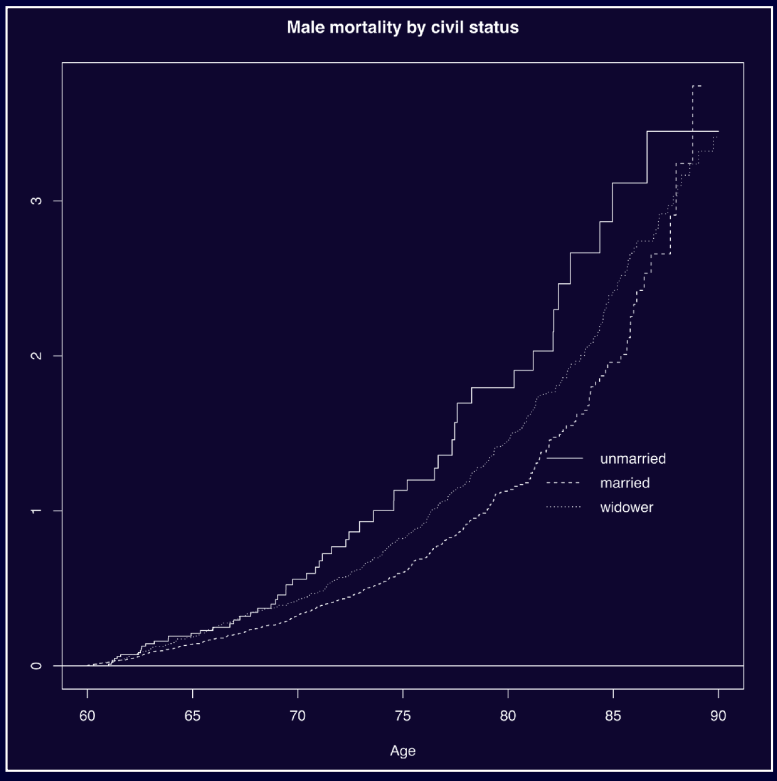
The R Series

Event History Analysis with R



Göran Broström

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Event History Analysis with R

Chapman & Hall/CRC
The R Series

Series Editors

John M. Chambers
Department of Statistics
Stanford University
Stanford, California, USA

Torsten Hothorn
Institut für Statistik
Ludwig-Maximilians-Universität
München, Germany

Duncan Temple Lang
Department of Statistics
University of California, Davis
Davis, California, USA

Hadley Wickham
Department of Statistics
Rice University
Houston, Texas, USA

Aims and Scope

This book series reflects the recent rapid growth in the development and application of R, the programming language and software environment for statistical computing and graphics. R is now widely used in academic research, education, and industry. It is constantly growing, with new versions of the core software released regularly and more than 2,600 packages available. It is difficult for the documentation to keep pace with the expansion of the software, and this vital book series provides a forum for the publication of books covering many aspects of the development and application of R.

The scope of the series is wide, covering three main threads:

- Applications of R to specific disciplines such as biology, epidemiology, genetics, engineering, finance, and the social sciences.
- Using R for the study of topics of statistical methodology, such as linear and mixed modeling, time series, Bayesian methods, and missing data.
- The development of R, including programming, building packages, and graphics.

The books will appeal to programmers and developers of R software, as well as applied statisticians and data analysts in many fields. The books will feature detailed worked examples and R code fully integrated into the text, ensuring their usefulness to researchers, practitioners and students.

Published Titles

Customer and Business Analytics: Applied Data Mining for Business Decision Making Using R, *Robert E. Krider and Daniel S. Putler*

Event History Analysis with R, *Göran Broström*

Programming Graphical User Interfaces with R, *John Verzani and Michael Lawrence*

R Graphics, Second Edition, *Paul Murrell*

Statistical Computing in C++ and R, *Randall L. Eubank and Ana Kupresanin*

The R Series

Event History Analysis with R

Göran Broström
Professor of Statistics
Centre for Population Studies
Umeå University
Umeå, Sweden

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20120106

International Standard Book Number-13: 978-1-4398-3167-0 (eBook - PDF)

Contents

List of Figures **ix**

List of Tables **xiii**

Preface **xv**

1 Event History and Survival Data **1**
1.1 Introduction 1
1.2 Survival Data 1
1.3 Right Censoring 5
1.4 Left Truncation 6
1.5 Time Scales 7
1.6 Event History Data 9
1.7 More Data Sets 11

2 Single Sample Data **17**
2.1 Introduction 17
2.2 Continuous Time Model Descriptions 17
2.3 Discrete Time Models 22
2.4 Nonparametric Estimators 23
2.5 Doing it in **R** 27

3 Cox Regression **31**
3.1 Introduction 31
3.2 Proportional Hazards 31
3.3 The Log-Rank Test 33
3.4 Proportional Hazards in Continuous Time 39
3.5 Estimation of the Baseline Hazard 42
3.6 Explanatory Variables 42
3.7 Interactions 44
3.8 Interpretation of Parameter Estimates 47
3.9 Proportional Hazards in Discrete Time 47
3.10 Model Selection 48
3.11 Male Mortality 49

4 Poisson Regression **57**
4.1 Introduction . . . 57
4.2 The Poisson Distribution . . . 57
4.3 The Connection to Cox Regression . . . 59
4.4 The Connection to the Piecewise Constant Hazards Model . 62
4.5 Tabular Lifetime Data . . . 62

5 More on Cox Regression **67**
5.1 Introduction . . . 67
5.2 Time-Varying Covariates . . . 67
5.3 Communal covariates . . . 68
5.4 Tied Event Times . . . 71
5.5 Stratification . . . 74
5.6 Sampling of Risk Sets . . . 75
5.7 Residuals . . . 77
5.8 Checking Model Assumptions . . . 80
5.9 Fixed Study Period Survival . . . 83
5.10 Left- or Right-Censored Data . . . 84

6 Parametric Models **85**
6.1 Introduction . . . 85
6.2 Proportional Hazards Models . . . 85
6.3 Accelerated Failure Time Models . . . 112
6.4 Proportional Hazards or AFT Model? . . . 115
6.5 Discrete Time Models . . . 116

7 Multivariate Survival Models **127**
7.1 Introduction . . . 127
7.2 Frailty Models . . . 129
7.3 Parametric Frailty Models . . . 134
7.4 Stratification . . . 136

8 Competing Risks Models **139**
8.1 Introduction . . . 139
8.2 Some Mathematics . . . 140
8.3 Estimation . . . 140
8.4 Meaningful Probabilities . . . 140
8.5 Regression . . . 141
8.6 **R** Code for Competing Risks . . . 144

9 Causality and Matching **147**
9.1 Introduction . . . 147
9.2 Philosophical Aspects of Causality . . . 147
9.3 Causal Inference . . . 148
9.4 Aalen's Additive Hazards Model . . . 150
9.5 Dynamic Path Analysis . . . 152

9.6 Matching 153
9.7 Conclusion 157

A Basic Statistical Concepts **159**
A.1 Introduction 159
A.2 Statistical Inference 159
A.3 Asymptotic theory 161
A.4 Model Selection 163

B Survival Distributions **165**
B.1 Introduction 165
B.2 Relevant Distributions in **R** 165
B.3 Parametric Proportional Hazards and Accelerated Failure Time Models 172

C A Brief Introduction to R **177**
C.1 **R** in General 177
C.2 Some Standard **R** Functions 182
C.3 Writing Functions 187
C.4 Graphics 193
C.5 Probability Functions 194
C.6 Help in **R** 197
C.7 Functions in `eha` and `survival` 197
C.8 Reading Data into **R** 202

D Survival Packages in R **205**
D.1 Introduction 205
D.2 eha 205
D.3 survival 206
D.4 Other Packages 207

Bibliography **209**

This page intentionally left blank

List of Figures

1.1 Survival data. 5
1.2 Right censoring. 6
1.3 A Lexis diagram 8
1.4 Marital fertility 9
1.5 The illness-death model. 12

2.1 Life table and survival function 19
2.2 Interpretation of the density function 20
2.3 Interpretation of the hazard function 21
2.4 The geometric distribution 24
2.5 A simple survival data set. 24
2.6 Preliminaries for estimating the hazard function. 25
2.7 Nonparametric estimation of the hazard function. 26
2.8 The Nelson–Aalen estimator. 27
2.9 The Kaplan–Meier estimator 27
2.10 Male mortality, Nelson–Aalen and Kaplan–Meier plots 28
2.11 Male mortality, Weibull fit. 30

3.1 Proportional hazard functions 32
3.2 Proportional hazard functions, log scale 33
3.3 Two-sample data 34
3.4 Old age mortality, women vs. men, cumulative hazards. . . . 37
3.5 Old age mortality by birthplace, cumulative hazards. 39
3.6 Effect of proportional hazards 40
3.7 Estimated cumulative baseline hazard function. 52
3.8 Effects, interaction with centering 55
3.9 Effects, interaction without centering 55

4.1 The *Poisson* cdf 58
4.2 Number of children beyond one for married women 59
4.3 Theoretical *Poisson* distribution 59
4.4 Hazard functions for males and females, model based 65
4.5 Hazard functions for males and females, raw data 66

5.1 A time-varying covariate 68
5.2 Log rye price deviations from trend 69
5.3 Nelson–Aalen plot with the aid of the function `risksets` . . 73

5.4 Survival plot with the aid of the function `risksets` 74
5.5 Cumulative hazards by socioeconomic status, birth intervals. 75
5.6 Residuals in linear regression. 77
5.7 Martingale residuals from a fit of the kidney data. 79
5.8 Martingale residuals from a fit of the male mortality data. . . 79
5.9 Stratification on socio-economic status. 81
5.10 The distribution of `age`, birth interval data. 82
5.11 Plot of residuals against the explanatory variable x. 83
5.12 Plot of residuals against the explanatory variable x^2. 84

6.1 Selected hazard functions. 86
6.2 *Weibull* birth intervals 90
6.3 *Lognormal* birth intervals 92
6.4 Check of the *Weibull* model, birth intervals. 93
6.5 Check of the *Weibull* model, censored birth intervals 94
6.6 Check of the *lognormal* model, censored birth intervals 95
6.7 Check of the *pch* model, uncensored birth intervals. 96
6.8 Check of the *pch* model with 13 constant hazard intervals . . 97
6.9 Estimated hazard function for birth intervals, *pch* 97
6.10 Check of a *Gompertz* fit to old age mortality 108
6.11 Estimated *Gompertz* hazard function for old age mortality . . 108
6.12 Distribution of remaining life at 60, *Weibull* model 110
6.13 Check of a *Weibull* fit to old age mortality 110
6.14 Check of a *Gompertz* fit to old age mortality 112
6.15 Distribution of remaining life at 60, *Gompertz* model 112
6.16 Proportional hazards and accelerated failure time models . . 113
6.17 Barplot of the number of records per person. 118
6.18 Baseline hazards, old age mortality. 123
6.19 The cumulative hazards, from the coxreg fit. 124
6.20 The survival function, from the coxreg fit. 124

