2  1       2

### Create survival object:
ovarian.surv <- Surv(time=ovarian$futime, event=ovarian$fustat)

### look at survival object:
ovarian.surv

## [1] 59 115 156 421+ 431 448+ 464 475 477+ 563 638
## [12] 744+ 769+ 770+ 803+ 855+ 1040+ 1106+ 1129+ 1206+ 1227+ 268
## [23] 329 353 365 377+

# Estimate and plot the Kaplan-Meier curve:
ovarian.fit <- survfit(ovarian.surv ~ ovarian$rx)
plot(ovarian.fit, lwd=2, col=c("blue", "red"), mark=3, xlab="Time (days)")
legend("bottomleft", c("Chemotherapy 1", "Chemotherapy 2"),
       lwd=2, col=c("blue", "red"), bty="n")

```

Now each of the individual terms \hat{p}_k can be treated as in a binomial setting. Collecting the terms and using the delta-method we have

$$\text{se}(\hat{S}(t)) \approx \hat{S}(t) \left(\sum_{k:t_{(k)} \leq t} \frac{d_k}{n_k(n_k - d_k)} \right)^{1/2}. \quad (9.16)$$

Notice that the uncertainties are pointwise. Hence, comparing uncertainty bands for different survivor curves is not straightforward. We next discuss a better approach for comparing two different survivor functions.

9.3.2 Testing differences in Survival Curves

A typical aim when analyzing survival data is to compare survival rates under different treatments (such as treatment with different drugs) and testing if one of them better. A simple non-parametric test to be used in such a situation is the log-rank test. Its basic idea is again to look at the ordered event times $0 < t_{(1)} < t_{(2)} < \dots < t_{(m-1)} < t_{(m)}$. We use the number of individuals at risk and the number of events at each time point $t_{(k)}$ for each treatment group:

- $n_{1,k}$:= number of individuals in Group 1 at risk at (or shortly before) $t_{(k)}$
- $n_{2,k}$:= number of individuals in Group 2 at risk at (or shortly before) $t_{(k)}$



7 min

- $d_{1,k} :=$ number of events in Group 1 at $t_{(k)}$
- $d_{2,k} :=$ number of events in Group 2 at $t_{(k)}$.

The setting can be summarized by the following 2×2 table:

Group	# of events	# surviving beyond $t_{(k)}$	# at risk at $t_{(k)}$
1	$d_{1,k}$	$n_{1,k} - d_{1,k}$	$n_{1,k}$
2	$d_{2,k}$	$n_{2,k} - d_{2,k}$	$n_{2,k}$
Total	d_k	$n_k - d_k$	n_k

(9.17)

The null hypothesis of the log-rank test is

$$H_0 : S_1(t) = S_2(t), \quad (9.18)$$

and the corresponding alternative hypothesis is

$$H_1 : S_1(t) \neq S_2(t). \quad (9.19)$$

If survival is the same in both groups and based on fixed margins, the four entries of the table in (9.17) are determined by any single entry, say $d_{1,k}$. We therefore consider this number of deaths as a random variable and we compare the theoretical expected value with the observed one. If the null hypothesis is true, the number of deaths follows a hypergeometric distribution, so that the expected number of deaths in Group 1 is

$$e_{1k} = \frac{n_{1,k}}{n_k} d_k. \quad (9.20)$$

Now the observed and expected number of deaths in Group 1 is compared, leading to the test statistic

$$q = \frac{\left(\sum_{k=1}^m (d_{1,k} - e_{1,k}) \right)^2}{\sum_{k=1}^m v_{1,k}}, \quad (9.21)$$

where $v_{1,k}$ is the respective variance of the hypergeometric distribution. If H_0 is true, q follows a χ^2 distribution with one degree of freedom and we reject H_0 if $q \geq \chi_1^2(1 - \alpha)$, i.e., the $1 - \alpha$ quantile of a χ_1^2 distribution.



4 min

Example 9.5. (Log-rank test to compare treatment groups) We return to the *ovarian* dataset and conduct a log-rank test in order to see if the difference in survival between the two treatments is significant. The result shows no evidence of a significant difference between the two chemotherapies (see R-Code 9.3). The result is not surprising, if the Kaplan-Meier estimates are plotted with (pointwise) confidence bands. ♣

Remark 9.1. 1. The log-rank test has many alternative names, including *Mantel-Haenszel* or *Mantel-Cox* test.

R-Code 9.3 *ovarian* dataset: Testing for differences between survival curves from different treatments.

```
survdiff(ovarian.surv ~ rx, data=ovarian)

## Call:
## survdiff(formula = ovarian.surv ~ rx, data = ovarian)
##
##      N Observed Expected (O-E)^2/E (O-E)^2/V
## rx=1 13      7     5.23    0.596     1.06
## rx=2 13      5     6.77    0.461     1.06
##
##  Chisq= 1.1  on 1 degrees of freedom, p= 0.3
### Kaplan-Meier estimates are plotted with (pointwise) confidence bands:
# plot(ovarian.fit, col=c(4,2), mark=3, conf.int=TRUE)
```

2. There are two major problems associated with the application of the log-rank test. First, it can be used only to compare the categories of one categorical covariate, but not for several (possibly continuous) covariates. Second, no quantification of a covariate effect is possible, only a difference can be stated.
3. The Wilcoxon-test (or Breslow-test) weights the individual summands of (9.21) by n_k . Ultimately, under the null the test statistic has a chi-square distribution as well but is better adapted to settings where we do not have proportional hazards.



9.4 Parametric Modeling of Survival Data

9.4.1 Regression Models for Survival Data

By looking at the hazard function instead of the survivor function, one can quantify the effects of covariates and make statements such as “the risk in Group 1 is β times higher than the risk in Group 2.” The most common regression model for censored survival data is the proportional hazards model. The general idea is that the hazard function can be split into a baseline hazard component and covariate effects component. The covariates themselves specify group association and (possibly also) individual factors. The hazard function of individual i with covariates \mathbf{x}_i is:

$$h_i(t; \mathbf{x}_i) = h_0(t) \exp(\mathbf{x}_i^\top \boldsymbol{\beta}), \quad (9.22)$$

where $h_0(t)$ represents the baseline hazard rate which can be treated as a nuisance parameter (or rather nuisance function) and left completely unspecified, as is the case in a Cox regression model. Alternatively, a certain form of the baseline hazard can be assumed and prespecified. For example, a Weibull regression model assumes the baseline hazard follows a Weibull distribution.

When fitting a proportional hazards model, we have

$$\frac{h_i(t; \mathbf{x}_i^\top)}{h_j(t; \mathbf{x}_j^\top)} = \frac{h_0(t) \exp(\mathbf{x}_i^\top \boldsymbol{\beta})}{h_0(t) \exp(\mathbf{x}_j^\top \boldsymbol{\beta})} = \frac{\exp(\mathbf{x}_i^\top \boldsymbol{\beta})}{\exp(\mathbf{x}_j^\top \boldsymbol{\beta})} = \exp((\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\beta}), \quad (9.23)$$

for two distinct individuals i and j . This means that two individuals have proportional hazards, and this proportionality is independent of t . Therefore, the estimated coefficients $\boldsymbol{\beta}$ can be interpreted as hazard ratios. When any kind of proportional hazards model is fitted, it is crucial to check the proportional hazards assumption first. If this assumption is not met, the results will be wrong and cannot be correctly interpreted. Details on how to assess the proportional hazards assumption will be given later in this chapter.

From now on, we will talk about Cox proportional hazards regression models, by far the most common survival model. It is a semiparametric model, i.e., the baseline hazard function is not specified.

Example 9.6. (Cox regression models to compare treatment groups) Three different Cox regression models are fitted and used to compare the treatment groups in the `ovarian` dataset. The first model contains treatment only, whereas in the second model, we add residual disease presence as a covariate as well.

Model 1 leads to the same result as the log-rank test, i.e., no significant treatment effect can be found. The risk of Chemotherapy 2 is a factor of $\exp(\hat{\beta}_{rx}) = 0.551$ smaller than the risk of Chemotherapy 1. Adding the other covariates (the residual disease covariate has a p -value around 5%) does not alter the finding that the different therapies are not significantly different.

The age effect is typically modeled through the baseline $h_0(t)$. We can, of course, separate the age effect from the baseline. Model 3 includes patients' age. The effect of age itself is noteworthy: with each one-year increase in age and keeping all other covariates constant, the risk is multiplied by $\exp(\hat{\beta}_{age}) = 1.137$.



R-Code 9.4: Using Cox proportional hazards to model the `ovarian` dataset.

```
ovarian.cox1 <- coxph(ovarian.surv ~ rx, data=ovarian)
summary(ovarian.cox1)

## Call:
## coxph(formula = ovarian.surv ~ rx, data = ovarian)
##
##    n= 26, number of events= 12
##
##          coef exp(coef) se(coef)      z Pr(>|z|)
## rx -0.596     0.551     0.587 -1.02     0.31
##
##          exp(coef) exp(-coef) lower .95 upper .95
## rx     0.551      1.82     0.174      1.74
##
## Concordance= 0.608  (se = 0.07 )
```

```

## Likelihood ratio test= 1.05  on 1 df,   p=0.3
## Wald test          = 1.03  on 1 df,   p=0.3
## Score (logrank) test = 1.06  on 1 df,   p=0.3
(ovarian.cox2 <- coxph(ovarian.surv ~ rx + resid.ds, data=ovarian) )
## Call:
## coxph(formula = ovarian.surv ~ rx + resid.ds, data = ovarian)
##
##           coef exp(coef) se(coef)     z      p
## rx       -0.76      0.47     0.59 -1.3 0.20
## resid.ds  1.32      3.74     0.68  1.9 0.05
##
## Likelihood ratio test=5.4  on 2 df, p=0.066
## n= 26, number of events= 12
(ovarian.cox3 <- coxph(ovarian.surv ~ rx + resid.ds + age, data=ovarian))
## Call:
## coxph(formula = ovarian.surv ~ rx + resid.ds + age, data = ovarian)
##
##           coef exp(coef) se(coef)     z      p
## rx       -0.849     0.428     0.639 -1.3 0.184
## resid.ds  0.696     2.006     0.759  0.9 0.359
## age       0.129     1.137     0.047  2.7 0.007
##
## Likelihood ratio test=17  on 3 df, p=0.00079
## n= 26, number of events= 12

```

The proportional hazards assumption can be checked in several ways. In the following example, we show two of them: a formal statistical test and a graphical assessment. Note that a statistical test of the proportional hazards assumption assumes that if the null hypothesis is true, the proportional hazards assumption is fulfilled, i.e., we hope for a non-significant result. A significant result is evidence against the proportional hazards assumption.

Example 9.7. (Testing the proportional hazards assumption) R-Code 9.5 shows the assessment of proportional hazards for model 1 from the analysis of the `ovarian` dataset. The test (which is based on so-called Schönfeld residuals) has a p -value of 0.104, which means that the proportional hazards assumption can not be rejected.

Take a look at the corresponding plot, shown in Figure 9.4. If the proportional hazards assumption cannot be rejected, there should be a straight, horizontal line at about 0. Since the pointwise confidence band contains this value at almost all time points, there is not enough evidence to assume non-proportional hazards. ♣

Using the function `muhaz()` from the package `muhaz`, it is possible to estimate the hazard function. As shown by Property 9.1.1 an estimate for the density is required. While the non-parametric cdf estimation is straightforward, it is difficult to estimate the corresponding densities.

R-Code 9.5 Assessing the proportional hazards assumption with a formal statistical test and graphically. (See Figure 9.4.)

```
### Formal statistical test:
cox.zph(ovarian.cox1)

##      chisq df   p
## rx     2.68  1 0.1
## GLOBAL 2.68  1 0.1

### Graphical assessment:
plot(cox.zph(ovarian.cox1))
abline(h=0, col=3)
```

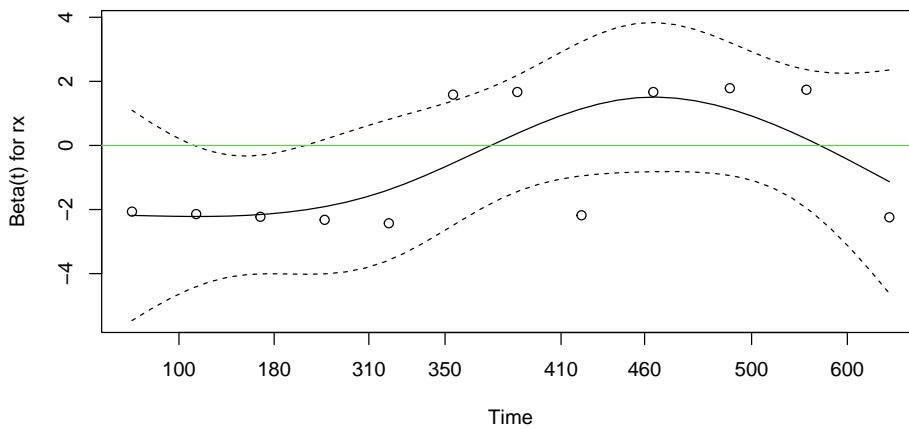


Figure 9.4: Graphical assessment of the proportional hazards assumption. (See R-Code 9.5.)

Typically, kernels are used (much in the spirit of Section 9.2) and additional parameters have to be specified.

As a general rule, if the hazard plots cross, the Cox proportional hazards model is not appropriate.

Example 9.8. R-Code 9.6 illustrates the result of hazard function estimation. Given the small dataset size, the results pictured in Figure 9.5 are difficult to interpret. ♣

Remark 9.2. 1. Although fitting a Cox proportional hazard model has the flavor of a GLM, it is not quite one. Under the hood, similar approaches are taken when estimating the parameters. However, it is at least as difficult assessing adequacy of the fit based on residuals. Comparing different models (nested or not) can be done using classical scores like AIC and BIC.

2. Naturally the “linear model” term can be extended to a “mixed model” idea, where each

R-Code 9.6 Estimation of the hazard function. (See Figure 9.5.)

```
library(muhaz)
### The following will produce a couple warnings, that we can ignore!
lam1 <- with(ovarian, muhaz(futime, fustat, rx==1, max.time=1106))
lam2 <- with(ovarian, muhaz(futime, fustat, rx==2, max.time=1227))

plot(lam1, xlim=c(0,1250))
lines(lam2, col=2)
rug(ovarian$futime[ovarian$rx==1])
rug(ovarian$futime[ovarian$rx==2], col=2)
```

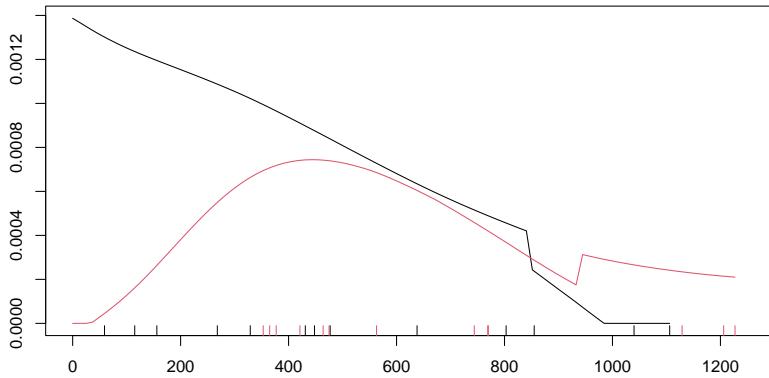


Figure 9.5: Plotting the estimated hazard functions. (See R-Code 9.6.)

subject would have an individual hazard, with some subjects being more *frail*. The individual terms are often not of interest and are modeled with a random effect in the linear component of the proportional hazard model and are often summarized by *frailty models*.

3. If the proportional hazards assumption is not fulfilled, care has to be taken when fitting regression models. Sometimes it is possible to solve this problem by conducting a stratified analysis or by creating an artificial time-dependent covariate. These measures, however, are beyond the scope of this course and are not discussed here.



9.4.2 Likelihood for Censored Data

Proper estimation of parameters requires censoring to be taken into account. The data contains the “exact” time to an event for some subjects, but for other subjects it is only known that the time to the event exceeds the observation time. As discussed at the beginning of the chapter, we assume that the censoring does not provide any information regarding the survival time of the subjects (see the sources listed in the bibliographic remarks for further details on censoring mechanisms and this assumption). Further, we do not model censoring stochastically.

Consider the survival times t_1, t_2, \dots, t_n for the n subjects where the associated random

variables T_i are iid according to some (parametric) density, say $f(t; \theta)$. We denote with c_i the censoring time for the i th subject and δ_i the indicator

$$\delta_i = \begin{cases} 1, & \text{the event was observed, i.e., } t_i \leq c_i, \\ 0, & \text{the event was not observed, i.e., } t_i > c_i. \end{cases} \quad (9.24)$$

The observed response is $y_i = \min\{t_i, c_i\}$ and thus (y_i, δ_i) for $i = 1, 2, \dots, n$ is given. Subjects experiencing the event contribute $f(y_i)$ to the likelihood (as classical) while censored subjects contribute $S(y_i)$ and thus the likelihood is

$$L(\theta, y_1, y_2, \dots, y_n) = \prod_{i=1, \delta_i=1}^n f(y_i) \times \prod_{i=1, \delta_i=0}^n S(y_i), \quad (9.25)$$

We now use the identities of Proposition 1 to avoid the differentiation of the censoring case and rewrite the likelihood and log-likelihood

$$L(\theta, y_1, y_2, \dots, y_n) = \prod_{i=1}^n h(y_i)^{\delta_i} \times S(y_i), \quad (9.26)$$

$$\ell(\theta, y_1, y_2, \dots, y_n) = \sum_{i=1}^n \delta_i \log(h(y_i)) - H(y_i). \quad (9.27)$$

Example 9.9. As an example, consider the exponential model with pdf

$$f(y) = \lambda \exp(-\lambda y), \quad y > 0. \quad (9.28)$$

The likelihood for n survival times is then

$$L(\lambda; y_1, y_2, \dots, y_n) = \prod_{i=1}^n f(y_i) = \lambda^n \exp(-\lambda \sum_{i=1}^n y_i). \quad (9.29)$$

It is straightforward to show that the maximum likelihood estimate of our parameter λ is $\hat{\lambda} = n / \sum_{i=1}^n y_i = 1/\bar{y}$. However, this assumes that all observed times are event times.

Suppose now that in addition to y_1, y_2, \dots, y_n , we also consider censoring. That is, for each subject, we have final observation time y_i and knowledge of whether that final observation time is an event time (e.g., death) or just the last time the subject was observed. Subjects experiencing the event contribute $f(y_i) = \lambda \exp(-\lambda y_i)$ to the likelihood while censored subjects contribute $S(y_i) = \exp(-\lambda c_i) = \exp(-\lambda y_i)$. Further, $h(y_i) = \lambda$ and $H(y_i) = \lambda y_i$. The log-likelihood for all observations is then

$$\ell(\lambda; y_1, y_2, \dots, y_n) = \sum_{i=1}^n \delta_i \log(\lambda) - \lambda y_i \quad (9.30)$$

with maximum likelihood estimate

$$\hat{\lambda} = \frac{\sum_{i=1}^n \delta_i}{n} \times \frac{n}{\sum_{i=1}^n y_i} = \bar{\delta} \frac{1}{\bar{y}}. \quad (9.31)$$

It is clear then that ignoring censoring yields biased results: we reduce the estimate by a factor of the proportion of observed events.

In R-Code 9.7 we use the *ovarian* data to show that ignoring censoring leads to a different estimate of the rate parameter λ . Using the exponential distribution with `survreg()` from the package *survival*, the ML estimate $\hat{\lambda}$ is given by $1/\exp(\text{Intercept})$. For simplicity, we ignore first the effect of the treatment (leading to the model above) and consider no censoring by indicating that all 26 subjects experience the event first.

In the second part of R-Code 9.7 we use two different hazard rates, again using an exponential model. Hence, the results are $h(t) = \lambda$ and $h(t) = \lambda + \Delta$ for the two treatments, where we need to specify the treatment as a factor. The numerical values are 0.001 and 0.0006 and ought to be compared with the non-parametric estimates in Figure 9.5. $\Delta = 0.5420$ is comparable to the value 0.5508 found in the proportional hazard models of R-Code 9.4. ♣

R-Code 9.7: Maximum likelihood estimation without and with censoring.

```

ovarian.surv.woCensoring <- Surv(time=ovarian$futime, event=rep(1, 26))
fit.woCensoring <- survreg(ovarian.surv.woCensoring ~ 1, data=ovarian,
                           dist="exponential")
c(1/exp(fit.woCensoring$coefficients), manual=1/mean(ovarian$futime) )
## (Intercept)      manual
## 0.0016679     0.0016679

fit.Censoring <- survreg(ovarian.surv ~ 1, data=ovarian, dist="exponential")
c(1/exp(fit.Censoring$coefficients),
  manual=mean(ovarian$fustat)/mean(ovarian$futime) )
## (Intercept)      manual
## 0.00076982    0.00076982

fit.woCensoring <- survreg(ovarian.surv.woCensoring ~ factor(rx),
                           data=ovarian, dist="exponential")
c(fit.tr1=exp(sum(fit.woCensoring$coefficients[1])),
  manual.tr1=mean(ovarian$futime[ovarian$rx==1]))

##      fit.tr1 manual.tr1
##      517.31    517.31

c(fit.tr2woCensoring=exp(sum(fit.woCensoring$coefficients)),
  manual.tr2.woCensoring=mean(ovarian$futime[ovarian$rx==2]))

##      fit.tr2woCensoring manual.tr2.woCensoring
##                      681.77          681.77

### to compare fit with previous results:
fit.Censoring <- survreg(ovarian.surv ~ factor(rx), data=ovarian,
                           dist="exponential")
c(fit.tr1=1/exp(sum(fit.Censoring$coefficients[1])),
  manual.tr2=1/exp(sum(fit.Censoring$coefficients)))

```

```

##    fit.tr1 manual.tr2
## 0.00104089 0.00056414
1/exp(fit.Censoring$coefficients[2])      # effect of treatment
## factor(rx)2
##      0.54198

```

Remark 9.3. Another commonly used parametric model is the log-logistic distribution, a random variable whose logarithm has a logistic distribution. While it has a similar shape as the log-normal distribution, a closed form for the cdf exists. The main advantage of the log-logistic model over the Weibull is the non-monotonic hazard function and it is the basis of accelerated failure time modeling. ♣

9.5 Bibliographic remarks

There are several good books about survival analysis, like Therneau and Grambsch (2010) or Klein and Moeschberger (2003) or the more classical ones like Cox and Oakes (1984) or Kalbfleisch and Prentice (2002). Moore (2016) and Kleinbaum and Klein (2012) contain a more practical and accessible description.

Collett (2014) is my favorite, he also provides all datasets through [this weblink](#).

An alternative way to represent survival data is through a so-called *Lexis* diagram where subjects are represented as lines (life line) in an age vs. time graph. The R package *Epi* manages Lexis diagrams. The CRAN task view [survival](http://cran.r-project.org/web/views/Survival.html) illustrates the activity of the field.

9.6 Exercises and Problems

Problem 9.1 (Log-logistic distribution) The log-logistic distribution has the following density

$$f(x; \alpha, \beta) = \frac{\left(\frac{\beta}{\alpha}\right)\left(\frac{x}{\alpha}\right)^{\beta-1}}{\left(1 + \left(\frac{x}{\alpha}\right)^{\beta}\right)^2}, \quad x \geq 0,$$

$$F(x; \alpha, \beta) = \frac{x^{\beta}}{\alpha^{\beta} + x^{\beta}}$$

where $\alpha > 0$ is a scale parameter and $\beta > 0$ a shape parameter.

- a) Plot the density, the distribution function, the survival function and the hazard function of the log-logistic distribution for the values $\alpha = 1$ and $\beta \in \{0.5, 1, 2, 4, 8\}$.
- b) The shape of the density of a Weibull random variable depends greatly on the value of κ . More specifically, there are three cases of interest: $0 < \kappa < 1$, $\kappa = 1$ and $\kappa > 1$. Let $\sigma = 1$: by using relevant plots in R, give a qualitative description of the obtained pattern for each of the aforementioned cases.

Problem 9.2 (Survival analysis) A randomized study was conducted to investigate the effect of a new treatment of *primary biliary cirrhosis* (PBC), a chronic and eventually fatal liver disease. The trial was designed to compare an active treatment (Azathioprine) ($n = 94$) versus placebo ($n = 90$). 31 of the patients suffered from *central cholestasis* (variable `cenc0`), a particular strong form of the disease. The primary endpoint was the survival time from entry into study. The data set `pbc.txt` is available from <https://www.math.uzh.ch/furrer/download/sta121/pbc.txt>.

- a) Load the data set `pbc.txt` containing the data from the PBC study, create a survival object and have a look at it.

Hint: Code the `treat` variable as factor.

- b) Compare the two groups defined by `treat=1` (placebo) and `treat=2` (treatment) based on graphical displays of the survival functions. Use the function `survfit()`, which computes a Kaplan-Meier estimate of the survival curve.

Do you think that there is a difference between treatment and placebo?

- c) Compare the two treatment groups with a log-rank test.

- d) Fit a Cox-model with `treat` as predictor.

- e) Fit a Cox-model with the additional predictors `age` and `cenc0`. How does the result of `treat` change compared to d)?

- f) Check the proportional hazards assumption in the Cox-models in d) and e).

Chapter 10

Time Series

Measurements of the same variables are often made at regular points in time and exhibit an inherent correlation. Appropriate methodological approaches must be used to correctly account for this dependency.

Learning goals for this chapter:

- ◊ Identify important features on a time series plot
- ◊ Identify and interpret an AR(p), MA(q)
- ◊ Interpret PACF, ACF
- ◊ Distinguish AR terms and MA terms from exploring an ACF and PACF
- ◊ Test if all residual autocorrelations are zero
- ◊ Predict values with ARIMA model using R, create and interpret confidence intervals for forecasts

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter10.R.

10.1 Introduction

We start with a classical dataset.

Example 10.1. The concentration of carbon dioxide (CO₂) in Earth's atmosphere has increased in the last century and is likely to increase in the future. CO₂ has a greenhouse effect and thus contributes to global warming. The left panel of Figure 10.1 shows recent monthly mean carbon dioxide levels measured at Mauna Loa Observatory, Hawaii. The right panel shows annual averages with a cubic fit. The data is [available here](#) with [more information here](#). See also [this article here](#).

R-Code 10.1 Monthly and annual mean carbon dioxide levels measured at Mauna Loa Observatory, Hawaii. (See Figure 10.1.)

```

path <- "http://user.math.uzh.ch/furrer/download/sta121/"
raw <- read.csv(paste0(path, "co2-mlo-monthly-noaa-esrl.csv"))
dim(raw)
## [1] 56 13
craw <- c(as.matrix(t(raw[,-1])))[-c(1,2,671,672)]
(n <- length(craw))
## [1] 668
mnts <- rep(1:12,56)[-c(1,2,671,672)]
time <- (1:n)/n
co2 <- ts(craw, start=c(1958, 3), frequency=12)
str(co2)
## Time-Series [1:668] from 1958 to 2014: 316 317 318 317 316 ...
plot(co2, ylab="CO2 [ppm]")
lines(lowess(co2, f=.2), col=2) # robust wrt f value!
am <- read.csv(paste0(path, "co2-mlo-annual-noaa-esrl.csv"))
dim(am)
## [1] 54 2
# apply(raw[,-1],1,weighted.mean,na.rm=T) is a different approach
plot(co2~year, data=am, cex=.5)
lines(am$year, fitted(lm(co2~poly(year,3), data=am)), col=3)

```

R-Code 10.2 Fitting an annual and a cubic trend. (See Figure 10.2.)

```

lm1 <- lm(craw ~ poly(time,3) + as.factor(mnts))
plot(resid(lm1))
### lot of structure!!
indi <- c(rep(0,400), rep(1,268))
lm2 <- lm(craw ~ indi * (poly(time,3)) + as.factor(mnts))
plot(resid(lm2))
plot(resid(lm2)[-1], resid(lm2)[-n]) # lag one scatter plot
AIC(lm1, lm2)
##      df      AIC
## lm1 16 1462.1503
## lm2 20  842.4133

```

The inconvenient truth is that the rate is increasing. The annual means can be modeled by a third order polynomial (see R-Code 10.2). The residuals of this fit are shown in Figure 10.2 and exhibit a strange behavior occurring after 400 months (reason?).

This effect is due to the eruption of the volcano Mount Pinatubo in June 1991 (see [here](#) for details). Including indicator functions (through the vector `indi`) in the model eliminates a lot of the dominant structure in the residuals (middle panel in Figure 10.2). However, the residuals remain correlated. This is also reflected in the lag-one scatter plot (i.e., plot of the pairs (r_{t-1}, r_t) , where r_t is the residual at time t) in the right panel of Figure 10.2. ♣

The iid assumptions of the noise in the regression model of the previous example is clearly violated. In this chapter we introduce a few models for such time series data where we exploit this correlation for more appropriate uncertainty estimates and better prediction models.

10.2 ARMA Models

In this section we set the stage for a very large class of time series models. For simplicity, we start by assuming that the time series has a zero mean.

10.2.1 Autoregressive Models

The following introduces an intuitive model for time series: the so-called autoregressive model

$$Y_t = \phi Y_{t-1} + \varepsilon_t, \quad \phi \neq 0 \quad (10.1)$$

with Gaussian white noise ε_t , i.e., $\varepsilon_t \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. It is called autoregressive because the current observation depends on the previous one. This model is denoted with AR(1), where the number in the parentheses indicates the number of previous terms the current observation depends on.

In practice, t varies between 1 and T and some initial condition has to be assumed, for example, $Y_0 = 0$ or $Y_1 \sim \mathcal{N}(0, \sigma^2)$. Further, we assume that $|\phi| < 1$, to ensure stationarity. In the case of $\phi > 1$, it quickly becomes clear that the series “explodes”.

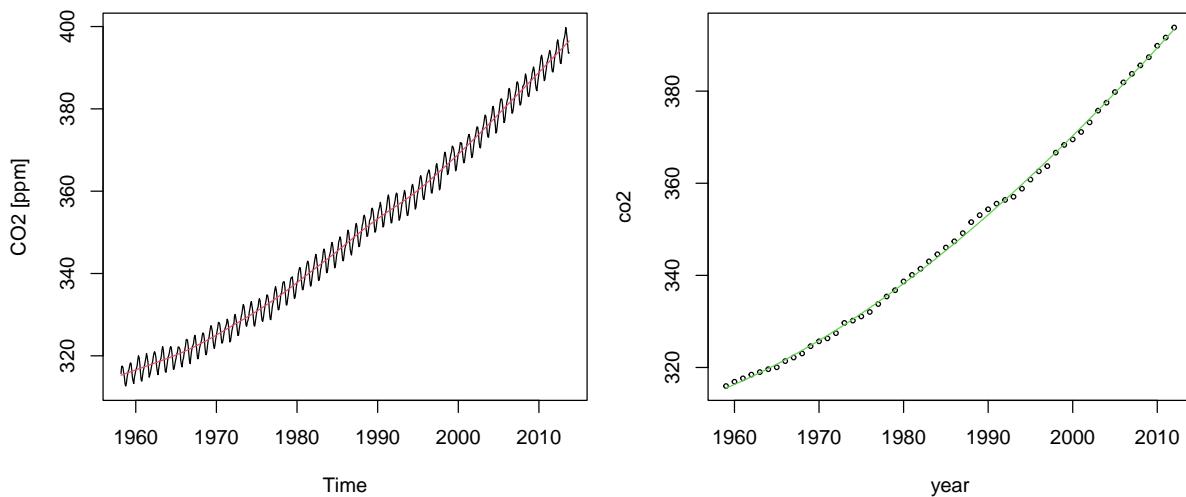


Figure 10.1: Monthly and annual mean carbon dioxide levels measured at Mauna Loa Observatory, Hawaii. The red and green curves are lowess smooth and third order polynomial fits, respectively. (See R-Code 10.1.)

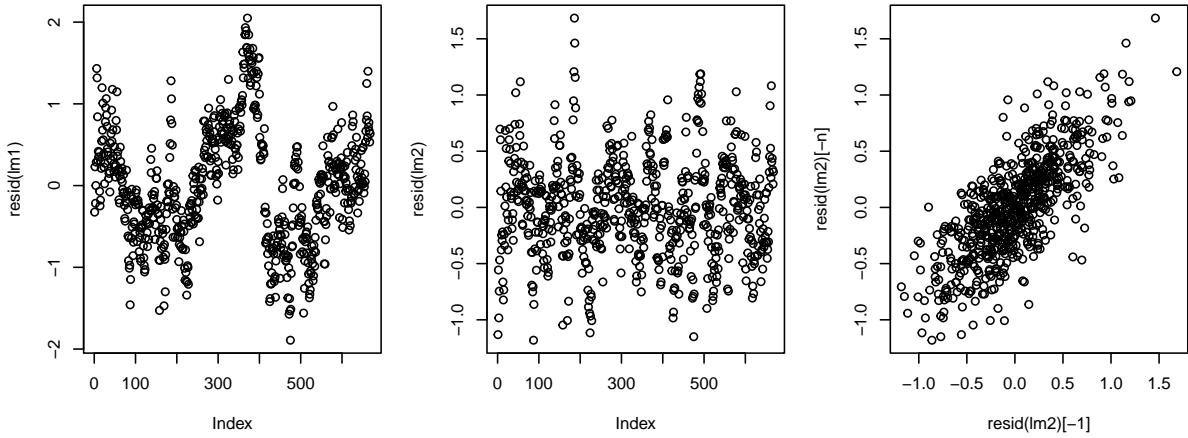


Figure 10.2: Residuals of cubic trend fits without (left) and with (middle) the Mount Pinatubo effect. Lag one scatter plot of residuals (right). (See R-Code 10.2.)

Assuming $Y_1 \sim \mathcal{N}(0, \sigma^2/(1 - \phi^2))$ allows for the connection between an AR(1) model (with $1 \leq t \leq T$) and a multivariate Gaussian distribution. More specifically, let $\mathbf{Y} = (Y_1, \dots, Y_T)^\top$ and $\mathbf{0}$ a zero T -vector. Then

$$\mathbf{Y} \sim \mathcal{N}_T(\mathbf{0}, \Sigma), \quad \Sigma = \left(\frac{\sigma^2}{1 - \phi^2} \phi^{|i-j|} \right)_{ij}. \quad (10.2)$$

The AR(1) model (10.1) can be extended to an AR(p) model, defined as

$$Y_t = \phi_1 Y_{t-1} + \dots + \phi_p Y_{t-p} + \varepsilon_t, \quad \phi_p \neq 0, \quad (10.3)$$

with appropriate assumptions on the parameters ϕ_1, \dots, ϕ_p and with “initial” conditions (e.g., distributions for Y_1, \dots, Y_p).

In the case of a non-zero but constant mean, Y_\bullet is replaced by $Y_\bullet - \mu$ in the AR(p) definition.

10.2.2 Moving Average Models

Similarly, we can define moving average model where the present observation depends on a weighted average of white noise components. More precisely, a MA(1) is defined by

$$Y_t = \varepsilon_t + \theta \varepsilon_{t-1}, \quad \theta \neq 0, \quad (10.4)$$

with, typically, $\varepsilon_t \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$, and a MA(q) is defined by

$$Y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q}, \quad \theta_q \neq 0. \quad (10.5)$$

In the case of a non-zero mean, Y_\bullet is replaced by $Y_\bullet - \mu$ in the MA(q) definition.

It is possible to combine AR(p) and MA(q) processes into so-called ARMA(p, q) models.

10.3 ACF and PACF and Their Estimation

One of the goals of time series modeling is to implement correlation structures in the observations. We recall here the covariance and correlation of a bivariate random variable.

The function

$$\gamma(s, t) = \text{Cov}(Y_s, Y_t), \quad s, t = 1, \dots, T, \quad (10.6)$$

is called the autocovariance function. The autocorrelation function is obtained by normalizing the autocovariance by $\sqrt{\gamma(s, s)\gamma(t, t)}$.

It is convenient to further restrict the class of time series by considering stationarity. A time series having constant mean and an autocovariance function that depends only on the magnitude $|s - t|$ is called stationary. More precisely, Y_t is (weakly) stationary if

$$E[Y_t] = \mu, \quad (10.7)$$

$$\text{Cov}(Y_s, Y_t) = \gamma(|s - t|), \quad s, t = 1, \dots, T. \quad (10.8)$$

We will consider only integer values of $k = |s - t|$ which we subscript from now on. For a stationary process, the autocorrelations, denoted by ρ_k , simplify to γ_k/γ_0 .

The autocorrelation and partial autocorrelation function can be used to infer the “nature” of the ARMA model, as illustrated in the following examples.

The autocovariance of an MA(1) process $Y_t = \theta\varepsilon_{t-1} + \varepsilon_t$ is obtained from

$$\gamma_h = \text{Cov}(\theta\varepsilon_{t-1} + \varepsilon_t, \theta\varepsilon_{t-1+h} + \varepsilon_{t+h}) = \begin{cases} (\theta^2 + 1)\sigma^2 & h = 0, \\ \theta\sigma^2 & h = 1, \\ 0 & h \geq 2. \end{cases} \quad (10.9)$$

For an AR(1) process, we have as autocorrelation function (ACF)

$$\rho_h = \phi^h. \quad (10.10)$$

Another, yet similar, concept is the partial autocorrelation function (PACF). It is essentially the correlation between Y_{t-k} and Y_t “conditional” on knowing $Y_{t-k+1}, \dots, Y_{t-1}$. The PACF is denoted with ϕ_{kk} and by definition $\phi_{11} = \rho_1$. For an AR(1) process, we have $\phi_{kk} = 0$ for all $k > 1$. For an MA(1) process, we have $\phi_{22} < 0$ and then levels off.

R-Code 10.3 Autocorrelation and partial autocorrelation of the residuals. (See Figure 10.3.)

```
acf(resid(lm2))
pacf(resid(lm2))
```

For an AR(p) model, the PACF vanishes after p lags. For an MA(q) model, the ACF vanishes after q lags. Hence, the PACF and ACF can be used to select an appropriate model.

Example 10.2. Figure 10.3 shows the ACF and PACF of the detrended time series (i.e., the residuals) and proposes an AR(2) model. ♣

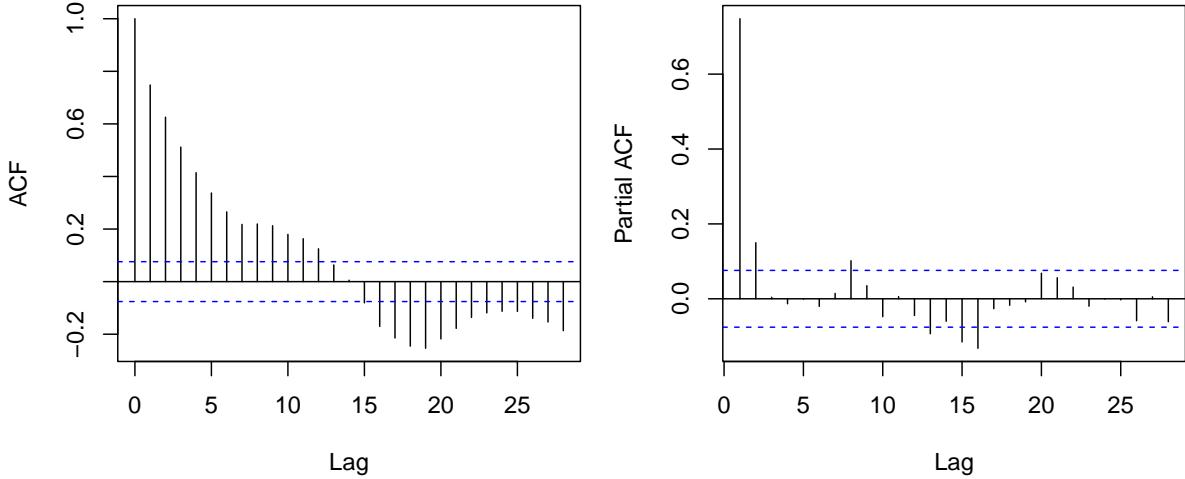


Figure 10.3: Autocorrelation and partial autocorrelation plots of the detrended time series (i.e., the residuals). (See R-Code 10.3.)

A useful estimate of the autocovariance function is

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-h} (y_t - \bar{y})(y_{t+h} - \bar{y}). \quad (10.11)$$

The estimator, however, is biased, even when replacing n by $n - h$.

Example 10.3. R-Code 10.4 and Figure 10.4 illustrate the estimation of the AR(2) parameters. The theoretical ACF and PACF values are compared with the empirical ones. The AR(2) represents a very good fit. ♣

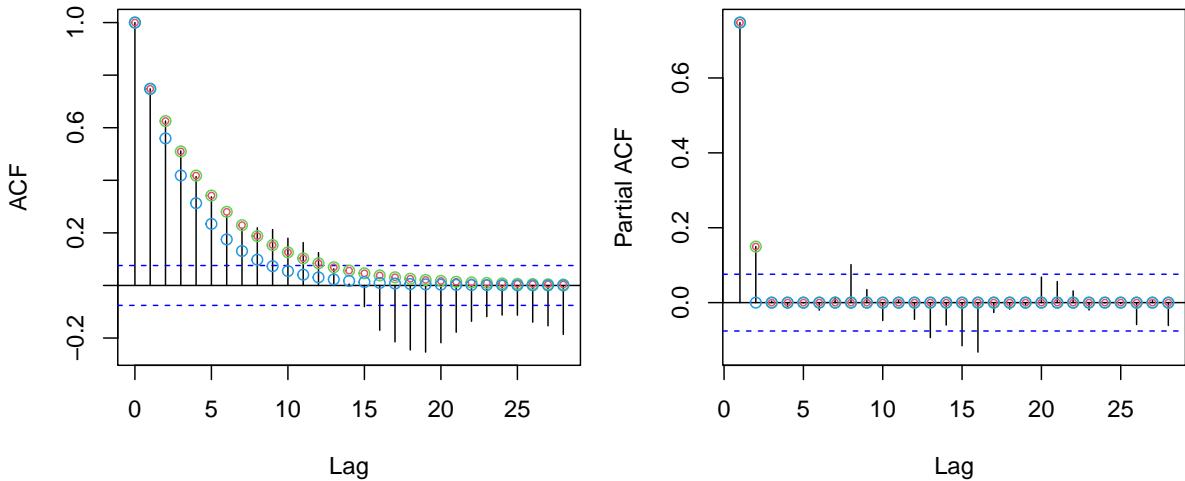


Figure 10.4: Autocorrelation and partial-autocorrelation plot of the detrended time series with superimposed theoretical values based on the estimated AR(p) model ($p = 2$ red, $p = 1$ blue, $p = 3$ green). (See R-Code 10.4.)

R-Code 10.4 ARMA fit. (See Figure 10.4.)

```

print(f1 <- ar(resid(lm2), order=2))

##
## Call:
## ar(x = resid(lm2), order.max = 2)
##
## Coefficients:
##      1       2
## 0.6360  0.1497
##
## Order selected 2  sigma^2 estimated as  0.0842
### AIC would select a too complicated model:
ar(resid(lm2))$order
## [1] 16

l <- 28
acf(resid(lm2))
points(0:1, ARMAacf(ar=f1$ar, l=l), col=2, cex=.6)
points(0:1, ARMAacf(ar=ar(resid(lm2), order=3)$ar, l=l), col=3)
points(0:1, ARMAacf(ar=ar(resid(lm2), order=1)$ar, l=l), col=4)
pacf(resid(lm2))
points(1:1, ARMAacf(ar=f1$ar, l=l, pacf=TRUE), col=2, cex=.6)
points(1:1, ARMAacf(ar=ar(resid(lm2), order=3)$ar, l=l, pacf=T), col=3)
points(1:1, ARMAacf(ar=ar(resid(lm2), order=1)$ar, l=l, pacf=T), col=4)

```

There are several ways to obtain the uncertainty in the estimates $\hat{\phi}_i$ as illustrated by R-Code 10.5. Figure 10.5 emphasizes that marginal confidence intervals may be misleadingly large.

R-Code 10.5: Bootstrapping. (See Figure 10.5.)

```

# a1 <- ar(resid(lm2), order=2) # does not allow the extraction of se!
a1 <- arima(resid(lm2), order=c(2,0,0))
th <- coef(a1)[1:2]
print(ci <- confint(a1) )

##                  2.5 %     97.5 %
## ar1          0.56789549 0.71833519
## ar2          0.07328981 0.22393823
## intercept -0.10904930 0.09901755

N <- 200
bootsam <- array(0, c(N,2))

```

```

res <- a1$resid[-c(1:2)]
for (i in 1:N) {
  ab <- arima.sim(n=length(res), list(ar=th), innov=sample(res))
  bootsam[i,] <- ar(ab, order=2, aic=F)$ar
}

plot(bootsam, xlab=expression(hat(phi)[1]), ylab=expression(hat(phi)[2]))
abline(v=th[1], h=th[2], col=3)
abline(v=ci[1,], h=ci[2,], col=4, lty=2)
hist(bootsam[,2], prob=T, xlim=c(0,1), main="", xlab=expression(hat(phi)))
abline(v=c(th[2],ci[2,]), col=c(3,4,4), lty=c(1,2,2))
hist(bootsam[,1], prob=T, add=T)
abline(v=c(th[1],ci[1,]), col=c(3,4,4), lty=c(1,2,2))

```

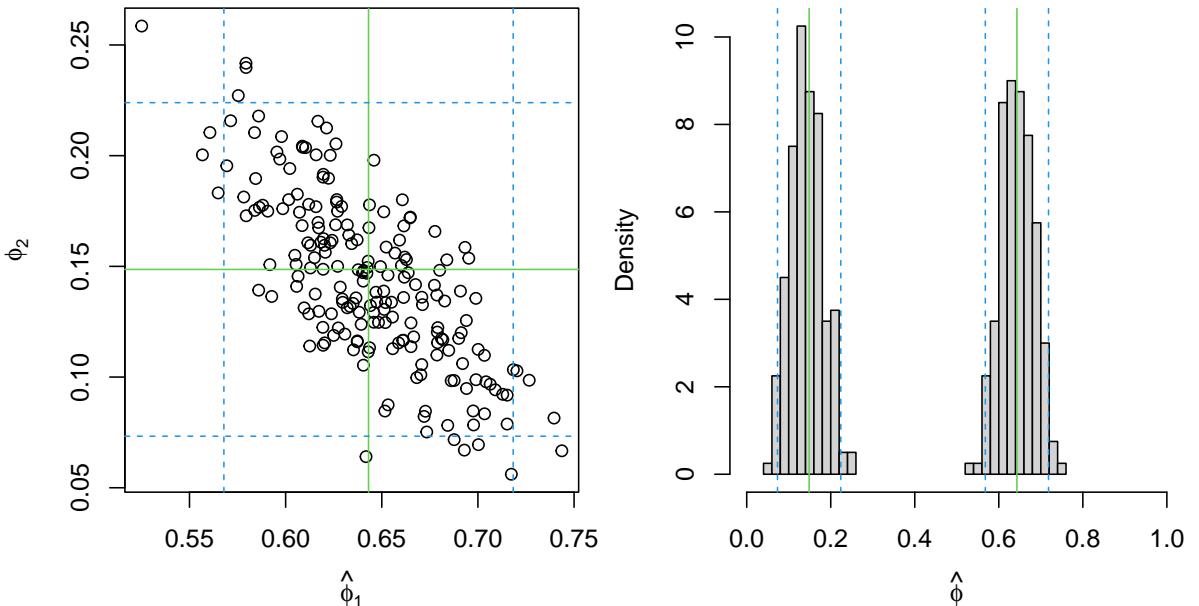


Figure 10.5: Bootstrap uncertainty of the AR(2) model. Left panel gives 200 bootstrapped estimates of an AR(2) model. Right panel gives the marginal histograms. Green lines indicate the estimates. Blue dotted lines give crude 95% confidence interval bounds. (See R-Code 10.5.)

It is possible to estimate the trend/“fixed effects” directly through the `arima()` function. The model specification is slightly more complicated but essentially equivalent to a `lm()` approach.

10.4 Prediction

After fitting a model, prediction can be performed. In the context of time series, prediction of observed values is called smoothing of future (unobserved) values or a forecast. Prediction is

essentially based on linear best unbiased predictor (BLUP) ideas.

For fitted `arima()` objects, the R-function `predict()` can be used. R-Code 10.6 illustrates estimation through the `arima` function, followed by a 24 month forecast. See Example 10.1 for a detailed description of the data and exploratory figures.

Notice that a cubic time and an offset are not required. A seasonal effect is almost as having a factor with individual fixed effects (see Figure 10.6).

R-Code 10.6: Prediction of CO₂ values with an AR(2) and different trend models. (See Figure 10.6.)

```
### We assume data objects co2, time and indi are available. See Example 10.1
set.seed(12)
### Pinatubo effect and cubic time:
X <- cbind(Pinatubo=indi, time, time2=I(time^2), time3=I(time^3))
f1 <- arima(co2, order=c(2,0,0), season=c(1,0,0), xreg=X)
### Eliminate Pinatubo and cubic factor
f2 <- arima(co2, order=c(2,0,0), season=c(1,0,0), xreg=X[,-c(1,4)])
### Manually include seasonal component
X3 <- model.matrix(~time+I(time^2)+factor(mnts))
f3 <- arima(co2, order=c(2,0,0), xreg=X3, include.mean=FALSE)

ahead <- ((n+1):(n+24))/n
newxreg <- cbind(0,ahead, ahead^2, ahead^3)
pr1 <- predict(f1, newxreg=newxreg, n.ahead=24)
newxreg <- cbind(ahead, ahead^2)
pr2 <- predict(f2, newxreg=newxreg, n.ahead=24)
factormat <- diag(12)[,-1] # just 11 factors (treatment contrasts)
newxreg <- cbind(1, ahead, ahead^2, factormat[c(11:12,1:12,1:10),])
pr3 <- predict(f3, newxreg=newxreg, n.ahead=24)

plot(co2, ylab="CO2 [ppm]", xlim=c(2010,2016), ylim=c(390,406))
lines(2013+(9+1:24)/12, pr1$pred, col=1)
lines(2013+(9+1:24)/12, pr1$pred+2*pr1$se, col=1, lty=2)
lines(2013+(9+1:24)/12, pr1$pred-2*pr1$se, col=1, lty=2)
# condensed:
matlines(2013+(9+1:24)/12, cbind(pr2$pred, pr2$pred+2*pr2$se,
                                    pr2$pred-2*pr2$se), col=2, lty=c(1,2,2))
matlines(2013+(9+1:24)/12, cbind(pr3$pred, pr3$pred+2*pr3$se,
                                    pr3$pred-2*pr3$se), col=3, lty=c(1,2,2))

lm2 <- lm(craw ~ (poly(time,2)) + as.factor(mnts))
lines(time*n/12+1958+1/12,predict(lm2, newdata=as.data.frame(X3)), col=5)
rbind(predictors=summary(pr1$pred-pr2$pred),
```

```

factors=summary(pr3$pred-pr2$pred))

##             Min.   1st Qu.    Median      Mean   3rd Qu.
## predictors 0.2421855 0.2517099 0.2709061 0.2673309 0.2803843
## factors     -0.9367029 -0.5942522 -0.4378734 -0.4679855 -0.3193980
##                  Max.
## predictors 0.29515327
## factors     -0.09481726

```

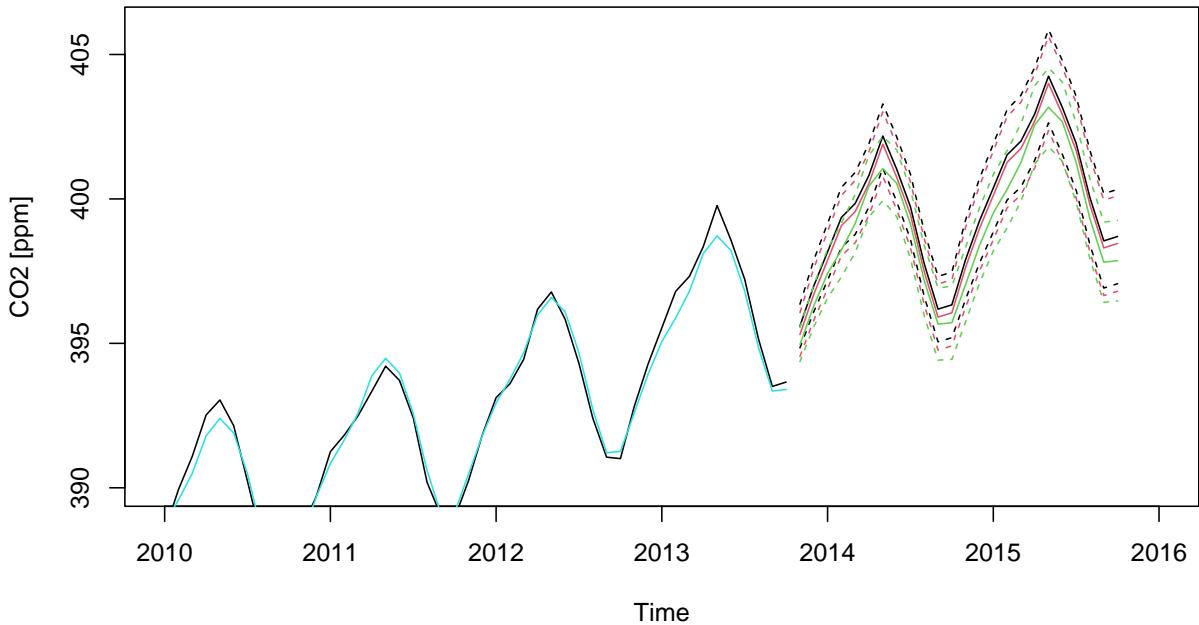


Figure 10.6: Prediction for an AR(2) model with seasonal components and associated crude uncertainties (dashed). Black: cubic trend with Pinatubo effect, red: quadratic trend, green: individual fixed levels for each month. Light blue is simple regression fit. (See R-Code 10.6.)

10.5 Example: Car Drivers Accidents

Consider the R dataset `UKDriverDeaths`, a time series giving monthly counts of car drivers in Great Britain killed or seriously injured from January 1969 to December 1984 ($n = 192$). Wearing a seat belt was made compulsory on 31 Jan 1983. The data is shown in Figure 10.7.

The series y_i exhibits a strong seasonal component and a (possibly) smooth trend. R-Code 10.7 illustrates the fitting of several simple models based on the raw data. A commonly performed square root transformation would yield completely different “optimal” models.

A similar result as that in R-Code 10.7 could be obtained by fitting a linear model with `belt` as a predictor and additional seasonal predictors, followed by a AR(2) model.

R-Code 10.7: Time series of the `UKDriverDeaths` dataset with seasonal AR fit. (See Figure 10.7.)

```

str(UKDriverDeaths)                      # from package:datasets
## Time-Series [1:192] from 1969 to 1985: 1687 1508 1507 1385 1632 ...
plot(UKDriverDeaths)

### There is a possibility to play with the order...
f1 <- arima(UKDriverDeaths, order=c(2,0,0),season=c(1,0,0))

belt <- c(rep(0,169),rep(1,23))
f2 <- arima(UKDriverDeaths, order=c(1,0,0),season=c(1,0,0), xreg=belt)
f3 <- arima(UKDriverDeaths, order=c(2,0,0),season=c(1,0,0), xreg=belt)
f1
##
## Call:
## arima(x = UKDriverDeaths, order = c(2, 0, 0), seasonal = c(1, 0, 0))
##
## Coefficients:
##          ar1      ar2      sar1  intercept
##          0.4347   0.2029   0.6671   1643.502
##  s.e.   0.0748   0.0728   0.0550     83.876
##
## sigma^2 estimated as 24943:  log likelihood = -1248.14,  aic = 2506.27
f2
##
## Call:
## arima(x = UKDriverDeaths, order = c(1, 0, 0), seasonal = c(1, 0, 0), xreg = belt)
##
## Coefficients:
##          ar1      sar1  intercept      belt
##          0.4446   0.6511   1710.1531  -347.6812
##  s.e.   0.0695   0.0564    53.3647    73.0634
##
## sigma^2 estimated as 23693:  log likelihood = -1242.86,  aic = 2495.71
f3
##
## Call:
## arima(x = UKDriverDeaths, order = c(2, 0, 0), seasonal = c(1, 0, 0), xreg = belt)
##
## Coefficients:

```

```

##          ar1      ar2     sar1 intercept       belt
##      0.3466  0.1966  0.6890  1714.8170 -365.0402
##  s.e.  0.0750  0.0740  0.0531    69.3883   80.6904
##
## sigma^2 estimated as 22720:  log likelihood = -1239.42,  aic = 2490.84
lines(UKDriverDeaths-f1$residuals, col=4)
lines(UKDriverDeaths-f2$residuals, col=3)
lines(UKDriverDeaths-f3$residuals, col=2)

```

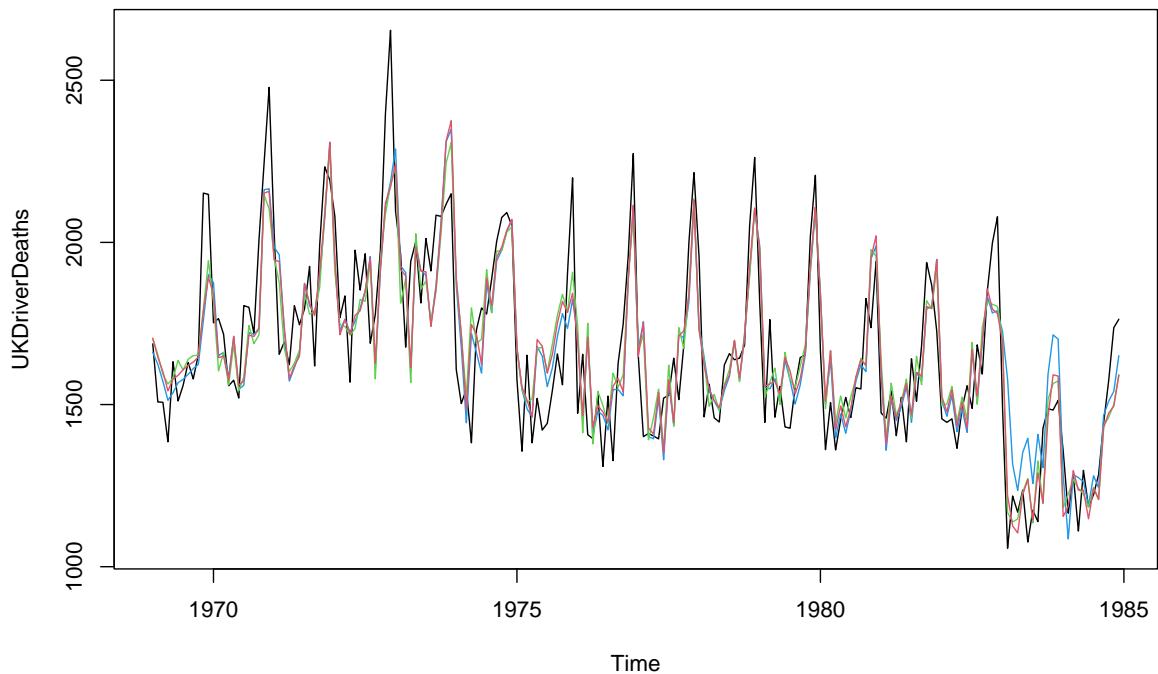


Figure 10.7: `UKDriverDeaths` data and fit. (See R-Code 10.7.)

10.6 Bibliographic remarks

There are plenty of good and valuable time series books, like Box *et al.* (2008), Brockwell and Davis (2010), and Shumway and Stoffer (2010). They are all quite formal in notation. For the latter one, there is a public pdf version available at www.stat.pitt.edu/stoffer/tsa4.

Brockwell and Davis (1991) is another classic, at the upper end with respect to technicalities.

10.7 Exercises and Problems

Problem 10.1 (Auto regressive process) Consider the AR(1) process defined as

$$Y_t = \phi Y_{t-1} + \epsilon_t, \quad \epsilon_t \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \quad \phi \in (0, 1), \quad Y_1 \sim \mathcal{N}(0, \sigma^2).$$

- a) Use R to simulate a realisation of an AR(1) process for $t = 1, \dots, 50$. Use (i) a sequential simulation and (ii) a direct approach (Exercise class). Use e.g., $\sigma^2 = 2$, $\phi = 0.5$. What is the effect of choosing different values for σ^2 and ϕ ?
- b) Calculate $E(Y_{t+1} | Y_1, \dots, Y_t)$ and $\text{Var}(Y_{t+1} | Y_1, \dots, Y_t)$. Use this result to predict Y_{51} together with a 95% confidence interval for the simulated chain in a.
- c) For $n \geq 1$, calculate $E(Y_{t+n} | Y_t)$ and $\text{Var}(Y_{t+n} | Y_t)$. Use this result to predict Y_{55} together with a 95% confidence interval for the simulated chain in a.
- d) How many times does the AR process as specified in a change sign (cross the zero line) in average?
Hint: Use simulation techniques.
- e) Use R to sequentially simulate a chain with $\phi = 1.1$ and describe its behaviour.

Problem 10.2 (Working with `ts` object) The data set `Meatprice.txt` is available from <https://www.math.uzh.ch/furrer/download/sta121/Meatprice.txt>. It was published by the Swiss Farmer's Union and consists of a time series that describes the realised monthly price of cow meat.

- a) Look at the data.
- b) Create a time series object and plot it.
Hint: Use the R function `ts()` and set the arguments `frequency` and `start`.
- c) Decompose the time series into a seasonal trend, an overall trend and residuals.
Hint: Use `stl()` with argument `s.window="periodic"`.
- d) Are the residuals correlated?
Hint: `acf()`, `pacf()`.
- e) Adjust the time series for seasonality. Plot and compare it to the original time series.
- f) Estimate the coefficients of an AR(1) process of this time series using `ar()`.

Chapter 11

Spatial Statistics

Measurements are often taken in some predefined spatial domain and exhibit an inherent correlation that needs to be taken into account through appropriate methodology approaches.

Learning goals for this chapter:

- ◊ Visualize geostatistical data in R, check if there is a spatial structure in the data and the spatial methods should be used to analyse the data
- ◊ Define stationarity of a process, additive decompositions of a process
- ◊ Explain a covariance functions
- ◊ Visualize the spatial structure
- ◊ Make prediction using kriging in R, define kriging

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter11.R.

11.1 Introduction

Spatial data summarizes measurements, observations, or data in general that have an inherent geographical location attribute attached. Examples are:

- Cadmium content in the sediment of Lake Geneva (Example 11.1)
- Total precipitation in October 2017 at all SwissMetNet stations (www.meteoschweiz.admin.ch/home/mess-und-prognosesysteme/bodenstationen/automatisches-messnetz.html)
- Number of Bluetongue (en.wikipedia.org/wiki/Bluetongue_disease) cases in cattle in the 150+ districts of Switzerland
- Number of births in the municipalities of the Canton of Zurich from 2000 to 2010

- Improvised Explosive Devices (IEDs) or roadside bombs along a US supply road in Baghdad between February 13th and March 10th 2007.
- Farms on which Bluetongue disease has been diagnosed

In these examples, a correlation between the data seems to exist, reflecting W. Tobler's (http://en.wikipedia.org/wiki/Waldo_Tobler) *first law of geography*: “Everything is related to everything else, but near things are more related than distant things” (Tobler, 1970).

In all six examples, the data is associated with space. Although we have some time attribute, it cannot be resolved. However, there are fundamental differences in these examples. In the first two examples, we can obtain measurements at any arbitrary spatial location within our spatial domain (lake Geneva, Switzerland). With additional effort, it would be possible to obtain more measurements, thus increasing the density of observations. Often, the measurement is representative of a tiny area or volume, which can be neglected compared to the entire spatial domain. In the second set of examples, a value represents a fixed geographic region (districts or municipalities of Switzerland) and it is often impossible to obtain additional readings (at a sub-regional scale or for a differently defined region). The regions are often induced by political boundaries (counties, cantons, ...) or defined historically (US ZIP codes). The last two examples represent recordings of spatial location for some sort of “event”. Other classical types of such data are the position of a tree in a certain study area, sightings of animals, etc. The occurrence of the event defines the location and often there we have no influence of where the event occurs.

We often refer to these three types of data as geostatistical data, lattice data, and spatial point patterns. All three types have similar underlying modeling approaches. In this chapter, we mainly discuss geostatistical data and shortly comment on lattice data.

11.2 Geostatistical Data

As seen in the last chapter, in time series analysis one of the main concerns is forecast (prediction of an unobserved quantity in the future). In the analysis of spatial processes, similarly, one of the main concerns is interpolation, prediction of an unobserved quantity in the domain. However, typically only a single “sample” is available. This implies different methodologies compared to classical multivariate statistics (as seen in earlier chapters).

Throughout the chapter, we use the following dataset.

Example 11.1. (*leman* data) We consider mercury content in the sediment of Lake Geneva taken in 1983. The data has been discussed in Furrer and Genton (1999). Figure 11.1 shows the bathymetry of Lake Geneva as well as major cities and rivers. Figure 11.2 shows the 293 sampling locations as well as the observed values (color coded). There is one large value measured next to the stream mouth of the river Rhone. Additionally, there is also a large value measured next to a large purification plant in the greater Lausanne area. The spatial structure in the data is clearly visible.



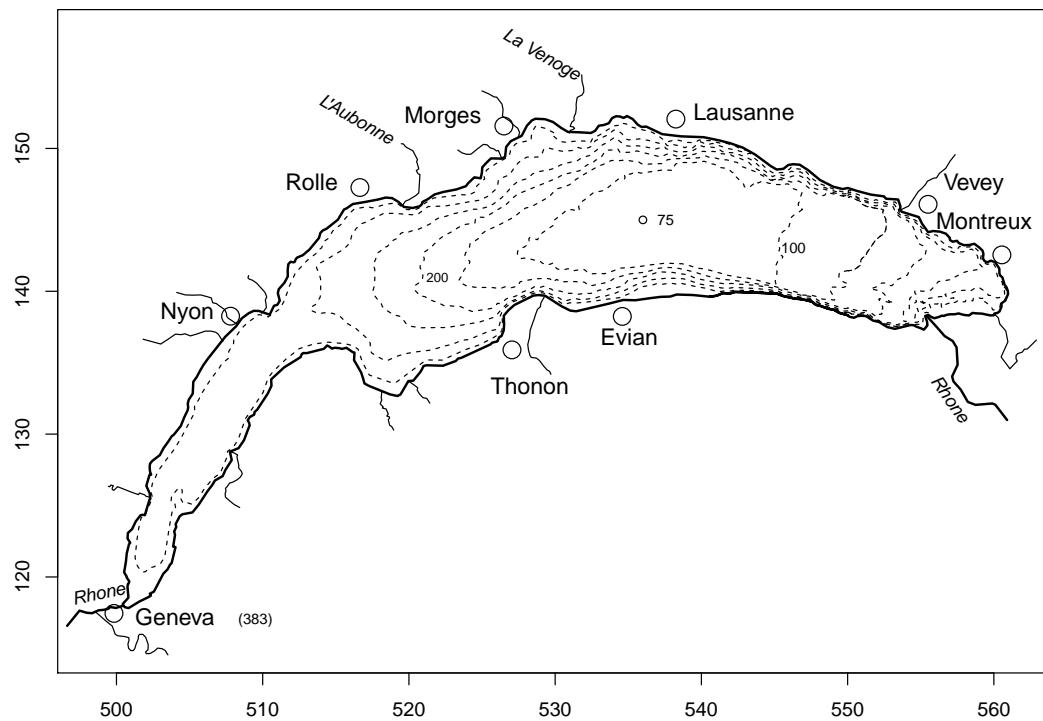


Figure 11.1: Lake Geneva bathymetry (Swiss coordinate system CH1903).

R-Code 11.1 `leman` data: EDA (see Figure 11.2).

```

load("download/sta121/leman83_red.RData")
str(leman)

## 'data.frame': 293 obs. of 10 variables:
## $ x : num 501 502 501 503 502 ...
## $ y : num 119 120 121 121 122 ...
## $ Zn: num 72.2 98.2 81.6 131 160 125 48 131 127 106 ...
## $ Cr: num 49.3 45.5 72.8 63.6 66.8 55 11.1 59.4 67.1 43.6 ...
## $ Cd: num 0.23 0.37 0.14 0.3 0.56 0.3 0.17 0.44 0.39 0.26 ...
## $ Co: num 22.1 8.5 14 4.6 5.7 9.5 7.5 4.5 19.6 6.6 ...
## $ Sr: num 345 308 240 383 356 349 350 374 345 419 ...
## $ Hg: num 0.17 0.21 0.06 0.24 0.35 0.14 0.08 0.26 0.23 0.18 ...
## $ Pb: num 11 15 15 18 24 22 7 19 23 7 ...
## $ Ni: num 32.5 33.8 55.9 47.1 52.5 41.8 11.7 45.5 50.6 32.8 ...

library(fields)
plot(y ~ x, col=tim.colors()[cut(Hg,64)], pch=19, data=leman)
# simpler alternative is quilt.plot
lines(lake.data)
with(hist(Hg, breaks=20, main=""), data=leman)

```

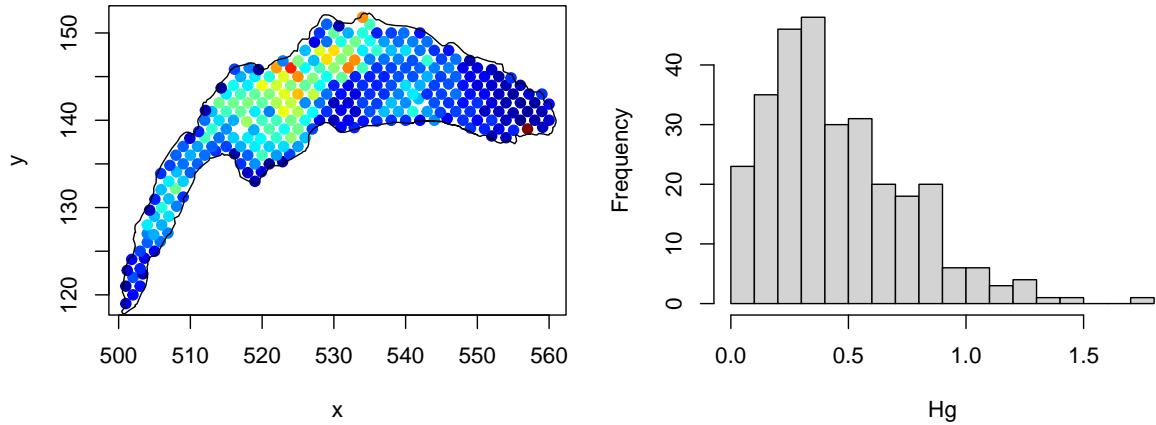


Figure 11.2: `leman` data: Visualization of the variable Hg .

We start by defining a stochastic process, which extends the concept of a random vector. The set of random variables

$$\{Z(s) : s \in \mathcal{D} \subset \mathbb{R}^d, d \geq 1\} \quad (11.1)$$

is called a spatial process (or random process or simply process). We call \mathcal{D} the spatial domain.

The spatial process $Z(\cdot)$ defines, for every $s \in \mathcal{D}$, a random variable $Z(s)$. The domain \mathcal{D} can be seen as an index set and, in practice, is often a region (a subset of \mathbb{R}^2), i.e., $d = 2$.

A realization of the spatial process $Z(\cdot)$, denoted by $z(\cdot)$, is often called the sample surface, or the sample path. In practice, we observe the sample surface at a finite number of locations in the domain. We denote with

$$\{z(s_i) : s_1, \dots, s_n \in \mathcal{D}\}, \quad (11.2)$$

the set of data measured at the locations s_1, \dots, s_n in a domain \mathcal{D} . The sample (11.2) can be seen as a subset of a potentially infinite number of measurements $\{z(s) : s \in \mathcal{D}\}$.

It is not trivial to construct genuine spatial processes. While finite dimensional joint distributions would be available, these need to satisfy specific properties to form a process. These properties ensure that finite dimensional subsets of the spatial process are coherent. In this chapter as well as very often in practice, we consider Gaussian spatial processes. That means that for every $s_0 \in \mathcal{D}$, $Z(s_0)$ is a normal random variable.

11.2.1 Stationarity

Even after fixing the distributions of a spatial process to normal random variables, we still have a lot of flexibility for means, variances and covariances. Stationarity helps to reduce this flexibility by adding structure to the first two moments, i.e., to the mean, variance and covariances. As in time series, the concept of stationarity is important.

Second-order stationarity means constant mean and variance and a covariance structure that is translation invariant. More formally:

Definition 11.1. A spatial process $Z(\cdot)$ is stationary if the following moments exist and satisfy

$$\mathbb{E}(Z(s_1)) \equiv \mu, \quad (11.3)$$

$$\text{Cov}(Z(s_1), Z(s_2)) = c(s_1 - s_2), \quad s_1, s_2 \in \mathcal{D}. \quad (11.4)$$

The function $c(\cdot)$ is called the covariance function and is a function of $s_1 - s_2$ only. \diamond

This restriction on the first two moments of a stationary process is needed as we often observe a single observation, i.e., we observe the sample surface *once* at a finite subset of spatial locations. Often, we further restrict the covariance function, as illustrated in the next definition.

Definition 11.2. If the covariance function $c(\cdot)$ is a function of the distance $\|s_1 - s_2\|$ only, then the spatial process is called isotropic, otherwise it is called anisotropic. The difference $\mathbf{h} = s_1 - s_2$ is called the (spatial) lag and we let $h = \|\mathbf{h}\|$. \diamond

Remark 11.1. 1. Stationarity as introduced above is often referred to as *second-order stationarity* because it only constraints the first two moments; some literature also uses the term *weak stationarity*. If for any location s in the domain the distribution of $Z(s)$ is identical we use the term *strong stationarity*.

2. In geostatistics, we often relax the assumption given in (11.4) to

$$\text{Var}(Z(s_1) - Z(s_2)) = 2\gamma(s_1 - s_2), \quad s_1, s_2 \in \mathcal{D}. \quad (11.5)$$

That is, if the variance of the differences can be written as a function of $s_1 - s_2$ only. Such a process is said to be intrinsically stationary (or to satisfy the intrinsic hypothesis). The functions $2\gamma(\cdot)$ and $\gamma(\cdot)$ are called the variogram and the semivariogram respectively. If $\gamma(\cdot)$ is a function of $\|\mathbf{h}\|$ only, then the process as well as $\gamma(\cdot)$ is called isotropic, otherwise anisotropic.

3. Strong stationarity implies second-order stationarity if the second moment of $Z(\cdot)$ is finite. Second-order stationarity implies intrinsic stationarity. For the inverse to hold, we need to impose that $\gamma(\mathbf{h})$ is bounded for all \mathbf{h} .
4. Stationarity is an assumption made by the statistician and it cannot be proven or rejected with the data itself — at most we could show that we cannot reject the hypothesis.
5. Traditionally, we distinguish between geometric anisotropy and zonal anisotropy. An anisotropy is said to be geometrical when it can be reduced to isotropy by a linear transformation; any other form of anisotropy is zonal. Geostatisticians often suppose a geometric anisotropy when the contours of the (empirical) variogram form an ellipsoid (here the variogram as a function of $\mathbf{h} = (h_x, h_y)$). Such processes can be transformed into isotropic processes by means of matrix multiplication: if the matrix \mathbf{O} describes the corresponding ellipsoid then the process $Z(\mathbf{O}s)$ is isotropic.



11.2.2 Additive Decomposition

When applying statistical procedures it is often necessary to decompose the process $Z(\cdot)$ into several parts in order to obtain stationary elements. The most commonly used and widely accepted decomposition for spatial processes is based on the additive separation according to different scales.

$$Y(\mathbf{s}) = \mu(\mathbf{s}) + Z(\mathbf{s}) + \varepsilon(\mathbf{s}), \quad \mathbf{s} \in \mathcal{D}, \quad (11.6)$$

where each term accounts for a variation at a certain scale, namely:

- $\mu(\cdot) = E(Y(\cdot))$ is the deterministic mean structure, called large-scale variation;
- $Z(\cdot)$ is a zero-mean, stationary process. The process $Y(\cdot)$ is called smooth small-scale variation;
- $\varepsilon(\cdot)$ is a zero-mean white-noise process, independent of $Z(\cdot)$ and is considered as measurement error. We often assume $\varepsilon(\mathbf{s}) \sim \mathcal{N}(0, \sigma^2)$.

The large-scale variation $\mu(\cdot)$ is also called trend or drift.

This additive decomposition is very much in the spirit of mixed models. Equation (11.6) has *two* stochastic components on the right-hand-side. Compared to a classical mixed model setup, as given in equation (6.4), we have only one level, but a multivariate observation. For identifiability, similar to the mean of the random effects, we have to assume zero-mean for $Z(\cdot)$.

The estimation of $\mu(\cdot)$ is crucial and often involves subjective decisions. There exist basically two ways of treating the mean structure $\mu(\cdot)$. The first supposes that $\mu(\cdot)$ is a linear combination of some functions $\{x_k(\cdot)\}_{k=0}^p$. This means, that for a particular location \mathbf{s}_0 , the trend is $\mu(\mathbf{s}_0) = \beta_0 + \beta_1 x_1(\mathbf{s}_0) + \beta_2 x_2(\mathbf{s}_0) + \dots + \beta_p x_p(\mathbf{s}_0)$.

The second approach consists in estimating $\mu(\cdot)$ and subtracting it from $y(\cdot)$. In this case, the estimation neglects the dependency of the spatial process, but the resulting difference is a stationary process. The trend $\mu(\cdot)$ can be modeled via a parametric or a non-parametric approach.

11.2.3 Covariance Functions

If the process is stationary we can use the covariance function $c(\cdot)$ defined in (11.4) to describe the spatial dependence structure of the process.

We cannot use an arbitrary function as a possible covariance function because a valid covariance has to be a positive (semi-)definite function in order to ensure

$$\text{Var}\left(\sum_{i=1}^n \alpha_i Z(\mathbf{s}_i)\right) = \sum_{i,j=1}^n \alpha_i \alpha_j \text{Cov}(Z(\mathbf{s}_i), Z(\mathbf{s}_j)) = \sum_{i,j=1}^n \alpha_i \alpha_j c(\mathbf{s}_i - \mathbf{s}_j) \geq 0 \quad (11.7)$$

for all $\alpha_1, \dots, \alpha_n$.

There is a lot of mathematical theory on how to construct valid covariance functions. In practice only a few are used. One example is the so-called exponential covariance function, given by

$$c(\mathbf{h}; \boldsymbol{\theta}) = \theta_1 \exp(-\|\mathbf{h}\|/\theta_2), \quad \theta_1 > 0, \quad \theta_2 > 0. \quad (11.8)$$

The parameters of the covariance function have particular names. θ_1 is called the partial sill and θ_2 is related to the range, i.e., the distance at which the covariance vanishes or essentially vanishes. For covariance functions that only asymptotically reach zero, we often use the term practical range, which is defined as the distance at which the covariance function is at 5% of the partial sill.

Hence, we have for an arbitrary location $\text{Var}(Y(\mathbf{s}_0)) = \text{Var}(Z(\mathbf{s}_0)) + \text{Var}(\varepsilon(\mathbf{s}_0)) = \theta_1 + \sigma^2$, termed the total sill. The variance of the measurement error has been historically called the nugget or nugget effect (Matheron, 1962).

The covariance function of the spatial process $Y(\cdot)$ is as for $Z(\cdot)$, except at 0, where it takes the value $\theta_1 + \sigma^2$. Hence, in the presence of measurement error, the covariance function is discontinuous.

R-Code 11.2 and Figure 11.3 illustrate typical covariance functions. Notice that the functions look all very similar and that choosing a model may not be trivial. The so-called spherical covariance function is zero beyond the practical range. The different Matérn have a different behavior at lag zero, which has implications on the smoothness of the sample surface.

R-Code 11.2 Often used covariance functions. (See Figure 11.3.)