7.1 Population hazard function 128

8.1 Competing risks: causes of death. 139
8.2 Death and emigration, Skellefteå, 1800–1870. 142
8.3 Cause-specific exit probabilities 143

9.1 Local dependence. 149
9.2 Cumulative regression coefficients, additive hazards. 151
9.3 Dynamic path analysis 152

B.1 The exponential distribution 167
B.2 Piecewise constant hazards distribution. 168
B.3 Weibull plot of male mortality data. 170
B.4 Loglogistic hazard function 171

C.1 Example of the use of "..." in function calls. 190
C.2 Output from the function `plot2`. 194

This page intentionally left blank

List of Tables

2.1 Life table for Swedish females 2010 18

3.1 Log rank test summary 34
3.2 Logrank test: Summary table 35
3.3 Data in a simple example 35
3.4 Interaction, two factors 45
3.5 Interaction, one factor and one continuous covariate 46
3.6 Interaction, two continuous covariates 46

5.1 The coding of a time-varying covariate 68
5.2 Comparison of methods of handling ties 72

6.1 Distribution assumptions for the birth intervals data 98

8.1 Competing risks in Skellefteå 143

A.1 General table in a log rank test 160

C.1 Precedence of operators, from highest to lowest 182

This page intentionally left blank

Preface

This book is about the analysis of event history and survival data, with special emphasis on how to do it in the statistical computing environment **R** (R Development Core Team 2011). The main applications in mind are demography and epidemiology, but also other applications, where durations are of primary interest, fit in to this framework. Ideally, the reader will have taken a first course in statistics, but some necessary basics may be found in Appendix A. The basis for this book is courses in event history and survival analysis that I have given and developed over the years since the mid-eighties and the development of software for the analysis of this kind of data. This development has during the last ten to fifteen years taken place in the environment of **R** in the package `eha` (Broström 2012).

There are already several good textbooks in the field of survival and event history analysis on the market: (Aalen, Borgan & Gjessing 2008), (Andersen, Borgan, Gill & Keiding 1993), (Cox & Oakes 1984), (Hougaard 2000), (Kalbfleisch & Prentice 2002), and (Lawless 2003) to mention a few. Some of these are already classical, but they all are aimed at an audience with solid mathematical and statistical background. On the other hand, the (Parmar & Machin 1995), (Allison 1984), (Allison 1995), (Klein & Moeschberger 2003), (Collett 2003), and (Elandt-Johnson & Johnson 1999) books are all more basic but lack the special treatment demographic applications needs today, and also the connection to **R**.

In the late seventies, large databases with individual life histories began to appear. One of the absolutely first was *The Demographic Data Base* (DDB) at Umeå University. I started working there as a researcher in 1980, and at that time the database contained individual data for seven Swedish parishes scattered all over the country. Statistical software for Cox regression (Cox 1972) did not exist at that time, so we began the development, with the handling of large data sets in mind, of FORTRAN programs for analyzing censored survival data. The starting point was Appendix 3 of Kalbfleisch & Prentice (1980). It contained "Fortran programs for the proportional hazards model." The beginning of the eighties was also important because then the first text books on survival analysis began to appear. Of special importance (for me) were the books by Kalbfleisch & Prentice (1980), and, four years later, Cox & Oakes (1984).

The time period covered by the DDB data is approximately the nineteenth century. Today the geographical content has been expanded to cover four large regions, with more than 60 parishes in all.

This book is closely tied to the **R** Environment for Statistics and Computing (R Development Core Team 2011). This fact has one specific advantage: Software and this book can be totally synchronized, not only by adapting the book to the software, but also vice versa. This is possible because **R** and its packages are *open source*, and one of the survival analysis packages (Broström 2012) in **R** and this book have the author in common. The `eha` package contains some research results not found in other software (Broström & Lindkvist 2008, Broström 2002, Broström 1987). However, it is important to emphasize that the real workhorse in survival analysis in **R** is the recommended package `survival` by Terry Therneau (Therneau & original Splus to R port by T. Lumley 2011).

The mathematical and statistical theory underlying the content of this book is kept to a minimum; the main target audience is social science researchers and students who are interested in studying demographically oriented research questions. However, the apparent limitation to demographic applications in this book is not really a limitation, because the methods described here are equally useful whenever durations are of primary interest, for instance, in epidemiology and econometrics.

In Chapter 1 event history and survival data are presented, and the specific problems data of this kind pose for statistical analysis. Of special importance are the concepts of censoring and truncation, or in other words, incomplete observations. The dynamic nature of this kind of data is emphasized. The data sets used throughout the book are presented here for the first time.

How to analyze homogeneous data is discussed in Chapter 2. The concept of a survival distribution is introduced, including the hazard function, which is the fundamental concept in survival analysis. The functions that can be derived from the hazard function, the survival and the cumulative hazard functions, are introduced. Then the methods for estimating the fundamental functions nonparametrically are introduced, most important being the Kaplan–Meier and Nelson–Aalen estimators. By "nonparametric" is meant that in the estimation procedures, no restrictions are imposed on the class of allowed distributions except that it must consist of distributions on the positive real line; a life length cannot be negative. Some effort is put into giving an understanding of the intuitive reasoning behind the estimators, and the goal is to show that the ideas are really simple and elementary.

Cox regression is introduced in Chapter 3 and expanded on in Chapter 5. It is based on the property of proportional hazards, which makes it possible to analyze the effects of covariates on survival without specifying a family of survival distributions. Cox regression is in this sense a nonparametric method, but correct is perhaps to call it semiparametric, because the proportionality constant is parametrically determined. The introduction to Cox regression goes via the log-rank test, which can be regarded as a special case of Cox regression. A fairly detailed discussion of different kinds of covariates are given, together with an explanation of how to interpret regression parameters con-

nected to these covariates. Discrete time versions of the proportional hazards assumption are introduced.

In Chapter 4, a break from Cox regression is taken, and Poisson regression is introduced. The real purpose of this, however, is to show the equivalence (in some sense) between Poisson and Cox regressions. For tabular data, the proportional hazards model can be fitted via Poisson regression. This is also true for the continuous time piecewise constant hazards model, which is explored in more detail in Chapter 6.

The Cox regression thread is taken up again in Chapter 5. Time-varying and so-called communal covariates are introduced, and it is shown how to prepare a data set for these possibilities. The important distinction between internal and external covariates is discussed. Stratification as a tool to handle nonproportionality is introduced. How to check model assumptions, in particular the proportional hazards one, is discussed, and finally some less common features, like sampling of risk sets and fixed study period, are introduced.

Chapter 6 introduces fully parametric survival models. They come in three flavors, proportional hazards, accelerated failure time, and discrete time models. The parametric proportional hazards models have the advantages compared to Cox regression that the estimation of the baseline hazard function comes for free. This is also part of the drawback with parametric models; they are nice to work with but often too rigid to fit complex real-world data. It is, for instance, not possible to find simple parametric descriptions of human mortality over the full life span; this would require a U-shaped hazard function, and no standard survival distribution has that property. However, when studying shorter segments of human life span, very good fits may be achieved with parametric models. So it is for instance possible to model accurately old age mortality, say above the age of 60, with the Gompertz distribution. Another important general feature of parametric models is that they convey a simple and clear message about the properties of a distribution or relation, thus easily adding to the accumulated knowledge about the phenomenon it is related to.

The accelerated failure time (AFT) model is an alternative to the proportional hazards (PH) one. While in the PH model, relative effects are assumed to remain constant over time, the AFT model allows for effects to shrink towards zero with age. This is sometimes a more realistic scenario, for instance, in a medical application, where a specific treatment may have an instant, but transient effect.

With modern registration data, exact event times are often not available. Instead, data are grouped in one-year intervals. This is generally not because exact information is missing at the government agencies, but a result of integrity considerations. With access to birth date information, a person may be easy to identify. Anyway, as a result it is only possible to measure age in (completed) years, and heavily tied data sets will be the result. That opens up the use of discrete-time models, similar to uses for logistic and Poisson

regression. In fact, as is shown in Chapter 4, there is a close relation between Cox regression and binary and count data regression models.

Finally, in the framework of Chapter 6 and parametric models, it is important to mention the piecewise constant hazard (PCH) model. It constitutes an excellent compromise between the nonparametric Cox regression and the fully parametric models discussed above. Formally, it is a fully parametric model, but it is easy to adapt to given data by changing the number of cut points (and thus the number of unknown parameters). The PCH model is especially useful in demographic and epidemiological applications where there is a huge amount of data available, often in tabular form.

Chapters 1–6 are suitable for an introductory course in survival analysis, with a focus on Cox regression and independent observations. In the remaining chapters, various extensions to the basic model are discussed.

In Chapter 7, a simple dependence structure is introduced, the so-called shared frailty model. Here, it is assumed that data are grouped into clusters or litters, and that individuals within a cluster share a common risk. In demographic applications, the clusters are biological families, people from the same geographical area, and so on. This pattern creates dependency structures in the data, which are necessary to consider in order to avoid biased results.

Competing risks models are introduced in Chapter 8. Here the simple survival model is extended to include of failures of several types. In a mortality study, it may be deaths due to different causes. The common feature in these situations is that for each individual, many events may occur, but at most one will occur. The events are competing with each other, and at the end there is only one winner. It turns out that in these situations the concept of cause-specific hazards is meaningful, while the discussion of cause-specific survival functions is problematic.

During the last few decades, causality has become a hot topic in statistics. In Chapter 9 a review of parts relevant to event history analysis is given. The concept of matching is emphasized. It is also an important technique in its own right, not necessarily tied to causal inference.

There are four appendices. They contain material that is not necessary to read in order to be able to follow the core of the book, given proper background knowledge. Browsing through Appendix A is recommended to all readers, at least so that common statistical terminology is agreed upon. Appendix B contains a description of relevant statistical distributions in **R**, and also a presentation of the modeling behind the parametric models in the **R** package `eha`. The latter part is not necessary for an understanding of the rest of the book. For readers new to **R**, Appendix C is a good starting point, but it is recommended to complement it with one of the many introductory text books on **R** or online sources. Consult the home of **R**, `http://www.r-project.org` and search under *Documentation*. Appendix C also contains a separate section with a selection of useful functions from the `eha` and the `survival` packages. This is, of course, recommended reading for everyone. It is also valuable as a

reference library to functions used in the examples of the book. They are not always documented in the text.

Finally, Appendix D contains a short description of the most important packages for event history analysis in **R**.

As a general reading instruction, the following is recommended. Install the latest version of **R** from CRAN (`http://cran.r-project.org`), and replicate the examples in the book by yourself. You need to install the package `eha` and `load` it. All the data in the examples are then available online in **R**.

Many thanks go to people who have read early and not-so-early versions of the book; they include Tommy Bengtsson, Kristina Broström, Bendix Carstensen, Renzo Derosas, Sören Edvinsson, Kajsa Fröjd, Ingrid Svensson, and students at the Departments of Statistics and Mathematical Statistics at the University of Umeå. Many errors have been spotted and improvements suggested, and the remaining errors and weaknesses are solely my responsibility.