```
library(geoR) # providing the function `cov.spatial`
curve(cov.spatial(x, cov.pars=c(1, .25)),
       xlab="Distance", ylab=expression(c(h)))
curve(cov.spatial(x, cov.pars=c(1, .75), cov.model="sph"),
       add=TRUE, lty=2,col=2)
curve(cov.spatial(x, cov.pars=c(1, 0.188), kappa=1), add=TRUE, lty=3,col=3)
curve(cov.spatial(x, cov.pars=c(1, 0.14), kappa=2), add=TRUE, lty=4,col=4)
curve(cov.spatial(x, cov.pars=c(1, 0.117), kappa=3), add=TRUE, lty=5,col=5)
abline(v=.75, col="gray")
legend("topright", c("Exponential", "Spherical", "Matern (v=1)",
                     "Matern (v=2)", "Matern (v=3)"), lty=1:5, col=1:5, bty="n")
```

11.2.4 Estimation of the Second Moment Structure

If the underlying distribution of the random field is known and if the covariance matrix is an explicit function of the parameter $\boldsymbol{\theta}$, maximum likelihood estimation (MLE) methods can be used to determine all parameters. This approach is computationally quite intensive and *method-of-moments* type estimation approaches are often used. For simplicity assume that the process Z is stationary.

An alternative approach is to estimate the covariance function at certain lags and then fitting a parametric function to the estimates. For example, the covariance function at lag \mathbf{h} can be

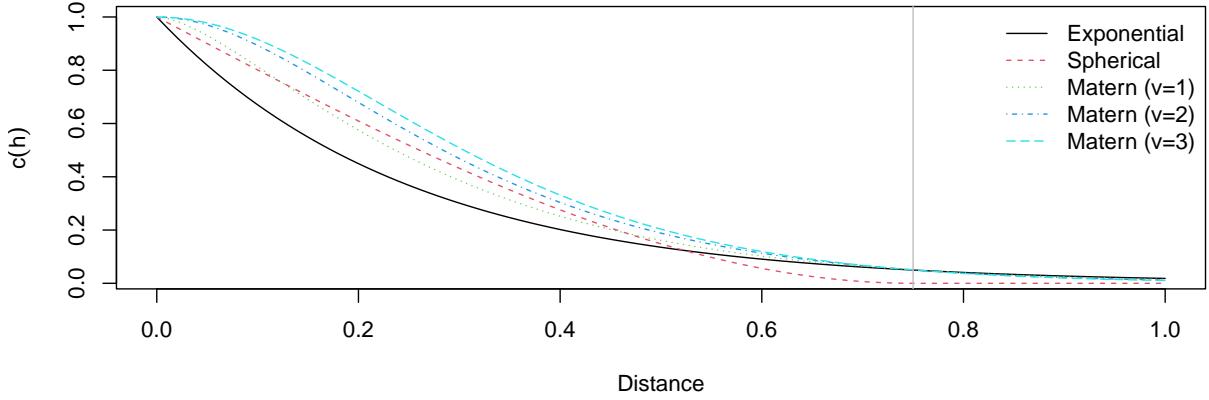


Figure 11.3: Examples of different covariance functions. The black curve is given by (11.8). The covariance functions have nugget effect 0, partial sill 1 and (practical) range 0.75. (See R-Code 11.2.)

estimated by

$$\hat{c}(\mathbf{h}) = \frac{1}{N_J} \sum_{(i,j) \in J} (Z(\mathbf{s}_i) - \bar{Z})(Z(\mathbf{s}_j) - \bar{Z}), \quad (11.9)$$

where J contains all pairs (i, j) for which \mathbf{s}_i and \mathbf{s}_j are approximately \mathbf{h} apart, N_J is the number of such pairs and $\bar{Z} = \sum_i Z(\mathbf{s}_i)/n$ (see Remark 11.2 below).

However, in practice a different and more convenient approach is used, which is based on a function called the semivariogram, which is linked to the covariance function. Section 11.2.5 contains more details, which are not essential for the understanding of the remaining part of the chapter.

Definition 11.3. For a stationary process $Z(\cdot)$ with covariance function $c(\cdot)$, the semivariogram $\gamma(\cdot)$ is defined by

$$\gamma(\mathbf{h}) = c(\mathbf{0}) - c(\mathbf{h}), \quad (11.10)$$

where $\mathbf{0}$ is the null vector. \diamond

In other words, the semivariogram is a reflection of the covariance function (at the line $(\theta_1 + \sigma^2)/2$) and can also be defined by

$$\gamma(\mathbf{s}_1 - \mathbf{s}_2) = \frac{1}{2} \text{Var}(Z(\mathbf{s}_1) - Z(\mathbf{s}_2)) = E((Z(\mathbf{s}_1) - Z(\mathbf{s}_2))^2), \quad (11.11)$$

because a stationary process has constant mean. Based on this last expression, the following estimator is quite intuitive:

$$\hat{\gamma}(\mathbf{h}) = \frac{1}{2} \frac{1}{N_J} \sum_{(i,j) \in J} (Z(\mathbf{s}_i) - Z(\mathbf{s}_j))^2, \quad (11.12)$$

where J is again an index set, similar as used in (11.9).

This estimator is often called Matheron's classical estimator of the semivariogram (Matheron, 1962) and has appealing properties. For example, if the vector \mathbf{z} denotes the observations, we can write an estimate of (11.12) as $\hat{\gamma}(\mathbf{h}) = \mathbf{z}^\top \mathbf{A}(\mathbf{h}) \mathbf{z}$ for some matrix $\mathbf{A}(\mathbf{h})$.

Many other estimators exist and are implemented in common statistical software routines. These usually try to overcome the classical estimator's sensitivity towards outliers.

Remark 11.2. 1. The index set J is formally described by

$$J = J(\mathbf{h}) = \{(i, j) : \mathbf{s}_i - \mathbf{s}_j \in T(\mathbf{h})\}, \quad N_J = \text{card}\{J\} \quad (11.13)$$

and $T(\mathbf{h})$ is some specified tolerance region in \mathbb{R}^d . A typical tolerance region is $T(\mathbf{h}) = T_{l,a}(\mathbf{h}) = \{\mathbf{k} : \|\mathbf{k} - \mathbf{h}\| \leq l, \arccos(\mathbf{k}^\top \mathbf{h} / (\|\mathbf{k}\| \|\mathbf{h}\|)) \leq a\}$.

2. It is recommended that variograms should not be estimated for lags bigger than half the diameter of the domain. Further, all lags \mathbf{h} should be chosen such that at least thirty pairs should be included ([Journel and Huijbregts, 1978](#)).
3. We can estimate the variance along a given axis \mathbf{h} , i.e., estimate a directional variogram, by using an appropriate tolerance region $T(\mathbf{h})$. Directional variograms are an indispensable tool in exploratory variography. Besides the indication of trends and other types of nonstationarity, they allow for calculation of the transformation matrix \mathbf{O} of geometrical anisotropy.



Once the empirical covariance or semivariogram has been estimated, we fit an appropriately chosen parametric model to the empirical counterparts. This so-called semivariogram fitting approach is as follows. Suppose we have an empirical semivariogram $\hat{\gamma}(\mathbf{h}_k)$, for different lags \mathbf{h}_k , $k = 1, \dots, K$, and an appropriate family $\gamma(\mathbf{h}; \boldsymbol{\theta})$. The method of ordinary least squares estimates $\boldsymbol{\theta}$ by means of

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{k=1}^K (\hat{\gamma}(\mathbf{h}_k) - \gamma(\mathbf{h}_k; \boldsymbol{\theta}))^2. \quad (11.14)$$

to get a good match between the estimated and the theoretical counterpart. Of course more complicated approaches exist, e.g., by incorporating weights that take into account the number of pairs in the individual bins, etc.

Example 11.2. R-Code 11.3 illustrates the estimation of the semivariogram of the [leman](#) data. We assume an exponential model (which would be the default for many of the functions). Graphically, the resulting fit seems adequate (see Figure 11.4).

The estimates are $\sigma^2 = 0.039$ ([tausq](#)), $\theta_1 = 0.077$ ([sigmasq](#)), and $\theta_2 = 11.64$ ([phi](#)).

Note that the variance of the variable ([var\(leman\\$Hg\)](#)) is smaller than the total sill. The total variance is a weighted average of the empirical semivariogram.



R-Code 11.3: [leman](#) data semivariogram estimation and fitting. (See Figure 11.4.)

```
library(geoR)
vg <- variog(coords=leman[, c("x", "y")], data=leman$Hg, max.dist=40)
```

```

## variog: computing omnidirectional variogram
plot(vg)           # visualize empirical variogram
abline(h=var(leman$Hg), col="gray") # sample variance

### Fitting the empirical variogram
fit <- variofit(vg, ini.cov.pars=c(.1, .1), cov.model="exponential")

## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim

fit

## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: exponential
## parameter estimates:
##   tausq sigmasq     phi
## 0.0392 0.0771 11.6394
## Practical Range with cor=0.05 for asymptotic range: 34.86849
##
## variofit: minimised weighted sum of squares = 2.3477

lines(fit)          # adding fitted variogram
points(0, fit$nugget+fit$cov.pars[1], pch=20, col=4)
curve( cov.spatial(x, cov.pars=fit$cov.pars), xlim=c(0,35),
       add=T, col=4)          # adding fitted covariance function

```

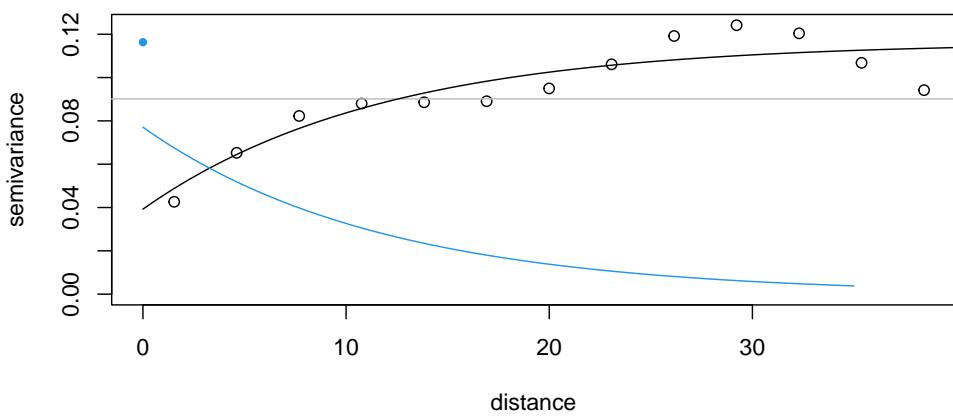


Figure 11.4: Spatial dependence structure for `leman` data: black circles represent the empirical semivariogram. Black line is the fitted exponential model. Blue line (including the dot) is the corresponding covariance function. Gray horizontal line represents the marginal variance of the data. (See R-Code 11.3.)

11.2.5 Semivariograms

For every intrinsically stationary process the semivariogram exists and is uniquely defined. However, a process for which $\gamma(\cdot)$ is its semivariogram does not exist for every function $\gamma(\cdot)$. A semivariogram $\gamma(\cdot)$ has to satisfy certain properties. In practice, we choose among a few parameterized families (as we choose between different families of distributions).

Similarly to the covariance function, the parameters of a semivariogram are linked to the partial sill (if it exists), and the range (if it exists).

Consider the following widely used, isotropic semivariograms, parameterized using partial sill $\theta_1 > 0$ and range parameter $\theta_2 > 0$.

- The spherical semivariogram

$$\gamma(\mathbf{h}; \boldsymbol{\theta}) = \begin{cases} \theta_1 \left(3/2(\|\mathbf{h}\|/\theta_2) - 1/2(\|\mathbf{h}\|/\theta_2)^3 \right), & \text{if } \|\mathbf{h}\| \leq \theta_2, \\ \theta_1, & \text{otherwise.} \end{cases} \quad (11.15)$$

- The exponential semivariogram

$$\gamma(\mathbf{h}; \boldsymbol{\theta}) = \theta_1 (1 - \exp(-\|\mathbf{h}\|/\theta_2)), \quad (11.16)$$

- The Matérn semivariogram

$$\gamma(\mathbf{h}; \boldsymbol{\theta}) = \theta_1 (1 - (\|\mathbf{h}\|/\theta_2)^\nu \mathcal{K}_\nu(\|\mathbf{h}\|/\theta_2)), \quad (11.17)$$

where \mathcal{K}_ν is the modified Bessel function of the second kind of order $\nu > 0$ ([Abramowitz and Stegun, 1970](#)).

For certain ν the Matérn model (11.17) has appealing forms. For example, if $\nu = 0.5$, it reduces to an exponential model. The practical range of the Matérn model depends on the parameter θ_2 and ν . As ν decreases it approaches a pure nugget effect model.

The last two semivariograms attain their maximum value only asymptotically, i.e., have an infinite range.

11.2.6 Spatial Prediction: Kriging

If we wish to make predictions at a location not included in the observed locations, for the purpose of establishing a pollution map for example, all measures will have to be taken into account in computing the predicted value. Of course, their contributions will be weighted by the strength of their correlation with the location of interest.

Hence, a reasonable prediction of $Z(\mathbf{s}_0)$, for simplicity noted by $\widehat{Z}(\mathbf{s}_0)$ is a linear combination of all observations. More specifically, we assume the linear predictor

$$\widehat{Z}(\mathbf{s}_0) = \sum_{i=1}^n \lambda_i Z(\mathbf{s}_i) = \boldsymbol{\lambda}^\top \mathbf{Z}, \quad (11.18)$$

where the weights $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)^\top$ are chosen such that the predictor is unbiased and that the mean squared prediction error

$$E\left(\left(Z(\mathbf{s}_0) - \sum_{i=1}^n \lambda_i Z(\mathbf{s}_i)\right)^2\right) \quad (11.19)$$

is minimized. With $\boldsymbol{\lambda}$ chosen under these constraints the estimator (11.18) is called the ordinary kriging predictor and is the best linear unbiased predictor (BLUP).

The optimal weights sum to one (i.e., satisfy $\mathbf{1}^\top \boldsymbol{\lambda} = 1$) to guarantee unbiasedness and minimizing equation (11.19) (under the constraint $\mathbf{1}^\top \boldsymbol{\lambda} = 1$ by means of Lagrange multipliers) leads to

$$\boldsymbol{\lambda} = \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\sigma} + \mathbf{1} \frac{1 - \mathbf{1}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\sigma}}{\mathbf{1}^\top \boldsymbol{\Sigma}^{-1} \mathbf{1}} \right). \quad (11.20)$$

where $\boldsymbol{\Sigma} = (c(s_i - s_j))$, $\boldsymbol{\sigma} = (c(s_0 - s_i))$.

The exact form of the ordinary kriging weights (11.20) is intimidating. To get a better understanding, recall the formula of conditional expectation for a Gaussian random vector (D.18) and replace the unknown mean with the generalized least squares (GLS) estimator. The GLS estimator for the mean is

$$\hat{\mu}_{\text{GLS}} = \mathbf{1}^\top \frac{1}{\mathbf{1}^\top \boldsymbol{\Sigma}^{-1} \mathbf{1}} \mathbf{Z}. \quad (11.21)$$

Hence

$$\hat{Z}(s_0) = \hat{\mu}_{\text{GLS}} + \boldsymbol{\sigma}^\top \boldsymbol{\Sigma}^{-1} (\mathbf{Z} - \hat{\mu}_{\text{GLS}} \mathbf{1}). \quad (11.22)$$

Now, placing (11.21) in placing (11.22) collecting the terms, we find the weights (11.20).

In a pure nugget effect model, the ordinary kriging predictor is

$$\hat{Z}(s_0) = \begin{cases} \sum_i Z(s_i)/n, & \text{if } s_0 \notin \{s_1, \dots, s_n\}, \\ Z(s_i), & \text{if } s_0 = s_i, \quad i = 1, \dots, n. \end{cases} \quad (11.23)$$

The minimized mean squared prediction error (11.19) is often called the kriging variance and is given by

$$\text{Var}(\hat{Z}(s_0)) = c(0) - \boldsymbol{\sigma}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\sigma} + \frac{(1 - \mathbf{1}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\sigma})^2}{\mathbf{1}^\top \boldsymbol{\Sigma}^{-1} \mathbf{1}}. \quad (11.24)$$

Remark 11.3. 1. If the mean $\mu(\cdot)$ and the covariance function $c(\cdot)$ of the spatial process $Z(\cdot)$ are known functions, the optimal predictor simplifies to

$$\hat{Z}(s_0) = \mu(s_0) + \boldsymbol{\sigma}^\top \boldsymbol{\Sigma}^{-1} (\mathbf{Z} - \boldsymbol{\mu}). \quad (11.25)$$

where $\boldsymbol{\mu} = (\mu(s_i))$. The predictor (11.25) is called the simple kriging predictor. For Gaussian processes, simple kriging coincides with $E(Z(s_0) \mid Z(s_1), \dots, Z(s_n))$ and is therefore optimal amongst all unbiased predictors.

2. If the deterministic mean structure can be expressed as

$$\mu(\cdot) = \sum_{k=0}^K \beta_k x_k(\cdot), \quad (11.26)$$

for known functions $x_0(\cdot)$, then the kriging weights can be generalized as follows. The vector $\mathbf{1}$ is replaced by the design matrix \mathbf{X} induced by the functions $\{x_k(\cdot)\}_{k=0}^K$ evaluated at the spatial locations. This type of prediction is often referred to as universal kriging.



Theoretically kriging is an optimal predictor in the sense that it minimizes the estimation variance under the unbiasedness constraint when the variogram is known and the process is second-order stationary. In practice, these conditions are rarely met. Nevertheless, kriging has become a popular interpolator in the statistical community.



Example 11.3. R-Code 11.4 illustrates kriging using the package *fields*. Estimation and fitting is done in a almost blackbox-fashion in the function *Krig()*. We assume a linear trend surface (here linear in the coordinates). Figure 11.4 shows the resulting interpolated (seemingly smooth) surface, which is decomposed into the trend and the spatial process.



R-Code 11.4 *leman* prediction. (See Figure 11.5.)

```
### Construct a grid within the lake boundaries:
library(splancs)
xr <- seq(min(lake.data[,1]), to=max(lake.data[,1]), l=100)
yr <- seq(min(lake.data[,2]), to=max(lake.data[,2]), by=xr[2]-xr[1])

locs <- data.frame(x=lake.data[,1], y=lake.data[,2])
grid <- expand.grid(x=xr, y=yr)
pts <- pip(grid, locs, bound=TRUE) # get Point-In-Polygon
#
### Perform kriging:
with({
  out <- Krig(cbind(x,y), Hg, theta=11.64, give.warnings=FALSE)
  pout <- predict(out, pts)
  quilt.plot(pts, pout, nx=length(xr)-2, ny=length(yr)-1)

  fit0 <- predict(out, pts, just.fixed=TRUE) # trend
  quilt.plot(pts, fit0, nx=length(xr)-2, ny=length(yr)-1)

  fit1 <- pout - fit0                      # smooth spatial
  quilt.plot(pts, fit1, nx=length(xr)-2, ny=length(yr)-1)
}, data=leman)
```

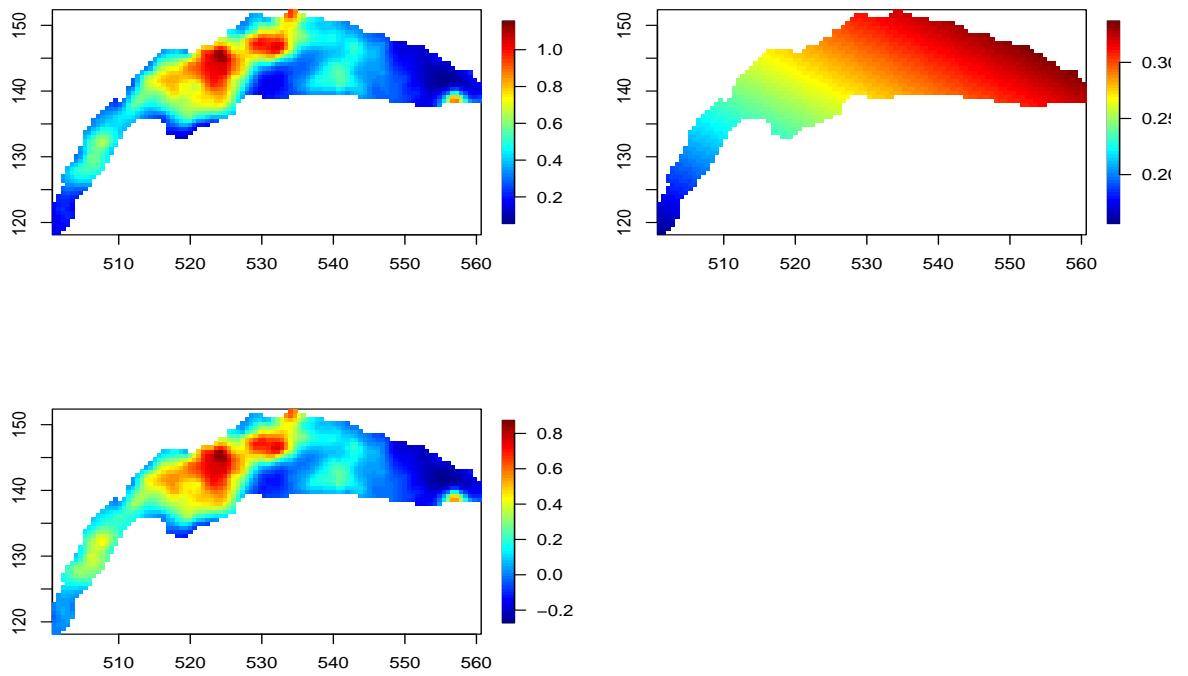


Figure 11.5: [leman](#): prediction (top left), trend (top right), and smooth spatial component (bottom). (See R-Code 11.4.)

11.3 Lattice Data

In the case of lattice data, the lattice structure does not change. Hence, we can work with “classical” multivariate distributions and we do not have to work with processes because the dimension is finite and fixed. Further, we typically have a datum or observation for all the lattice elements and prediction is seen as smoothing. Often the data are modeled according to some distribution where the parameters thereof vary smoothly over space. Classical analysis evaluate if such spatially structured terms are present and if the structure can be attributed to other factors.

11.4 Bibliographic remarks

The book by [Cressie \(1993\)](#) is often seen as a bible. Other older titles are [Journel and Huijbregts \(1978\)](#) and [Ripley \(1981\)](#). At the end of last century many books in spatial statistics were published, including [Chilès and Delfiner \(1999\)](#), [Olea \(1999\)](#), [Webster and Oliver \(2001\)](#), and [Schabenberger and Gotway \(2005\)](#).

Multivariate geostatistics is covered in [Wackernagel \(1995\)](#). Space-time geostatistics is covered, e.g., in [Sherman \(2011\)](#).

[Stein \(1999\)](#) is a very theoretical gem.

Kriging is a minimum mean squared error method of spatial prediction. Matheron (1963) named this method of optimal spatial linear prediction after D. G. Krige, a South African mining engineer who developed empirical methods for determining ore-grade distributions from samples (Krige, 1951). However, the formulation of optimal linear prediction did not come from him. Wold (1938), Kolmogoroff (1941) and Wiener (1949) developed optimal linear prediction, where closer points obtained more weight than distant points. Extensions from simple kriging to universal kriging are covered in Goldberger (1962) and Matheron (1969). For a more thorough overview of the historical origins of kriging, refer to Cressie (1990).

There are many R packages that deal with spatial data. See the CRAN Task View: [Analysis of Spatial Data](#) for an overview. We personally work a lot with `fields` and `geoR`.

11.5 Exercises and Problems

Problem 11.1 (Visualizing spatial data) The (artificial) data set `spatialSim.RData` is available from <https://www.math.uzh.ch/furrer/download/sta121/spatialSim.RData>. It contains two spatial fields on a regular 100×100 grid.

- a) Load the data and plot both fields using `image.plot()` from package `fields`. Describe and compare both fields via these plots and summary statistics.
- b) Sketch covariance functions for `sim1` and `sim2` by hand. Write in a few words in what way the two curves differ.
Hint: x-axis: distance between points h , y-axis: covariances $c(h)$.
- c) Use linear regression to investigate if there is an overall trend in `sim1` with respect to the coordinates `x1` and `x2`.
Hint: Convert the data to long format, i.e., a `data.frame` having the column names `c("x1", "x2", "z")`. Then regress `z` on the other variables.
- d) Discuss the assumptions of a linear models. Which ones are violated in c.?
- e) How could the regression setting from c be extended to take into account the spatial dependency of `z`? Give a model formulation (equation) and describe the used components.

Problem 11.2 (Simple kriging on a transect) The data set `transect.txt` is available from <https://www.math.uzh.ch/furrer/download/sta121/transect.txt>. It consist of observations of a spatial process $Z(s_i)$ at locations s_i , where $i = 1, \dots, 5$. s_i are locations along a transect between 0 and 10.

- a) Have a look at the data and plot them in the interval $[0, 10]$.
- b) Write an R function `exp.cov(h, theta2, theta3)` that implements the exponential covariance function as defined in the script. Plot the function for the distances $h \in [0, 4]$, with nugget $\theta_1 = 0$, sill $\theta_2 = 3$ and range $\theta_3 = 1$. Can you identify these parameters in the plot?
Hint: $\theta_1 = 0$, hence no case discrimination is needed.

- c) Write a function `dist.matrix(x, y)` that returns a matrix of pairwise (Euclidean) distances between each elements of the vectors `x` and `y`. Use this function to calculate `DIST.MAT`, a matrix containing all possible distances of the locations in the transect dataset.
Hint: Use for-loops or `outer()`, and `abs()`.
- d) Use the exponential covariance function as specified in **c.** and `DIST.MAT` from **c.** to calculate a covariance matrix `SIGMA`.
- e) Define a sequence of s_{new} location where the process $Z(s)$ should be predicted. Use (c) to calculate all possible distances between the observed points and the s_{new} . Then calculate `sigma`, the corresponding covariance matrix, using `exp.cov()` from **b.**
- f) Assume known constant mean $\mu = 0$, second-order stationarity, and known covariance function. Calculate the simple kriging prediction for the locations s_{new} . Plot the predicted values of $Z(s)$ as line and the observed values as points.
Hint: Use `solve()` for matrix inversion, and `t()` to get the matrices in an appropriate shape for multiplication.
- g) Repeat **b.** to **f.** with different parameters of to covariance function. What is the influence of θ_2 ?

Chapter 12

Extreme Value Theory

To model extreme events (i.e., the size of the largest flood in a decade) we cannot simply estimate the location and spread of daily runoff and derive relevant statistics. We have to use a different set of statistical tools that do not rely on the central limit theorem. This is the topic of extreme value theory.

Learning goals for this chapter:

- ◊ Explain the motivation for applying extreme value theory
- ◊ Explain the modeling the maxima approach
- ◊ Explain Modeling Peaks over Thresholds approach
- ◊ Explain Return Level Plots
- ◊ Give some examples of applications of the EVT
- ◊ Conduct EVT analysis in R

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter12.R.

12.1 Motivation

Example 12.1. R-Code 12.1 shows the maximum monthly discharge of Rhone river at Porte du Scex since 1970.

Suppose we need to design a dam for flood protection. Based on discharge data, we are not interested in predicting the center of the distribution(mean, mode, . . . , i.e., not something about the red line in the bottom panel) but the maximum discharge (in a possibly distant future).

More precisely, in the 45 years of data, the largest discharge was 202 m^3 . What is the largest discharge that we can expect in the next 50 years with reasonable probability? Of course we need to be more precise what we mean with ‘largest’ and ‘reasonable probability’.



R-Code 12.1 Maximum monthly discharge of Rhone river at Porte du Scex. (See Figure 12.1.)

```

rhone <- read.csv("download/sta121/PorteduScex70.csv")
str(rhone)

## 'data.frame': 549 obs. of 3 variables:
## $ year : int 1970 1970 1970 1970 1970 1970 1970 1970 1970 1970 ...
## $ month: int 1 2 3 4 5 6 7 8 9 10 ...
## $ m3   : num 14.8 12.6 13.5 13.7 14.2 51.8 61.9 51.8 40.7 40.6 ...
plot(rhone$m3, xlab="Time (since 1970-01-01)", ylab="Discharge [m3]")
plot(m3~month, data=rhone, xlab="Month", ylab="Discharge [m3]")
lines(1:12, c(by(rhone$m3, rhone$month, median)), col=2, lwd=2)
rhone[which.max(rhone$m3),]

##      year month      m3
## 285 1993     9 202.16

```

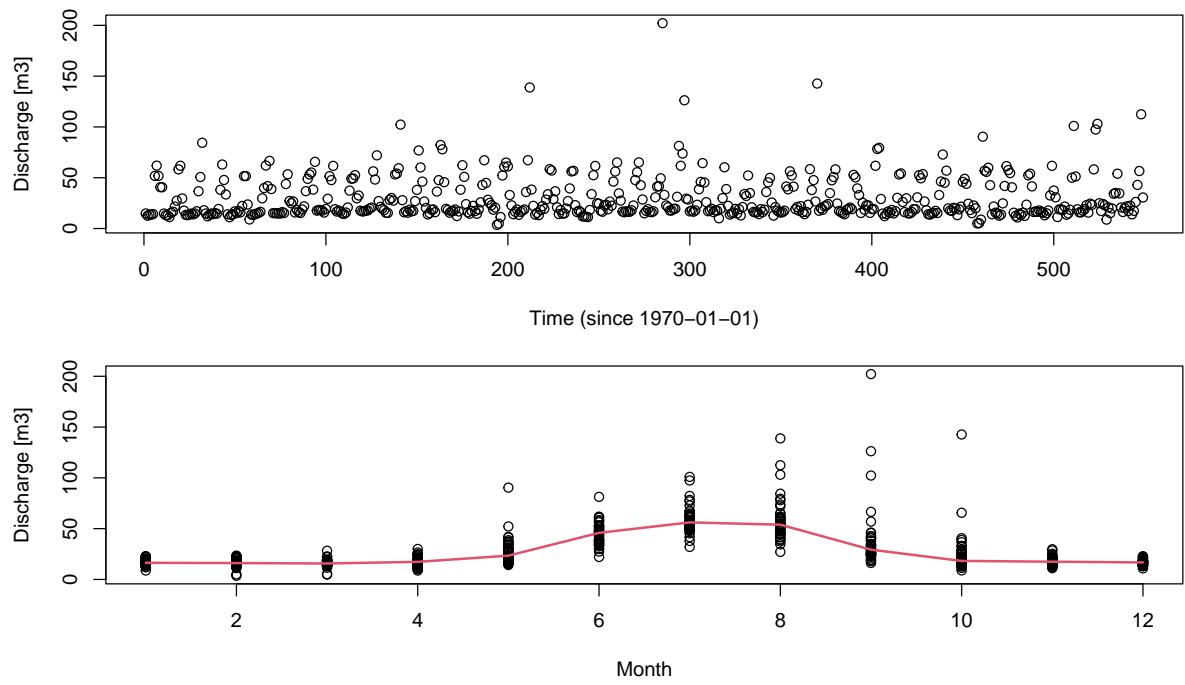


Figure 12.1: Monthly maxima discharge of Rhone river at Porte du Scex from January 1970 to September 2015. Top: time series, bottom: by month. The red line indicates the monthly medians. (See R-Code 12.1.)

When modeling maximum values, it is dangerous to model distributions from first and second order quantities and then extrapolate to quantiles, i.e., we should not perform inference obtained from a model that fits the bulk of the data. Consider the following example. Suppose we observe normal data, say x_1, \dots, x_n from the random sample $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2)$. For the illustration

here, assume that we have zero mean and standard deviation one. The probability that a single value is larger than 4 is $P(X_i > 4) = 1 - \Phi(4) = 3.167 \cdot 10^{-5}$. However, the probability that out of a sample of size $n = 10, 100, 1000$ the maximum exceeds 4 is 0.0003 0.0032 0.0312 and thus (naturally) strongly depends on the sample size.

Denoting M_n the maximum of X_1, \dots, X_n its distribution depends on the distribution and the sample size, as

$$P(M_n \leq x) = P\left(\max_i\{X_i\} \leq x\right) = P(X_1 \leq x, X_2 \leq x, \dots, X_n \leq x) = F_X(x)^n. \quad (12.1)$$

Albeit, this result should not be too surprising, as the distribution of \bar{X} also depends on n : $\bar{X} \sim \mathcal{N}(\mu, \sigma^2/n)$. The next example illustrates this along the minimum, which is – when reversing the “direction” of the axis – conceptually not too different compared to the maximum.

Example 12.2. Let $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} \text{Exp}(\lambda)$ and $M_n = \min_i\{X_i\}$. The distribution of M_n is

$$P(M_n \leq x) = 1 - P(M_n > x) = 1 - (1 - F_X(x \leq x))^n = 1 - \exp(-n\lambda x), \quad (12.2)$$

that means the minimum is again an exponential random variable with rate $n\lambda$. Thus $E(M_n) = 1/(n\lambda)$, which vanishes as the sample size n increases. ♣

The statistical theory of extreme values (maximum, minimum or exceedances of a large threshold) is called extreme value theory (EVT). At the core of extreme value theory is a limit theorem appropriate for the extreme values. These large observations behave asymptotically differently than the central values. More specifically but loosely said, the central limit theorem states that averages converge to a normal distribution whereas the maximum converges to the so-called generalized extreme value distribution (GEV). In both settings we need some weak assumptions such such that the limit exists. However, we do not need to elaborate these technicalities here.

EVT comprises three different approaches that are all interlinked. The first models only the maximum (Section 12.2), the second models data exceeding a threshold (peak over threshold, POT, Section 12.3) and the third uses a Poisson process model for POTs (not discussed here). All three models are interlinked, but depending on the situation some may be more suitable than others. For technical details see the accessible text of [Coles \(2001\)](#).

12.2 Modeling the Maxima

Let X_1, \dots, X_n be iid. We are interested in describing the distribution of the maximum of X_i , i.e., the distribution of $M_n = \max_i X_i$. Under very mild assumptions, the (scaled) distribution of $M_n = \max_i X_i$ converges to a so-called GEV distribution with cdf

$$F(x; \mu, \sigma, \xi) = \exp\left(-\left(1 + \xi\left(\frac{x - \mu}{\sigma}\right)\right)^{-1/\xi}\right), \quad (12.3)$$

for all x such that $1 + \xi(x - \mu)/\sigma > 0$, where $\mu \in \mathbb{R}$ is the location parameter, $\sigma > 0$ the scale parameter and $\xi \in \mathbb{R}$ the shape parameter.

The GEV distribution is actually a family of distribution families. The three sub-families are defined by $\xi > 0$, $\xi < 0$ and $\xi = 0$, and correspond to the Fréchet, Weibull and Gumbel families, respectively. The shape parameter ξ imposes the support of the distribution, specifically for $\xi > 0$, the distribution has lower endpoint as $x > \mu - \sigma/\xi$, and for $\xi < 0$ it has an upper endpoint $x < \mu + \sigma/(-\xi)$, i.e., the support of the random variable has an upper bound. For $\xi = 0$, we do not have any restrictions on the support but the distribution specified in (12.3) should be interpreted by taking the limit as $\xi \rightarrow 0$. The shape parameter ξ also governs the tail behavior of the distribution. The case $\xi > 0$ means that the density has heavy tails, for which not all moments are finite anymore. More specifically, $E(X^k)$ is finite for $k < 1/\xi$ only. Hence, the estimation of ξ is thus crucial.

Example 12.3. Consider $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1)$ for three different $n = 10, 100, 1000$. R-Code 12.2 samples $R = 10000$ times a corresponding sample and calculates the maximum. The resulting empirical densities of M_n are given in Figure 12.2. The mean of these maxima increases from 1.6, to 2.5, to 3.2 as n increases. ♣

R-Code 12.2 Empirical densities of the maximum of Gaussian samples. (See Figure 12.2.)

```
set.seed(1)
R <- 10000
breaks <- seq(-.5, to=5.5, by=.25)
for (n in c(10,100,1000)) {
  samples <- matrix(rnorm(n*R), nrow=R, ncol=n)
  Mn <- apply(samples, 1, max)
  hist(Mn, main="", xlab=paste("n =",n), prob=TRUE, breaks=breaks,
        ylim=c(0,1.2), border=NA, col="lightblue")
  abline(v=mean(Mn), col="blue")
}
```

In practice we hardly ever have replicates of an entire series to estimate the maximum. To circumvent this lack, one subdivides an entire sequence into several shorter sub-sequences, called blocks. From each of these blocks, we take the maxima. Hence, one has to balance block length versus sample size (number of the blocks). For some settings, the choice of the block is natural, as the following example illustrates.

Example 12.4. In the setting of maximum monthly discharge of the Rhone river, we calculate the maximum annual discharge (block length 12). We consider these as independent. R-Code 12.3 illustrates the fitting of a GEV distribution to the annual maxima with maximum likelihood using the R package `extRemes`. The package can handle many different settings and we will only extract relevant information, here parameter estimates and standard errors thereof. A crude Wald-type confidence interval for the shape parameters ξ is obtained via the function `ci()` and yields [0.258, 0.899]; implying “heavy” tails as the lower endpoint is larger than zero. The tails are comparable to a t_2 distribution as the confidence interval suggest finite mean but non-existing variance.

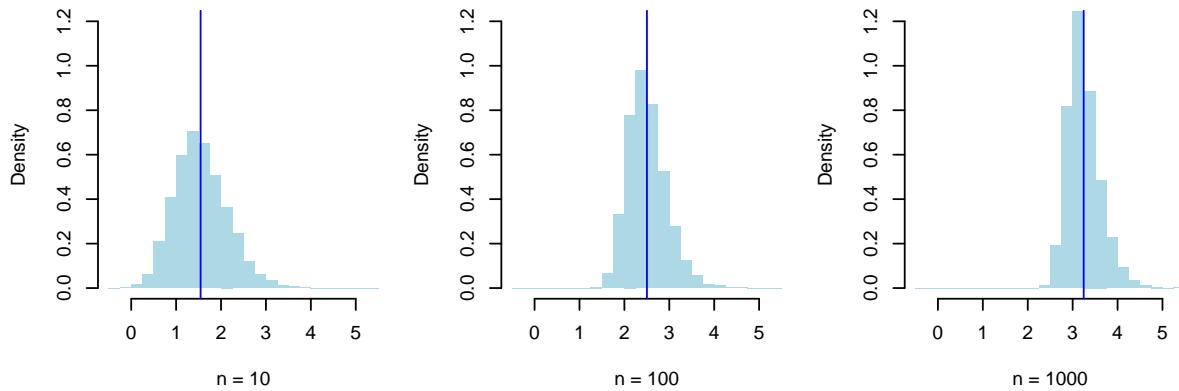


Figure 12.2: Distribution of the maximum of sequences of Gaussian random variables.
(See R-Code 12.2.)

The fitting is typically assessed graphically by comparing empirical and fitted densities and with quantile (Q-Q) and probability (P-P) plots of the fitted model, see Figure 12.3. On the x -axis are the model fits, on the y -axis the empirical values. The diagonal line has intercept zero and slope one. Q-Q plots are scaled to fit the observations, P-P plots have range $[0, 1]$. Because of possibly large values, probability plots are easiest to interpret. ♣

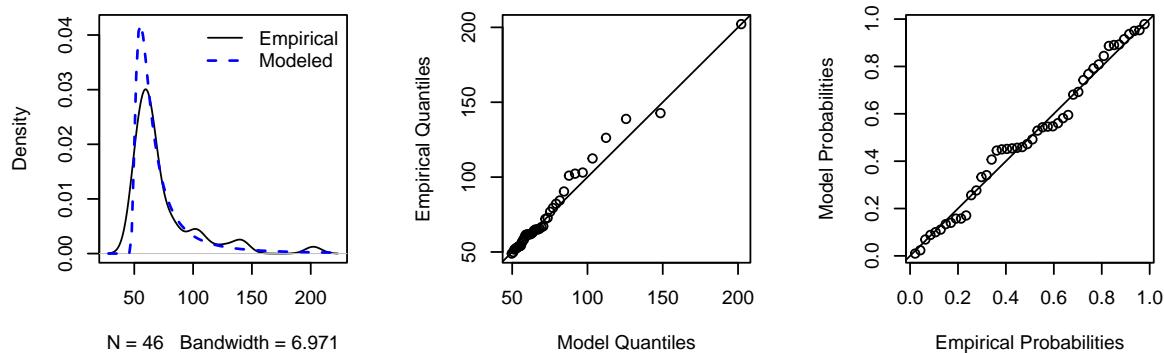


Figure 12.3: GEV fit of the `rhone` data with densities, probability and quantile plots (left to right). (See R-Code 12.3.)

12.3 Modeling Peaks over Thresholds

Estimating GEV distribution parameters via block maxima seems a ‘data wasting’ approach. We keep only a single observation from each year or block and thus would not consider two exceptionally large events in a year or, alternatively, might consider a very small observation from a particular year. As an alternative, we might consider using all data that exceed a certain large value, i.e., keeping all exceptionally large values, independent of their occurrence. However, when using this approach, the limit theorem and the limiting distribution (12.3) does not hold

R-Code 12.3 GEV analysis of `rhone` data. (See Figure 12.3.)

```

year <- aggregate(m3~year, rhone, max)$m3
library(extRemes)
gevfit <- fevd(year) # we can use all the default values
summary(gevfit, silent=TRUE)[c(1,3)]
## $par
## location      scale      shape
## 59.17419 10.06745 0.57834
##
## $se.theta
## location      scale      shape
## 1.72203 1.73698 0.16341
ci(gevfit, type="parameter")
## fevd(x = year)
##
## [1] "Normal Approx."
##
##         95% lower CI Estimate 95% upper CI
## location      55.79906 59.17419      62.54931
## scale        6.66304 10.06745     13.47186
## shape        0.25806 0.57834      0.89861
plot(gevfit, type="density", main="")
plot(gevfit, type="qq", main="")
plot(gevfit, type="prob", main="")

```

anymore. But as it turns out, a similar result holds in the case of such exceedances.

Under rather weak assumptions, the distribution function of $X - u \mid X > u$ (i.e., conditional on X exceeding the threshold u) for large u is approximately

$$P(X - u > y \mid X > u) \approx 1 - \left(1 + \frac{\xi y}{\sigma}\right)^{-1/\xi}, \quad (12.4)$$

for appropriate y . It is important to keep in mind that this is a limiting/asymptotic result. The approximation gets “better” for larger thresholds u and we have asymptotic equality. The distribution specified by the right-hand-side of (12.4) is called the generalized Pareto distribution (GPD, or GP) and σ and ξ are called the scale and shape parameters respectively. The distribution has an upper end-point if $\xi < 0$ (bounded distributions) and a lower end-point if $\xi > 0$ (heavy tailed distributions). The limiting case $\xi = 0$ corresponds to an exponential distribution.

Example 12.5. The conditional distribution on the left-hand side of (12.4) can be written as

$$P(X - u > y \mid X > u) = \frac{P(X - u > y \cap X > u)}{P(X > u)} = \frac{1 - F(u + y)}{1 - F(u)} \quad (12.5)$$

For an exponential random variable $X \sim \text{Exp}(\lambda)$ the latter expression simplifies to $\exp(-\lambda(u+y))/\exp(-\lambda u) = \exp(-\lambda y) = 1 - F(y)$. More specifically, the distribution is again exponential with parameter λ . This result is not surprising, as the exponential random variable is *memoryless*. ♣

Remark 12.1. There is an asymptotic link between a GEV and a GPD. It can be shown that (asymptotically) $\xi_{\text{GEV}} = \xi_{\text{GP}}$ and $\sigma_{\text{GP}} = \sigma_{\text{GEV}} + \xi_{\text{GEV}}(u - \mu_{\text{GEV}})$.

Similarly to the GEV, the GPD is also a family of distributions, discriminated by ξ . For example the case $\xi = 0$ leads to a (shifted) exponential distribution. ♣

12.3.1 Return Level Plots

One of the key questions in EVT is the magnitude of an event that occurs very rarely, which is often quantified by the so-called *return level*. The concept of return level and return period has been coined in the context of floods. As illustrated here, for a so-called 100-year flood, the return level has 1% chance of being exceeded in any given year. The return period (or ‘average recurrence interval’) corresponding to this probability of 0.01 is 1/0.01 or 100 years.

Lets denote the generalized Pareto distribution with $G(\cdot)$. Let z_p the return level with return period $1/p$, that means that $G(z_p) = 1 - p$ and by (12.4) we have

$$z_p = u + \frac{\sigma}{\xi}(p^\xi - 1), \quad (\xi \neq 0). \quad (12.6)$$

In practice we need of course estimates for the values above.

Additionally, if we are interested in calculating the return level z_t that is exceeded on average every t observations, we need to take into account that we observe many observations below the threshold. Lets denote $\zeta_u = P(X > u)$, i.e., the probability that an observation is larger than the threshold. Then (for independent observations), the m -observation return level is

$$z_m = u + \frac{\sigma}{\xi}((m\zeta_u)^\xi - 1). \quad (12.7)$$

If we would calculate the n -observation return level (for a series of length n) this would imply that we have on average (if we would observe many of these n sized series) one exceedance thereof.

Example 12.6. R-Code 12.4 illustrates the fitting of a GPD. We have chosen (somewhat arbitrarily) a threshold of 55, corresponding essentially to the 90th percentile with 63 exceeding observations exceeding. As we have monthly data, we need to specify this in the fitting procedure to be able to calculate return levels based on years and not observations. The shape estimate ξ is similar as in the case of the GEV fitting. The resulting return level has quite large uncertainties, reflecting the difficulty estimating the shape parameter and the projection into the future.

Of course return level plots can be calculated for any distribution, including for the GEV. As a matter of fact, it is possible to compare the return levels for both approaches shown here. Using `ci(gevfit, type="return.level", return.period=50)` we get an estimate of 208.03, with confidence interval [67.55, 348.5], yielding an even larger uncertainty. ♣

R-Code 12.4 GPD analysis of *rhone* data. (See Figure 12.4.)

```

gpdfit <- fevd(rhone$m3, threshold=55, type="GP", time.units="months")
summary(gpdfit, silent=TRUE)[c(1)]

## $par
##      scale      shape
## 10.53230 0.44421

ci(gpdfit, type="parameter" )

## fevd(x = rhone$m3, threshold = 55, type = "GP", time.units = "months")
##
## [1] "Normal Approx."
##
##      95% lower CI Estimate 95% upper CI
## scale      5.997237 10.53230      15.06735
## shape      0.074801  0.44421      0.81361

plot(gpdfit, type="density", main="")
plot(gpdfit, type="qq", main="")
plot(gpdfit, type="prob", main="")

### Return period is automatically calculated to years by 'time.units':
ci(gpdfit, type="return.level", return.period=50)

## fevd(x = rhone$m3, threshold = 55, type = "GP", time.units = "months")
##
## [1] "Normal Approx."
##
## [1] "50-year return level: 186.655"
##
## [1] "95% Confidence Interval: (78.2432, 295.0668)"

```

In practice, we do not always have independent observations. For example, hourly, daily or even weakly run-off values are typically serially dependent and special modeling approaches are required.

Remark 12.2. With dependent data, we have to take the dependency into account. Loosely speaking, we look at clusters (a series of exceedances defines a cluster) and we take the typical cluster size into account. The cluster size is often linked to the event duration.

More formally, for an EVT approach, we need a *stationary series* satisfying a so-called $D(u_n)$ criterion. The $D(u_n)$ condition states (essentially) that extreme events are “near-independent” if they are sufficiently far apart (here in time). The dependence does not have any influence on the limit laws for extremes.



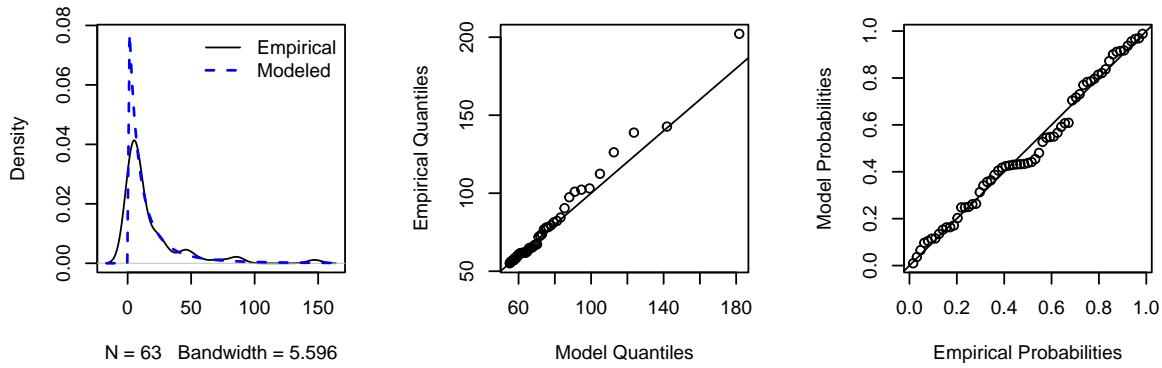


Figure 12.4: GPD fit of the `rhone` data with densities, probability and quantile plots (left to right). (See R-Code 12.4.)

12.3.2 Threshold Selection

The key question is selecting a threshold u for the generalized Pareto distribution. There exist essentially two approaches, a ‘mean residual life plot’ and parameter estimates against threshold plot. The latter is more conclusive.

Example 12.7. R-Code 12.5 illustrates threshold selection for a GPD, with a mean residual life plot and through fitting models to a range of thresholds. The middle and bottom panel of Figure 12.5 gives parameter estimates against threshold. The uncertainties have been indicated with vertical lines. We select the lowest threshold that yields roughly the same parameter estimates as any higher threshold. The choice of 55 in the previous example seems adequate. ♣

R-Code 12.5 Graphical threshold selection for GPD. (See Figure 12.5.)

```
mrlplot(rhone$m3)
threshrange.plot(rhone$m3, r=c(30, 100), nint=15)
```

As illustrated, both with a GEV or a GPD approach it is possible to calculate return levels and suchlike. At first, it seems that a GPD approach has the advantage of estimating only two parameters compared to the three of a GEV. However, for a GPD we require a threshold which, to a certain degree, picks up the role of this missing parameter.

The advantages of a peak over threshold approach over a classical GEV is that the estimation uncertainties are typically smaller due to larger “sample” sizes. However, the threshold selection is not straightforward and requires a certain amount of care.

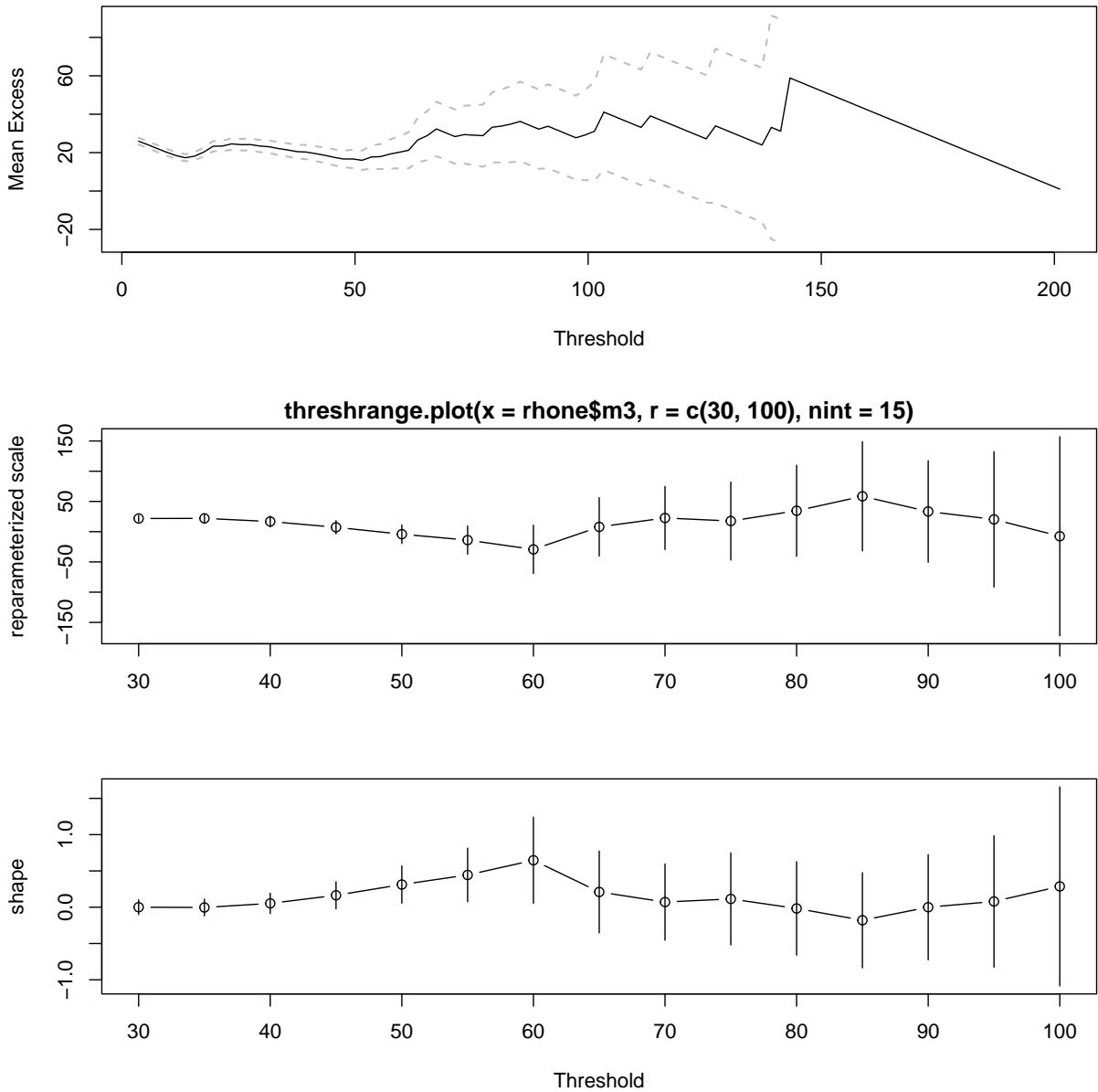


Figure 12.5: Top: Mean residual life plot. Bottom: Parameter estimates against threshold. The dashed lines in the top plot and the vertical bars in the lower ones indicate the 95% uncertainty bounds. (See R-Code 12.5.)

12.4 *Comparing the Approaches Through Profile Estimates

The previous two sections calculated crude Wald confidence intervals of ξ (estimate $\pm 2 \cdot$ standard error). Because we estimate the parameters with a likelihood approach, we can get better uncertainty estimates by *profiling*. Profiling is a technique that calculates for a specific value of one parameter the maximum likelihood over all other parameters.

R-Code 12.6 illustrates the result and compares the relative-log-likelihoods of the GEV and POT estimation approach. As we want to superimpose the two curves, we need to manually plot the profile likelihoods. Moreover, we need to compare the relative profile log-likelihoods and thus subtract the maximum of the log-likelihood. Confidence intervals are constructed by comparing

the relative profile log-likelihoods with the quantile of a chisquared distribution with one degree of freedom.

No surprise, we get similar answers, also compared to Wald type estimates.

R-Code 12.6: Profiling the parameter ξ based on GEV and GPD likelihood estimation. (See Figure 12.6.)

```

xi1 <- profliker(gevfit, type="parameter", which.par=3, xrange=c(0,1),
                  nint=50, plot=FALSE) + gevfit$results$value
xi2 <- profliker(gpdfit, type="parameter", which.par=2, xrange=c(0,1),
                  nint=50, plot=FALSE) + gpdfit$results$value
x <- seq(0, to=1, length=length(xi1))
plot(x, xi1, type="l", ylim=c(-2.5,0),
      ylab="Relative profile\nlog-likelihood", xlab="xi")
lines(x, xi2, col=2)
abline(h=-0.5*qchisq(0.95, 1), col=4)
abline(v=c(x[which.max(xi1)],x[which.max(xi2)]), col=c(1,2))
tmp <- cbind(gev.profile=range(x[xi1 > -0.5*qchisq(0.95, 1) ] ),
              gev.Wald=ci(gevfit, type="parameter", which.par=3)[c(1,3)],
              gpd.profile=range(x[xi2 > -0.5*qchisq(0.95, 1) ] ),
              gpd.Wald=ci(gpdfit, type="parameter", which.par=2)[c(1,3)])
rug(tmp[,2], col=1, ticksize=.25)
rug(tmp[,4], col=2, ticksize=.25)
tmp <- rbind(tmp, "Width of CI"= tmp[2,]-tmp[1,])
print(tmp)

##          gev.profile gev.Wald gpd.profile gpd.Wald
## 95% lower CI    0.30612  0.25806    0.14286  0.074801
## 95% upper CI    0.93878  0.89861    0.89796  0.813612
## Width of CI     0.63265  0.64055    0.75510  0.738811

```

12.5 Bibliographic remarks

Coles (2001) is very accessible and has a lot of emphasis on (environmental) applications. Embrechts *et al.* (1997) is more theoretical but clearly outlined and structured. For mathematically more advanced treatises see Resnick (2008) and Haan and Ferreira (2006).

The article by Davison *et al.* (2012) gives an excellent overview of statistical modeling of spatial extremes.

12.6 Exercises and Problems

Problem 12.1 (EVT) Load the `dataWind` data from the package `extremefit`. The data comes from Airport of Brest (France) and represents the average wind speed per day from 1976 to 2005.

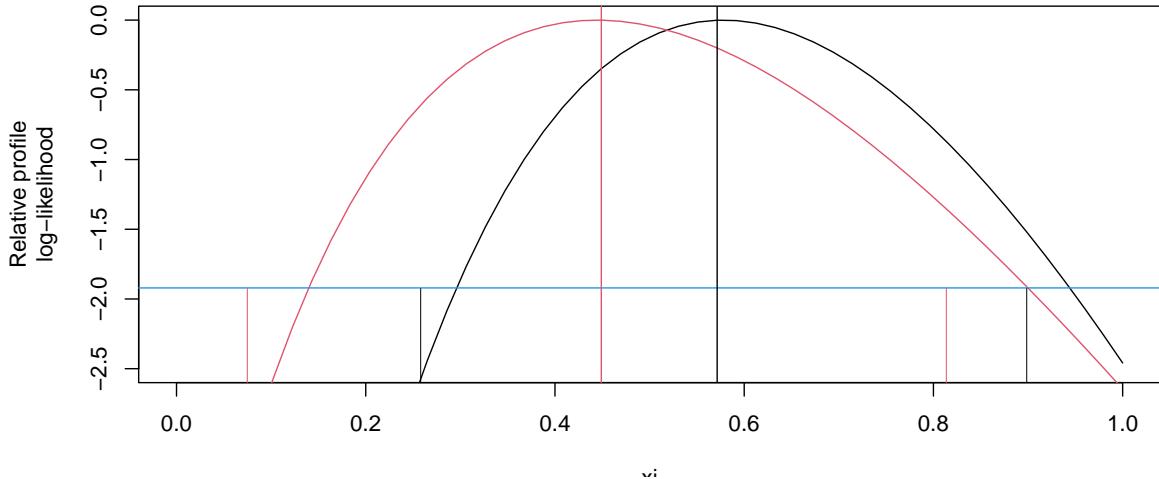


Figure 12.6: Profile relative log-likelihoods for GEV (black) and GPD (red) fits. The corresponding Wald-type confidence intervals are given by the ticks. (See R-Code 12.6.)

- Conduct EDA for the data and plot maximum monthly wind speed.
- Using the monthly maxima of the wind speed, conduct the GEV analysis of the wind data. Assess the fit using the P-P and Q-Q plots.
- Apply Modeling Peaks over Thresholds for the same data set (first choose a threshold).
- Compare return levels for the considered two methods.

Problem 12.2 (ML of GPD) In this problem we use the density

$$f(x; \sigma, \xi) = \begin{cases} \frac{1}{\sigma} \left(\frac{\xi x}{\sigma} + 1 \right)^{-\frac{\xi+1}{\xi}}, & \text{for } \xi \neq 0, \\ \frac{1}{\sigma} e^{-x/\sigma}, & \text{for } \xi = 0, \end{cases}$$

where $x \geq 0$ for $\xi \geq 0$ and $0 \leq x \leq -\sigma/\xi$ for $\xi < 0$.

- For a random variable X with density given above, calculate the density of $c \cdot X$, $c > 0$.
- Use the data posted on <https://www.math.uzh.ch/furrer/download/sta121/gpd.RData> to determine numerically the ML estimate of $\boldsymbol{\theta} = (\sigma, \xi)^\top$.
- Visualize the relative log-likelihood as a function of $\boldsymbol{\theta}$ and, based on this figure, summarize the evidence against $H_0 : \sigma = 9, \xi = 0.2$.
- Graphically visualize the profile log-likelihood $\ell_p(\xi)$. Compare the estimation uncertainty based on Wald and profile log-likelihood.

Chapter 13

Neural Networks and Deep Learning

In recent years there has been an explosion of “new methodologies” centered around machine learning, deep learning and artificial intelligence. At their base they are very similar to classical statistical methodologies but upscaled for large datasets deployed to massive computing resources.

Learning goals for this chapter:

- ◊ Understand the concept of (machine) learning
- ◊ Describe the concept of a neural network (including input layer, hidden layers, output layer, neuron, activation function, weights, forward propagation, backward propagation, error function, hyperparameters of the network)
- ◊ Classify images using a neural network in R (train the network, interpret the results)
- ◊ Get some understanding of the structure of large ML models

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta121/chapter14.R.

In Section 4.4, we introduced the concept of machine learning (ML). We also compared statistical and ML terminology in Table 4.1. This chapter delves into new, yet archetypal, ML areas that are not easily captured by statistical terminology. As a result, we will primarily use ML-specific language. The goal of this chapter is to provide an introduction to neural networks and deep learning.

Building models with hundreds of thousands of parameters—or weights—poses challenges. Traditional statistical methods, like the likelihood framework, cannot easily accommodate this complexity. Even using R functions like `optim()` is insufficient for these tasks. Despite the need for additional software components, it is still possible to execute these operations within the RStudio environment.

13.1 Learning as in Machine Learning

We are all familiar with the term *learning* in the context of traditional education. However, *machine learning* is a term that even experts interpret differently. Opinions about what ML is — and what it is not — appear frequently in various forms, such as social media discussions among professionals.

In the context of this chapter, we define learning in ML as model fitting. This includes both constructing the model's structure and optimizing its parameters. This broad definition makes the term particularly useful in ML settings. Learning often involves an automatic iterative process guided by a loss function and a rule for model updating, as depicted in Figure 13.1.

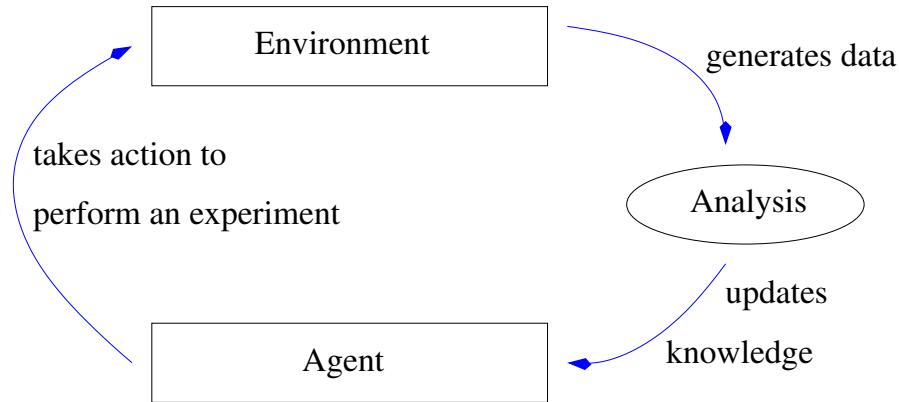


Figure 13.1: Statistical learning as an iterative process between an agent and environment.

Traditionally, the agent in this loop was a human researcher conducting experiments to collect data. Now, in the ML paradigm, this agent is an algorithm. It often conducts computational experiments, generating output that is then analyzed. Based on performance metrics like RMSE or loss values, the algorithm adjusts model parameters.

One effective context for machine learning is the reinforcement learning framework. Here, the algorithm learns to make decisions that maximize a defined “reward”, gradually improving its performance. This is why algorithms for games like Chess and Go are so powerful; they operate within a finite set of moves and clear rules for success or failure.

Besides reinforcement learning, other learning strategies include supervised learning, where the algorithm learns a function mapping inputs to outputs based on labeled data. In unsupervised learning, the algorithm identifies patterns in unlabeled data.

The next example show how to “automatically” learn an unknown curve. As presented, the approach is intuitive (based on the knowledge of Chapter 11) but it should be also intuitive how such an approach can be used in higher dimensions and with more complex curves.

Example 13.1. Suppose we aim to learn an unknown curve g . We don't require mathematical rigor here, but let's assume that the curve is smooth. Evaluating the curve at specific locations is possible; that is, we can compute $g(x)$ for certain values of x . The R-Code 13.1 initially constructs this “unknown truth” and then proceeds to estimate the curve based on six randomly

selected observation points, as shown in the top panel of Figure 13.2. It becomes evident that the estimation significantly deviates from the actual curve towards the right end of the domain.

In this basic scenario, the prediction error largely depends on how close the observation points are to each other. While more points generally yield a better curve approximation, the prediction error can still be used as a metric to quantify our understanding of the curve.

If we employ a reinforcement learning approach, we would strategically choose the next observation point where the prediction error is highest. In essence, we aim to minimize the maximum prediction error across the curve. For example, after the initial six evaluations, the highest prediction error occurs at the origin. Therefore, the next observation point is chosen there. The algorithm, being unaware of the “unknown truth”, cannot recognize that the initial estimate at the origin is already quite accurate due to chance.

By iteratively selecting subsequent points based on maximizing the prediction error, we can refine our curve estimate. After 14 such observations, our estimation significantly improves, as depicted in the bottom panel of Figure 13.2. ♣

R-Code 13.1: Illustration of learning a curve.

```

library(fields)
library(spam)
set.seed(16)

nobs <- 6           # starting with a few observations
ngrid <- 100        # grid resolution
x1 <- runif(nobs)   # selecting a random set of locations to start
grid <- cbind((1:ngrid)/(ngrid+1), 0)
truth <- c(rmvnorm.spam(1, Sigma=wendland.cov(grid, theta=.2)))
### `truth` represents the curve that we want to find
sel <- sample(ngrid, nobs, replace=FALSE) # we select the first obs
x <- grid[, sel]      # x values for observations
y <- truth[, sel]      # starting observations
mKout <- mKrig(x, y, Covariance="Wendland2.2", m=1, theta=.2,
                lambda=0.0001) # is essentially zero but prevents numeric issues
out <- predict(mKout, xnew=grid)          # predicting the curve
outSE <- predictSE(mKout, xnew=grid)       # uncertainties
plot(grid[,1], truth, type="l", ylim=c(-3.2,2.2), xaxt="n", yaxt="n")
polygon(c(grid[,1], rev(grid[,1])), c(out+2*outSE,
                                         rev(out-2*outSE)), col="gray93", border=NA)
lines(grid[,1], truth, type="l", ylim=c(-2,2)) # unkown truth
lines(grid[,1], c(out), col=4)                 # first guess
points(x[,1], y, pch=19, col=2)                # nobs observations
# this completes the top panel

for (i in 1:8) {
  sel <- c(sel, which.max(outSE)) # successively adding obs

```

```

mKout <- mKrig(grid[,sel], truth[,sel], Cov="Wendland2.2", m=1, theta=.2,
                  lambda=0.0001)
out <- predict(mKout, xnew=grid)
outSE <- predictSE(mKout, xnew=grid)
### Plotting (superimposing with alpha level)
polygon(c(grid[,1],rev(grid[,1])), c(out+2*outSE, rev(out-2*outSE)),
         col=rgb(.9-i/25, .9-i/25, .9-i/25, .4), border=NA )
lines(grid[,1], c(out), pch=20, col=4)
}
lines(grid[,1], truth)
points(grid[sel,1], truth[sel], pch=19, col=2)
text(grid[sel[-(1:nobs)],1], truth[sel[-(1:nobs)]]+.3, paste(1:8))

```

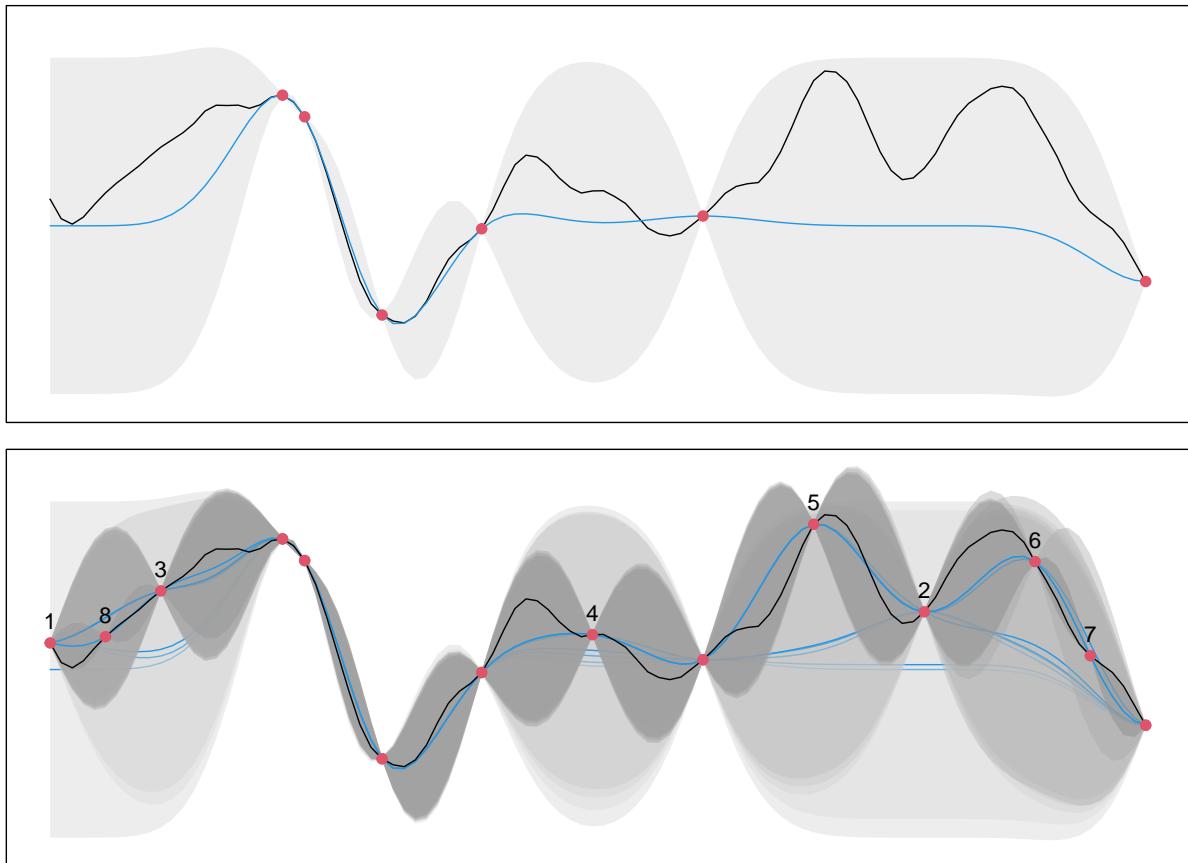


Figure 13.2: Illustration of “learning” a curve. Top: the blue line is the best guess of the unknown black curve based on six observations (red dots). Bottom: best guess with eight additional observations. The shading represents the uncertainty of the corresponding iteration. The black line is the (unknown) truth, red dots are the observations. The numbers indicate the order of the new locations. (See R-Code 13.1.)

13.2 Concept of Neural Networks

The concept of neural networks is inspired by the workings of the biological nervous system and, more specifically, the brain's network of neurons. This computational paradigm has its historical origins in Frank Rosenblatt's perceptron model, dating back to 1957 (Rosenblatt, 1957). However, the foundational principles took several decades to be applied widely, largely due to computational limitations. Advances in both hardware capabilities (particularly Graphics Processing Units (GPUs)) and algorithmic efficiencies have rejuvenated the field in the 21st century. Neural networks aim to create a simplified, mathematical abstraction of biological neurons to perform tasks ranging from classification and regression to natural language understanding and computer vision.

In a neural network, the most elemental computing unit is known as a neuron or node. A neuron takes in a set of inputs $\{x_1, x_2, \dots, x_p\}$, applies a linear transformation by summing the weighted inputs and adding a bias term, and then passes the result through a non-linear activation function. Symbolically, the output y of a single neuron can be represented as:

$$y = \phi\left(\sum_{i=1}^p w_i x_i + b\right), \quad (13.1)$$

where w_i are the weights, x_i are the input features, b is the bias and $\phi(\cdot)$ is the activation function. A single neuron mimics a biological neuron as it also receives many inputs of varying strength which will trigger a single output (possibly as soon as a certain critical value is exceeded). Figure 13.3 illustrates a single neuron (often called perceptron). While a single artificial neuron mimics the basic functionality of a biological neuron by integrating multiple inputs to produce an output, it is a simplified abstraction. Biological neurons are involved in more complex processes like spike-timing-dependent plasticity and exhibit more intricate dynamics, which are not captured in the artificial models.

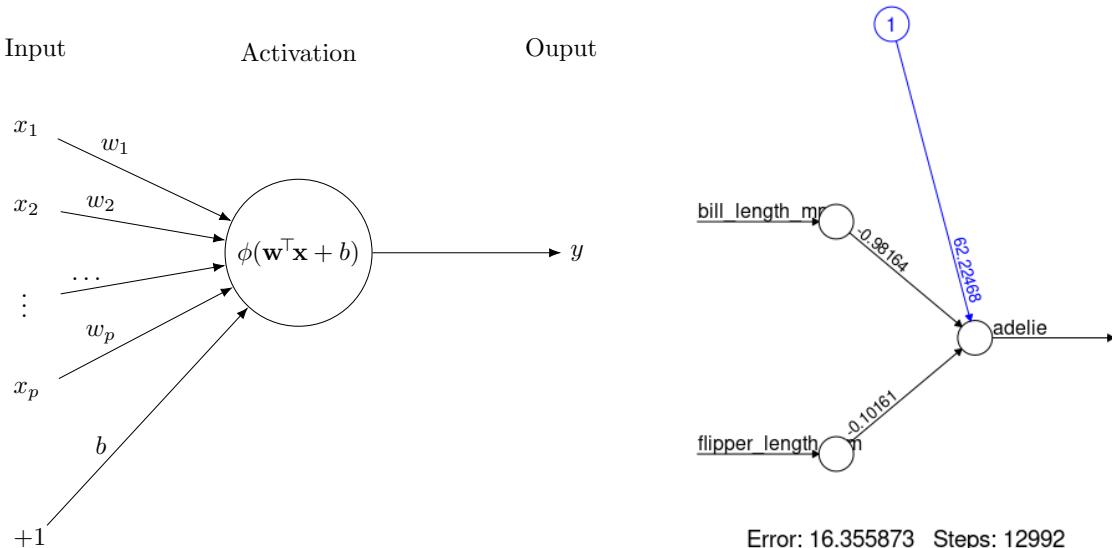


Figure 13.3: Single neuron as building block of neural networks. Left simple generic representation, right fitted single neuron classifier with $p = 2$ (see R-Chunk 13.2).

Example 13.2. R-Code 13.2 showcases a minimalist classification model applied to the *penguin* dataset. The aim is to classify the species Adelie based on their bill and flipper lengths. We opt for a single-neuron setting with no hidden layers (`hidden=0`). The loss function used is cross-entropy, denoted by `err.fun="ce"`, which in this context simplifies the neural network to a generalized linear model (GLM). It's worth noting that the estimated parameters are not exact but approximately align with what one might expect from a GLM (refer to the right panel of Figure 13.3). The code also generates a confusion matrix to evaluate the model's prediction accuracy.



R-Code 13.2 Minimalist classification model based on *penguins* dataset.