I also had invaluable support from the publisher, CRC Press, Ltd. I especially want to thank Sarah Morris and Rob Calver for their interest in my work and their encouragement in the project.

Of course, without the excellent work of the **R** community, especially the **R** Core Team, a title like the present one would have been impossible. Especially I want to thank Terry Therneau for his inspiring work on the `survival` package; the `eha` package depends heavily on it. Also, Friedrich Leisch, author of the `Sweave` package, deserves many thanks. This book was entirely written with the aid of the package `Sweave` and the typesetting system LaTeX, wonderful companions.

Finally, I want to thank Tommy Bengtsson, director of the *Centre for Economic Demography*, Lund University, and Anders Brändström, director of the Demographic Data Base at Umeå University, for kindly letting me use real-world data in this book and in the **R** package `eha`. It greatly enhances the value of the illustrations and examples.

Umeå, October 2011

Göran Broström

This page intentionally left blank

1

Event History and Survival Data

1.1 Introduction

What characterizes event history and survival data the most is its *dynamic* nature. Individuals are followed over time, and during that course, the timings of events of interest are noted. Naturally, things may happen that makes it necessary to interrupt an individual follow-up, such as the individual suddenly disappearing for some reason. With classical statistical tools, such as linear regression, these observations are difficult or impossible to handle in the analysis. The methods discussed in this book aim among other things, at solving such problems.

In this introductory chapter, the use of the special techniques that constitute survival and event history analysis are motivated. The concepts of *right censoring* and *left truncation* are defined and discussed. The data sets used throughout the book are also presented in this chapter.

The environment **R** is freely (under a *GPL license*) available for download from `http://cran.r-project.org`. There you will find precompiled versions for Linux, MacOS X, and Microsoft Windows, as well as the full source code, which is open. See *Appendix C* for more information.

1.2 Survival Data

Survival data (survival times) constitute the simplest form of event history data. A survival time is defined as the time it takes for an event to occur, measured from a well-defined start event. Thus, there are three basic elements which must be well defined: a time origin, a scale for measuring time, and an event. The response in a statistical analysis of such data is the exact time elapsed from the time origin to the time at which the event occurs. The challenge, which motivates special methods, is that in most applications, this duration is often not possible to observe exactly.

As an introduction to the research questions that are suitable for handling with event history and survival analysis, let us look at a data set found in the `eha` package (Broström 2012) in **R** (R Development Core Team 2011).

Example 1 *Old age mortality*

The data set `oldmort` in `eha` contains survival data from the parish Sundsvall in the mid-east of 19th century Sweden. The name `oldmort` is an acronym for *old age mortality*. The source is digitized information from historical parish registers and church books. More information about this can be found at the web page of *The Demographic Data Base* at Umeå University (DDB), `http://www.ddb.umu.se`.

The sampling was done as follows: Every person who was present and alive and 60 years of age or above anytime between 1 January 1860 and 31 December 1879 was followed from the entrance age (for most people that would be 60) until the age when last seen, determined by death, out-migration, or surviving until 31 December 1879. Those born during the eighteenth century would enter observation at an age above 60, given that they lived long enough, that is, at least until 1 January 1860.

Two types of finishing the observation of a person are distinguished: Either it is by *death* or it is by something else, out-migration or end of study period. In the first case we say that the event of interest has occurred, in the second case not.

After installing the `eha` package and starting an **R** session (see Appendix C), the data set is loaded as follows.

```
> library(eha)
> data(oldmort)
```

The first line loads the package `eha` into the *workspace* of **R**, and the second line loads the data set `oldmort` found in `eha`. Let us look at the first few lines of `oldmort`. It is conveniently done with the aid of the **R** function `head`:

```
> head(oldmort)
         id  enter   exit event birthdate m.id f.id    sex
1 765000603 94.510 95.813  TRUE  1765.490   NA   NA female
2 765000669 94.266 95.756  TRUE  1765.734   NA   NA female
3 768000648 91.093 91.947  TRUE  1768.907   NA   NA female
4 770000562 89.009 89.593  TRUE  1770.991   NA   NA female
5 770000707 89.998 90.211  TRUE  1770.002   NA   NA female
6 771000617 88.429 89.762  TRUE  1771.571   NA   NA female
        civ  ses.50 birthplace imr.birth   region
1     widow unknown     remote  22.20000    rural
2 unmarried unknown     parish  17.71845 industry
3     widow unknown     parish  12.70903    rural
4     widow unknown     parish  16.90544 industry
5     widow  middle     region  11.97183    rural
6     widow unknown     parish  13.08594    rural
```

The variables in `oldmort` have the following definitions and interpretations:

id A unique id number for each individual.

enter, exit The *start age* and *stop age* for this record (spell). For instance, in row No. 1, individual No. 765000603 enters under observation at age 94.51 and exits at age 95.813. Age is calculated as the number of days elapsed since birth, and this number is then divided by 365.25 to get age in years. The denominator is the average length of a year, taking into account that (almost) every fourth year is 366 days long.

The first individual was born around 1 July 1765, and so almost 95 years of age when the study started. Suppose that this woman had died at age 94; then she had not been in our study at all. This property of our sampling procedure is a special case of a phenomenon called *length-biased sampling*. That is, of those born in the eighteeenth century, only those who live well beyond 60 will be included. This bias must be compensated for in the analysis, and it is accomplished by *conditioning* on the fact that these persons were alive at 1 January 1860. This technique is called *left truncation*.

event A *logical* variable (taking values `TRUE` or `FALSE`) indicating if the exit is a death (`TRUE`) or not (`FALSE`). For our first individual, the value is `TRUE`, indicating that she died at the age of 95.813 years.

birthdate The birth date expressed as the time (in years) elapsed since January 1, year 0 (which by the way does not exist). For instance, the (pseudo) date 1765.490 is really June 27, 1765. The fraction 0.490 is the fraction of the year 1765 that elapsed until the birth of individual No. 765000603.

m.id Mother's `id`. It is unknown for all the individuals listed above. That is the symbol `NA`, which stands for `Not Available`. The oldest people in the data set typically have no links to parents.

f.id Father's `id`. See `m.id`.

sex A *categorical* variable with the levels `female` and `male`.

civ Civil status. A categorical variable with three levels; `unmarried`, `married`, and `widow(er)`.

ses.50 Socioeconomic status (SES) at age 50. Based on occupation information. There is a large proportion of `NA` (missing values) in this variable. This is quite natural, because this variable was of secondary interest to the record holder (the priest in the parish). The occupation is only noted in connection to a vital event in the family (such as a death, birth, marriage, or in- or out-migration). For those who were above 50 at the start of the period there is no information on SES at 50.

birthplace A categorical variable with two categories, `parish` and `remote`, representing *born in parish* and *born outside parish*, respectively.

imr.birth A rather specific variable. It measures the *infant mortality rate* in the birth parish at the time of birth (per cent).

region Present geographical area of residence. The parishes in the region are grouped into three regions, `Sundsvall town`, `rural`, and `industry`. The industry is the sawmill one, which grew rapidly in this area during the late part of the 19th century. The Sundsvall area was, in fact, one of the largest sawmill areas in Europe at this time.

Of special interest is the triple `(enter, exit, event)`, because it represents *the response variable*, or what can be seen of it. More specifically, the sampling frame is all persons observed to be alive and above 60 years of age between 1 January 1860 and 31 December 1879. The start event for these individuals is their 60th anniversary, and the stop event is death. Clearly, many individuals in the data set did not die before 1 January 1880, so for them we do not know the full duration between the start and stop events; such individuals are said to be *right censored* (the exact meaning of which will be given soon). The third component in the *survival object* `(enter, exit, event)`; that is, `event` is a *logical* variable taking the value `TRUE` if `exit` is the true duration (the interval ends with a death) and `FALSE` if the individual is still alive at the duration "last seen."

Individuals aged 60 or above between 1 January 1860 and 31 December 1879 are included in the study. Those who are above 60 at this start date are included only if they did not die between the age of 60 and the age at 1 January 1860. If this is not taken into account, a bias in the estimation of mortality will result. The proper way of dealing with this problem is to use *left truncation*, which is indicated by the variable `enter`. If we look at the first rows of `oldmort`, we see that the `enter` variable is very large; it is the age for each individual at 1 January 1860. You can add `enter` and `birthdate` for the first six individuals to see that:

```
> oldmort$enter[1:6] + oldmort$birthdate[1:6]
[1] 1860 1860 1860 1860 1860 1860
```

The statistical implication (description) of left truncation is that its presence forces the analysis to be *conditional* on survival up to the age `enter`.

A final important note: In order to get the actual duration at exit, we must subtract 60 from the value of `exit`. When we actually perform a survival analysis in **R**, we should subtract 60 from both `enter` and `exit` before we begin. It is not absolutely necessary in the case of Cox regression, because of the flexibility of the baseline hazard in the model (it is, in fact, left unspecified!). However, for parametric models, it may be important in order to avoid dealing with truncated distributions.

Now let us think of the research questions that could be answered by analyzing this data set. Since the data contain individual information on the length of life after 60, it is quite natural to study what determines a long life and what are the conditions that are negatively correlated with long life.

Obvious questions are: (i) Do women live longer than men? (Yes), (ii) Is it advantageous for a long life to be married? (Yes), (iii) Does socioeconomic status play any role for a long life? (Don't know), and (iv) Does place of birth have any impact on a long life, and if so, is it different for women and men?

The answers to these, and other, questions will be given later. The methods in later chapters of the book are all illustrated on a few core eamples. They are all introduced in this chapter. □

The data set `oldmort` contained only two states, referred to as *Alive* and *Dead*, and one possible transition, from Alive to Dead; see Figure 1.1. The

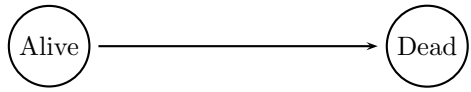


FIGURE 1.1
Survival data.

ultimate study object in survival analysis is the time it takes from entering state *Alive* (e.g., becoming 60 years of age) until entering state *dead* (e.g., death). This time interval is defined by the exact time of two events, which we may call *birth* and *death*, although in practice these two events may be almost any kind of events. Economists, for instance, are interested in the duration of out-of-work spells, where "birth" refers to the event of losing the job, and "death" refers to the event of getting a job. In a clinical trial for treatment of cancer, the starting event time may be time of operation, and the final event time is time of relapse (if any).

1.3 Right Censoring

When an individual is lost to follow-up, we say that she is *right censored*; see Figure 1.2. As indicated in Figure 1.2, the true age at death is T, but due to right censoring, the only information available is that death age T is larger than C. The number C is the age at which this individual was last seen. In ordinary, classical regression analysis, such data are difficult, if not impossible, to handle. Discarding such information may introduce bias. The modern theory of survival analysis offers simple ways to deal with right-censored data.