```

library(palmerpenguins)
library(neuralnet)

peng <- as.data.frame(penguins[complete.cases(penguins[,c(1,3:6)]),c(1,3:6)])

peng$adelie <- peng$species == "Adelie" # binary classification
set.seed(16)

train <- sort(sample(nrow(peng), 150)) # train based on 150 random observations
peng1N <- neuralnet(adelie ~ bill_length_mm + flipper_length_mm,
                     hidden=0,           # single neuron
                     err.fct="ce",       # cross-entropy as criterion
                     data=peng[train,], linear.output=FALSE)

plot(peng1N)

pred1N <- predict(peng1N, peng[-train,])
predicted <- c("others", "Adelie")[1 + round(pred1N)]
table(peng[-train,]$adelie, predicted) # row true, column predicted

##      predicted
##      Adelie others
## FALSE      3     98
## TRUE      87      4

coef(glm(adelie ~ bill_length_mm + flipper_length_mm, data=peng[train,],
         family=binomial))

##          (Intercept)    bill_length_mm flipper_length_mm
##          62.2482176      -0.9814299       -0.1017747

```

In a neural network, neurons are organized into layers, and these layers are connected to each other by weighted edges or links that represent the flow of information between them. The output of one neuron becomes the input to one or more neurons in the subsequent layer. These connections are weighted, meaning that the value of the output is scaled by a weight factor before being passed on. The architecture of a neural network, which specifies how neurons are organized into layers and how these layers are connected, is a crucial factor in determining the network's performance and capabilities.

Weights modulate the strength of the connection between neurons, whereas biases allow for flexibility in the activation function, effectively shifting it along the input axis. More specifically, *layer* in a neural network is a collection of neurons that operate in parallel and are connected to neurons in the adjacent layers. Each neuron in a layer receives potentially multiple inputs from neurons in the previous layer and sends its output to multiple neurons in the next layer. Therefore, a layer serves as a vector of neurons, each executing its own set of calculations but sharing the same input or output space with other neurons in that layer. During a so-called *feedforward* step, data flows from the input layer through the hidden layers to the output layer, with each layer performing calculations defined by the weights and biases of its neurons.

Neural networks architecturally consist of three types of layers:

1. *Input Layer*: This layer serves to interface with the dataset, receiving p -dimensional vectors as input features. Often, this layer is simply a pass-through for the data, with each neuron corresponding to a single feature.
2. *Hidden Layers*: Situated between the input and output layers, these layers perform the majority of the computation. Neural networks may contain one or more hidden layers, and the choice of architecture often depends on the complexity of the task at hand.
3. *Output Layer*: This layer emits the final output of the neural network, which can be either a vector or a single value, depending on the application. For instance, in classification tasks, the output layer commonly contains as many neurons as there are classes.

Example 13.3. In R-Code 13.3, we return to the `palmerpenguins` dataset to classify all three species using a neural network with a single hidden layer containing five neurons. We employ the logistic activation function, specified by `act.fct="logistic"` as the default. The model performs remarkably well on the training data, correctly classifying all penguins, and misses only two in the validation set. For numerical stability, it is essential to transform the body mass measurements to kilograms. Additionally, the model's performance appears to be sensitive to the initial random seed (`set.seed()`), as evidenced by the variability in results across five repetitive fits with `lifesesign="minimal"`.

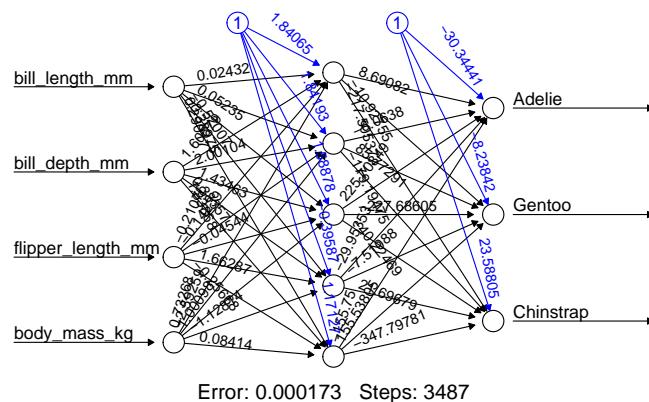


Figure 13.4: Fitted neural net. (See R-Code 13.3.)

R-Code 13.3 *peng* (See Figure 13.4.)

```

library(palmerpenguins)
library(neuralnet)
peng$body_mass_kg <- peng$body_mass_g/1000
set.seed(12)
train <- sort(sample(nrow(peng), nrow(peng)/2)) # based on half of the obs
pengNN <- neuralnet(species ~ bill_length_mm + bill_depth_mm +
                      flipper_length_mm + body_mass_kg, rep=5, hidden=c(5),
                      err.fct="ce",    # lifesign = "minimal",
                      data=peng[train,], linear.output=FALSE)
# fitNN <- predict(pengNN, peng[train,]) # equivalent to pengNN$net.result[[1]]
# table(peng[train,]$species, apply(fitNN, 1, which.max))
plot(pengNN, rep="best")
predNN <- predict(pengNN, peng[-train,])
predicted <- c("Adelie", "Chinstrap", "Gentoo") [apply(predNN, 1, which.max)]
table(predicted, peng[-train,]$species)

##
## predicted Adelie Chinstrap Gentoo
##   Adelie      67        2       0
##   Chinstrap     3       29       1
##   Gentoo       2        0      67

```

13.2.1 Training the Model Parameters

Once the architecture is set up, the network “learns” through a two-step process: feedforward and backpropagation. In the feedforward phase, data moves from the input layer through the hidden layers to the output layer. The backpropagation phase is where the actual learning occurs; errors computed at the output layer are propagated back to adjust the weights and biases.

During the feedforward phase, data flows in a unidirectional manner from the input layer through the hidden layers and finally to the output layer. As data progresses through the network, each layer performs its respective computations, applying both linear and non-linear transformations to the inputs. This cascade of computations can be represented mathematically by a series of matrix multiplications and activation function applications, effectively mapping the input space to the output space through multiple transformations.

After the feedforward phase, the network computes the error between its output and the actual target values. This error is then propagated backward through the network; hence the term “backpropagation”. During backpropagation, the network adjusts its weights and biases in an attempt to minimize the error. When activation functions are differentiable, these algorithms can easily adjust model parameters to minimize the loss function. Backpropagation uses the chain rule of calculus to compute gradients for each layer, starting from the output layer and moving backward to the input layer. These gradients are then used to update the weights and biases.

The Backpropagation algorithm essentially consists of the following steps:

1. *Initial Forward Pass*: Perform a feedforward pass to get the predicted output.
2. *Compute the Loss*: Calculate the error (or loss) between the predicted and actual output. Commonly used loss functions include mean squared error (MSE) for regression problems and cross-entropy loss (defined by $-\sum_{i=1}^n y_i \log(\hat{y}_i)$) for classification problems.
3. *Backward Pass*: Start from the output layer and move backward through the hidden layers. Compute the gradient of the loss function with respect to each weight by taking the derivative of the loss function. To do this, the chain rule of calculus is applied to break down complex derivatives into simpler components.
4. *Update Weights and Biases*: Use the gradients computed during the backward pass to update the weights and biases. This is done using an optimization algorithm, most commonly Gradient Descent or one of its variants (like Stochastic Gradient Descent, Adam, etc.). The formula for a basic weight update is:

$$w_{\text{new}} = w_{\text{old}} - \alpha \times \text{gradient}$$

where α is the learning rate, a hyperparameter that controls how much we are adjusting the weights in the direction of the gradient. Biases are updated in a similar manner.

There are no rules on how many layers with respective sizes are needed. A typical approach is starting from a specific setup of the neural net. Then the weights ($\{w_{ij}\}$) and biases (b_j) of the neurons are initially set to random values and are updated via optimization algorithms like stochastic gradient descent (SGD) to minimize a predefined loss function. Depending on the model performance, the structure of the neural net is adapted.

13.2.2 Role of Activation Functions

Activation functions introduce the much-needed non-linearity into the system, enabling the network to capture complex relationships in the data. These functions are differentiable almost everywhere, a property that is critical for the backpropagation algorithm to update weights and biases effectively.

Classical activation functions are

$$\text{Sigmoid or SoftStep or logistic:} \quad \phi(x) = \frac{1}{1 + e^{-x}}, \quad (13.2)$$

$$\text{Rectified Linear Unit (ReLU):} \quad \phi(x) = \max(0, x), \quad (13.3)$$

$$\text{SoftPlus:} \quad \phi(x) = \log(1 + e^x), \quad (13.4)$$

$$\text{TanH:} \quad \phi(x) = \tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (13.5)$$

Of course many variants of the previous transfer functions exist.

The Sigmoid function squashes the output between 0 and 1 and is often used in the output layer for binary classification problems. ReLU is computationally more efficient. Due to its

piecewise-linear nature, it helps mitigate the vanishing gradient problem in deep networks. Researchers and practitioners often consider ReLU's non-differentiability at zero to be a minor issue in practice, and many optimization algorithms handle it quite well. In multi-class classification, the Softmax function is often used in the output layer. It converts a vector of raw scores (also called logits) into a probability distribution. TanH is similar to the sigmoid but maps inputs to values between -1 and 1. While less commonly used than ReLU for deep networks, it still sees application where centered outputs are desirable.

The *Universal Approximation Theorem* is a fundamental theorem in the theory of artificial neural networks. It states that a feedforward neural network with a single hidden layer containing a finite number of neurons can approximate any continuous function to an arbitrary degree of accuracy, provided the activation function for the neurons is a non-constant, bounded, and monotonically-increasing continuous function (see. e.g. [Hornik, 1991](#)).

If we interpret an n -dimensional data vector \mathbf{y} as the output of an unknown function $f(\cdot)$, the theorem states that with very weak assumptions, a NN will find an approximation $\hat{\mathbf{y}}$ which is arbitrarily close to \mathbf{y} . Or more generally, for every $\epsilon > 0$, there exists a neural network $g(x; w)$ such that $|f(x) - g(x; w)| < \epsilon$ for all x in the domain of interest. Note that $f(\cdot)$ has to be a smooth function in (compact sub-spaces of) \mathbb{R}^n .

The theorem is often cited to justify the use of neural networks in a wide range of approximation tasks, such as function approximation, classification, and regression. However, it's crucial to understand that while the theorem provides theoretical backing for the capability of neural networks, practical implementation involves many other considerations like the architecture of the network, the efficiency of training algorithms, and the avoidance of overfitting. The theorem does not tell us how to train the neural network to achieve this approximation, nor does it provide insights into how many neurons are needed for a given level of approximation.

13.2.3 Learning and Addressing Overfitting

To effectively train a Deep Neural Network (DNN), it's essential to understand the key terms often used in this context:

- *Epoch*: An epoch involves one complete forward and backward pass of all training examples.
- *Batch Size*: This specifies the number of training examples utilized in one forward and backward pass.
- *Number of Iterations*: This refers to the total number of passes, each involving a subset of training examples equal to the 'batch size'.

Increasing the number of epochs or iterations allows for a longer learning process, which can result in higher accuracy. However, a larger batch size requires more computing memory.

A crucial challenge in training machine learning models is the risk of overfitting. Overfitting occurs when a model performs well on the training data but poorly on new, unseen data. To diagnose overfitting, cross-validation is commonly used. The dataset is divided into two subsets: a training set and a validation set. Metrics such as accuracy and loss are monitored for both subsets during the training process.

If the training accuracy continually increases while the validation accuracy begins to decline, the model is likely overfitting. This indicates that the model is becoming overly specialized to the training set and losing its generalization ability.

To mitigate overfitting, especially in larger network architectures, a technique known as *dropout* is often employed. Dropout involves setting a fraction of the input units to zero at each training step, effectively “dropping out” certain nodes in the network during training. This prevents the model from becoming too reliant on any individual feature and promotes better generalization. In the context of the network, dropout can be visualized as temporarily removing certain connections between layers.

By carefully tuning the epochs, batch size, and dropout rate, among other hyperparameters, one can achieve a balanced model that performs well not just on the training data but also on unseen data.

13.3 Deep Learning

Deep learning is a subfield of machine learning that employs deep neural networks with multiple hidden layers. This added complexity enables the model to learn more intricate features, thus outperforming traditional neural networks with fewer layers. Each layer, or set of layers, in a deep network is specialized to recognize different features of the input, leading to hierarchical learning.

One key advantage of deep learning is its capability for automatic feature extraction. Unlike traditional machine learning algorithms, which may require manual selection of important features, deep learning automatically identifies these features, contributing to its high accuracy in complex tasks like image and speech recognition.

Furthermore, deep learning models are scalable and tend to perform better as data volume increases. However, these benefits come with challenges, such as the significant computational resources required and the potential for overfitting on small or unrepresentative datasets. They are also often viewed as "black-box" models due to their complex internal mechanics, which can make them difficult to interpret.

In summary, deep learning has revolutionized machine learning, particularly in tasks that were previously considered too complex. Its strengths in automatic feature extraction, accuracy, and scalability make it highly effective, but it's crucial to be aware of its computational demands and risk of overfitting.

13.3.1 Types of Deep Neural Networks

There are several specific types of architectures of deep neural networks that have gained prominence for particular kinds of tasks. Here's a summary of some of the most important ones:

1. *Convolutional Neural Networks* (CNNs): Primarily used in image recognition tasks, CNNs take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. They are highly effective in tasks like image and video recognition, image classification, and object detection.

2. *Recurrent Neural Networks* (RNNs): Designed for sequence prediction problems, RNNs are ideal for tasks that involve sequential data like time series analysis, natural language processing, and speech recognition. The unique feature of RNNs is their “memory”, which captures information about what has been calculated so far.
3. *Long Short-Term Memory Networks* (LSTMs): A specialized kind of RNN, LSTMs are designed to remember past information for long periods of time, making them highly suitable for tasks that require learning from context, like machine translation.
4. *Generative Adversarial Networks* (GANs): These are used to generate data that is similar to some input data. In a GAN, two networks are trained simultaneously: a generator that produces fake data and a discriminator that tries to distinguish the fake data from real data. GANs are widely used in tasks like image generation, super-resolution, and style transfer.
5. *Autoencoders*: Used primarily for unsupervised learning tasks, autoencoders aim to encode input data as vectors. They are often used for anomaly detection or for reducing the dimensions of data for other machine learning tasks.
6. *Transformer Networks*: Originally designed for natural language processing tasks, transformers have been found to be highly effective in a wide range of applications. They excel at handling a wide range of sequence lengths, making them ideal for tasks like text summarization, translation, and classification.
7. *Graph Neural Networks* (GNNs): These networks are used for learning on graph-structured data. They are commonly used in social network analysis, citation networks, and biological data representation.

Each of these architectures is best suited for specific types of tasks and data, and they often outperform traditional machine learning algorithms for those specific tasks. The choice of architecture is often driven by the problem you’re trying to solve, the nature and structure of your data, and the computational resources available.

13.3.2 Large Language Models

In recent years, large language models like GPT (Generative Pre-trained Transformer), BERT (Bidirectional Encoder Representations from Transformers), and their derivatives have emerged as a groundbreaking advancement in the field of natural language understanding and generation. These models are often trained on vast datasets comprising terabytes of text data and are capable of performing a wide range of tasks directly from the raw text.

One of the defining features of these models is their scale. With hundreds of billions of parameters, these models are able to capture complex relationships in the data. These large models are generally pre-trained on a comprehensive corpus of text and then fine-tuned for specific tasks, allowing for transfer learning. This enables the use of a single model for multiple natural language processing (NLP) tasks, from text completion and summarization to translation and question-answering. Unlike traditional machine learning models, which treat words as independent features, large language models consider the context of each word relative to the others.

in a sentence. Note that LLM rather use tokens instead of words. A token is a unit of text that has been extracted from a larger corpus during the process of tokenization. In natural language processing, a token could be a word, a subword, or even a single character, depending on the granularity of the tokenization method used.

Example 13.4. I have asked a famous LLM (<https://chat.openai.com/?model=gpt-4>) to “give me details on the internal working of large language models”. Here is its output provided in Markdown format:

Certainly! The internal workings of large language models like GPT, BERT, and their variants can be quite intricate, but I'll do my best to summarize the key concepts:

Internal Workings of Large Language Models

1. **Architecture**: These models typically rely on transformer architectures. In its most basic form, a transformer consists of an encoder and a decoder, although some models like GPT only use the decoder part. The transformer is essentially a stack of layers where each layer comprises self-attention mechanisms and feed-forward neural networks.
2. **Parameters**: These models have a large number of parameters—weights and biases—often numbering in the hundreds of millions or even billions. These parameters are tuned during the training process.
3. **Self-Attention Mechanism**: One of the key innovations in these models is the self-attention mechanism, which allows the model to weigh the importance of different parts of the input text when making a prediction. This is crucial for capturing long-range dependencies and contextual nuances in text.
4. **Tokenization**: The input text is usually broken down into smaller pieces called tokens. These tokens are then transformed into vectors using embedding layers. These vectors serve as the initial input for the model.
5. **Positional Encoding**: Since transformers don't have a built-in sense of order or sequence, positional encodings are added to the embeddings to give the model information about the positions of the tokens in the sequence.
6. **Forward Pass**: During inference, the tokenized, embedded, and positionally encoded text is passed through the model layer by layer. Each layer's self-attention mechanisms and feed-forward networks modify the representations to capture both local and global contextual information.
7. **Task-Specific Heads**: For specific tasks like classification or question-answering, a task-specific "head" is usually attached to the final layer of the pre-trained model. This head transforms the complex representations learned by the model into output suitable for the specific task at hand.
8. **Fine-Tuning**: Although the base model is trained on a general corpus, it can be fine-tuned on a smaller, task-specific dataset to make it more effective for specific applications. This is possible due to the model's ability to transfer knowledge, a feature inherent to its large-scale, multi-layer architecture.

9. **Decoding Strategies**: For generative tasks, different decoding strategies like greedy decoding, beam search, or top-k sampling are employed to generate the most likely sequence of tokens as output.

10. **Computational Requirements**: The sheer size of these models often requires specialized hardware like TPUs (Tensor Processing Units) or large GPU clusters for both training and sometimes even for inference.

This should give you a deeper look into what goes on inside these large language models. Would you like more detail on any specific aspect?



At the time of updating this chapter, a new release of an open source LLM has been announced (<https://falconllm.tii.ae/falcon.html>). Falcon 180B sets a new state-of-the-art for open models. It is the largest openly available language model, with 180 billion parameters, and was trained on a massive 3.5 trillion tokens using TII's RefinedWeb dataset (Penedo *et al.*, 2023).

At the time of updating this chapter, the release of the open-source LLM *Falcon 180B* has been announced. Falcon 180B sets a new state-of-the-art benchmark for open models. It is the largest openly available language model (via <https://huggingface.co/>), consisting of 180 billion parameters, trained on a massive dataset comprising 3.5 trillion tokens from TII's RefinedWeb dataset (Penedo *et al.*, 2023).

13.4 Tensorflow and Keras

To construct and train deep learning models with ease, several specialized frameworks have been created. Among the most widely used are TensorFlow, Keras, and PyTorch. These frameworks offer not only pre-defined layers and architectures but also a suite of functions and training algorithms that streamline the entire model-building process. In this section, we focus on TensorFlow and its high-level API, Keras, illustrating their utility through a specific example.

TensorFlow is a versatile, open-source machine learning library initially developed by the Google Brain team (now Google DeepMind). It is particularly geared towards tasks that require heavy computational power, including but not limited to deep learning. TensorFlow provides a comprehensive ecosystem of tools, libraries, and community resources that facilitate the building and deployment of ML-powered applications.

Keras, on the other hand, is a high-level neural network API that was designed to be user-friendly and modular. Written in Python, it is capable of running on top of various lower-level APIs including TensorFlow. The primary design goal of Keras is to enable rapid prototyping and experimentation with deep neural networks, offering an environment that is not just powerful but also easy to use and extend.

The advantage of our software choice is the smooth integration thereof via RStudio. In case systemwide installation is missing, a local (virtual environment one) can be installed via `install.packages("keras"); keras::install_keras()`.

We illustrate the use of TensorFlow with a classical example, which is taken from [https:](https://)

//github.com/rstudio/keras/blob/master/vignettes/examples/cifar10_cnn.R. A similar example is presented at <https://poissonisfish.com/2018/07/08/convolutional-neural-networks-in-r/>.

The dataset we use is called CIFAR-10 (Canadian Institute for Advanced Research 10-class) and is a widely used dataset for training machine learning models, particularly in the domain of computer vision. The dataset consists of 60,000 $32 \times$ color images distributed across the 10 different classes airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset is divided into 50,000 training images and 10,000 test images, with each class having an equal number of instances (5,000 images for training and 1,000 images for testing per class). The images are low-resolution, which makes the dataset manageable yet challenging enough for training convolutional neural networks (CNNs) and other types of image classification algorithms, see Figure 13.5 shows examples for a subset of classes.

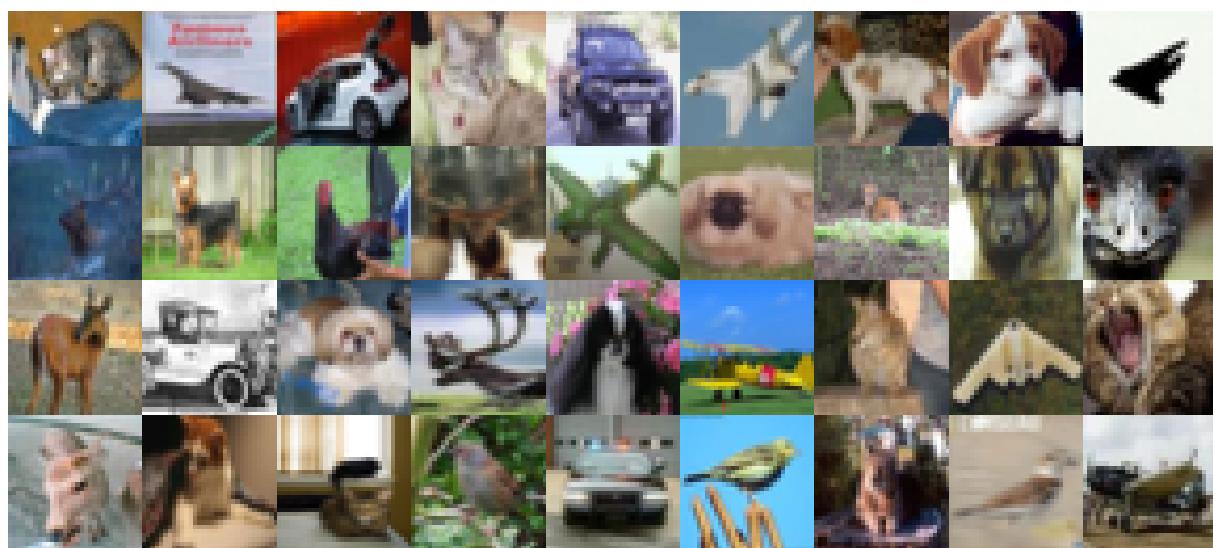


Figure 13.5: Sample test images, with 32×32 RGB structure, as constructed by R. (See R-Code 13.4.)

We start by downsizing the dataset `cifar10` (provided by the R-package `keras`) to the first six categories and to 30×30 grayscale pixels. R-Code 13.4 loads and downscals the images. At least for the human eye, the classification tasks seems now more difficult (Figure 13.6)

R-Code 13.4: Data preparation (See Figure 13.5.)

```

library(keras)
cifar10full <- dataset_cifar10() # See ?dataset_cifar10 for more info

sela <- (cifar10full$train$y < 6)      # select only first six categories
selb <- (cifar10full$test$y < 6)
cifar10 <- list(train=list(x=cifar10full$train$x[sela,,,],
                            y=cifar10full$train$y[sela]), test=list(x=cifar10full$test$x[selb,,,],
                            y=cifar10full$test$y[selb]))

# Scale RGB values in test and train inputs:

```

```

x_train <- cifar10$train$x/255
x_test <- cifar10$test$x/255
y_train <- to_categorical(cifar10$train$y, num_classes=6)
y_test <- to_categorical(cifar10$test$y, num_classes=6)

par(mfrow=c(4,9), mai=rep(0,4))
for (i in 1:36) {
  plot(0, xlim=c(1,32), ylim=c(1,32), type="n", axes=F, xlab="",
       ylab="", xaxs="i", yaxs="i")
  rasterImage(x_test[i,,,], 1, 1, 32, 32, interpolate=FALSE)
}
# Transform (manually) to grayscale
x_train2 <- 0.3 * x_train[,,,1] + 0.59 * x_train[,,,2] + 0.11 * x_train[,,,3]
x_test2 <- 0.3 * x_test[,,,1] + 0.59 * x_test[,,,2] + 0.11 * x_test[,,,3]
x_train2 <- x_train2[,2:31,2:31]      # omit a one pixel border
x_test2 <- x_test2[,2:31,2:31]
dim(x_train2) <- c(dim(x_train2), 1) # Fitting requires 4-dimensional array
dim(x_test2) <- c(dim(x_test2), 1)
rm(x_train, x_test, cifar10, cifar10full)      # not needed anymore

```



Figure 13.6: Sample test images, with 30×30 grayscale structure (same images as the top half of Figure 13.5).

R-Code 13.5 shows the model definition, the most difficult part, as there are no rules on how to choose the layers and their connectivity.

R-Code 13.5: TensorFlow model definition.

```

model <- keras_model_sequential() # Initialize sequential model
model %>%      # Start with hidden 2D convolutional layer being fed by images
  layer_conv_2d(filter=32, kernel_size=c(3,3), padding="same",
                input_shape=c(30, 30, 1)) %>%
  layer_activation("relu") %>%
  # Second hidden layer
  layer_conv_2d(filter=30, kernel_size=c(3,3)) %>%
  layer_activation("relu") %>%

```

```

# Use max pooling
layer_max_pooling_2d(pool_size=c(2,2)) %>%
layer_dropout(0.25) %>%
# 2 additional hidden 2D convolutional layers
layer_conv_2d(filter=32, kernel_size=c(3,3), padding="same") %>%
layer_activation("relu") %>%
layer_conv_2d(filter=32, kernel_size=c(3,3)) %>%
layer_activation("relu") %>%
# Use max pooling once more
layer_max_pooling_2d(pool_size=c(2,2)) %>%
layer_dropout(0.25) %>%
# Flatten max filtered output into feature vector & feed into dense layer
layer_flatten() %>%
layer_dense(32) %>%          # original proposal 512
layer_activation("relu") %>%
layer_dropout(0.25) %>%
# Outputs from dense layer are projected onto 6 unit output layer
layer_dense(6) %>%
layer_activation("softmax")

model %>% compile(loss="categorical_crossentropy",
                      optimizer="adam",      # stochastic gradient descend method
                      metrics="accuracy")
# summary(model) # to show the structure of the NN, see figure below

```

The call `summary(model)` summarizes the layers of the NN (see also the screenshot shown in Figure 13.7). Here, we have to fit 64'004 parameters!

Once the model has been defined, the call to fit it is straightforward, as illustrated in R-Code 13.6. For testing purposes, we have kept `epochs` unreasonably low. On a colab session, 2 epoch require a few seconds.

R-Code 13.6 Simple test setting.

```

fitsmall <- model %>% fit(x_train2, y_train, epochs=2,
                           validation_data=list(x_test2, y_test))

```

As NN models require a lot of data, it is often possible to artificially augment the data. Especially with images, this works nicely. Data augmentation consists of rotating or translating images and including these in the training dataset. R-Code 13.7 illustrates the additional code necessary. The additional improvement leads to higher accuracy and lower loss, compared to the original setting (see Figure 13.8). Note that the time to fit the model is now several minutes.

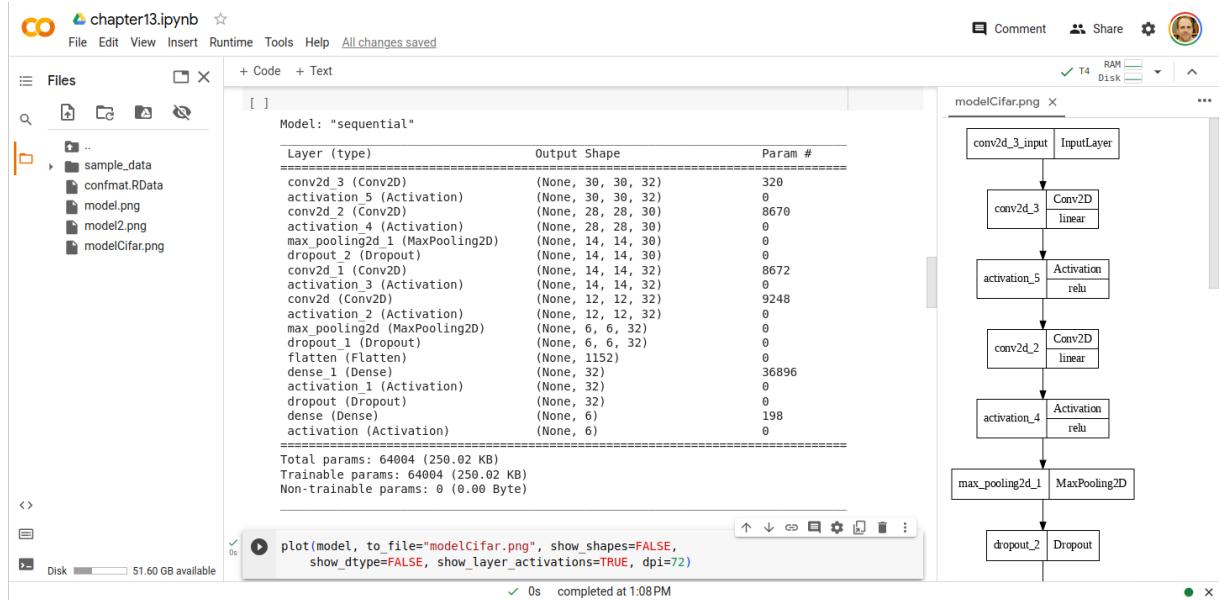


Figure 13.7: Screenshot of a *colab* session (<https://colab.research.google.com>). Middle: the DNN is summarized (as produced by the last line of R-Code 13.5); right: graphical summary of the first few layers of the NN.

Figure 13.9 provides the class and its classification probability of the same images as shown in Figure 13.5.

R-Code 13.7: Final model fit with data augmentation.

```
batch_size <- 50
epochs <- 100
datagen <- image_data_generator(rotation_range=20, width_shift_range=0.2,
                                  height_shift_range=0.2, horizontal_flip=TRUE)
datagen %>% fit_image_data_generator(x_train2)
fitfull <- model %>% fit(
  flow_images_from_data(x_train2, y_train, datagen, batch_size=batch_size),
  steps_per_epoch=as.integer(30000/batch_size),
  epochs=epochs, validation_data=list(x_test2, y_test))
plot(fitfull)
pred <- as.numeric(k_argmax( predict(model, x_test2))) + 1
prob <- predict(model, x_test2)
truth <- max.col(y_test)
obj <- c("airplane", "automobile", "bird", "cat", "deer", "dog")
conf.mat <- table(pred, truth)
dimnames(conf.mat) <- list(obj, obj)

conf.mat # confusion matrix
##      airplane automobile bird cat deer dog
```

```

## airplane      817      4   80   36   34   18
## automobile    78     992   34   71   18   13
## bird          57      1  594   74   43   44
## cat           12      0   30  482   30   73
## deer          29      1  160  103  846   69
## dog            7      2  102  234   29  783

par(mfrow=c(4,9), mai=rep(0,4), cex=.9) # lower panel of Figure
for (i in 1:36) {
  image(t(x_test2[i,,30:1,])[30:1,30:1], axes=F, xlab="", ylab="",
        col=gray.colors(64,0,1))
  legend("topleft", legend=obj[pred[i]],
         text.col=ifelse(pred[i]==truth[i], 4, 5), bty="n")
  legend("bottomleft", legend=round(prob[i,pred[i]],2), bty="n")
}

```

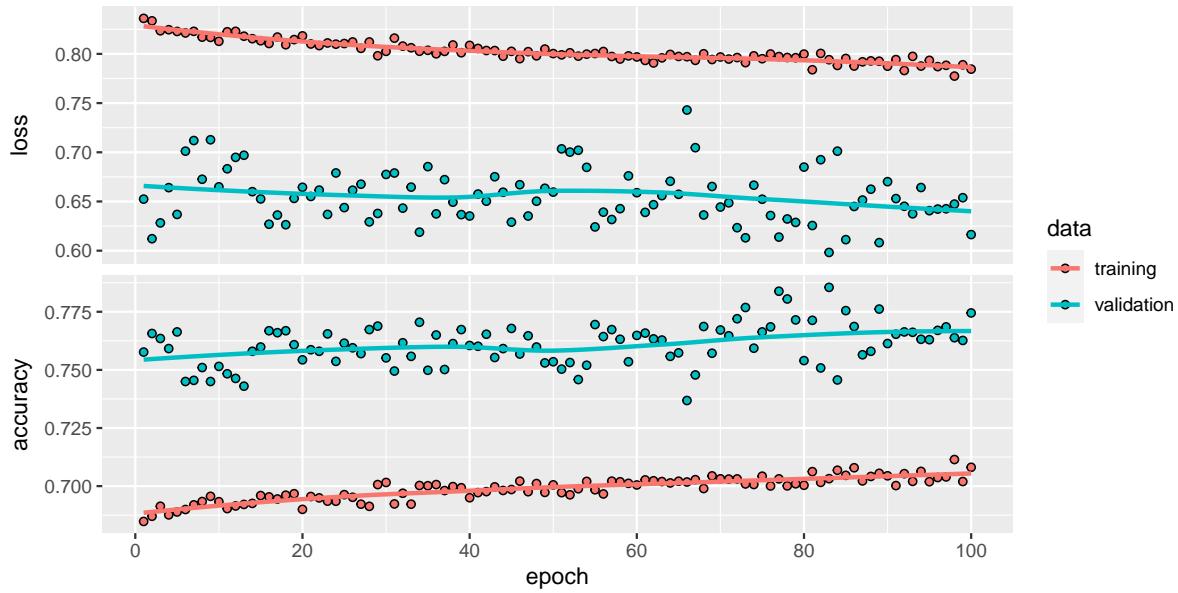


Figure 13.8: Loss and accuracy as a function of learned epochs for training and validation (test) datasets. Figure is based on `plot(fitfull)`.

Of course the accuracy or predictive performance depends on the networks structure. There are some overall rules (more layers, denser structure more data required), but often constructing the network is a trial-and-test situation. For the specific example, increasing dropout reduces performance and decreasing dropout does virtually not change it. Similarly, increasing the filter size in the two hidden 2D convolutional layers does not improve the performance. However, decreasing it does decrease.

A few remarks:

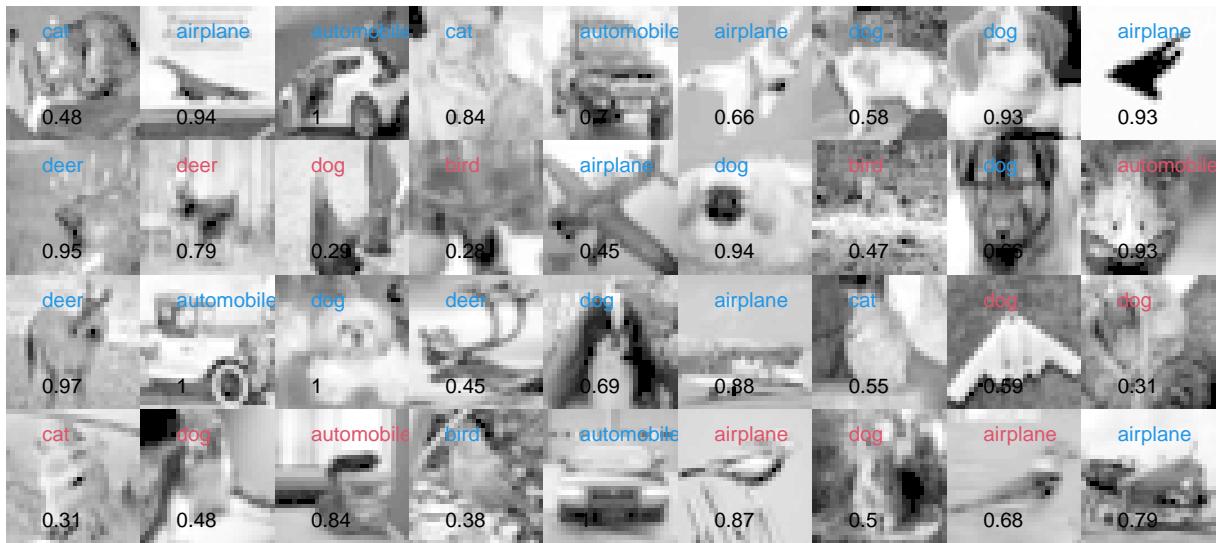


Figure 13.9: Sample test images with assigned categories and class probabilities as shown in Figure 13.5. Properly specified images are written in blue, wrongly in red. (See R-Code 13.7.)

- There are functions to convert a RGB image to a grayscale (e.g., `imager::grayscale()`) these are quite complex and involve additional overhead.
- Using the color 32×32 pixel RGB images with all categories, it is possible to tune the parameters to get to 75% validation accuracy in 25 epochs, and 79% after 50 epochs (batch size 32).

With larger machines, much higher performance can be achieved. See for example Figure 15 in <https://arxiv.org/pdf/1908.00709.pdf>. Note that the necessary time to train the network ranges between more than half a million GPU hours to less than ten hours. The high training time does not guarantee a high accuracy (here 96.35% and 96.47%). It is important to recognize the tremendous amount of parameters involved in all given examples.

13.5 Bibliographic remarks

Fleuret (2023) wrote very accessible book about deep learning, available also as pdf (<https://fleuret.org/public/lbdl.pdf>). The wikipedia page https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software summarizes different software frameworks for deep learning. The page <https://en.wikipedia.org/wiki/Backpropagation> gives an accessible account of backpropagation and illustrates the algorithm including actual gradient calculation for a single layer network.

There are several other R packages implementing neural nets. For example, `nnet` is very established and provides a feed-forward neural networks with a single hidden layer (Venables and Ripley, 2002). The package provides the handy function `multinom()` that fits multinomial log-linear models via neural networks. The R package `RWeka` (Hornik *et al.*, 2009) provides an R interface to the Java machine learning software Weka, which provides tools for data pre-processing, classification, regression, clustering, association rules, and visualization. The R package `RSNNS`

(Bergmeir and Benítez, 2012) provides the interface to the Stuttgart Neural Network Simulator (SNNS) (Zell *et al.*, 1994) which is a library implementing many standard forms of neural networks.

There are endless contributions to the topic of ML and NN. The following is a selective list that fit well into the framework of this chapter: <https://www.r-bloggers.com/understanding-the-magic-of-neural-networks/>, https://en.wikipedia.org/wiki/Deep_learning and <https://machinelearningmastery.com/what-is-deep-learning/>. The manuscript <https://arxiv.org/pdf/1908.00709.pdf> gives a nice summary of different types or learners, their different learning time and performance for several classical datasets (e.g., CIFAR10, CIFAR100, MNIST, . . .). The following are about more classical classification algorithms: <https://mark-borg.github.io/blog/2016/optdigits-classification/> and <https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm>,

13.6 Exercises and Problems

Problem 13.1 (Classification) Consider the classes `versicolor` and `virginica` of the `iris` data.

- a) Using a logistic GLM, construct a classifier for the two groups.
- b) Build a perceptron with logistic activation function to classify. Compare the result with the GLM from a).
- c) Construct a DNN and by monitoring loss and accuracy of training and validation data over epochs, illustrate overfitting.

Problem 13.2 (OCR with keras) Using the optical character recognition data set (See also Section 4.6), construct a neural net to predict the last ten digits. Discuss the result.

Hint: Start with R-Code 4.8.

Problem 13.3 Based on Section 13.4, try to confirm the following statements by adjusting the network structure accordingly.

Hint: testing a couple epochs are sufficient.

- a) Increasing dropout reduces performance. Decreasing does not change much.
- b) Increase filter size in the two hidden 2D convolutional layers does not improve the performance. However, decreasing does decrease.
- c) Increasing filter size in the second hidden layer does not improve the performance.
- d) Using less classes (reducing data sizes) increases accuracy.

Postscript

The following list gives the attached and loaded packages that were used to compile this document. If evaluating the code results in errors they might be due to outdated R-packages (many of these evolve quickly and regular updates are recommended).

R-Code 13.8: Session info of this document.

```
print(sessionInfo(), locale=FALSE)

## R Under development (unstable) (2023-01-31 r83741)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.3 LTS
##
## Matrix products: default
## BLAS:    /usr/lib/R-devel/lib/R/lib/libRblas.so
## LAPACK:  /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3;  LAPACK version 3.10.0
##
## attached base packages:
## [1] splines   parallel  grid       stats      graphics  grDevices utils
## [8] datasets  methods   base
##
## other attached packages:
##  [1] keras_2.11.0          neuralnet_1.44.2      evd_2.3-6.1
##  [4] ismev_1.42            mgcv_1.9-0           nlme_3.1-161
##  [7] extRemes_2.1-3        distillery_1.2-1     Lmoments_1.3-1
## [10] geoR_1.9-2            muhaz_1.2.6.4       survival_3.5-0
## [13] fma_2.5               forecast_8.20      wavethresh_4.7.2
## [16] gridExtra_2.3          plyr_1.8.8           lme4_1.1-31
## [19] Matrix_1.5-3          faraway_1.0.8      bootstrap_2019.6
## [22] ellipse_0.4.3         rattle_5.5.1        bitops_1.0-7
## [25] tibble_3.1.8          titanic_0.1.0      palmerpenguins_0.1.1
## [28] randomForest_4.7-1.1 adabag_4.2          doParallel_1.0.17
## [31] iterators_1.0.14      foreach_1.5.2       caret_6.0-93
## [34] lattice_0.20-45       ggplot2_3.4.0       rpart_4.1.19
## [37] partykit_1.2-16       libcoin_1.0-9       png_0.1-8
## [40] mvtnorm_1.1-3         kml_2.4.6          longitudinalData_2.4.5
```

```

## [43] misc3d_0.9-1           clv_0.3-2.2          class_7.3-21
## [46] cluster_2.1.4          mapdata_2.3.1        mclust_6.0.0
## [49] MASS_7.3-58.2          ade4_1.7-22          splancs_2.01-43
## [52] sp_1.6-0                deldir_1.0-6         maps_3.4.1
## [55] fields_14.1            viridis_0.6.2        viridisLite_0.4.1
## [58] spam_2.9-1              ncdf4_1.21          rgl_1.0.1
## [61] sm_2.2-5.7.1           knitr_1.42