A natural question to ask is: If there is right censoring, there should be

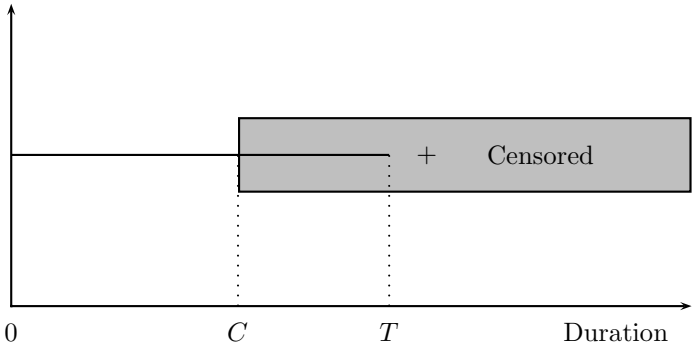


FIGURE 1.2
Right censoring.

something called *left* censoring, and if so, what is it? The answer to that is that yes, *left censoring* refers to a situation where the only thing known about a death age is that it is *less than* a certain value C. Note carefully that this is different from *left truncation*; see the next section.

1.4 Left Truncation

The concept of *left truncation*, or *delayed entry*, is well illustrated by the data set `oldmort` that was discussed in detail in Section 1.2. Please note the difference compared to *left censoring*. Unfortunately, you may still see articles where these two concepts are confused.

It is illustrative to think of the construction of the data set `oldmort` as a statistical follow-up study, starting on 1 January 1860. At that day, all persons present in the parish and 60 years of age *or above*, are included in the study. It is decided that the study will end at 31 December 1879; that is, the study period (follow-up time) is 20 years. The interesting event in this study is *death*. This means that the start event is the sixtieth anniversary of birth, and the final event is death. Due to the calendar time constraints (and migration), all individuals will not be observed to die (especially those who live long), and moreover, some individuals will enter the study after the "starting" event, the sixtieth anniversary. A person who entered late—say he is 65 on January 1, 1860—would not have been included had he died at age 63 (say). Therefore, in the analysis, we must *condition* on the fact that he was alive at 65. Another way of putting this is to say that this observation is *left truncated* at age 65.

People being too young at the start date will be included from the day

they reach 60, if that happens before the closing date, 31 December 1879. They are not left truncated, but will have a higher and higher probability of being right censored, the later they enter the study.

1.5 Time Scales

In demographic applications *age* is often a natural time scale, that is, time is measured from birth. In the old age data just discussed, time was measured from age 60 instead. In a case like this, where there is a common "late start age," it doesn't matter much, but in other situations it does. Imagine, for instance, that interest lies in studying the time it takes for a woman to give birth to her first child after marriage. The natural way of measuring time is to start the clock at the day of marriage, but a possible (but not necessarily recommended!) alternative is to start the clock at some (small) common age of the women, for instance, at birth. This would give left truncated (at marriage) observations, since women were sampled at marriage. There are two clocks ticking, and you have to make a choice. Generally, it is important to realize that there often are alternatives, and that the result of an analysis may depend strongly on the choice made.

1.5.1 The Lexis Diagram

Two time scales are nearly always present in demographic research: age (or duration) and calendar time. For instance, an investigation of mortality may be limited in these two directions. In Figure 1.3 this is illustrated for a study of old age mortality during the years 1829 and 1895. "Old age mortality" is defined as mortality from age 50 and onwards to age 100. The Lexis diagram is a way of showing the interplay between the two time scales and (human) life lines. Age moves vertically and calendar time horizontally, which will imply that individual lives will move diagonally, from birth to death, from south-west to north-east, in the Lexis diagram. In our example study, we are only interested in the part of the life lines that appear inside the rectangle.

Assume that the data set at hand is saved in the text file 'lex.dat'. Note that this data set is not part of `eha`; it is only used here for the illustration of the Lexis diagram.

```
> lex <- read.table("Data/lex.dat", header = TRUE)
> lex
  id enter   exit event birthdate    sex
1  1     0 98.314     1  1735.333   male
2  2     0 87.788     1  1750.033   male
3  3     0 71.233     0  1760.003 female
4  4     0 87.965     1  1799.492   male
```

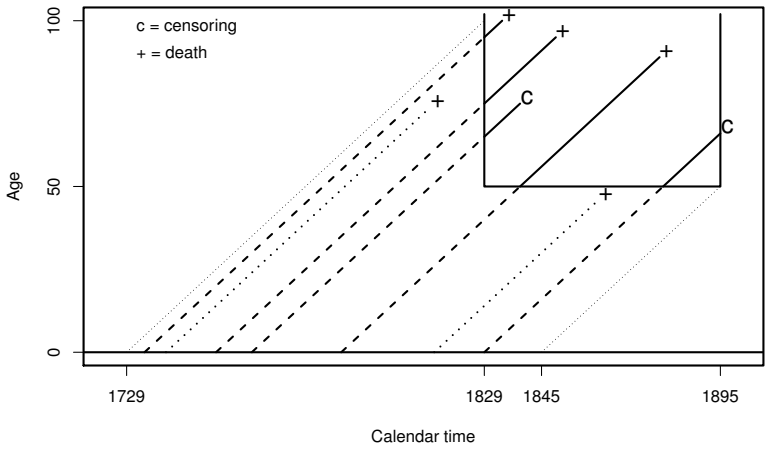


FIGURE 1.3
Lexis diagram; time period 1829–1894 and ages 50–100 under study.

```
5  5     0 82.338     1  1829.003 female
6  6     0 45.873     1  1815.329 female
7  7     0 74.112     1  1740.513 female
```

How do we restrict the data to fit into the rectangle given by the Lexis diagram in Figure 1.3? With the two functions `age.window` and `cal.window`, it is easy. The former fixes the 'age cut' while the latter makes the 'calendar time cut'.

The age cut:

```
> require(eha)
> lex <- age.window(lex, c(50, 100))
> lex
  id enter   exit event birthdate    sex
1  1    50 98.314     1  1735.333   male
2  2    50 87.788     1  1750.033   male
3  3    50 71.233     0  1760.003 female
4  4    50 87.965     1  1799.492   male
5  5    50 82.338     1  1829.003 female
7  7    50 74.112     1  1740.513 female
```

Note that individual No. 6 dropped out completely because she died too young. Then the calendar time cut:

```
> lex <- cal.window(lex, c(1829, 1895))
> lex
```

```
  id  enter   exit event birthdate    sex
1  1 93.667 98.314     1  1735.333   male
2  2 78.967 87.788     1  1750.033   male
3  3 68.997 71.233     0  1760.003 female
4  4 50.000 87.965     1  1799.492   male
5  5 50.000 65.997     0  1829.003 female
```

and here individual No. 7 disappeared because she died before 1 January 1829. Her death date is her birth date plus her age at death, $1740.513 + 74.112 = 1814.625$, or 17 August 1814.

1.6 Event History Data

Event history data arise, as the name suggests, by following subjects over time and making notes about what happens and when. Usually the interest is concentrated to a few specific kinds of events. The main application in this book is demography and epidemiology, and hence events of primary interest are *births*, *deaths*, *marriages*, and *migration*.

Example 2 *Marital fertility in 19th century Sweden*

As a rather complex example, let us look at *marital fertility* in 19th century Sweden; see Figure 1.4.

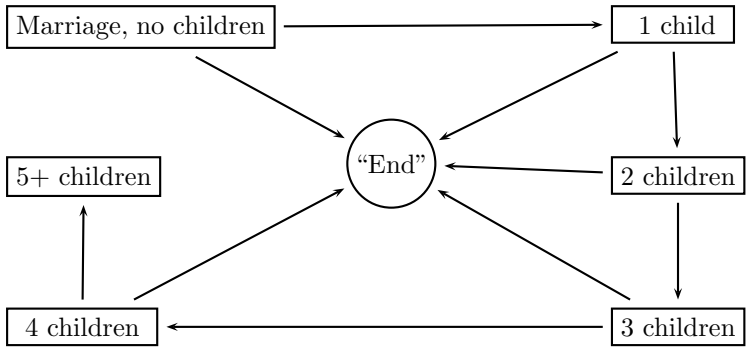


FIGURE 1.4
Marital fertility. Follow-up starts at marriage and ends when the marriage is dissolved or the woman has had her fifth birth or she becomes 50 years of age, whichever comes first.

In a marital fertility study, women are typically followed over time from the time of their marriage until the time the marriage is dissolved or her fertility

period is over, say at age 50, whichever comes first. The marriage dissolution may be due to the death of the woman or of her husband, or it may be due to a divorce. If the study is limited to a given geographical area, women may get lost to follow-up due to out-migration. This event gives rise to a *right-censored* observation.

During the follow-up, the exact timings of child births are recorded. Interest in the analysis may lie in investigating which factors, if any, affect the length of birth intervals. A data set may look like this:

```
  id parity age year next.ivl event prev.ivl     ses parish
1  1      0  24 1825    0.411     1       NA  farmer    SKL
2  1      1  25 1826   22.348     0    0.411  farmer    SKL
3  2      0  18 1821    0.304     1       NA unknown    SKL
4  2      1  19 1821    1.837     1    0.304 unknown    SKL
5  2      2  21 1823    2.546     1    1.837 unknown    SKL
6  2      3  23 1826    2.541     1    2.546 unknown    SKL
7  2      4  26 1828    2.431     1    2.541 unknown    SKL
8  2      5  28 1831    2.472     1    2.431 unknown    SKL
9  2      6  31 1833    3.173     0    2.472 unknown    SKL
```

This is the first 9 rows, corresponding to the first two mothers in the data file. The variable `id` is *mother's id*, a label that uniquely identifies each individual.

A birth interval has a start point (in time) and an end point. These points are the time points of births, except for the first interval, where the start point is time of marriage, and the last interval, which is open to the right. However, the last interval is stopped at the time of marriage dissolution or when the mother becomes 50, whatever comes first. The variable `parity` is zero for the first interval, between date of marriage and date of first birth, one for the next interval, and so forth. The last (highest) number is thus equal to the total number of births for a woman during her first marriage (disregarding twin births, etc.).

Here is a variable-by-variable description of the data set.

id The mother's unique id.

parity Order of *previous* birth; see above for details. Starts at zero.

age Mother's age at the event defining the start of the interval.

year Calendar year for the birth defining the start of the interval.

next.ivl The time in years from the birth at `parity` to the birth at `parity + 1`, or, for the woman's last interval, to the age of right censoring.

event An indicator for the interval ending with a birth. It is always equal to 1, except for the last interval, which always has event equal to zero.

prev.ivl The length of the interval preceding this one. For the first interval of a woman, it is always `NA` (Not Available).

ses Socioeconomic status (based on occupation data).

parish The home parish of mother at birth.

Just to make it clear: The first woman has id `1`. She is represented by two records, meaning that she gave birth to one child. She waited 0.411 years from marriage to the first birth, and 22.348 years from the first birth to the second, *which never happened*. The second woman `2` is represented by seven records, implying that she gave birth to six children. And so on.

Of course, in an analysis of birth intervals we are interested in causal effects; why are some intervals short while others are long? The dependence of the history can be modeled by `lengths of previous intervals` (for the same mother), `parity`, survival of earlier births, and so on. Note that all relevant covariate information *must refer to the past*. More about that later.

The first interval of a woman is different from the others, since it starts with marriage. It therefore makes sense to analyze these intervals separately. The last interval of a woman is also special; it always ends with a right censoring, at the latest when the woman is 50 years of age. You should think of data for a woman generated sequentially in time, starting at the day of her marriage. Follow-up is made to the next birth, as long as she is alive, the marriage is still alive, and she is younger than 50 years of age. If there is no next birth, that is, she reaches 50, or the marriage is dissolved (most often by death of one of the spouses), the interval is censored at the duration when she still was under observation. Censoring can also occur by emigration, and reaching the end of follow-up, in this case 5 November 1901. □

Example 3 *The illness-death model.*

Another useful setup is the so-called *illness-death* model; see Figure 1.5. Individuals may move back and forth between the states *Healthy* and *Diseased*, and from each of these two states there is a pathway to *Death*, which is an *absorbing state*, meaning that once in that state, you never leave it. □

1.7 More Data Sets

A few examples and data sets will be used repeatedly throughout the book, and we give a brief description of them here. They are all available in the **R** package `eha`, which is loaded into a running **R** session by the call

```
> require(eha)
```

This loads the `eha` package, if it wasn't loaded before. In the examples to follow, we assume that this is already done. The main data source is the

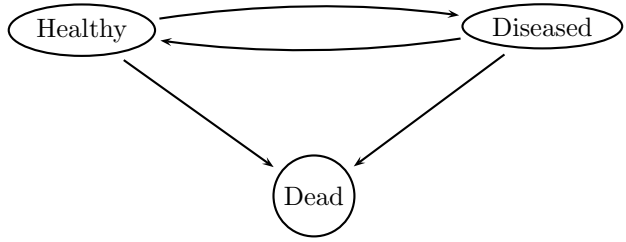


FIGURE 1.5
The illness-death model.

Demographic Data Base, Umeå University, Sweden. However, one data set is taken from the home page of *Statistics Sweden* (`http://www.scb.se`).

Example 4 *Survival of males aged 20*

This data set is included in the **R** (R Development Core Team 2011) package *eha* (Broström 2012). It contains information about 1023 males, age 20 between 1 January 1800 and 31 December 1819, and living in Skellefteå, a parish in the north-east of Sweden. The total number of records in the data frame is 1211; that is, some individuals are represented by more than one record in the data file. The reason for that is that the *socioeconomic status* (`ses`) is one of the covariates in the file, and it changes over time. Each time a change is recorded, a new record is created for that individual, with the new value of SES. For instance, the third and fourth rows in the data frame are

```
> library(eha)
> data(mort)
> mort[3:4, ]
  id  enter   exit event birthdate   ses
3  3  0.000 13.463     0  1800.031 upper
4  3 13.463 20.000     0  1800.031 lower
```

Note that the variable `id` is the same (3) for the two records, meaning that both records are information about individual No. 3. The variable `enter` is age (in years) that has elapsed since the 20th birth day anniversary, and `exit` likewise. The information about him is that he was born on 1800.031, or January 12, 1800, and he is followed from his 21th birth date, or from January 12, 1820. He is in an `upper` socioeconomic status until he is $20 + 13.46311 = 33.46311$ years of age, when he unfortunately is degraded to a `lower ses`. He

is then followed until 20 years have elapsed, or until his 41th birthday. The variable `event` tells us that he is alive we when stop observing him; the value zero indicates that the follow-up ends with *right censoring*.

In an analysis of male mortality with this data set, we could ask whether there is a socioeconomic difference in mortality, and also if it changes over time. That would typically be done by *Cox regression* or by a parametric *proportional hazards model*. More about that follows in later chapters.

Example 5 *Infant mortality*

This data set is taken from (Broström 1987) and concerns the interplay between infant and maternal mortality in 19th century Sweden (source: The Demographic Data Base, Umeå University, Sweden). More specifically, we are interested in estimating the effect of a mother's death on the infant's survival chances. Because maternal mortality was rare (around one per 200 births), matching is used. This is performed as follows: for each child experiencing the death of its mother (before age one), two matched controls were selected. The criteria were: same age as the case at the event, same sex, birth year, parish, socioeconomic status, and marital status of mother. The triplets so created were followed until age one, and eventual deaths of the infants were recorded. The data collected in this way is part of the `eha` package under the name `infants`, and the first row of the data frame are shown here:

```
> data(infants)
> head(infants)
  stratum enter exit event mother age  sex      parish   civst
1       1    55  365     0   dead  26  boy Nedertornea married
2       1    55  365     0  alive  26  boy Nedertornea married
3       1    55  365     0  alive  26  boy Nedertornea married
4       2    13   76     1   dead  23 girl Nedertornea married
5       2    13  365     0  alive  23 girl Nedertornea married
6       2    13  365     0  alive  23 girl Nedertornea married
     ses year
1 farmer 1877
2 farmer 1870
3 farmer 1882
4  other 1847
5  other 1847
6  other 1848
```

A short description of the variable follows.

stratum denotes the id of the triplets, 35 in all.

enter is the age in days of the case, when its mother died.

exit is the age in days when follow-up ends. It takes the value 365 (one year) for those who survived their first anniversary.

event indicates whether a death (1) or a survival (0) was observed.

mother has value `dead` for all cases and the value `alive` for the controls.

age Age of mother at infant's birth.

sex Sex of the infant.

parish Birth parish.

civst Civil status of mother, married or unmarried.

ses Socioeconomic status, often the father's, based on registrations of occupation.

year Calendar year of the birth.

This data set is discussed and analyzed in Chapter 8. □

Example 6 *Old age mortality, tabular data*

This data set is taken from Statistics Sweden. It is freely available on the web site `http://www.scb.se`. The aggregated data set contains information about population size and no. of deaths by sex and age for the ages 61 and above for the year 2007.

```
> data(swe07)
> head(swe07)
    pop deaths    sex age  log.pop
1 63483    286 female  61 11.05853
2 63770    309 female  62 11.06304
3 64182    317 female  63 11.06948
4 63097    366 female  64 11.05243
5 61671    387 female  65 11.02957
6 57793    419 female  66 10.96462
> tail(swe07)
     pop deaths  sex age  log.pop
35 31074    884 male  75 10.34413
36 29718    904 male  76 10.29951
37 29722   1062 male  77 10.29964
38 28296   1112 male  78 10.25048
39 27550   1219 male  79 10.22376
40 25448   1365 male  80 10.14439
```

The variables have the following meanings.

pop Average population size 2007 in the `age` and for the `sex` given on the same row. The average is based on the population at the beginning and end of the year 2007.

deaths The observed number of deaths in the `age` and for the `sex` given on the same row.

sex Female or male.

age Age in completed years.

log.pop The natural logarithm of `pop`. This variable is used as `offset` in a Poisson regression.

See Chapter 4 for how to analyze this data set. □

This page intentionally left blank

2

Single Sample Data

2.1 Introduction

The basic model descriptions of survival data are introduced. Basically, the distribution of survival data may be either *continuous* or *discrete*, but the nonparametric estimators of the distributions of survival time are discrete in any case. Therefore, when introducing the estimators, the discrete models are necessary to know.

In the present chapter, only nonparametric estimation is discussed. This means that no assumptions whatsoever are made about the true underlying distribution. Parametric models are presented in Chapter 5.

2.2 Continuous Time Model Descriptions

Traditional statistical model descriptions are based on the *density* and the *cumulative distribution* functions. These functions are not so suitable for use with censored and/or truncated data. The *survival* and *hazard* functions are better suited, as will be demonstrated here. It should, however, be acknowledged that all these functions are simple functions of each other, so the information they convey is in principle the same, it is only convenience that determines which one(s) to choose and use in a particular application.

2.2.1 The Survival Function

We start with a motivating example, the life table, and its connection to death risks and survival probabilities in a population; see Table 2.1.

Example 7 *Swedish female life table 2009.*

Statistics Sweden (SCB) is a government agency that produces statistics and has a coordinating role for the official statistics of Sweden. From their home page (http://www.scb.se) it is possible to download vital population statistics, like population size and number of deaths by age, gender, and year. From such

TABLE 2.1 Life table for Swedish females 2010

age(x)	pop	deaths	risk(%)	alive at age x
0	55 407	117	0.211	100 000
1	54 386	22	0.040	99 789
2	53 803	11	0.020	99 748
3	53 486	7	0.013	99 728
4	52 544	4	0.008	99 715
...	...	...	...	...
96	3 074	1 030	33.507	6 529
97	2 204	817	37.061	4 341
98	1 473	624	42.363	2 732
99	920	433	47.065	1 575
100+	1 400	837	59.786	834

data it is possible to construct a *life table*; see Table 2.1. It is constructed in the following way. The first column, headed by *age(x)*, contains the ages in completed years. The second column, headed *pop*, contains the numbers of women in Sweden in the corresponding ages. It is equal to the mean population size over the year. Since population numbers from the SCB are taken at the turn of the years, the numbers in column two are averages of two columns in the SCB file. The third column is the total numbers of deaths in each age category during the year. The fourth and fifth columns are derived from the previous ones. The *risk* is essentially the ratio between the number of deaths and population size for each age.

Column five, *alive at age x*, contains the actual construction of the life table. It depicts the actual size of a birth cohort, *year by year*, exposed to the death risks of column four. It starts with 100 000 newborn females. Under the first year of life the death risk is 0.211%, meaning that we expect $0.00211 \times 100\,000 = 211$ girls to die the first year. In other words, $100\,000 - 211 = 99\,789$ girls will remain alive at their first anniversary (at exact age one year). That is the number in the second row in the fifth column. The calculations then continue in the same way through the last two columns. Note, though, that the last interval (100+) is an open interval; that is, it has no fixed upper limit.

Note the difference in interpretation between the numbers in the second column ("pop") and those in the last ("alive at age x"); the former give the age distribution in a population a given year (referred to as *period data*), while the latter constitute *synthetic cohort data*; that is, a group of newborn are followed from birth to death, and the actual size of the (still alive) cohort is recorded age by age. That is, "individuals" in the synthetic cohort live their whole lives under the mortality conditions of the year 2009. Of course, no real individual can experience this scenario, hence the name "synthetic cohort." Nevertheless, it is a useful tool to illustrate the state of affairs regarding mortality in a given year.