##
## loaded via a namespace (and not attached):
## [1] jsonlite_1.8.2          magrittr_2.0.3        nloptr_2.0.3
## [4] vctrs_0.5.2              minqa_1.2.5          base64enc_0.1-3
## [7] htmltools_0.5.4          curl_5.0.0           Formula_1.2-4
## [10] pROC_1.18.0             TTR_0.24.3          parallelly_1.34.0
## [13] htmlwidgets_1.6.1        zoo_1.8-11           lubridate_1.8.0
## [16] whisker_0.4.1           lifecycle_1.0.3       pkgconfig_2.0.3
## [19] R6_2.5.1                fastmap_1.1.0        future_1.31.0
## [22] digest_0.6.31           colorspace_2.1-0     tfruns_1.5.1
## [25] fansi_1.0.4             compiler_4.3.0       withr_2.5.0
## [28] tseries_0.10-53         highr_0.10          tensorflow_2.11.0
## [31] lava_1.7.1              ModelMetrics_1.2.2.2 tools_4.3.0
## [34] lmtest_0.9-40            quantmod_0.4.20      future.apply_1.10.0
## [37] nnet_7.3-18              glue_1.6.2           quadprog_1.5-8
## [40] inum_1.0-4               reshape2_1.4.4       generics_0.1.3
## [43] recipes_1.0.4            gtable_0.3.1         data.table_1.14.6
## [46] utf8_1.2.3               pillar_1.8.1         stringr_1.5.0
## [49] dplyr_1.1.0              tidyselect_1.2.0     urca_1.3-3
## [52] stats4_4.3.0              xfun_0.37           hardhat_1.2.0
## [55] timeDate_4022.108        stringi_1.7.12       boot_1.3-28.1
## [58] evaluate_0.16             codetools_0.2-18     tcltk_4.3.0
## [61] cli_3.6.0                reticulate_1.28      munsell_0.5.0
## [64] Rcpp_1.0.10               globals_0.16.2       zeallot_0.1.0
## [67] fracdiff_1.5-2           gower_1.0.1          dotCall64_1.0-2
## [70] listenv_0.9.0             ipred_0.9-13         scales_1.2.1
## [73] xts_0.12.2               prodlim_2019.11.13   purrr_1.0.1
## [76] rlang_1.0.6

```

Appendix A

Calculus

In this chapter we present some of the most important ideas and concepts of calculus. For example, we will not discuss sequences and series. It is impossible to give a formal, mathematically precise exposition. Further, we cannot present all rules, identities, guidelines or even tricks.

A.1 Functions

We start with one of the most basic concepts, a formal definition that describes a relation between two sets.

Definition A.1. A function f from a set D to a set W is a rule that assigns a unique value element $f(x) \in W$ to each element $x \in D$. We write

$$f : D \rightarrow W \tag{A.1}$$

$$x \mapsto f(x) \tag{A.2}$$

The set D is called the domain, the set W is called the range (or target set or codomain).

The graph of a function f is the set $\{(x, f(x)) : x \in D\}$. \diamond

The function will not necessarily map to every element in W , and there may be several elements in D with the same image in W . These functions are characterized as follows.

Definition A.2.

1. A function f is called injective, if the image of two different elements in D is different.
2. A function f is called surjective, if for every element y in W there is at least one element x in D such that $y = f(x)$.
3. A function f is called bijective if it is surjective and injective. Such a function is also called a *one-to-one* function. \diamond

As an illustration, the first point can be ‘translated’ to $\forall x, z \in D, x \neq z \implies f(x) \neq f(z)$, which is equivalent to $\forall x, z \in D, f(x) = f(z) \implies x = z$.

By restricting the range, it is possible to render a function surjective. It is often possible to restrict the domain to obtain a locally bijective function.

In general, there is virtually no restriction on the domain and codomain. However, we often work with real functions, i.e., $D \subset \mathbb{R}$ and $W \subset \mathbb{R}$.

There are many different characterizations of functions. Some relevant one are as follows.

Definition A.3. A real function f is

1. periodic if there exists an $\omega > 0$ such that $f(x + \omega) = f(x)$ for all $x \in D$. The smallest value ω is called the period of f ;
2. called increasing if $f(x) \leq f(x + h)$ for all $h \geq 0$. In case of strict inequalities, we call the function strictly increasing. Similar definitions hold when reversing the inequalities. \diamond

The inverse $f^{-1}(y)$ of a bijective function $f : D \rightarrow W$ is defined as

$$\begin{aligned} f^{-1} : W &\rightarrow D \\ y &\mapsto f^{-1}(y), \text{ such that } y = f(f^{-1}(y)). \end{aligned} \tag{A.3}$$

Subsequently, we require the “inverse” of increasing functions by generalizing the previous definition. We call these function quantile functions.

To capture the behavior of a function locally, say at a point $x_0 \in D$, we use the concept of a *limit*.

Definition A.4. Let $f : D \rightarrow \mathbb{R}$ and $x_0 \in D$. The limit of f as x approaches x_0 is a , written as $\lim_{x \rightarrow x_0} f(x) = a$ if for every $\epsilon > 0$, there exists a $\delta > 0$ such that for all $x \in D$ with $0 < |x - x_0| < \delta \implies |f(x) - a| < \epsilon$. \diamond

The latter definition does not assume that the function is defined at x_0 .

It is possible to define “directional” limits, in the sense that x approaches x_0 from above (from the right side) or from below (from the left side). These limits are denoted with

$$\lim_{\substack{x \rightarrow x_0^+ \\ x \searrow x_0}} \quad \text{for the former; or} \quad \lim_{\substack{x \rightarrow x_0^- \\ x \nearrow x_0}} \quad \text{for the latter.} \tag{A.4}$$

We are used to interpret graphs and when we sketch an arbitrary function we often use a single, continuous line. This concept of not lifting the pen while sketching is formalized as follows and linked directly to limits, introduced above.

Definition A.5. A function f is continuous in x_0 if the following limits exist

$$\lim_{h \nearrow 0} f(x_0 + h) \quad \lim_{h \searrow 0} f(x_0 + h) \tag{A.5}$$

and are equal to $f(x_0)$. \diamond

There are many other approaches to define continuity, for example in terms of neighborhoods, in terms of limits of sequences.

Another very important (local) characterization of a function is the derivative, which quantifies the (infinitesimal) rate of change.

Definition A.6. The derivative of a function $f(x)$ with respect to the variable x at the point x_0 is defined by

$$f'(x_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h}, \quad (\text{A.6})$$

provided the limit exists. We also write $\frac{df(x_0)}{dx} = f'(x_0)$.

If the derivative exists for all $x_0 \in D$, the function f is differentiable. \diamond

Some of the most important properties in differential calculus are:

Property A.1. 1. *Differentiability implies continuity.*

2. (*Mean value theorem*) For a continuous function $f : [a, b] \rightarrow \mathbb{R}$, which is differentiable on (a, b) there exists a point $\xi \in (a, b)$ such that $f'(\xi) = \frac{f(b) - f(a)}{b - a}$.

The *integral* of a (positive) function quantifies the area between the function and the x -axis. A mathematical definition is a bit more complicated.

Definition A.7. Let $f(x) : D \rightarrow \mathbb{R}$ a function and $[a, b] \in D$ a finite interval such that $|f(x)| < \infty$ for $x \in [a, b]$. For any n , let $t_0 = a < t_1 < \dots < t_n = b$ a partition of $[a, b]$.

The integral of f from a to b is defined as

$$\int_a^b f(x) dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(t_i)(t_i - t_{i-1}). \quad (\text{A.7})$$

\diamond

For non-finite a and b , the definition of the integral can be extended via limits.

Property A.2. (*Fundamental theorem of calculus (I)*). Let $f : [a, b] \rightarrow \mathbb{R}$ continuous. For all $x \in [a, b]$, let $F(x) = \int_a^x f(u) du$. Then F is continuous on $[a, b]$, differentiable on (a, b) and $F'(x) = f(x)$, for all $x \in (a, b)$.

The function F is often called the antiderivative of f . There exists a second form of the previous theorem that does not assume continuity of f but only Riemann integrability, that means that an integral exists.

Property A.3. (*Fundamental theorem of calculus (II)*). Let $f : [a, b] \rightarrow \mathbb{R}$. And let F such that $F'(x) = f(x)$, for all $x \in (a, b)$. If f is Riemann integrable then $\int_a^b f(u) du = F(b) - F(a)$.

There are many ‘rules’ to calculate integrals. One of the most used ones is called integration by substitution and is as follows.

Property A.4. Let I be an interval and $\varphi : [a, b] \rightarrow I$ be a differentiable function with integrable derivative. Let $f : I \rightarrow \mathbb{R}$ be a continuous function. Then

$$\int_{\varphi(a)}^{\varphi(b)} f(u) du = \int_a^b f(\varphi(x))\varphi'(x) dx. \quad (\text{A.8})$$

A.2 Functions in Higher Dimensions

We denote with \mathbb{R}^m the vector space with elements $\mathbf{x} = (x_1, \dots, x_m)^\top$, called vectors, equipped with the standard operations. We will discuss *vectors* and *vector notation* in more details in the subsequent chapter.

A natural extension of a real function is as follows. The set D is subset of \mathbb{R}^m and thus we write

$$\begin{aligned} f : D \subset \mathbb{R}^m &\rightarrow W \\ \mathbf{x} &\mapsto f(\mathbf{x}). \end{aligned} \tag{A.9}$$

Note that we keep $W \subset \mathbb{R}$.

The concept of limit and continuity translates one-to-one. Differentiability, however, is different and slightly more delicate.

Definition A.8. The partial derivative of $f : D \subset \mathbb{R} \rightarrow W$ with respect to x_j is defined by

$$\frac{\partial f(\mathbf{x})}{\partial x_j} = \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_{j-1}, x_j + h, x_{j+1}, \dots, x_m) - f(x_1, \dots, x_m)}{h}, \tag{A.10}$$

(provided it exists). \diamond

The derivative of f with respect to all components is thus a vector

$$\mathbf{f}'(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_m} \right)^\top \tag{A.11}$$

Hence $\mathbf{f}'(\mathbf{x})$ is a vector valued function from D to \mathbb{R}^m and is called the gradient of f at \mathbf{x} , also denoted with $\text{grad}(f(\mathbf{x})) = \nabla f(\mathbf{x})$.

Remark A.1. The existence of partial derivatives is not sufficient for the differentiability of the function f . \clubsuit

In a similar fashion, higher order derivatives can be calculated. For example, taking the derivative of each component of (A.11) with respect to all components is an matrix with components

$$\mathbf{f}''(\mathbf{x}) = \left(\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \right), \tag{A.12}$$

called the Hessian matrix.

It is important to realize that the second derivative constitutes a set of derivatives of f : all possible double derivatives.

A.3 Approximating Functions

Quite often, we want to approximate functions.

Property A.5. *Let $f : D \rightarrow \mathbb{R}$ with continuous $m + 1$ derivatives. Then there exists $\xi \in [a, x]$ such that*

$$\begin{aligned} f(x) &= f(a) + f'(a)(x - a) + \frac{1}{2}f''(a)(x - a)^2 + \dots \\ &\quad + \frac{1}{m!}f^{(m)}(a)(x - a)^m + \frac{1}{(m+1)!}f^{(m+1)}(\xi)(x - a)^{m+1} \end{aligned} \tag{A.13}$$

We call (A.13) Taylor's formula and the last term, often denoted by $R_n(x)$, as the remainder of order n . Taylor's formula is an extension of the mean value theorem.

If the function has bounded derivatives, the remainder $R_n(x)$ converges to zero as $x \rightarrow a$.

Hence, if the function is at least twice differentiable in a neighborhood of a then

$$f(a) + f'(a)(x - a) + \frac{1}{2}f''(a)(x - a)^2 \tag{A.14}$$

is the best quadratic approximation in this neighborhood.

If all derivatives of f exist in an open interval I with $a \in I$, we have for all $x \in I$

$$f(x) = \sum_{r=0}^{\infty} \frac{1}{r!} f^{(r)}(a)(x - a)^r \tag{A.15}$$

Often the approximation is for $x = a + h$, h small.

Taylor's formula can be expressed for multivariate real functions. Without stating the precise assumptions we consider here the following example

$$f(\mathbf{a} + \mathbf{h}) = \sum_{r=0}^{\infty} \sum_{i:i_1+\dots+i_n=r} \frac{1}{i_1! i_2! \dots i_n!} \frac{\partial^r f(\mathbf{a})}{\partial x_{i_1} \dots \partial x_{i_n}} h_1^{i_1} h_2^{i_2} \dots h_n^{i_n}, \tag{A.16}$$

extending (A.15) with $\mathbf{x} = \mathbf{a} + \mathbf{h}$.

Appendix B

Linear Algebra

In this chapter we cover the most important aspects of linear algebra, namely of notational nature.

B.1 Vectors, Matrices and Operations

A collection of p real numbers is called a vector, an array of $n \times m$ real numbers is called a matrix. We write

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix}, \quad \mathbf{A} = (a_{ij}) = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix}. \quad (\text{B.1})$$

Providing the dimensions are coherent, vector and matrix addition (and subtraction) is performed componentwise, as is scalar multiplication. That means, for example, that $\mathbf{x} \pm \mathbf{y}$ is a vector with elements $x_i \pm y_i$ and $c\mathbf{A}$ is a matrix with elements ca_{ij} .

The $n \times n$ identity matrix \mathbf{I} is defined as the matrix with ones on the diagonal and zeros elsewhere. We denote the vector with solely one elements with $\mathbf{1}$ similarly, $\mathbf{0}$ is a vector with only zero elements. A matrix with entries d_1, \dots, d_n on the diagonal and zero elsewhere is denoted with $\text{diag}(d_1, \dots, d_n)$ or $\text{diag}(d_i)$ for short and called a diagonal matrix. Hence, $\mathbf{I} = \text{diag}(\mathbf{1})$.

To indicate the i th- j th element of \mathbf{A} , we use $(\mathbf{A})_{ij}$. The transpose of a vector or a matrix flips its dimension. When a matrix is transposed, i.e., when all rows of the matrix are turned into columns (and vice-versa), the elements a_{ij} and a_{ji} are exchanged. Thus $(\mathbf{A}^\top)_{ij} = (\mathbf{A})_{ji}$. The vector $\mathbf{x}^\top = (x_1, \dots, x_p)$ is termed a row vector. We work mainly with column vectors as shown in (B.1).

In the classical setting of real numbers, there is only one type of multiplication. As soon as we have several dimensions, several different types of multiplications exist, notably scalar multiplication, matrix multiplication and inner product (and actually more such as the vector product, outer product).

Let \mathbf{A} and \mathbf{B} be two $n \times p$ and $p \times m$ matrices. Matrix multiplication \mathbf{AB} is defined as

$$\mathbf{AB} = \mathbf{C} \quad \text{with} \quad (\mathbf{C})_{ij} = \sum_{k=1}^p a_{ik} b_{kj}. \quad (\text{B.2})$$

This last equation shows that the matrix \mathbf{I} is the neutral element (or identity element) of the matrix multiplication.

Definition B.1. The inner product between two p -vectors \mathbf{x} and \mathbf{y} is defined as $\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^p x_i y_i$. There are several different notations used: $\mathbf{x}^\top \mathbf{y} = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{x} \cdot \mathbf{y}$.

If for an $n \times n$ matrix \mathbf{A} there exists an $n \times n$ matrix \mathbf{B} such that

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}, \quad (\text{B.3})$$

then the matrix \mathbf{B} is uniquely determined by \mathbf{A} and is called the inverse of \mathbf{A} , denoted by \mathbf{A}^{-1} .

B.2 Linear Spaces and Basis

The following definition formalizes one of the main spaces we work in.

Definition B.2. A vector space over \mathbb{R} is a set V with the following two operations:

1. $+ : V \times V \rightarrow V$ (vector addition)
2. $\cdot : \mathbb{R} \times V \rightarrow V$ (scalar multiplication).

◇

Typically, V is \mathbb{R}^p , $p \in \mathbb{N}$.

In the following we assume a fixed d and the usual operations on the vectors.

Definition B.3. 1. The vectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ are linearly dependent if there exists scalars a_1, \dots, a_k (not all equal to zero), such that $a_1 \mathbf{v}_1 + \dots + a_k \mathbf{v}_k = \mathbf{0}$.

2. The vectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ are linearly independent if $a_1 \mathbf{v}_1 + \dots + a_k \mathbf{v}_k = \mathbf{0}$ cannot be satisfied by any scalars a_1, \dots, a_k (not all equal to zero). ◇

In a set of linearly dependent vectors, some vector can be expressed as a linear combination of the others.

Definition B.4. The set of vectors $\{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ is a basis of a vectors space V if the set is linearly independent and any other vector $\mathbf{v} \in V$ can be expressed by $\mathbf{v} = v_1 \mathbf{b}_1 + \dots + v_d \mathbf{b}_d$. ◇

The following proposition summarizes some of the relevant properties of a basis.

Property B.1. 1. *The decomposition of a vector $\mathbf{v} \in V$ in $\mathbf{v} = v_1 \mathbf{b}_1 + \dots + v_d \mathbf{b}_d$ is unique.*

2. *All bases of V have the same cardinality, which is called the dimension of V , $\dim(V)$.*
3. *If there are two bases $\{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ and $\{\mathbf{e}_1, \dots, \mathbf{e}_d\}$ then there exists a $d \times d$ matrix \mathbf{A} such that $\mathbf{e}_i = \mathbf{Ab}_i$, for all i .*

Definition B.5. The standard basis, or canonical basis of $V = \mathbb{R}^d$ is $\{\mathbf{e}_1, \dots, \mathbf{e}_d\}$ with $\mathbf{e}_i = (0, \dots, 0, 1, 0, \dots)^\top$, i.e., the vector with a one at the i th position and zero elsewhere. ◇

Definition B.6. Let \mathbf{A} be a $n \times m$ matrix. The column rank of the matrix is the dimension of the subspace that the m columns of \mathbf{A} span and is denoted by $\text{rank}(\mathbf{A})$. A matrix is said to have full rank if $\text{rank}(\mathbf{A}) = \min(n, m)$.

The row rank is the column rank of \mathbf{A}^\top . ◇

Some fundamental properties of the rank are as follows.

Property B.2. Let \mathbf{A} be a $n \times m$ matrix.

1. The column rank and row rank are identical.
2. $\text{rank}(\mathbf{A}^\top \mathbf{A}) = \text{rank}(\mathbf{A}\mathbf{A}^\top) = \text{rank}(\mathbf{A})$.
3. $\text{rank}(\mathbf{A}) \leq \dim(V)$.
4. $\text{rank}(\mathbf{A}) \leq \min(m, n)$.
5. For an appropriately sized matrix \mathbf{B} $\text{rank}(\mathbf{A} + \mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B})$ and $\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$.

B.3 Projections

We consider classical Euclidean vector spaces with elements $\mathbf{x} = (x_1, \dots, x_p)^\top \in \mathbb{R}^p$ with Euclidean norm $\|\mathbf{x}\| = (\sum_i x_i^2)^{1/2}$.

To illustrate projections, consider the setup illustrated in Figure B.1, where \mathbf{y} and \mathbf{a} are two vectors in \mathbb{R}^2 . The subspace spanned by \mathbf{a} is

$$\{\lambda \mathbf{a}, \lambda \in \mathbb{R}\} = \{\lambda \mathbf{a}/\|\mathbf{a}\|, \lambda \in \mathbb{R}\} \quad (\text{B.4})$$

where the second expression is based on a normalized vector $\mathbf{a}/\|\mathbf{a}\|$. By the (geometric) definition of the inner product (dot product),

$$\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^\top \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta \quad (\text{B.5})$$

where θ is the angle between the vectors. Classical trigonometric properties state that the projection is $\mathbf{a}/\|\mathbf{a}\| \cdot \|\mathbf{y}\| \cos(\theta)$. Hence, the projected vector is

$$\frac{\mathbf{a}}{\|\mathbf{a}\|} \frac{\mathbf{a}^\top}{\|\mathbf{a}\|} \mathbf{y} = \mathbf{a} (\mathbf{a}^\top \mathbf{a})^{-1} \mathbf{a}^\top \mathbf{y}. \quad (\text{B.6})$$

In statistics we often encounter expressions like this last term. For example, ordinary least squares (“classical” multiple regression) is a projection of the vector \mathbf{y} onto the column space spanned by \mathbf{X} , i.e., the space spanned by the columns of the matrix \mathbf{X} . The projection is $\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$. Usually, the column space is in a lower dimension.

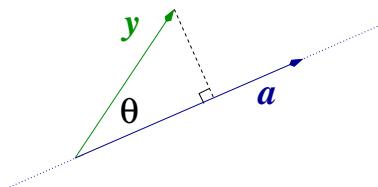


Figure B.1: Projection of the vector \mathbf{y} onto the subspace spanned by \mathbf{a} .

Remark B.1. Projection matrices (like $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$) have many nice properties such as being symmetric, being idempotent, i.e., $\mathbf{H} = \mathbf{HH}$, having eigenvalues within $[0, 1]$, (see next section), $\text{rank}(\mathbf{H}) = \text{rank}(\mathbf{X})$, etc. ♣

B.4 Matrix Decompositions

In this section we elaborate representations of a matrix as a product of two or three other matrices.

Let \mathbf{x} be a non-zero n -vector (i.e., at least one element is not zero) and \mathbf{A} an $n \times n$ matrix. We can interpret $\mathbf{A}(\mathbf{x})$ as a function that maps \mathbf{x} to \mathbf{Ax} . We are interested in vectors that change by a scalar factor by such a mapping

$$\mathbf{Ax} = \lambda \mathbf{x}, \quad (\text{B.7})$$

where λ is called an eigenvalue and \mathbf{x} an eigenvector.

A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ has n eigenvalues, $\{\lambda_1, \dots, \lambda_n\}$, albeit not necessarily different and not necessarily real. The set of eigenvalues and the associated eigenvectors denotes an eigendecomposition.

For all square matrices, the set of eigenvectors span an orthogonal basis, i.e., are constructed that way.

We often denote the set of eigenvectors with $\gamma_1, \dots, \gamma_n$. Let Γ be the matrix with columns γ_i , i.e., $\Gamma = (\gamma_1, \dots, \gamma_n)$. Thus $\mathbf{A}\Gamma = (\mathbf{A}\gamma_1, \dots, \mathbf{A}\gamma_n) = (\lambda_1\gamma_1, \dots, \lambda_n\gamma_n)$. Therefore, $\Gamma \text{diag}(\lambda_1, \dots, \lambda_n) = \mathbf{A}\Gamma$. Since the inverse of Γ is Γ^\top (due to the orthogonality property of the eigenvectors $\Gamma^\top \Gamma = \mathbf{I}$), we get $\Gamma^\top \Gamma \text{diag}(\lambda_1, \dots, \lambda_n) = \Gamma^\top \mathbf{A}\Gamma$ and, hence,

$$\Gamma^\top \mathbf{A}\Gamma = \text{diag}(\lambda_1, \dots, \lambda_n) \text{ and } \mathbf{A} = \Gamma \text{diag}(\lambda_1, \dots, \lambda_n) \Gamma^\top. \quad (\text{B.8})$$

In cases of non-square matrices, an eigendecomposition is not possible and a more general approach is required. The so-called singular value decomposition (SVD) works for any $n \times m$ matrix \mathbf{B} ,

$$\mathbf{B} = \mathbf{UDV}^\top \quad (\text{B.9})$$

where \mathbf{U} is an $n \times \min(n, m)$ orthogonal matrix (i.e., $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_n$), \mathbf{D} is an diagonal matrix containing the so-called singular values and \mathbf{V} is an $\min(n, m) \times m$ orthogonal matrix (i.e., $\mathbf{V}^\top \mathbf{V} = \mathbf{I}_m$).

We say that the columns of \mathbf{U} and \mathbf{V} are the left-singular vectors and right-singular vectors, respectively.

Note however, that the dimensions of the corresponding matrices differ in the literature, some write \mathbf{U} and \mathbf{V} as square matrices and \mathbf{D} as a rectangular matrix.

Remark B.2. Given an SVD of \mathbf{B} , the following two relations hold:

$$\mathbf{BB}^\top = \mathbf{UDV}^\top (\mathbf{UDV}^\top)^\top = \mathbf{UDV}^\top \mathbf{VDU}^\top = \mathbf{UDDU}^\top \quad (\text{B.10})$$

$$\mathbf{B}^\top \mathbf{B} = (\mathbf{UDV}^\top)^\top \mathbf{UDV}^\top = \mathbf{VDU}^\top \mathbf{UDV}^\top = \mathbf{VDDV}^\top \quad (\text{B.11})$$

and hence the columns of \mathbf{U} and \mathbf{V} are eigenvectors of \mathbf{BB}^\top and $\mathbf{B}^\top\mathbf{B}$, respectively, and most importantly, the elements of \mathbf{D} are the square roots of the (non-zero) eigenvalues of \mathbf{BB}^\top or $\mathbf{B}^\top\mathbf{B}$. \clubsuit

Besides an SVD there are many other matrix factorization. We often use the so-called Cholesky factorization, as - to a certain degree - it generalizes the concept of a square root for matrices. Assume that all eigenvalues of \mathbf{A} are strictly positive, then there exists a unique lower triangular matrix \mathbf{L} with positive entries on the diagonal such that $\mathbf{A} = \mathbf{LL}^\top$. There exist very efficient algorithm to calculate \mathbf{L} and solving large linear systems is often based on a Cholesky factorization.

The determinant of a square matrix essentially describes the change in “volume” that associated linear transformation induces. The formal definition is quite complex but it can be written as $\det(\mathbf{A}) = \prod_{i=1}^n \lambda_i$ for matrices with real eigenvalues.

The trace of a matrix is the sum of its diagonal entries.

B.5 Positive Definite Matrices

Besides matrices containing covariates, we often work with variance-covariance matrices, which represent an important class of matrices as we see now.

Definition B.7. A $n \times n$ matrix \mathbf{A} is positive definite (pd) if

$$\mathbf{x}^\top \mathbf{Ax} > 0, \quad \text{for all } \mathbf{x} \neq \mathbf{0}. \quad (\text{B.12})$$

Further, if $\mathbf{A} = \mathbf{A}^\top$, the matrix is symmetric positive definite (spd). \diamond

Relevant properties of spd matrices $\mathbf{A} = (a_{ij})$ are given as follows.

Property B.3. 1. $\text{rank}(\mathbf{A}) = n$

2. *the determinant is positive, $\det(\mathbf{A}) > 0$*

3. *all eigenvalues are positive, $\lambda_i > 0$*

4. *all elements on the diagonal are positive, $a_{ii} > 0$*

5. $a_{ii}a_{jj} - a_{ij}^2 > 0, i \neq j$

6. $a_{ii} + a_{jj} - 2|a_{ij}| > 0, i \neq j$

7. \mathbf{A}^{-1} is spd

8. *all principal sub-matrices of \mathbf{A} are spd.*

If

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \in \mathbb{R}^{p+q \times p+q} \quad (\text{B.13})$$

with $\mathbf{A}_{11} \in \mathbb{R}^{p \times p}$ nonsingular (an inverse exists), then \mathbf{A} is nonsingular if and only if $\mathbf{C} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}$ is nonsingular. Further,

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}_{11}^{-1} - \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C}^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C}^{-1} \\ -\mathbf{C}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{C}^{-1} \end{pmatrix}. \quad (\text{B.14})$$

It also holds that $\det(\mathbf{A}) = \det(\mathbf{A}_{11}) \det(\mathbf{C})$.

Appendix C

Random Variables

C.1 Basics of Probability Theory

In order to define the term random variable, some basic principles are needed first.

The set of all possible outcomes of an experiment is called the sample space, denoted by Ω . Each outcome of an experiment $\omega \in \Omega$ is called an elementary event. A subset of the sample space Ω is called an event, denoted by $A \subset \Omega$.

Informally, a probability P can be considered the value of a function defined for an event of the sample set and assuming values in the interval $[0, 1]$, that is, $P(A) \in [0, 1]$.

In order to introduce probability theory formally one needs some technical terms (σ -algebra, measure space, \dots). However, the axiomatic structure of A. Kolmogorov can also be described accessibly as follows and is sufficient as a basis for our purposes.

A probability measure must satisfy the following axioms:

1. $0 \leq P(A) \leq 1$, for every event A ,
2. $P(\Omega) = 1$,
3. $P(\cup_i A_i) = \sum_i P(A_i)$, for $A_i \cap A_j = \emptyset$, $i \neq j$.

Informally, a probability function P assigns a value in $[0, 1]$, i.e., the probability, to each event of the sample space constraint to:

1. the probability of an event is never smaller than 0 or greater than 1,
2. the probability of the whole sample space is 1,
3. the probability of several events is equal to the sum of the individual probabilities, if the events are mutually exclusive.

Probabilities are often visualized with Venn diagrams (Figure C.1), which clearly and intuitively illustrate more complex facts, such as:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B), \quad (\text{C.1})$$

$$P(A | B) = \frac{P(A \cap B)}{P(B)}. \quad (\text{C.2})$$

We consider a random variable as a *function* that assigns values to the outcomes (events) of a random experiment, that is, these values or values in the interval are assumed with certain probabilities. These values are called realizations of the random variable.

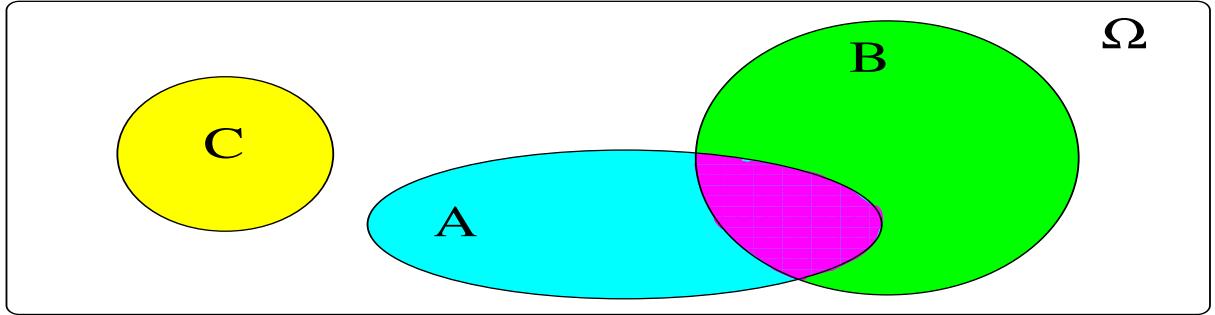


Figure C.1: Venn diagram

The following definition gives a (unique) characterization of random variables. In subsequent sections, we will see additional characterizations. These, however, will depend on the what type of values the random variable can take.

Definition C.1. The distribution function (cumulative distribution function, cdf) of a random variable X is

$$F(x) = F_X(x) = \Pr(X \leq x), \quad \text{for all } x. \quad (\text{C.3})$$

◊

Property C.1. *A distribution function $F_X(x)$ is*

1. *monotonically increasing, i.e. for $x < y$, $F_X(x) \leq F_X(y)$.*
2. *right-continuous, i.e. $\lim_{\epsilon \searrow 0} F_X(x + \epsilon) = F_X(x)$, for all $x \in \mathbb{R}$.*
3. *normalized, i.e. $\lim_{x \rightarrow -\infty} F_X(x) = 0$ and $\lim_{x \rightarrow \infty} F_X(x) = 1$.*

Random variables are denoted with uppercase letters (e.g. X , Y), while realizations are denoted by the corresponding lowercase letters (x , y). This means that the theoretical concept, or the random variable as a function, is denoted by uppercase letters. Actual values or data, for example the columns in your dataset, would be denoted with lowercase letters. For further characterizations of random variables, we need to differentiate according to the sample space of the random variables.

C.2 Discrete Distributions

A random variable is called discrete when it can assume only a finite or countably infinite number of values, as illustrated in the following two examples.

Example C.1. Let X be the sum of the roll of two dice. The random variable X assumes the values $2, 3, \dots, 12$. Figure C.2 illustrates the probabilities and distribution function. The distribution function (as for all discrete random variables) is piece-wise constant with jumps equal to the probability of that value. ♣

Example C.2. A boy practices *free throws*, i.e., foul shots to the basket standing at a distance of 15 ft to the board. Let the random variable X be the number of throws that are necessary until the boy succeeds. Theoretically, there is no upper bound on this number. Hence X can take the values $1, 2, \dots$. ♣

Another way of describing discrete random variables is the probability mass function, defined as follows.

Definition C.2. The probability mass function (pmf) of a discrete random variable X is defined by $f_X(x) = P(X = x)$. ◇

In other words, the pmf gives probabilities that the random variables takes one single value, the cdf gives probabilities that the random variables takes that or any smaller value.

Property C.2. Let X be a discrete random variable with mass function $f_X(x)$ and distribution function $F_X(x)$. Then:

1. The probability mass function satisfies $f_X(x) \geq 0$ for all $x \in \mathbb{R}$.
2. $\sum_i f_X(x_i) = 1$.
3. The values $f_X(x_i) > 0$ are the “jumps” in x_i of $F_X(x)$.
4. $F_X(x_i) = \sum_{k; x_k \leq x_i} f_X(x_k)$.

The last two points show that there is a one-to-one relation (also called a bijection) between the cdf and pmf. Given one, we can construct the other.

Figure C.2 illustrates the pmf and cdf of the random variable X as given in Example C.1. The jump locations and sizes (discontinuities) of the cdf correspond to probabilities given in the left panel. Notice that we have emphasized the right continuity of the cdf (see Proposition C.1.2) with the additional dot.

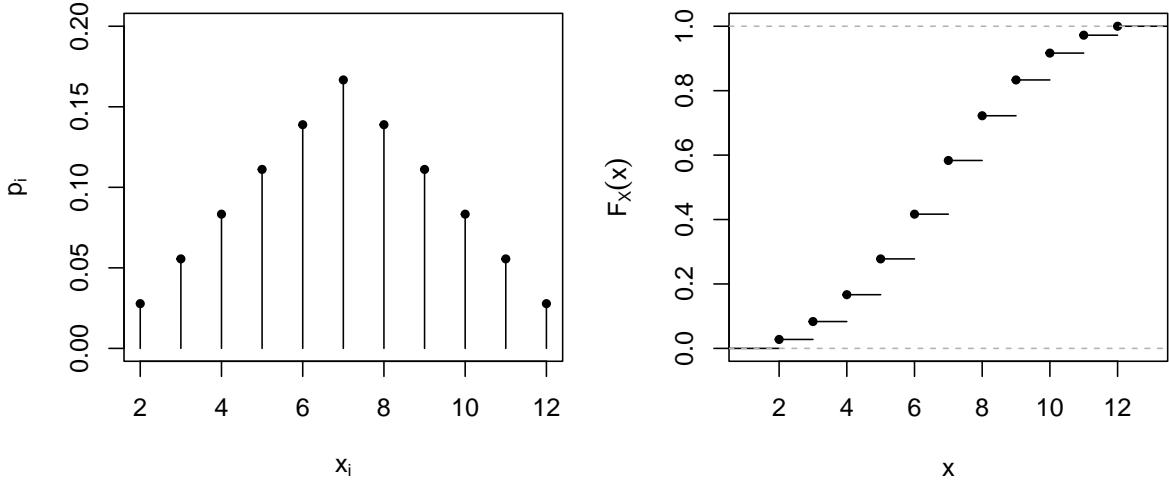


Figure C.2: Probability mass function (left) and cumulative distribution function (right) of $X = \text{the sum of the roll of two dice}$.

C.3 Continuous Distributions

A random variable is called continuous if it can (theoretically) assume any value from one or several intervals. This means that the number of possible values within the sample space is uncountably infinite. Therefore, it is impossible to assign one probability value to one (elementary) event. Or in other words, given an infinite amount of possible outcomes, the likeliness of one particular value being the outcome becomes zero. For this reason, we need to consider outcomes that are contained in some interval. Hence, the probability is described by an integral, as areas under the probability density function, which is formally defined as follows.

Definition C.3. The probability density function (density function, pdf) $f_X(x)$, or density for short, of a continuous random variable X is defined by

$$P(a < X \leq b) = \int_a^b f_X(x)dx, \quad a < b. \quad (\text{C.4})$$

◊

The density function does not give directly a probability and thus cannot be compared to the probability mass function. The following properties are nevertheless similar to Property C.2.

Property C.3. Let X be a continuous random variable with density function $f_X(x)$ and distribution function $F_X(x)$. Then:

1. The density function satisfies $f_X(x) \geq 0$ for all $x \in \mathbb{R}$ and $f_X(x)$ is continuous almost everywhere.
2. $\int_{-\infty}^{\infty} f_X(x)dx = 1$.

$$3. f_X(x) = F'_X(x) = \frac{dF_X(x)}{dx}.$$

$$4. F_X(x) = \int_{-\infty}^x f_X(y) dy.$$

5. The cumulative distribution function $F_X(x)$ is continuous everywhere.

$$6. P(X = x) = 0.$$

As given by Property C.3.3 and 4, there is again a bijection between the density function and the cumulative distribution function: if we know one we can construct the other. Actually, there is a third characterization of random variables, called the quantile function, which is essentially the inverse of the cdf. That means, we are interested in values x for which $F_X(x) = p$.

Definition C.4. The quantile function $Q_X(p)$ of a random variable X with (strictly) monotone cdf $F_X(x)$ is defined by

$$Q_X(p) = F_X^{-1}(p), \quad 0 < p < 1, \tag{C.5}$$

i.e., the quantile function is equivalent to the inverse of the distribution function. \diamond

For discrete random variables the cdf is not continuous (see the plateaus in the right panel of Figure C.2) and the inverse does not exist. The quantile function returns the minimum value of x from amongst all those values with probability $p \leq P(X \leq x) = F_X(x)$, more formally,

$$Q_X(p) = \inf_{x \in \mathbb{R}} \{p \leq F_X(x)\}, \quad 0 < p < 1. \tag{C.6}$$

Definition C.5. The median ν of a continuous random variable X with cdf $F_X(x)$ is defined by

$$\nu = Q_X(1/2). \tag{C.7}$$

\diamond

Example C.3. The continuous uniform distribution $\mathcal{U}(a, b)$ is defined by a constant density function over the interval $[a, b]$, $a < b$, i.e. $f(x) = \begin{cases} \frac{1}{b-a}, & \text{if } a \leq x \leq b, \\ 0, & \text{otherwise.} \end{cases}$

The quantile function is $Q_X(p) = a + p(b - a)$ for $0 < p < 1$. Figure C.3 shows the density and cumulative distribution function of the uniform distribution $\mathcal{U}(0, 1)$. \clubsuit

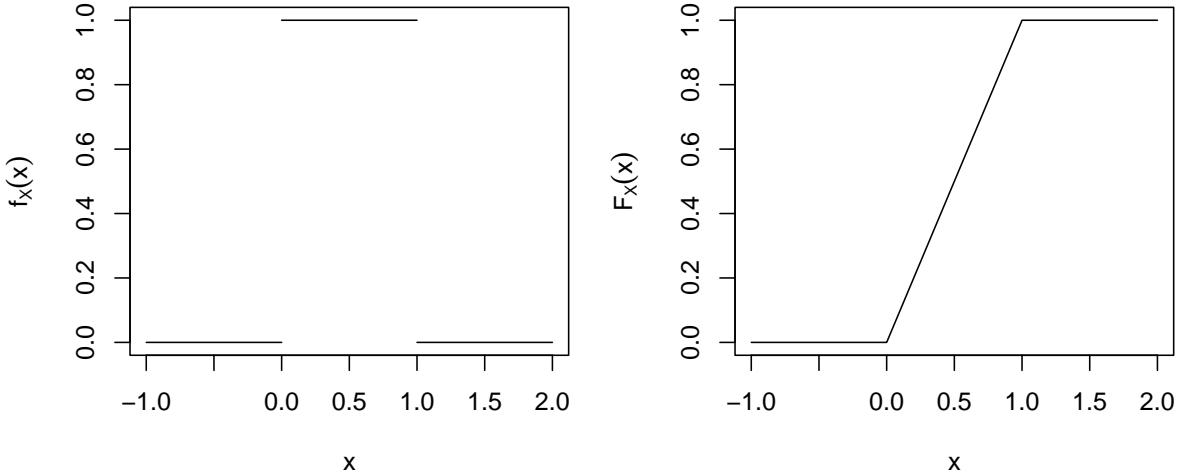


Figure C.3: Density and distribution function of the uniform distribution $\mathcal{U}(0, 1)$.

C.4 Expectation, Variance and Moments

Density, cumulative distribution function or quantile function uniquely characterize random variables. Often we do not require such a complete definition and “summary” values are sufficient. We introduce a measure of location and scale.

Definition C.6. The expectation of a discrete random variable X is defined by

$$\mathbb{E}(X) = \sum_i x_i P(X = x_i). \quad (\text{C.8})$$

The expectation of a continuous random variable X is defined by

$$\mathbb{E}(X) = \int_{\mathbb{R}} x f_X(x) dx, \quad (\text{C.9})$$

where $f_X(x)$ denotes the density of X . \diamond

The expectation is “linked” to the average (or empirical mean, **mean**) if we have a set of realizations thought to be from the particular random variable. Similarly, the variance, the expectation of the squared deviation of a random variable from its expected value is “linked” to the empirical variance (**var**). This link will be more formalized in later chapters.

Definition C.7. The variance of X is defined by:

$$\text{Var}(X) = \mathbb{E}((X - \mathbb{E}(X))^2). \quad (\text{C.10})$$

The standard deviation of X is defined by

$$\text{SD}(X) = \sqrt{\text{Var}(X)}. \quad (\text{C.11})$$

\diamond

Property C.4. For an “arbitrary” real function g we have:

$$\begin{aligned} 1. \quad \mathbb{E}(g(X)) &= \sum_i g(x_i) P(X = x_i), \quad X \text{ discrete}, \\ &\mathbb{E}(g(X)) = \int_{\mathbb{R}} g(x) f_X(x) dx, \quad X \text{ continuous}. \end{aligned}$$

Regardless of whether X is discrete or continuous, the following rules apply:

2. $\text{Var}(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2,$
3. $\mathbb{E}(a + bX) = a + b\mathbb{E}(X),$ for a and b given constants,
4. $\text{Var}(a + bX) = b^2 \text{Var}(X),$ for a and b given constants,
5. $\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y),$ for a random variable $Y.$

The second but last property seems somewhat surprising. But starting from the definition of the variance, one quickly realizes that the variance is not a linear operator:

$$\text{Var}(a + bX) = \mathbb{E}((a + bX - \mathbb{E}(a + bX))^2) = \mathbb{E}((a + bX - (a + b\mathbb{E}(X)))^2). \quad (\text{C.12})$$

Example C.4. We consider again the setting of Example C.1, and straightforward calculation shows that

$$\mathbb{E}(X) = \sum_{i=2}^{12} i P(X = i) = 7, \quad \text{by equation (C.8)}, \quad (\text{C.13})$$

$$= 2 \sum_{i=1}^6 i \frac{1}{6} = 2 \cdot \frac{7}{2}, \quad \text{by using Property C.4.5 first.} \quad (\text{C.14})$$



C.5 Independent Random Variables

Often we not only have one random variable but many of them. Here, we see a first way to characterize a set of random variables.

Definition C.8. Two random variables X and Y are independent if

$$P(X \in A \cap Y \in B) = P(X \in A) P(Y \in B). \quad (\text{C.15})$$

for $A, B \subset \mathbb{R}.$ The random variables X_1, \dots, X_n are independent if

$$P(\bigcap_{i=1}^n X_i \in A_i) = \prod_{i=1}^n P(X_i \in A_i). \quad (\text{C.16})$$

for $A_1, \dots, A_n \subset \mathbb{R}.$



The definition of independent random variables also implies that the joint density and joint cumulative distribution is simply the product of the individual ones, also called marginal ones.

We will often use many independent random variables with a common distribution function.

Definition C.9. A random sample X_1, \dots, X_n consists of n independent random variables with the same distribution F . We write $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} F$ where iid stands for “independent and identically distributed.” The number n of random variables is called the sample size or sample range. \diamond

The iid assumption is very crucial and relaxing it has severe implications on the statistical modeling. Luckily, independence also implies a simple formula for the variance of the sum of two or many random variables.

Property C.5. *Let X and Y be two independent random variables. Then*

$$1. \text{Var}(aX + bY) = a^2 \text{Var}(X) + b^2 \text{Var}(Y).$$

Let $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} F$ with $E(X_1) = \mu$ and $\text{Var}(X_1) = \sigma^2$. Denote $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$. Then

$$2. E(\bar{X}) = E\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = E(X_1) = \mu.$$

$$3. \text{Var}(\bar{X}) = \text{Var}\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = \frac{1}{n} \text{Var}(X_1) = \frac{\sigma^2}{n}.$$

The latter two properties will be used when we investigate statistical properties of the sample mean, i.e., linking the empirical mean $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ with the random sample mean $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$.

C.6 Common Discrete Distributions

There is of course no limitation on the number of different random variables. In practice, we can reduce our framework to some common distributions.

C.6.1 Binomial Distribution

A random experiment with exactly two possible outcomes (for example: success/failure, heads/tails, male/female) is called a Bernoulli trial. For simplicity, we code the sample space with ‘1’ (success) and ‘0’ (failure).

$$P(X = 1) = p, \quad P(X = 0) = 1 - p, \quad 0 < p < 1. \quad (\text{C.17})$$

Thus

$$E(X) = p, \quad \text{Var}(X) = p(1 - p). \quad (\text{C.18})$$

Note that we exclude the cases $p = 0$ and $p = 1$, which are typically not relevant anyway.

If a Bernoulli experiment is repeated n times (resulting in n -tuples of zeros and ones), the random variable X = “number of successes” is intuitive. The distribution of X is called the binomial distribution, denoted with $X \sim \text{Bin}(n, p)$ and the following applies:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad 0 < p < 1, \quad k = 0, 1, \dots, n, \quad (\text{C.19})$$

and

$$\text{E}(X) = np, \quad \text{Var}(X) = np(1-p). \quad (\text{C.20})$$

C.6.2 Poisson Distribution

The Poisson distribution gives the probability of a given number of events occurring in a fixed interval of time if these events occur with a known and constant rate over time. One way to formally introduce such a random variable is giving its probability mass.

Definition C.10. A random variable X , whose probability function is given by

$$P(X = k) = \frac{\lambda^k}{k!} \exp(-\lambda), \quad 0 < \lambda, \quad k = 0, 1, \dots, \quad (\text{C.21})$$

is said to follow a Poisson distribution, denoted by $X \sim \text{Pois}(\lambda)$. Further,

$$\text{E}(X) = \lambda, \quad \text{Var}(X) = \lambda. \quad (\text{C.22})$$

◊

The Poisson distribution is also a good approximation for the binomial distribution with large n and small p .

C.7 Common Continuous Distributions

Example C.3 introduced a first common continuous distribution. In this section we see first the most commonly used one, i.e., the normal distribution, and then some distributions that are derived as functions from normally distributed random variables. We will encounter these later, for example, the t distribution when discussing tests about means and the F distribution when discussing tests about variances.

C.7.1 The Normal Distribution

The normal or Gaussian distribution is probably the most known distribution, having the omnipresent “bell-shaped” density. Its importance is mainly due the fact that the sum of many random variables (under iid or more general settings) is distributed as a normal random variable. This is due to the celebrated central limit theorem. As in the case of a Poisson random variable, we define the normal distribution by giving its density.

Definition C.11. The random variable X is said to be normally distributed if

$$F_X(x) = \int_{-\infty}^x f_X(y)dy \quad (\text{C.23})$$

with density function

$$f(x) = f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \cdot \frac{(x-\mu)^2}{\sigma^2}\right), \quad (\text{C.24})$$

for all x ($\mu \in \mathbb{R}$, $\sigma^2 > 0$). We denote this with $X \sim \mathcal{N}(\mu, \sigma^2)$.

The random variable $Z = (X - \mu)/\sigma$ (the so-called z -transformation) is standard normal and its density and distribution function are usually denoted with $\varphi(z)$ and $\Phi(z)$, respectively. ◇

While the exact form of the density (C.24) is not important, a certain recognizing factor will be very useful. Especially, for a standard normal random variable, the density is proportional to $\exp(-z^2/2)$.

The following property is essential and will be consistently used throughout the work. We justify the first one later in this chapter. The second one is a result of the particular form of the density.

- Property C.6.**
1. Let $X \sim \mathcal{N}(\mu, \sigma^2)$, then $\frac{X - \mu}{\sigma} \sim \mathcal{N}(0, 1)$. Conversely, if $Z \sim \mathcal{N}(0, 1)$, then $\sigma Z + \mu \sim \mathcal{N}(\mu, \sigma^2)$.
 2. Let $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ be independent, then $aX_1 + bX_2 \sim \mathcal{N}(a\mu_1 + b\mu_2, a^2\sigma_1^2 + b^2\sigma_2^2)$.

The cumulative distribution function Φ has no closed form and the corresponding probabilities must be determined numerically. In the past, so-called “standard tables” were often used and included in statistics books. Table C.1 gives an excerpt of such a table. Now even “simple” pocket calculators have the corresponding functions. It is probably worthwhile to remember $84\% = \Phi(1)$, $98\% = \Phi(2)$, $100\% \approx \Phi(3)$, as well as $95\% = \Phi(1.64)$ and $97.5\% = \Phi(1.96)$.