It makes sense to plot the life table, see Figure 2.1. By dividing all numbers by the original cohort size (in our case 100,000), they may be interpreted as *probabilities*. □

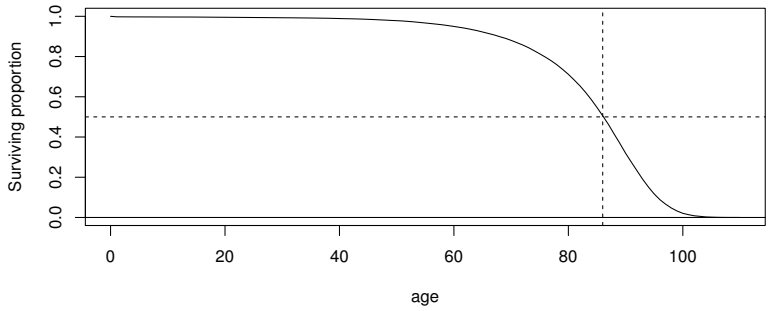


FIGURE 2.1
Life table and survival function, females, Sweden 2009. Median life length is 86; see construction in figure.

The *survival function* $S(t),\ t > 0$, is defined as the probability of surviving past t, $t > 0$, or with a formula,

$$S(t) = P(T \geq t), \quad t > 0,$$

where T is the (random) life length under study. The symbol $P(T \geq t)$ reads *"the probability that T is equal to or larger than t"*; here t is a fixed number while T denotes a "random quantity." In statistical language, T is called a *random variable*. In our example, T is the (future) life length of a randomly chosen newborn female, of course, unknown at birth.

We will, for the time being, assume that S is a "nice" function, smooth and *differentiable*. That will ensure that the following definitions are unambiguous.

2.2.2 The Density Function

The *density function* f is defined as minus the derivative of the survival function, or

$$f(t) = -\frac{\partial}{\partial t} S(t), \quad t > 0.$$

The intuitive interpretation of this definition is that, for small enough s, the following approximation is close to the true value:

$$P(t_0 \leq T < t_0 + s) \approx s f(t_0)$$

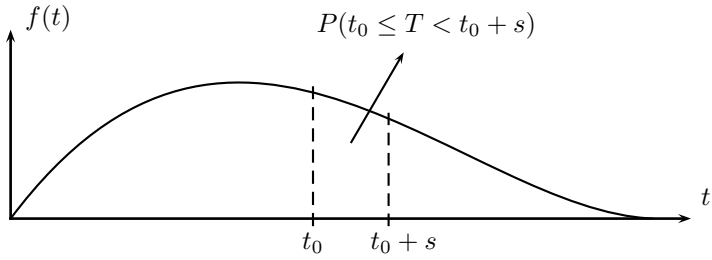


FIGURE 2.2
Interpretation of the density function f.

This is illustrated in Figure 2.2; for a short enough interval $(t_0, t_0 + s]$, the probability of an observation falling in that interval is well approximated by the area of a rectangle with sides of lengths s and $f(t_0)$, or $sf(t_0)$. The formal mathematical definition as a limit is given in equation (2.1).

$$f(t) = \lim_{s \to 0} \frac{P(t \leq T < t + s)}{s}, \quad t > 0. \tag{2.1}$$

2.2.3 The Hazard Function

The *hazard function* is central to the understanding of survival analysis, so the reader is recommended to get at least an intuitive understanding of it. One way of thinking of it is as an "instant probability"; at a given age t, it measures the risk of dying in a short interval $(t, t+s)$ immediately after t, for an individual *who still is alive at* t.

$$h(t) = \lim_{s \to 0} \frac{P(t \leq T < t + s \mid T \geq t)}{s}$$
$$P(t_0 \leq T < t_0 + s \mid T \geq t_0) \approx sh(t_0)$$

Note the difference between the density and the hazard functions. The former is (the limit of) an *unconditional* probability, while the latter is (the limit of) a *conditional* probability. Otherwise, the hazard function is also a kind of density function, with a similar graphical interpretation; see Figure 2.3.

2.2.4 The cumulative hazard function

The *cumulative hazard function* is defined as the integral of the hazard function,

$$H(t) = \int_0^t h(s)ds, \quad t \geq 0.$$

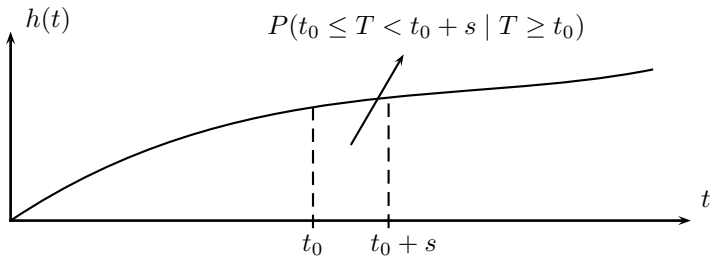


FIGURE 2.3
Interpretation of the hazard function h.

That is, an intuitive interpretation is that the cumulative hazard function successively accumulates the instant risks.

The cumulative hazard function is important because it is fairly easy to estimate nonparametrically (i.e., without any restrictions), in contrast to the hazard and density functions.

Example 8 *The exponential distribution*

Perhaps the simplest continuous life length distribution is the *exponential distribution*. It is simple because its hazard function is constant:

$$h(t) = \lambda, \quad \lambda > 0,\ t \geq 0.$$

From this, it is easy to calculate the other functions that characterize the exponential distribution. The cumulative hazards function is

$$H(t) = \lambda t, \quad \lambda > 0,\ t \geq 0,$$

the survival function is

$$S(t) = e^{-\lambda t}, \quad \lambda > 0,\ t \geq 0,$$

and the density function is

$$f(t) = \lambda e^{-\lambda t}, \quad \lambda > 0,\ t \geq 0.$$

The property of constant hazard implies *no aging*. This is not a realistic property for human mortality, but, as we will see, a useful benchmark, and a useful model for modelling mortality over short time intervals (*piece-wise constant hazard*). The exponential distribution is described in detail in Appendix B.□

2.3 Discrete Time Models

So far we have assumed, implicitly or explicitly, that time is continuous. We will now introduce discrete time survival models, and the reason is two-fold: (i) Even if data are generated from truly continuous time models, nonparametric estimation of these models will, as will be shown later, give rise to estimators corresponding to a discrete time model. This is an inherent property of nonparametric maximum likelihood estimators. Thus, in order to study the properties of these estimators, we need some knowledge of discrete time models. (ii) Data are discrete, usually through grouping. For instance, life lengths may be measured in full years, introducing *tied data*.

It is important to realize that in practice all data are discrete. For instance, it is impossible to measure time with infinite precision. Therefore, all data are more or less rounded. If data are so much rounded that the result is heavily tied data, true discrete-data models are called for.

Discrete time models will now be introduced. Let R be a discrete random variable with

- support $(r_1, r_2, \ldots, r_k)$ (positive real numbers, usually $1, 2, \ldots$ or $0, 1, 2, \ldots$),
- probability mass function

$$p_i = P(R = r_i), \quad i = 1, \ldots, k,$$

with $p_i > 0, \quad i = 1, \ldots, k$ and $\sum_{i=1}^{k} p_i = 1$.

Then

$$F(t) = \sum_{i:r_i \leq t} p_i, \quad x - \infty < t < \infty,$$

is the cumulative distribution function, and

$$S(t) = \sum_{i:r_i \geq x} p_i, \quad x - \infty < t < \infty,$$

is the survival function.

The discrete time hazard function is defined as

$$h_i = P(R = r_i \mid R \geq r_i) = \frac{p_i}{\sum_{j=i}^{k} p_j}, \quad i = 1, \ldots, k.$$

Note that here, the hazard at any given time point is a *conditional probability*, so it must always be bounded to lie between zero and one. In the continuous case, on the other hand, the hazard function may take any positive value. Further note that if, like here, the support is finite, the last "hazard atom" is always equal to one (having lived to the "last station," one is bound to die).

The system (2.3) of equations has a unique solution, easily found by recursion:

$$p_i = h_i \prod_{j=1}^{i-1}(1-h_j), \quad i = 1, \ldots, k. \tag{2.2}$$

From this, we get the discrete time survival function at each support point as

$$S(r_i) = \sum_{j=i}^{k} p_j = \prod_{j=1}^{i-1}(1-h_j), \quad i = 1, \ldots, k,$$

and the general definition

$$S(t) = \prod_{j:r_j<t} (1-h_j)\,, \quad t \geq 0 \tag{2.3}$$

It is easily seen that S is decreasing, $S(0) = 1$, and $S(\infty) = 0$, as it should be.

Example 9 *The geometric distribution*

The *geometric* distribution has support on $\{1, 2, \ldots\}$ (another version also includes zero in the support; this is the case for the one in **R**), and the hazard function h is constant:

$$h_i = h, \quad 0 < h < 1,\ i = 1, 2, \ldots.$$

Thus, the geometric distribution is the discrete analogue to the exponential distribution in that it implies no aging. The probability mass function is, from (2.3),

$$p_i = h(1-h)^{i-1}, i = 1, 2, \ldots,$$

and the survival function becomes, from (2.3),

$$S(t) = (1-h)^{[t]}, \quad t > 0,$$

where $[t]$ denotes the largest integer smaller than or equal to t (rounding downwards). In Figure 2.4 the four functions for a geometric distribution with $h = 0.25$ are plotted. □

2.4 Nonparametric Estimators

As an introductory example, look at an extremely simple data set: 4, 2*, 6, 1, 3* (starred observations are right censored; in the figure, deaths are marked with +, censored observations with a small circle); see Figure 2.5.

How should the survival function be estimated? The answer is that we take

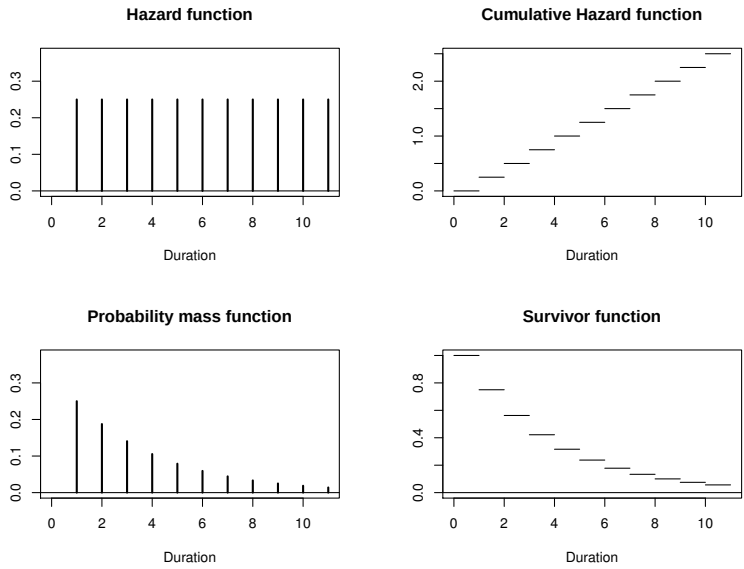


FIGURE 2.4
The geometric distribution with success probability 0.25.

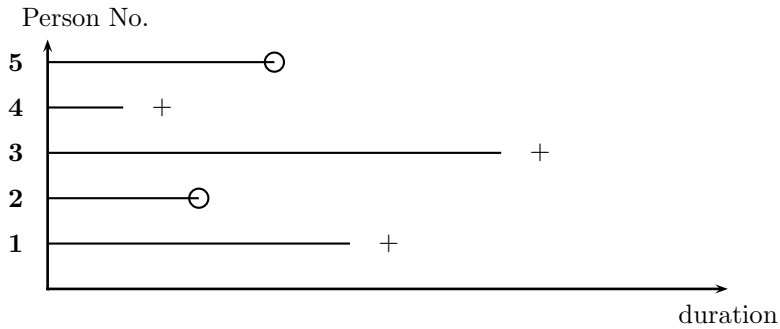


FIGURE 2.5
A simple survival data set.

it in steps. First, the hazard "atoms" are estimated. It is done nonparametrically, and the result as such is not very useful. Its potential lies in that it is used as the building block in constructing estimates of the cumulative hazards and survival functions.

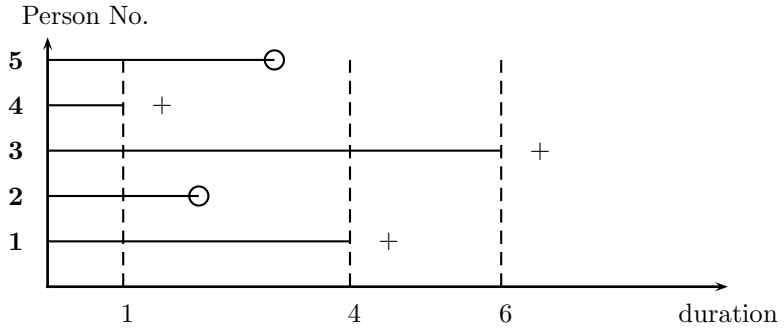


FIGURE 2.6
Preliminaries for estimating the hazard function.

2.4.1 The Hazard Atoms

In Figure 2.6, the observed event times in Figure 2.5 are marked, the vertical dashed lines at durations 1, 4, and 6, respectively. In the estimation of the hazard atoms, the concept of *risk set* is of vital importance.

The risk set $R(t)$ at duration t, $t > 0$ is defined mathematically as

$$R(t) = \{\text{all individuals under observation at } t-\}, \tag{2.4}$$

or in words, the risk set at time t consists of all individuals present and under observation just prior to t. The reason we do not say "present at time t" is that it is vital to include those individuals who have an event or are right censored at the exact time t. In our example, the risk sets $R(1)$, $R(4)$, and $R(6)$ are the interesting ones. They are:

$$\begin{aligned} R(1) &= \{1, 2, 3, 4, 5\} \\ R(4) &= \{1, 3\} \\ R(6) &= \{3\} \end{aligned}$$

The estimation of the hazard atoms is simple. First, we assume that the probability of an event at times where no event is observed, is zero. Then, at times where events do occur, we count the number of events and divides that number by the size of the corresponding risk set. The result is (see also Figure 2.7)

$$\begin{aligned} \hat{h}(1) &= \frac{1}{5} = 0.2 \\ \hat{h}(4) &= \frac{1}{2} = 0.5 \\ \hat{h}(6) &= \frac{1}{1} = 1 \end{aligned}$$

As is evident from Figure 2.7, the estimated hazard atoms will be too irregular to be of practical use; they need *smoothing*. The simplest way of smoothing

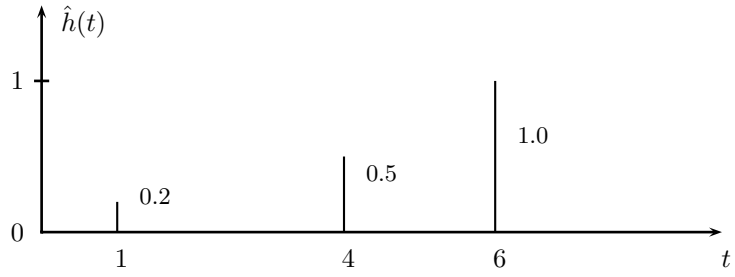


FIGURE 2.7
Nonparametric estimation of the hazard function.

them is to calculate the cumulative sums, which leads to the *Nelson–Aalen* estimator of the cumulative hazards function; see the next section. There are more direct smoothing techniques to get reasonable estimators of the hazard function itself, for example, *kernel estimators*, but they will not be discussed here. See, for example, (Silverman 1986) for a general introduction to kernel smoothing.

2.4.2 The Nelson–Aalen Estimator

From the theoretical relation, we immediately get

$$\hat{H}(t) = \sum_{s \le t} \hat{h}(s), \quad t \ge 0$$

which is the *Nelson–Aalen* estimator (Nelson 1972, Aalen 1978); see Figure 2.8. The sizes of the jumps are equal to the heights of the "spikes" in Figure 2.7.

2.4.3 The Kaplan–Meier Estimator.

From the theoretical relation (2.3), we get

$$\hat{S}(t) = \prod_{s<t} \big(1 - \hat{h}(s)\big), \quad t \ge 0; \tag{2.5}$$

see also Figure 2.9. Equation (2.5) may be given a heuristic interpretation: In order to survive time t, one must survive *all* "spikes" (or shocks) that come before time t. The multiplication principle for conditional probabilities then gives equation (2.5).

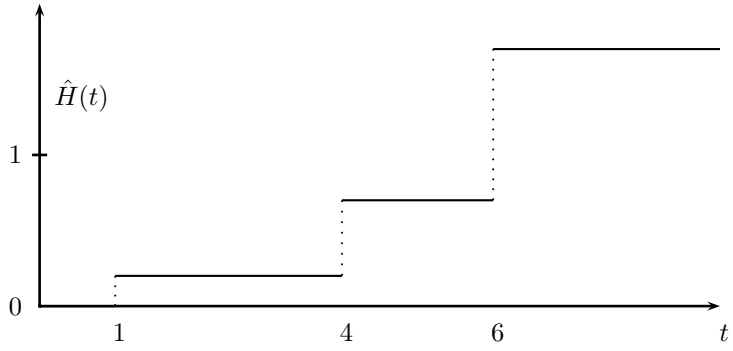


FIGURE 2.8
The Nelson–Aalen estimator.

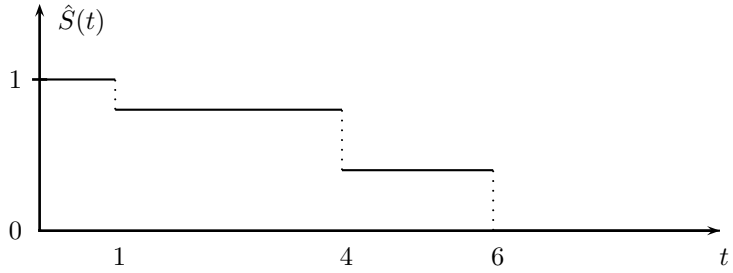


FIGURE 2.9
The Kaplan–Meier estimator

2.5 Doing it in R

For the male mortality data set `mort`,

```
> require(eha)
> data(mort)
> head(mort)
```

```
  id  enter   exit event birthdate   ses
1  1  0.000 20.000     0  1800.010 upper
2  2  3.478 17.562     1  1800.015 lower
3  3  0.000 13.463     0  1800.031 upper
4  3 13.463 20.000     0  1800.031 lower
5  4  0.000 20.000     0  1800.064 lower
6  5  0.000  0.089     0  1800.084 lower
```

an indicator for *death* is introduced, called `event`. The value 1 indicates that the corresponding life time is fully observed, while the value 0 indicates a right-censored lifetime. Another common coding scheme is `TRUE` and `FALSE`, respectively.

2.5.1 Nonparametric Estimation

In the **R** package `eha`, the `plot` function can be used to plot both Nelson–Aalen curves and Kaplan–Meier curves. The code for the Nelson–Aalen and Kaplan–Meier plots is

```
> par(mfrow = c(1, 2))# Two plots, "one row, two columns".
> with(mort, plot(Surv(enter, exit, event), fn = "cum"))
> with(mort, plot(Surv(enter, exit, event), fn = "surv"))
```

and the result is seen in Figure 2.10 The package `eha` must be loaded for this

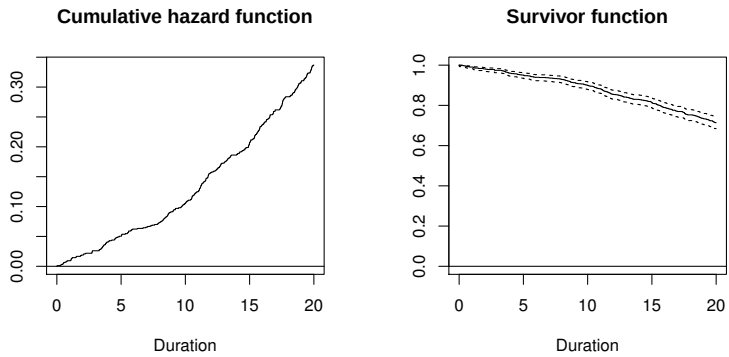


FIGURE 2.10
Nelson–Aalen plot (left panel) and Kaplan–Meier plot (right panel), male mortality data. The dashed lines in the Survival function plot are 95% confidence limits.

to work. Note the use of the function *with*; it tells the `plot` function that it should get its data (`enter, exit, event`) from the data frame `mort`. The function `Surv` from the package `survival` creates a "survival object," which

is used in many places. It is, for instance, *the response* in all functions that perform regression analysis on survival data.

Note that the "Duration" in Figure 2.10 is duration (in years) since the day each man became 20 years of age. They are followed until death or age 40, whichever comes first. The right hand figure shows that approximately 25% of the men alive at age 20 died before they became 40.

If the Kaplan–Meier estimator is wanted as numbers, use `survfit` in combination with `coxreg` or `coxph`. Since the output will have approximately as many rows as there are distinct event times, we illustrate it by taking a (small) random sample from `mort`.

```
> indx <- sample(NROW(mort), size = 50, replace = FALSE)
> rsa <- mort[indx, ]
> fit <- coxph(Surv(enter, exit, event) ~ 1, data = rsa)
> s.fit <- survfit(fit)
> options(width=70)
> summary(s.fit)
Call: survfit(formula = fit)
   time n.risk n.event survival std.err lower 95% CI upper 95% CI
  0.526     46       1    0.978  0.0213        0.938        1.000
  3.615     39       1    0.954  0.0321        0.893        1.000
  4.036     38       1    0.929  0.0397        0.854        1.000
  5.334     36       1    0.904  0.0460        0.818        0.998
  8.653     30       1    0.874  0.0532        0.776        0.985
  9.690     30       1    0.845  0.0587        0.738        0.968
 10.139     29       1    0.817  0.0633        0.702        0.951
 10.462     28       1    0.788  0.0672        0.667        0.931
 13.515     29       1    0.761  0.0701        0.636        0.912
 13.584     28       1    0.735  0.0725        0.605        0.891
```

The functions `coxreg` and `coxph` both fit *Cox regression* models, which is the topic of the next chapter. Here we fit a regression model with just a constant term and no covariates. This results in estimation of the "baseline hazards" which `survfit` transforms to elements of the Kaplan–Meier estimator seen above. Note also that, by default, a 95% confidence band is produced around the estimator.