Example C.5. Let $X \sim \mathcal{N}(4, 9)$. Then

1. $P(X \leq -2) = P\left(\frac{X - 4}{3} \leq \frac{-2 - 4}{3}\right)$
 $= P(Z \leq -2) = \Phi(-2) = 1 - \Phi(2) = 1 - 0.977 = 0.023.$
 2. $P(|X - 3| > 2) = 1 - P(|X - 3| \leq 2) = 1 - P(-2 \leq X - 3 \leq 2)$
 $= 1 - (P(X - 3 \leq 2) - P(X - 3 \leq -2)) = 1 - \Phi\left(\frac{5 - 4}{3}\right) + \Phi\left(\frac{1 - 4}{3}\right) \approx 0.5281.$
- ♣

The following theorem is of paramount importance and many extensions thereof exist.

Table C.1: Probabilities of the standard normal distribution. The table gives the value of $\Phi(z_p)$. For example, $\Phi(0.2 + 0.04) = 0.595$.

z_p	0.00	0.02	0.04	0.06	0.08
0.0	0.500	0.508	0.516	0.524	0.532
0.1	0.540	0.548	0.556	0.564	0.571
0.2	0.579	0.587	0.595	0.603	0.610
0.3	0.618	0.626	0.633	0.641	0.648
0.4	0.655	0.663	0.670	0.677	0.684
0.5	0.691	0.698	0.705	0.712	0.719
:					
1.0	0.841	0.846	0.851	0.855	0.860
:					
1.6	0.945	0.947	0.949	0.952	0.954
1.7	0.955	0.957	0.959	0.961	0.962
1.8	0.964	0.966	0.967	0.969	0.970
1.9	0.971	0.973	0.974	0.975	0.976
2.0	0.977	0.978	0.979	0.980	0.981
:					
3.0	0.999	0.999	...		

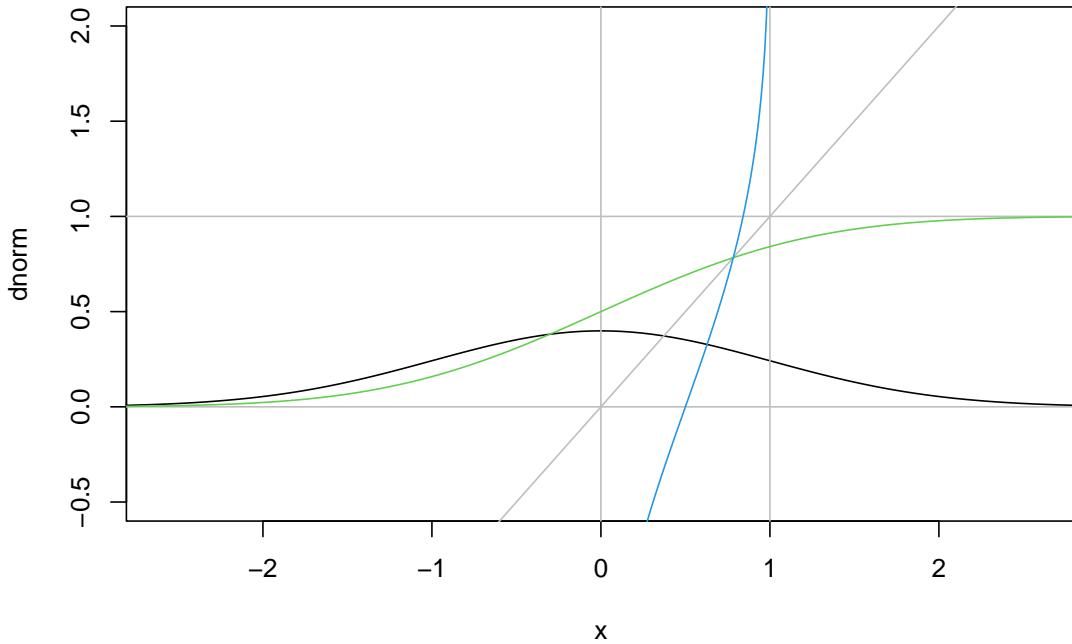


Figure C.4: Density (black), distribution function (green), and quantile function (blue) of the standard normal distribution.

Property C.7. (*Central Limit Theorem, classical version*) Let X_1, X_2, X_3, \dots an infinite sequence of iid random variables with $E(X_i) = \mu$ and $\text{Var}(X_i) = \sigma^2$. Then

$$\lim_{n \rightarrow \infty} P\left(\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \leq z\right) = \Phi(z) \quad (\text{C.25})$$

where we kept the subscript n for the sample mean to emphasize its dependence on n .

Using the central limit theorem argument, we can show that the distribution of a binomial random variable $X \sim \text{Bin}(n, p)$ converges to a distribution of a normal random variable as $n \rightarrow \infty$. Thus, the distribution of a normal random variable $\mathcal{N}(np, np(1 - p))$ can be used as an approximation for the binomial distribution $\text{Bin}(n, p)$. For the approximation, n should be larger than 30 for $p \approx 0.5$. For p closer to 0 and 1, n needs to be much larger.

To calculate probabilities, we often apply a so-called continuity correction, as illustrated in the following example.

Example C.6. Let $X \sim \text{Bin}(30, 0.5)$. Then $P(X \leq 10) = 0.049$, “exactly”. However,

$$P(X \leq 10) \approx P\left(\frac{X - np}{\sqrt{np(1 - p)}} \leq \frac{10 - np}{\sqrt{np(1 - p)}}\right) = \Phi\left(\frac{10 - 15}{\sqrt{30/4}}\right) = 0.034, \quad (\text{C.26})$$

$$P(X \leq 10) \approx P\left(\frac{X + 0.5 - np}{\sqrt{np(1 - p)}} \leq \frac{10 + 0.5 - np}{\sqrt{np(1 - p)}}\right) = \Phi\left(\frac{10.5 - 15}{\sqrt{30/4}}\right) = 0.05. \quad (\text{C.27})$$



C.7.2 Chi-Square Distribution

Let $Z_1, \dots, Z_n \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1)$. The distribution of the random variable

$$\mathcal{X}_n^2 = \sum_{i=1}^n Z_i^2 \quad (\text{C.28})$$

is called the chi-square distribution (χ^2 distribution) with n degrees of freedom. The following applies:

$$E(\mathcal{X}_n^2) = n; \quad \text{Var}(\mathcal{X}_n^2) = 2n. \quad (\text{C.29})$$

Here and for the next two distributions, we do not give the densities as they are very complex. Similarly, the expectation and the variance here and for the next two distributions are for reference only.

The chi-square distribution is used in numerous statistical tests.

The definition of a chi-square random variable is based on a sum of independent random variables. Hence in view of the central limit theorem, the following result are not surprising. If $n > 50$, we can approximate the chi-square distribution with a normal distribution, i.e., \mathcal{X}_n^2 is distributed approximately $\mathcal{N}(n, 2n)$. Furthermore, for \mathcal{X}_n^2 with $n > 30$ the random variable $X = \sqrt{2\mathcal{X}_n^2}$ is approximately normally distributed with expectation $\sqrt{2n-1}$ and standard deviation of 1.

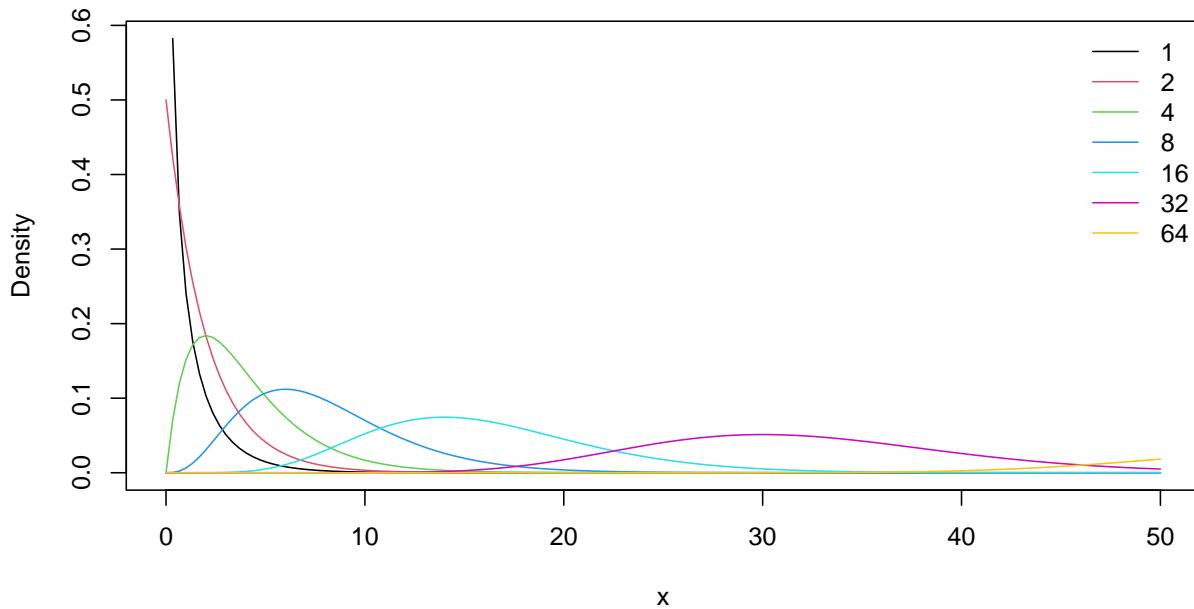


Figure C.5: Densities of the Chi-square distribution for various degrees of freedom.

C.7.3 Student's t -Distribution

In later chapters, we use the so-called t -test when, for example, comparing empirical means. The test will be based on the distribution that we define next.

Let $Z \sim \mathcal{N}(0, 1)$ and $X \sim \mathcal{X}_m^2$ be two independent random variables. The distribution

$$T_m = \frac{Z}{\sqrt{X/m}} \quad (\text{C.30})$$

is called the t -distribution (or Student's t -distribution) with m degrees of freedom. We have:

$$\mathbb{E}(T_m) = 0, \quad \text{for } m > 1; \quad (\text{C.31})$$

$$\text{Var}(T_m) = \frac{m}{(m-2)}, \quad \text{for } m > 2. \quad (\text{C.32})$$

The density is symmetric around zero and as $m \rightarrow \infty$ the density converges to the standard normal density $\varphi(x)$ (see Figure C.6).

Remark C.1. For $m = 1, 2$ the density is *heavy-tailed* and the variance of the distribution does not exist. Realizations of this random variable occasionally manifest with extremely large values.



C.7.4 F -Distribution

The F -distribution is mainly used to compare two empirical variances with each other.

Let $X \sim \mathcal{X}_m^2$ and $Y \sim \mathcal{X}_n^2$ be two independent random variables. The distribution

$$F_{m,n} = \frac{X/m}{Y/n} \quad (\text{C.33})$$

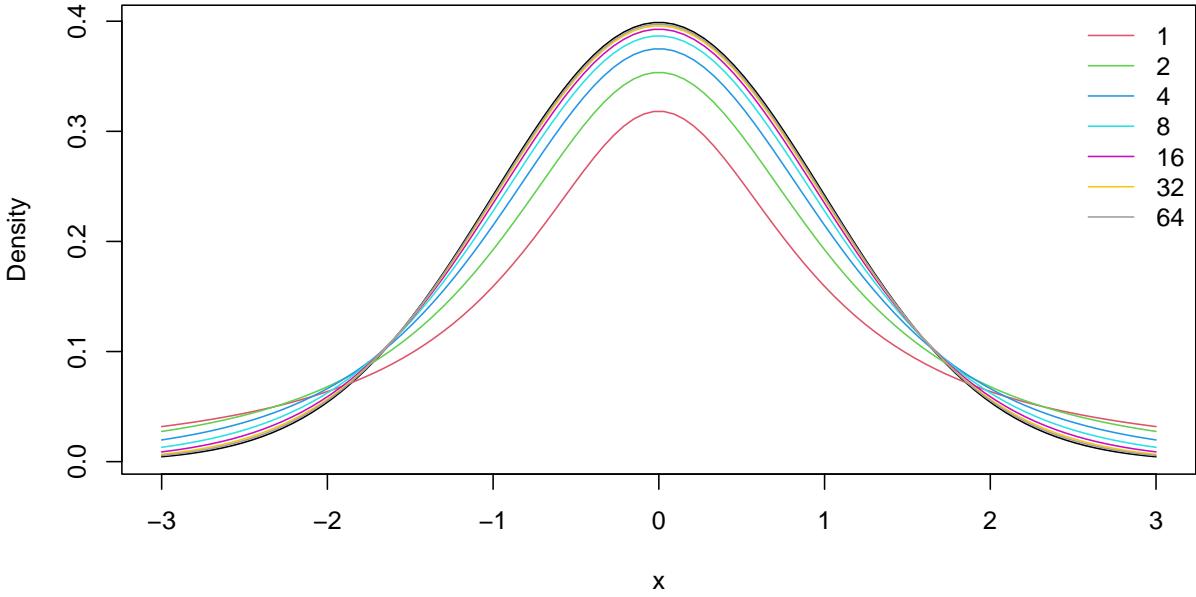


Figure C.6: Densities of the t -distribution for various degrees of freedom. The normal distribution is in black. A density with $2^7 = 128$ degrees of freedom would make the normal density function appear thicker.

is called the F -distribution with m and n degrees of freedom. It holds that:

$$\mathbb{E}(F_{m,n}) = \frac{n}{n-2}, \quad \text{for } n > 2; \quad (\text{C.34})$$

$$\text{Var}(F_{m,n}) = \frac{2n^2(m+n-2)}{m(n-2)^2(n-4)}, \quad \text{for } n > 4. \quad (\text{C.35})$$

Figure C.7 shows the density for various degrees of freedom.

C.7.5 Beta Distribution

A random variable X with density

$$f_X(x) = c \cdot x^{\alpha-1} (1-x)^{\beta-1}, \quad x \in [0, 1], \alpha > 0, \beta > 0, \quad (\text{C.36})$$

where c is a normalization constant, is called the beta distributed with parameters α and β . We write this as $X \sim \text{Beta}(\alpha, \beta)$. The normalization constant cannot be written in closed form for all parameters α and β . For $\alpha = \beta$ the density is symmetric around 1/2 and for $\alpha > 1, \beta > 1$ the density is concave with mode $(\alpha-1)/(\alpha+\beta-2)$. For arbitrary $\alpha > 0, \beta > 0$ we have:

$$\mathbb{E}(X) = \frac{\alpha}{\alpha+\beta}; \quad (\text{C.37})$$

$$\text{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta+1)(\alpha+\beta)^2}. \quad (\text{C.38})$$

Figure C.8 shows densities of the beta distribution for various pairs of (α, β) .

The beta distribution is mainly used to model probabilities and thus often encountered in Bayesian modelling.

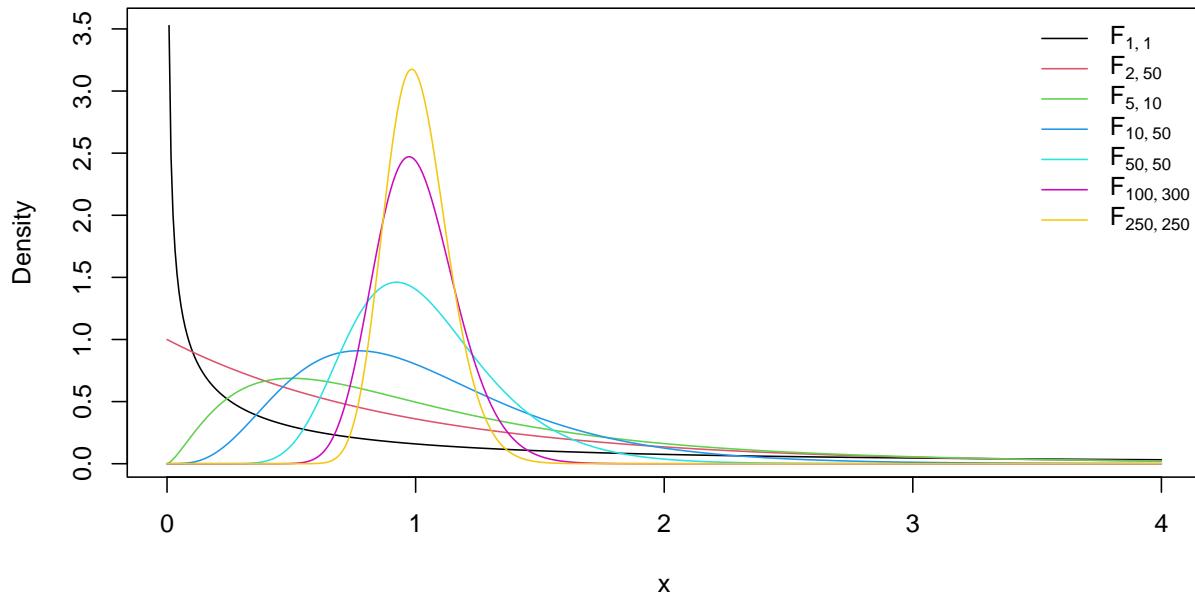


Figure C.7: Density of the F -distribution for various degrees of freedom.

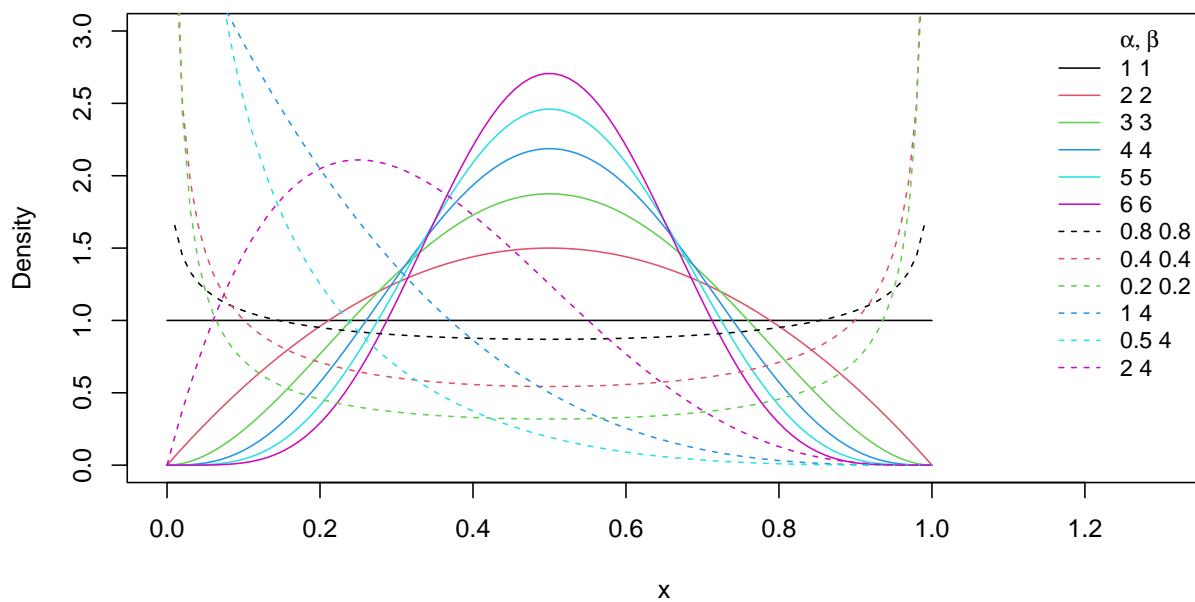


Figure C.8: Densities of beta distributed random variables for various pairs of (α, β) .

C.8 Functions of Random Variables

In the previous sections we saw different examples of often used, classical random variables. These examples are often not enough and through a modeling approach, we need additional ones. In this section we illustrate how to construct the cdf and pdf of a random variable that is the square of one from which we know the density.

Let X be a random variable with distribution function $F_X(x)$. We define a random variable

$Y = g(X)$. The cumulative distribution function of Y is written as

$$F_Y(y) = P(Y \leq y) = P(g(X) \leq y). \quad (\text{C.39})$$

In many cases $g(\cdot)$ is invertable (and differentiable) and we obtain

$$F_Y(y) = \begin{cases} P(X \leq g^{-1}(y)) = F_X(g^{-1}(y)), & \text{if } g^{-1} \text{ monotonically increasing,} \\ P(X \geq g^{-1}(y)) = 1 - F_X(g^{-1}(y)), & \text{if } g^{-1} \text{ monotonically decreasing.} \end{cases} \quad (\text{C.40})$$

To derive the probability mass function we apply Property C.2.3. In the more interesting setting of continuous random variables, the density function is derived by Property C.3.3 and is thus

$$f_Y(y) = \left| \frac{d}{dy} g^{-1}(y) \right| f_X(g^{-1}(y)). \quad (\text{C.41})$$

Example C.7. Let X be a random variable with cdf $F_X(x)$ and pdf $f_X(x)$. We consider $Y = a + bX$, for $b > 0$ and a arbitrary. Hence, $g(\cdot)$ is a linear function and its inverse $g^{-1}(y) = (y-a)/b$ is monotonically increasing. The cdf of Y is thus $F_X((y-a)/b)$ and the pdf is $f_X((y-a)/b) \cdot 1/b$. This fact has already been stated in Property C.6 for the Gaussian random variables. ♣

Example C.8. Let $X \sim \mathcal{U}(0, 1)$ and for $0 < x < 1$, we set $g(x) = -\log(1-x)$, thus $g^{-1}(y) = 1 - \exp(-y)$. Then the distribution and density function of $Y = g(X)$ is

$$F_Y(y) = F_X(g^{-1}(y)) = g^{-1}(y) = 1 - \exp(-y), \quad (\text{C.42})$$

$$f_Y(y) = \left| \frac{d}{dy} g^{-1}(y) \right| f_X(g^{-1}(y)) = \exp(-y), \quad (\text{C.43})$$

for $y > 0$. This random variable is called the exponential random variable (with rate parameter one). Notice further that $g(x)$ is the quantile function of this random variable. ♣

As we are often interested in summarizing a random variable by its mean and variance, we have a very convenient short-cut.

The expectation and the variance of a transformed random variable Y can be approximated by the so-called *delta method*. The idea thereof consists of a Taylor expansion around the expectation $E(X)$:

$$g(X) \approx g(E(X)) + g'(E(X)) \cdot (X - E(X)) \quad (\text{C.44})$$

(two terms of the Taylor series). Thus

$$E(Y) \approx g(E(X)), \quad (\text{C.45})$$

$$\text{Var}(Y) \approx g'(E(X))^2 \cdot \text{Var}(X). \quad (\text{C.46})$$

Example C.9. Let $X \sim \mathcal{B}(1, p)$ and $Y = X/(1 - X)$. Thus,

$$\mathbb{E}(Y) \approx p/(1 - p); \quad \text{Var}(Y) \approx \left(\frac{1}{(1-p)^2}\right)^2 \cdot p(1-p) = \frac{p}{(1-p)^3}. \quad (\text{C.47})$$



Example C.10. Let $X \sim \mathcal{B}(1, p)$ and $Y = \log(X)$. Thus

$$\mathbb{E}(Y) \approx \log(p), \quad \text{Var}(Y) \approx \left(\frac{1}{p}\right)^2 \cdot p(1-p) = \frac{1-p}{p}. \quad (\text{C.48})$$



Of course, in the case of a linear transformation (as, e.g., in Example C.7), equation (C.44) is an equality and thus relations (C.45) and (C.46) are exact, which is in sync with Property C.6.

Appendix D

Multivariate Normal Distribution

In Chapter C we have introduced univariate random variables. We now extend the framework to random vectors (i.e., multivariate random variables). We will mainly focus on continuous random vectors, especially Gaussian random vectors.

D.1 Random Vectors

A random vector is a (column) vector $\mathbf{X} = (X_1, \dots, X_p)^\top$ with p random variables as components. The following definition is the generalization of the univariate cumulative distribution function (cdf) to the multivariate setting (see Definition C.1).

Definition D.1. The multidimensional (or multivariate) distribution function of a random vector \mathbf{X} is defined as

$$F_{\mathbf{X}}(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x}) = P(X_1 \leq x_1, \dots, X_p \leq x_p), \quad (\text{D.1})$$

where the list in the right-hand-side is to be understood as the intersection (\cap). ◊

The multivariate distribution function generally contains more information than the set of marginal distribution functions, because (D.1) only simplifies to $F_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^p P(X_i \leq x_i)$ under independence of all random variables X_i (compare to Equation (C.3)).

A random vector \mathbf{X} is a continuous random vector if each random variable X_i is continuous. The probability density function for a continuous random vector is defined in a similar manner as for univariate random variables.

Definition D.2. The probability density function (or density function, pdf) $f_{\mathbf{X}}(\mathbf{x})$ of a p -dimensional continuous random vector \mathbf{X} is defined by

$$P(\mathbf{X} \in A) = \int_A f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}, \quad \text{for all } A \subset \mathbb{R}^p. \quad (\text{D.2})$$

◊

For convenience, we summarize here a few facts of random vectors with two continuous components, i.e., for a bivariate random vector $(X, Y)^\top$. The univariate counterparts are stated in Properties C.1 and C.3.

- The distribution function is monotonically increasing:
for $x_1 \leq x_2$ and $y_1 \leq y_2$, $F_{X,Y}(x_1, y_1) \leq F_{X,Y}(x_2, y_2)$.

- The distribution function is normalized:

$$\lim_{x,y \rightarrow \infty} F_{X,Y}(x, y) = F_{X,Y}(\infty, \infty) = 1.$$

We use the slight abuse of notation by writing ∞ in arguments without a limit.

- $F_{X,Y}(-\infty, -\infty) = F_{X,Y}(x, -\infty) = F_{X,Y}(-\infty, y) = 0$.
- $F_{X,Y}(x, y)$ ($f_{X,Y}(x, y)$) are continuous (almost) everywhere.
- $f_{X,Y}(x, y) = \frac{\partial^2}{\partial x \partial y} F_{X,Y}(x, y)$.

$$\begin{aligned} \bullet \quad P(a < X \leq b, c < Y \leq d) &= \int_a^b \int_c^d f_{X,Y}(x, y) dy dx \\ &= F_{X,Y}(b, d) - F_{X,Y}(b, c) - F_{X,Y}(a, d) + F_{X,Y}(a, c). \end{aligned}$$

In the multivariate setting there is also the concept termed marginalization, i.e., reduce a higher-dimensional random vector to a lower dimensional one. Intuitively, we “neglect” components of the random vector in allowing them to take any value. In two dimensions, we have

- $F_X(x) = P(X \leq x, Y \text{ arbitrary}) = F_{X,Y}(x, \infty)$;
- $f_X(x) = \int_{\mathbb{R}} f_{X,Y}(x, y) dy$.

Definition D.3. The expected value of a random vector \mathbf{X} is defined as

$$E(\mathbf{X}) = E \left(\begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \right) = \begin{pmatrix} E(X_1) \\ \vdots \\ E(X_p) \end{pmatrix}. \quad (\text{D.3})$$

◇

Hence the expectation of a random vector is simply the vector of the individual expectations. Of course, to calculate these, we only need the marginal univariate densities $f_{X_i}(x)$ and thus the expectation does not change whether (D.1) can be factored or not. The variance of a random vector requires a bit more thought and we first need the following.

Definition D.4. The covariance between two arbitrary random variables X_1 and X_2 is defined as

$$\text{Cov}(X_1, X_2) = E((X_1 - E(X_1))(X_2 - E(X_2))). \quad (\text{D.4})$$

◇

Using the linearity properties of the expectation operator, it is possible to show the following handy properties.

Property D.1. *We have for arbitrary random variables X_1 , X_2 and X_3 :*

1. $\text{Cov}(X_1, X_2) = \text{Cov}(X_2, X_1)$,
2. $\text{Cov}(X_1, X_1) = \text{Var}(X_1)$,
3. $\text{Cov}(a + bX_1, c + dX_2) = bd \text{Cov}(X_1, X_2)$, for arbitrary values a , b , c and d ,
4. $\text{Cov}(X_1, X_2 + X_3) = \text{Cov}(X_1, X_2) + \text{Cov}(X_1, X_3)$.

The covariance describes the *linear* relationship between the random variables. The correlation between two random variables X_1 and X_2 is defined as

$$\text{Corr}(X_1, X_2) = \frac{\text{Cov}(X_1, X_2)}{\sqrt{\text{Var}(X_1) \text{Var}(X_2)}} \quad (\text{D.5})$$

and corresponds to the *normalized* covariance. It holds that $-1 \leq \text{Corr}(X_1, X_2) \leq 1$, with equality only in the *degenerate* case $X_2 = a + bX_1$ for some a and $b \neq 0$.

Definition D.5. The variance of a p -variate random vector $\mathbf{X} = (X_1, \dots, X_p)^\top$ is defined as

$$\text{Var}(\mathbf{X}) = E((\mathbf{X} - E(\mathbf{X}))(\mathbf{X} - E(\mathbf{X}))^\top) \quad (\text{D.6})$$

$$= \text{Var} \left(\begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \right) = \begin{pmatrix} \text{Var}(X_1) & \dots & \text{Cov}(X_i, X_j) \\ & \ddots & \\ \text{Cov}(X_j, X_i) & \dots & \text{Var}(X_p) \end{pmatrix}, \quad (\text{D.7})$$

called the covariance matrix or variance–covariance matrix. \diamond

The covariance matrix is a symmetric matrix and – except for degenerate cases – a positive definite matrix. We will not consider degenerate cases and thus we can assume that the inverse of the matrix $\text{Var}(\mathbf{X})$ exists.

Similar to Properties C.4, we have the following properties for random vectors.

Property D.2. *For an arbitrary p -variate random vector \mathbf{X} , vector $\mathbf{a} \in \mathbb{R}^q$ and matrix $\mathbf{B} \in \mathbb{R}^{q \times p}$ it holds:*

1. $\text{Var}(\mathbf{X}) = E(\mathbf{X}\mathbf{X}^\top) - E(\mathbf{X})E(\mathbf{X})^\top$,
2. $E(\mathbf{a} + \mathbf{B}\mathbf{X}) = \mathbf{a} + \mathbf{B}E(\mathbf{X})$,
3. $\text{Var}(\mathbf{a} + \mathbf{B}\mathbf{X}) = \mathbf{B}\text{Var}(\mathbf{X})\mathbf{B}^\top$.

D.2 Multivariate Normal Distribution

We now consider a special multivariate distribution: the multivariate normal distribution, by first considering the bivariate case.

D.2.1 Bivariate Normal Distribution

Definition D.6. The random variable pair (X, Y) has a bivariate normal distribution if

$$F_{X,Y}(x, y) = \int_{-\infty}^x \int_{-\infty}^y f_{X,Y}(u, v) dv du \quad (\text{D.8})$$

with density

$$\begin{aligned} f(x, y) &= f_{X,Y}(x, y) \\ &= \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\right)\right), \end{aligned} \quad (\text{D.9})$$

for all x and y and where $\mu_x \in \mathbb{R}$, $\mu_y \in \mathbb{R}$, $\sigma_x > 0$, $\sigma_y > 0$ and $-1 < \rho < 1$. \diamond

The role of some of the parameters μ_x , μ_y , σ_x , σ_y and ρ might be guessed. We will discuss their precise meaning after the following example.

Example D.1. Figure D.1 show the density of a bivariate normal distribution with $\mu_x = \mu_y = 0$, $\sigma_x = 1$, $\sigma_y = \sqrt{5}$, and $\rho = 2/\sqrt{5} \approx 0.9$. Because of the quadratic form in (D.9), the contour lines (isolines) are ellipses whose axes are given by the eigenvectors of the covariance matrix. \clubsuit

The bivariate normal distribution has many nice properties.

Property D.3. For the bivariate normal distribution we have: The marginal distributions are $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$ and

$$\mathbb{E}\left(\begin{pmatrix} X \\ Y \end{pmatrix}\right) = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \quad \text{Var}\left(\begin{pmatrix} X \\ Y \end{pmatrix}\right) = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}. \quad (\text{D.10})$$

Thus,

$$\text{Cov}(X, Y) = \rho\sigma_x\sigma_y, \quad \text{Corr}(X, Y) = \rho. \quad (\text{D.11})$$

If $\rho = 0$, X and Y are independent and vice versa.

Note, however, that the equivalence of independence and uncorrelatedness is specific to jointly normal variables and cannot be assumed for random variables that are not jointly normal.

Example D.2. Figure D.2 shows realizations from a bivariate normal distribution for various values of correlation ρ . Even for large sample shown here ($n = 500$), correlations between -0.25 and 0.25 are barely perceptible. \clubsuit

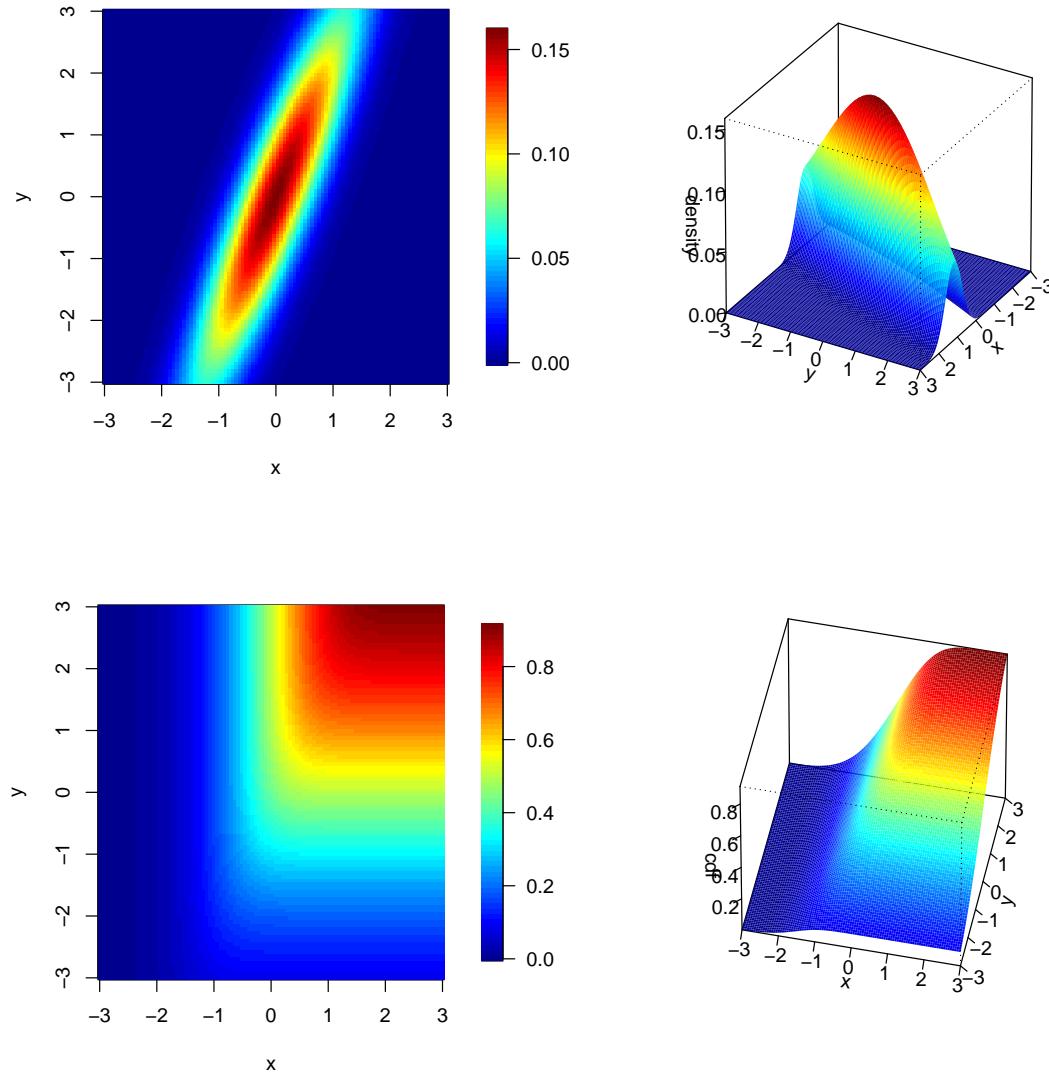


Figure D.1: Density of a bivariate normal distribution.

D.2.2 Multivariate Normal Distribution

In the general case we have to use vector notation. Surprisingly, we gain clarity even compared to the bivariate case.

Definition D.7. The random vector $\mathbf{X} = (X_1, \dots, X_p)^\top$ is multivariate normally distributed if

$$F_{\mathbf{X}}(\mathbf{x}) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_p} f_{\mathbf{X}}(u_1, \dots, u_p) du_1 \dots du_p \quad (\text{D.12})$$

with density

$$f_{\mathbf{X}}(x_1, \dots, x_p) = f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \quad (\text{D.13})$$

for all $\mathbf{x} \in \mathbb{R}^p$ ($\boldsymbol{\mu} \in \mathbb{R}^p$ and symmetric, positive-definite $\boldsymbol{\Sigma}$). We denote this distribution with $\mathbf{X} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. \diamond

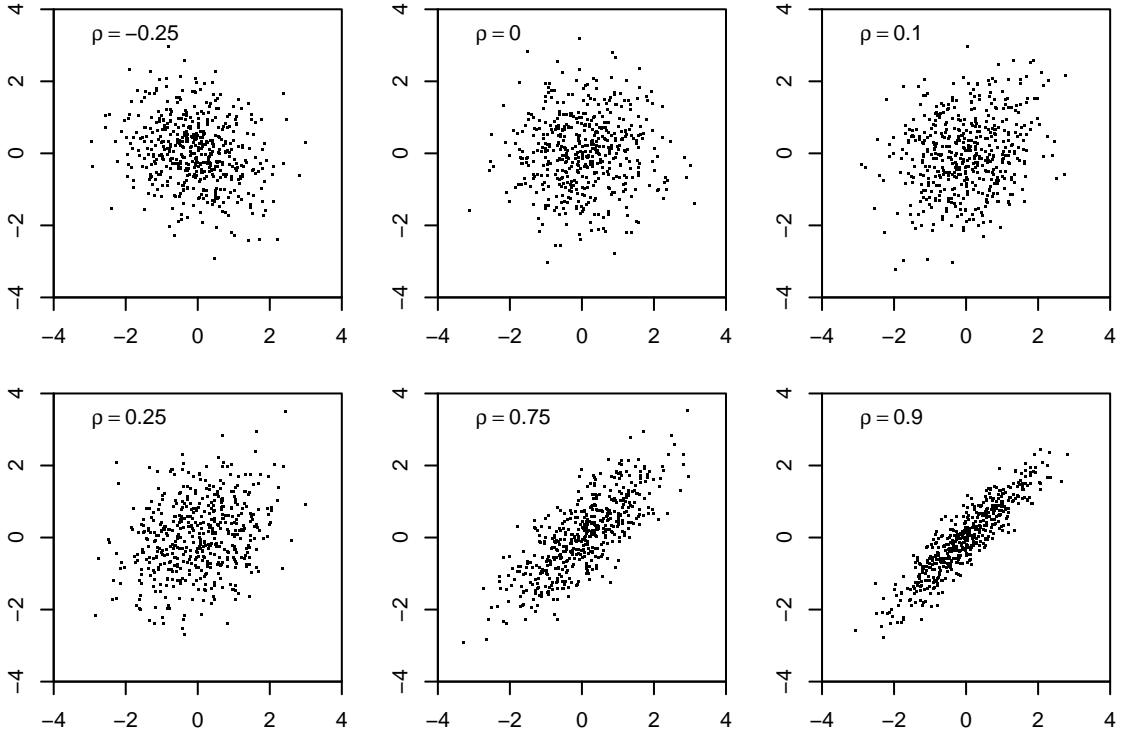


Figure D.2: Realizations from a bivariate normal distribution.

Property D.4. *For the multivariate normal distribution we have:*

$$\mathbb{E}(\mathbf{X}) = \boldsymbol{\mu}, \quad \text{Var}(\mathbf{X}) = \boldsymbol{\Sigma}. \quad (\text{D.14})$$

Property D.5. *Let $\mathbf{a} \in \mathbb{R}^q$, $\mathbf{B} \in \mathbb{R}^{q \times p}$, $q \leq p$, $\text{rank}(\mathbf{B}) = q$ and $\mathbf{X} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then*

$$\mathbf{a} + \mathbf{B}\mathbf{X} \sim \mathcal{N}_q(\mathbf{a} + \mathbf{B}\boldsymbol{\mu}, \mathbf{B}\boldsymbol{\Sigma}\mathbf{B}^\top). \quad (\text{D.15})$$

This last property has profound consequences. It also asserts that the one-dimensional marginal distributions are again Gaussian with $X_i \sim \mathcal{N}((\boldsymbol{\mu})_i, (\boldsymbol{\Sigma})_{ii})$, $i = 1, \dots, p$. Similarly, any subset of random variables of \mathbf{X} is again Gaussian with appropriate subset selection of the mean and covariance matrix.

We now discuss how to draw realizations from an arbitrary Gaussian random vector, much in the spirit of Property C.6.2. Let $\mathbf{I} \in \mathbb{R}^{p \times p}$ be the identity matrix, a square matrix which has only ones on the main diagonal and only zeros elsewhere, and let $\mathbf{L} \in \mathbb{R}^{p \times p}$ so that $\mathbf{L}\mathbf{L}^\top = \boldsymbol{\Sigma}$. That means, \mathbf{L} is like a “matrix square root” of $\boldsymbol{\Sigma}$.

To draw a realization \mathbf{x} from a p -variate random vector $\mathbf{X} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, one starts with drawing p values from $Z_1, \dots, Z_p \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1)$, and sets $\mathbf{z} = (z_1, \dots, z_p)^\top$. The vector is then (linearly) transformed with $\boldsymbol{\mu} + \mathbf{L}\mathbf{z}$. Since $\mathbf{Z} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$ Property D.5 asserts that $\mathbf{X} = \boldsymbol{\mu} + \mathbf{L}\mathbf{Z} \sim \mathcal{N}_p(\boldsymbol{\mu}, \mathbf{L}\mathbf{L}^\top)$.

In practice, the Cholesky decomposition of $\boldsymbol{\Sigma}$ is often used. This decomposes a symmetric positive-definite matrix into the product of a lower triangular matrix \mathbf{L} and its transpose. It holds that $\det(\boldsymbol{\Sigma}) = \det(\mathbf{L})^2 = \prod_{i=1}^p (\mathbf{L})_{ii}^2$.

D.2.3 Conditional Distributions

We now consider properties of parts of the random vector \mathbf{X} . For simplicity we write

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix}, \quad \mathbf{X}_1 \in \mathbb{R}^q, \quad \mathbf{X}_2 \in \mathbb{R}^{p-q}. \quad (\text{D.16})$$

We divide the matrix Σ in 2×2 blocks accordingly:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix} \sim \mathcal{N}_p \left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \right) \quad (\text{D.17})$$

Both (multivariate) marginal distributions \mathbf{X}_1 and \mathbf{X}_2 are again normally distributed with $\mathbf{X}_1 \sim \mathcal{N}_q(\boldsymbol{\mu}_1, \Sigma_{11})$ and $\mathbf{X}_2 \sim \mathcal{N}_{p-q}(\boldsymbol{\mu}_2, \Sigma_{22})$ (this can be seen again by Property D.5).

\mathbf{X}_1 and \mathbf{X}_2 are independent if $\Sigma_{21} = \mathbf{0}$ and vice versa.

Property D.6. *If one conditions a multivariate normally distributed random vector (D.17) on a subvector, the result is itself multivariate normally distributed with*

$$\mathbf{X}_2 | \mathbf{X}_1 = \mathbf{x}_1 \sim \mathcal{N}_{p-q} \left(\boldsymbol{\mu}_2 + \Sigma_{21} \Sigma_{11}^{-1} (\mathbf{x}_1 - \boldsymbol{\mu}_1), \Sigma_{22} - \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12} \right). \quad (\text{D.18})$$

The expected value depends linearly on the value of \mathbf{x}_1 , but the variance is independent of the value of \mathbf{x}_1 . The conditional expected value represents an update of \mathbf{X}_2 through $\mathbf{X}_1 = \mathbf{x}_1$: the difference $\mathbf{x}_1 - \boldsymbol{\mu}_1$ is normalized by the variance and scaled by the covariance. Notice that for $p = 2$, $\Sigma_{21} \Sigma_{11}^{-1} = \rho \sigma_y / \sigma_x$.

Equation (D.18) is probably one of the most important formulas you encounter in statistics albeit not always explicit.

Bibliography

- Abraham, B. and Ledolter, J. (2006). *Introduction To Regression Modeling*. Duxbury applied series. Thomson Brooks/Cole. [20](#), [161](#)
- Abramowitz, M. and Stegun, I. A., editors (1970). *Handbook of Mathematical Functions*. Dover. [213](#)
- Barnston, A. G. and Livezey, R. E. (1987). Classification, seasonality and persistence of low-frequency atmospheric circulation patterns. *Month. Weath. Rev.*, **115**, 1083–1126. [47](#)
- Bergmeir, C. and Benítez, J. M. (2012). Neural networks in R using the stuttgart neural network simulator: RSNNS. *Journal of Statistical Software*, **46**, 1–26. [251](#)
- Box, G. E. P. and Draper, N. R. (1987). *Empirical Model-building and Response Surfaces*. Wiley. [2](#)
- Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (2008). *Time Series Analysis, Forecasting, and Control*. Wiley, fourth edition. [200](#)
- Breiman, L. (2001). Random Forest. *Machine Learning*, **45**, 5–32. [96](#)
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth. [96](#)
- Brockwell, P. J. and Davis, R. A. (1991). *Time Series: Theory and Methods*. Springer-Verlag, second edition. [200](#)
- Brockwell, P. J. and Davis, R. A. (2010). *Introduction to time series and forecasting*. Springer texts in statistics. Springer, second edition. [200](#)
- Buehlmann, P. and Hothorn, T. (2007). Boosting algorithms: regularization, prediction and model fitting. *Statistical Science*, **22**, 477–505. [96](#)
- Butler, D. G., Cullis, B. R., Gilmour, A. R., and Gogel, B. J. (2009). *ASReml-R reference manual*. [130](#)
- Chilès, J.-P. and Delfiner, P. (1999). *Geostatistics: Modeling Spatial Uncertainty*. John Wiley & Sons Inc. [216](#)
- Choi, Y., Taylor, J., and Tibshirani, R. (2015). Selecting the number of principal components: estimation of the true rank of a noisy matrix. arXiv:1410.8260, to appear in *Annals of Statistics*. [47](#)

- Christensen, R. (2011). *Plane Answers to Complex Questions: The Theory of Linear Models*. Springer Texts in Statistics. Springer. [112](#)
- Coles, S. (2001). *An Introduction to Statistical Modeling of Extreme Values*. Springer series in statistics. Springer. [221](#), [229](#)
- Collett, D. (2014). *Modelling Survival Data in Medical Research*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, third edition. [187](#)
- Cox, D. R. and Oakes, D. (1984). *Analysis of Survival Data*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. [187](#)
- Cressie, N. A. C. (1990). The origins of kriging. *Mathematical Geology*, **22**, 239–252. [217](#)
- Cressie, N. A. C. (1993). *Statistics for Spatial Data*. Wiley, revised edition. [216](#)
- Dalal, S. R., Fowlkes, E. B., and Hoadley, B. (1989). Risk Analysis of the Space Shuttle: Pre-Challenger Prediction of Failure. *Journal of the American Statistical Association*, **84**, 945–957. [168](#)
- Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press. [20](#)
- Davison, A. C., Padoan, S. A., and Ribatet, M. (2012). Statistical Modeling of Spatial Extremes. *Statistical Science*, **27**, 161–186. [229](#)
- Diggle, P., Heagerty, P., Liang, K.-Y., and Zeger, S. (2002). *Analysis of Longitudinal Data*. Oxford University Press, second edition. [119](#)
- Dobson, A. J. (2001). *An Introduction to Generalized Linear Models, Second Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis. [168](#)
- Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *Annals of Statistics*, **7**, 1–26. [20](#)
- Embrechts, P., Klüppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance*. Springer. [229](#)
- Euromonitor (1979). *European Marketing Data and Statistics*. London: Euromonitor Publications. [27](#)
- Fahrmeir, L., Kneib, T., Lang, S., and Marx, B. (2013). *Regression: Models, Methods and Applications*. Springer. [112](#), [168](#)
- Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and Its Applications: Monographs on Statistics and Applied Probability 66*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. [153](#)
- Faraway, J. J. (2006). *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. CRC Press. [168](#)

- Fleuret, F. (2023). *The Little Book of Deep Learning*. ISBN 9781447678618. 250
- Freund, Y. and Shapire, R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, 23–37. 91
- Furrer, R. and Genton, M. G. (1999). Robust Spatial Data Analysis of Lake Geneva Sediments with S+SpatialStats. *Systems Research and Information Science*, **8**, 257–272. 204
- Gavish, M. and Donoho, D. L. (2014). The optimal hard threshold for singular values is $\sqrt{4/\sqrt{3}}$. *IEEE Transaction on Information Theory*, **60**, 5040–5053. 47
- Gilmour, A. R., Cullis, B. R., Welham, S. J., Gogel, B. J., and Thompson, R. (2004). An efficient computing strategy for prediction in mixed linear models. *Computational Statistics and Data Analysis*, **44**, 571–586. 130
- Gilmour, A. R., Thompson, R., and Cullis, B. R. (1995). Average information REML: An efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*, **51**, 1440–1450. 130
- Goldberger, A. S. (1962). Best linear unbiased prediction in the generalized linear regression model. *Journal of the American Statistical Association*, **57**, 369–375. 217
- Green, P. J. and Silverman, B. W. (1994). *Nonparametric Regression and Generalized Linear Models*. Chapman & Hall. 153, 168
- Haan, L. d. and Ferreira, A. (2006). *Extreme Value Theory : an Introduction*. Springer series in operations research and financial engineering. Springer. 229
- Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. 153, 168
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition. 20, 96
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**, 251–257. 240
- Hornik, K., Buchta, C., and Zeileis, A. (2009). Open-source machine learning: R meets Weka. *Computational Statistics*, **24**, 225–232. 250
- Hurrell, J. W., Kushnir, Y., Ottersen, G., and Visbeck, M. (2013). *An Overview of the North Atlantic Oscillation*, 1–35. American Geophysical Union. 47
- Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer-Verlag, second edition. 47
- Journel, A. G. and Huijbregts, C. J. (1978). *Mining Geostatistics*. Academic Press. 211, 216
- Kalbfleisch, J. D. and Prentice, R. L. (2002). *The Statistical Analysis of Failure Time Data*. John Wiley & Sons, Inc. 187

- Kirchner, N., Furrer, R., Jakobsson, M., Zwally, H. J., and Robbins, J. W. (2013). Statistical modeling of a former Arctic Ocean ice shelf complex using Antarctic analogies. *Journal of Geophysical Research: Earth Surface*, **118**, 1105–1117. [16](#)
- Klein, J. P. and Moeschberger, M. L. (2003). *Survival Analysis: Techniques for Censored and Truncated Data*. Springer, second edition. [187](#)
- Kleinbaum, D. and Klein, M. (2012). *Survival Analysis: A Self-Learning Text*. Springer, New York, 3rd edition. [187](#)
- Kolmogoroff, A. (1941). Interpolation und Extrapolation von stationären zufälligen Folgen. *Bull. Acad. Sci. URSS Sér. Math. [Izvestia Akad. Nauk. SSSR]*, **5**, 3–14. [217](#)
- Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, **52**, 119–139. [217](#)
- Makridakis, S. G., Wheelwright, S. C., and Hyndman, R. J. (1998). *Forecasting: Methods and Applications*. John Wiley & Sons: New York. [2](#), [135](#)
- Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press. [38](#), [47](#), [96](#), [112](#)
- Matheron, G. (1962). Trait? de G?ostatistique Appliqu?e, Tome I. *M?moires du Bureau de Recherches G?ologique et Mini?res*, **No. 14**, Editions Technip, Paris. [209](#), [210](#)
- Matheron, G. (1963). Trait? de G?ostatistique Appliqu?e, Tome II: Le Krigeage. *Mémoires du Bureau de Recherches Géologique et Minières*, **No. 24**, Editions Technip, Paris. [217](#)
- Matheron, G. (1969). Le Krigeage Universel. *Cahiers du Centre de Morphologie Math?matique*, **No. 1**, Fontainebleau, France. [217](#)
- McCullagh, P. and Nelder, J. A. (1983). *Generalized Linear Models*. Chapman & Hall. [168](#)
- Minka, T. P. (2000). Automatic choice of dimensionality for PCA. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *NIPS*, 598–604. MIT Press. [47](#)
- Moore, D. F. (2016). *Applied Survival Analysis Using R (Use R)*. Springer Verlag. [187](#)
- Nason, G. (2010). *Wavelet Methods in Statistics with R*. Use R! Springer. [153](#)
- North, G. R., Bell, T. L., Cahalan, R. F., and Moeng, F. J. (1982). Sampling errors in the estimation of empirical orthogonal functions. *Monthly Weather Review*, **110**, 699–706. [34](#)
- Olea, R. A. (1999). *Geostatistics for Engineers and Earth Scientists*. Kluwer Academic Publishers. [216](#)
- Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. (2023). The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. [244](#)

- Poole, J. H. (1989). Mate guarding, reproductive success and female choice in African elephants. *Animal Behavior*, **37**, 842–849. [163](#)
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 2nd edition. [96](#)
- Resnick, S. I. (2008). *Extreme Values, Regular Variation and Point Processes*. Springer series in operations research and financial engineering. Springer, 1987 edition. [229](#)
- Ripley, B. D. (1981). *Spatial Statistics*. John Wiley & Sons Inc. [216](#)
- Roche, J. R., Friggens, N. C., Kay, J., Fisher, M. W., Stafford, K. J., and Berry, D. P. (2009). Invited review: Body condition score and its association with dairy cow productivity, health, and welfare. *Journal of Dairy Science*, **92**, 92(12):5769–5801. [116](#)
- Rosenblatt, F. (1957). The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York. [235](#)
- Rubin, D. B. (1976). Inference and missing data. *Biometrika*, **63**, 581–592. [69](#)
- Sabates-Bellver, J., Van der Flier, L. G., de Palo, M., Cattaneo, E., Maake, C., Rehrauer, H., Laczkó, E., Kurowski, M. A., Bujnicki, J. M., Menigatti, M., Luz, J., Ranalli, T. V., Gomes, V., Pastorelli, A., Faggiani, R., Anti, M., Jiricny, J., Clevers, H., and Marra, G. (2007). Transcriptome profile of human colorectal adenomas. *Molecular Cancer Research*, **5**, 1263–1275. [69](#)
- Schabenberger, O. and Gotway, C. A. (2005). *Statistical Methods for Spatial Data Analysis*. Chapman & Hall/CRC. [216](#)
- Self, S. G. and Liang, K. (1987). Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association*, **82**, 605–610. [129](#)
- Sherman, M. (2011). *Spatial Statistics and Spatio-Temporal Data: Covariance Functions and Directional Properties*. Wiley. [216](#)
- Shumway, R. H. and Stoffer, D. S. (2010). *Time Series Analysis and Its Applications. With R Examples*. Springer, third edition. [200](#)
- Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric curve fitting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **47**, 1–52. [136](#)
- Stein, M. L. (1999). *Interpolation of Spatial Data*. Springer-Verlag. [216](#)
- Strobl, C., Malley, J., and Tutz, G. (2009). An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classification and Regression Trees, Bagging and Random Forests. *Psychological Methods*, **14**, 323–348. [96](#)

- Suhr, D. (2009). Principal component analysis vs. exploratory factor analysis. In *SUGI 30 Proceedings*, Paper 203-30, 1–11. <http://www2.sas.com/proceedings/sugi30/203-30.pdf>. 42
- Therneau, T. M. and Grambsch, P. M. (2010). *Modeling Survival Data: Extending the Cox Model*. Springer. 187
- Tobler, W. (1970). A computer movie simulating urban growth in the Detroit region. *Economic Geography*, **46**, 234–240. 204
- Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. 250
- von Storch, H. and Zwiers, F. W. (1999). *Statistical Analysis in Climate Research*. Cambridge University Press. 47
- Wackernagel, H. (1995). *Multivariate Geostatistics. An Introduction with Applications*. Springer. 216
- Wahba, G. (1990). *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM. 153
- Webster, R. and Oliver, M. (2001). *Geostatistics for Environmental Scientists*. John Wiley & Sons Inc. 216
- Wiener, N. (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series. With Engineering Applications*. The Technology Press of the Massachusetts Institute of Technology, Cambridge. 217
- Wilks, D. S. (2005). *Statistical Methods in the Atmospheric Sciences*. Academic Press, second edition. 47
- Wold, H. O. A. (1938). *A Study in the Analysis of Stationary Time Series*. Almqvist and Wiksell. 217
- Wood, S. (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis. 153, 168
- Zell, A., Mache, N., Hübner, R., Mamier, G., Vogt, M., Schmalzl, M., and Herrmann, K. (1994). Snns (stuttgart neural network simulator). In Skrzypek, J., editor, *Neural Network Simulation Environments*, volume 254, 165–186. The Kluwer International Series in Engineering and Computer Science, Springer, Boston, MA. 251
- Zhu, J., Zou, H., Rosset, S., and Hastie, T. (2009). Multi-class adaboost. *Statistics and Its Interface*, **2**, 349–360. 96

Glossary

Throughout the document we tried to be consistent with standard mathematical notation. We write random variables as uppercase letters (X, Y, \dots), realizations as lower case letters (x, y, \dots), matrices as bold uppercase letters ($\Sigma, \mathbf{X}, \dots$), and vectors as bold italics lowercase letters ($\mathbf{x}, \boldsymbol{\beta}, \dots$). (The only slight confusion arises with random vectors and matrices.)

The following glossary contains a non-exhaustive list of the most important notation. Standard operators or products are not repeatedly explained.

$:=$	Define the left hand side by the expression on the other side.
\clubsuit, \diamond	End of example, end of definition
\int, \sum, \prod	Integration, summation and product symbol. If there is no ambiguity, we omit the domain in inline formulas.
\cup, \cap	Union, intersection of sets or events.
$\hat{\theta}$	Estimator or estimate of the parameter θ .
\bar{x}	Arithmetic mean of the sample: $\sum_{i=1}^n x_i/n$.
$ x $	Absolute value of the scalar x .
$\ \mathbf{x}\ $	Norm of the vector \mathbf{x} , Euclidean distance.
\mathbf{X}^\top	Transpose of an matrix \mathbf{X} .
$x_{(i)}$	Order statistics of the sample $\{x_i\}$: $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$.
$\mathbf{0}, \mathbf{1}$	Vector or matrix with components 0 respectively 1.
$\text{Cov}(X, Y)$	Covariance between two random variables X and Y .
$\text{Corr}(X, Y)$	Correlation between two random variables X and Y .
$\frac{d}{dx}, ', \frac{\partial}{\partial x}$	Derivative and partial derivative with respect to x .
$\text{diag}(\mathbf{A})$	Diagonal entries of an $(n \times n)$ -matrix \mathbf{A} .
$\varepsilon, \varepsilon_i$	Random variable or process, usually measurement error.
$\text{E}(X)$	Expectation of the random variable X .
$e, \exp(\cdot)$	Transcendental number $e = 2.71828\ 18284$, the exponential function.
$\mathbf{I}_n = \mathbf{I}$	Identity matrix, $\mathbf{I} = (\delta_{ij})$.
$I_{\{A\}}$	Indicator function, taking the value one if A is true and zero otherwise.
\lim	Limit.
$\log(\cdot)$	Logarithmic function to the base e .

$\max\{A\}, \min\{A\}$	Maximum, minimum of the set A .
\mathbb{N}, \mathbb{N}^d	Space of natural numbers, of d -vectors with natural elements.
$\mathcal{N}(\mu, \sigma^2)$	Normal (Gaussian) distribution with mean μ and variance σ^2 .
$\mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal p dimensional distribution with mean vector $\boldsymbol{\mu}$ and variance matrix $\boldsymbol{\Sigma}$.
$\varphi(x)$	Gaussian probability density function $\varphi(x) = (2\pi)^{-1/2} \exp(-x^2/2)$.
$\Phi(x)$	Gaussian cumulative distribution function $\Phi(x) = \int_{-\infty}^x \varphi(z) dz$.
π	Transcendental number $\pi = 3.14159 26535$.
$P(A)$	Probability of the event A .
$\mathbb{R}, \mathbb{R}^n, \mathbb{R}^{n \times m}$	Space of real numbers, real n -vectors and real $(n \times m)$ -matrices.
$\text{rank}(\mathbf{A})$	The rank of a matrix \mathbf{A} is defined as the number of linearly independent rows (or columns) of \mathbf{A} .
$\text{tr}(\mathbf{A})$	Trace of an matrix \mathbf{A} defined by the sum of its diagonal elements.
$\text{Var}(X)$	Variance of the random variable X .
\mathbb{Z}, \mathbb{Z}^d	Space of integers, of d -vectors with integer elements.

The following table contains the abbreviations of the statistical distributions (dof denotes degrees of freedom).

$\mathcal{N}(\mu, \sigma^2)$	Gaussian or normal random variable with parameters μ and σ^2 .
$\mathcal{N}(0, 1), z_p$	Standard standard normal random variable, p -quantiles thereof.
$\text{Bin}(n, p), b_{n,p,1-\alpha}$	Binomial random variable with n trials and success probability p , $1 - \alpha$ -quantile thereof.
$\chi^2_\nu, \chi^2_{\nu,p}$	Chi-squared distribution with ν dof, p -quantile thereof.
$T_n, t_{n,p}$	Student's t -distribution with n dof, p -quantile thereof.
$F_{m,n}, f_{m,n,p}$	F -distribution with m and n dof, p -quantile thereof.

The following table contains the abbreviations of the statistical methods, properties and quality measures.

EDA	Exploratory data analysis.
dof	Degrees of freedom.
MAD	Median absolute deviation.
MAE	Mean absolute error.
ML	Maximum likelihood (ML estimator or ML estimation).
MM	Method of moments.
MSE	Mean squared error.
OLS, LS	Ordinary least squares.

RMSE	Root mean squared error.
WLS	Weighted least squares.

Dataset Index

Page numbers in upright, slanted and italics refer to main explanation in text, use in R chunks and in figures, respectively.

bicoal, 3, 136, 139

elephant, 163, 164, 166

geyser, 65, 66

handwash, 162, 163

jobs, 28, 29, 57, 58, 58

leman, 204, 205, 206, 211, 212, 215, 216

LifeCycleSavings, 10, 12, 14, 36, 104, 105

mcycle, 136, 136, 139, 140, 141, 142, 143,
144, 145

orings, 156, 158, 159

peng, 238

Penguin, 54, 55, 55, 56, 57, 57, 59, 60, 61

penguin, 51, 62

penguins, 236

pigs, 129, 130

rehab, 109

rhone, 220, 223, 224, 226, 227

scor, 38, 40

Sitka, 126

Sitka89, 128

swiss, 33

UKDriverDeaths, 198, 199, 200

Command Index

Only first or main appearance listed.

acf, 193
aggregate, 224
ar, 195
arima, 197
arima.sim, 195
ARMAacf, 195

bagging, 88, 89
biplot, 33
boosting, 89, 91

cancor, 36
compile, 246
complete.cases, 236, 238
cooks.distance, 10
cov.spatial, 209
cox.zph, 183
coxph, 181
cmtree, 85

dataset_cifar10, 245

expand.grid, 215
extractAIC, 105

factanal, 38
fit, 247
fit_, 248

gev.fit, 224
glm, 158

hatvalues, 10

image.plot, 43

keras_model_sequential, 246
Krig, 215

layer_ *, 246

map, 43
mKrig, 233
mrlplot, 227
muhaz, 184

neuralnet, 236, 238

pacf, 193
pip, 215
prcomp, 33, 45
predict
 arima, 197
 bagging, 88, 89
 boosting, 91
 glm, 158
 Krig, 215
 krig, 233
 predict, 236, 238
 randomForest, 89
 predictSE, 233

quilt.plot, 215

randomForest, 89
rasterImage, 245
residuals, 10

rmvnorm.spam, 233
rstandard, 10

screeplot, 33
step, 105
Surv, 178
survdiff, 180
survfit, 178
survreg, 186

threshrange.plot, 227

vgram, 211

Package Index

Only first appearance in each chapter listed.

adabag, 88
asreml-R, 130
Epi, 187
extRemes, 224
fields, 43, 205, 233
fma, 3
geoR, 209
ggplot, 136
ggplot2, 129
imager, 250
keras, 244, 245
lattice, 123
lme4, 129, 130
maps, 43
MASS, 79
MCMCglmm, 130
mgcv, 130
muhaz, 184
mvtnorm, 79
ncdf4, 43
neuralnet, 236, 238
nlme, 130
nnet, 250
palmerpenguins, 54, 236, 238
partykit, 85
plyr, 131
png, 95
randomForest, 89
rpart, 95
RSNNS, 250
Rweka, 250
spam, 233
splancs, 215
splines, 136
survival, 178
wavethresh, 146

Video Index

The following gives a short description of the video, including a link. Make sure that you are currently, i.e., actively, authenticated at tube.switch.ch/.

Chapter 0

What are all these videos about?, [v](#)

Chapter 2

More insights into the factor analysis model, [39](#)

Chapter 5

Comments about deriving the SS decomposition, [111](#)

Distributions of various estimators in the regression setting, [101](#)

Some features of the hat matrix, [101](#)

Stacking a multivariate regression problem for a multiple regression setting, [112](#)

Chapter 9

Deriving the standard error of the Kaplan–Meier estimate, [178](#)

Deriving the variance of the Log-rank test, [179](#)