You can also plot the output of the fit (`s.fit`) by simply typing

```
> plot(s.fit)
```

The result is not shown here. It will look similar to the right-hand panel of Figure 2.10.

2.5.2 Parametric Estimation

It is also possible to fit a parametric model to data with the aid of the function `phreg` (it also works with the function `aftreg`). Just fit a "parametric proportional hazards model with no covariates." Note that the default distribution

```
> par(mfrow = c(1, 2))
> fit.w <- phreg(Surv(enter, exit, event) ~ 1, data = mort)
> plot(fit.w, fn = c("cum"))
> plot(fit.w, fn = c("sur"))
```

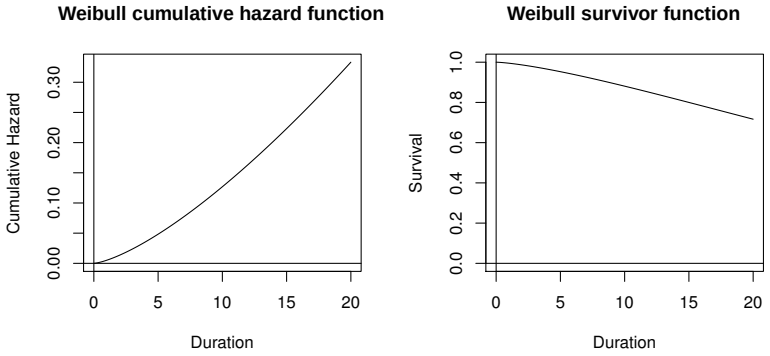


FIGURE 2.11
Weibull fit to the male mortality data.

in `phreg` is the *Weibull* distribution, which means that if no distribution is specified (through the argument `dist`), then the Weibull distribution is assumed.

We can also show the parameter estimates. For the interpretations of these, see Appendix B, the *Weibull* distribution.

```
> fit.w
Call:
phreg(formula = Surv(enter, exit, event) ~ 1, data = mort)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
log(scale)                    3.785    44.015     0.066     0.000
log(shape)                    0.332     1.393     0.058     0.000

Events                    276
Total time at risk         17038
Max. log. likelihood      -1399.3
```

More about this will be discussed in Chapter 6.

3

Cox Regression

3.1 Introduction

We move slowly from the two-sample and k-sample cases, where the *logrank test* is introduced, to the general regression situation. There the logrank test is generalized to *Cox regression* (Cox 1972). The fundamental concept of *proportional hazards* is introduced.

3.2 Proportional Hazards

The property of *proportional hazards* is fundamental in Cox regression. It is in fact the essence of Cox's simple yet ingenious idea. The definition is as follows.

Definition 1 *Proportional hazards*

If $h_1(t)$ and $h_0(t)$ are hazard functions from two separate distributions, we say that they are *proportional* if

$$h_1(t) = \psi h_0(t), \quad \text{for all } t \geq 0, \tag{3.1}$$

for some positive constant ψ and *all* $t \geq 0$. Further, if (3.1) holds, then the same property holds for the corresponding *cumulative hazard functions* $H_1(t)$ and $H_0(t)$.

$$H_1(t) = \psi H_0(t), \quad \text{for all } t \geq 0, \tag{3.2}$$

with the same proportionality constant ψ as in (3.1). □

Strictly speaking, the second part of this definition follows easily from the first (and vice versa), so more correct would be to state one part as a definition and the other as a corollary. The important part of this definition is "*for all* $t \geq 0$", and that the constant ψ *does not depend on* t. Think of the hazard functions as age-specific mortality for two groups, for example, women and men. It is "well known" that women have lower mortality than men in all ages. It would therefore be reasonable to assume proportional hazards in that case. It would mean that the female *relative* advantage is equally large in

all ages. See Example 4.5 for a demonstration of this example of *proportional hazards*.

It must be emphasized that this is an assumption that always must be carefully checked. In many other situations, it would not be reasonable to assume proportional hazards. If in doubt, check data by plotting the Nelson–Aalen estimates for each group in the same figure.

See Figure 3.1 for an example with two *Weibull* hazard functions and the proportionality constant $\psi = 2$. The proportionality is often difficult to judge

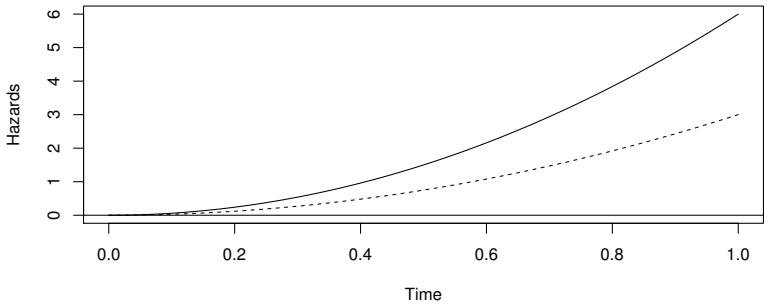


FIGURE 3.1
Two hazard functions that are proportional. The proportionality constant is $\psi = 2$.

by eye, so in order make it easier to see, the plot can be made on a *log scale*, see Figure 3.2. Note that both dimensions are on a log scale. This type of plot, constructed from empirical data, is called a *Weibull plot* in reliability applications: If the lines are straight lines, then data are well fitted by a Weibull distribution. Additionally, if the the slope of the line is 1 (45 degrees), then an exponential model fits well.

To summarize Figure 3.2: (i) The hazard functions are proportional because on the log-log scale, the vertical distance is constant, (ii) Both hazard functions represent a Weibull distribution, because both lines are straight lines, and (iii) neither represents an exponential distribution, because the slopes are *not* one. This latter fact may be difficult to see because of the different scales on the axes.

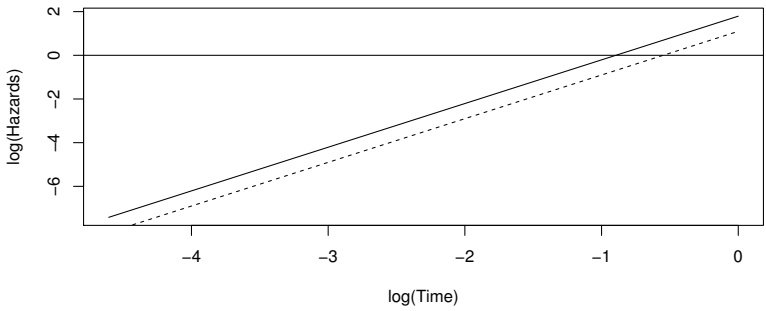


FIGURE 3.2
Two hazard functions that are proportional are on a constant distance from each other on a log-log scale. The proportionality constant is $\psi = 2$, corresponding to a vertical distance of $\log(2) \approx 0.693$.

3.3 The Log-Rank Test

The *log-rank test* is a k-sample test of equality of survival functions. We first look at the two-sample case, that is, $k = 2$.

3.3.1 Two Samples

Suppose that we have the small data set illustrated in Figure 3.3. There are two samples, the `letters` (A, B, C, D, E) and the `numbers` (1, 2, 3, 4, 5). We are interested in investigating whether `letters` and `numbers` have the same survival chances or not. Therefore, the hypothesis

$$H_0 : \text{No difference in survival between } \texttt{numbers} \text{ and } \texttt{letters}$$

is formulated. In order to test H_0, we make five tables, one for each observed *event time* (see Table 3.1). Let us look at the first table at failure time $t_{(1)}$, that is, the first three rows in Table 3.1, from the viewpoint of the `numbers`.

- The observed number of deaths among `numbers`: 1.
- The expected number of deaths among `numbers`: $2 \times 5/10 = 1$.

The *expected* number is calculated under H_0, that is, as if there is no difference between `letters` and `numbers`. It is further assumed that the two margins are given (fixed). Then, given two deaths in total and five out of ten observations

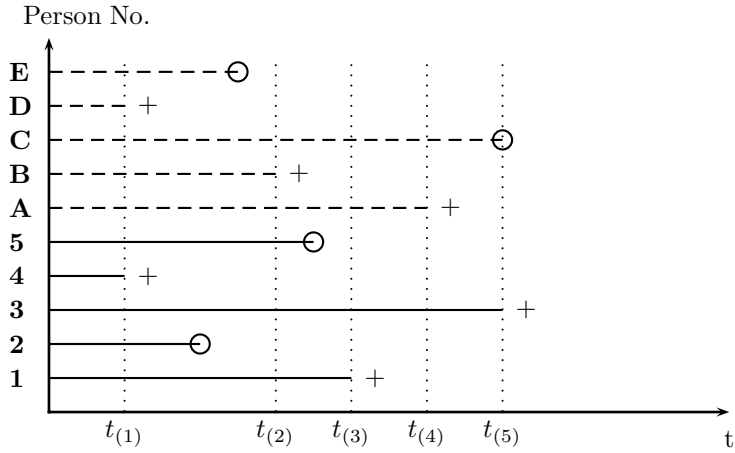


FIGURE 3.3
Two-sample data, the `letters` (dashed) and the `numbers` (solid). Circles denote censored observations, plusses uncensored.

TABLE 3.1
Five 2×2 tables. Each table reports the situation at one observed failure time point, i.e., at $t_{(1)}, \ldots, t_{(5)}$

Time	Group	Deaths	Survivals	Total
$t_{(1)}$	`numbers`	1	4	5
	`letters`	1	4	5
	Total	2	8	10
$t_{(2)}$	`numbers`	0	3	3
	`letters`	1	2	3
	Total	1	5	6
$t_{(3)}$	`numbers`	1	1	2
	`letters`	0	2	2
	Total	1	3	4
$t_{(4)}$	`numbers`	0	1	1
	`letters`	1	1	2
	Total	1	2	3
$t_{(5)}$	`numbers`	1	0	1
	`letters`	0	1	1
	Total	1	1	2

are from the group `numbers`, the expected number of deaths is calculated as above.

This procedure is repeated for each of the five tables, and the results are summarized in Table 3.2. Finally, the observed test statistic T is calculated as

TABLE 3.2
Observed and expected number of deaths for `numbers` at the five observed event times

Time	Observed	Expected	Difference	Variance
$t_{(1)}$	1	1.0	0.0	0.44
$t_{(2)}$	0	0.5	-0.5	0.25
$t_{(3)}$	1	0.5	0.5	0.25
$t_{(4)}$	0	0.3	-0.3	0.22
$t_{(5)}$	1	0.5	0.5	0.25
Sum	3	2.8	0.2	1.41

$$T = \frac{0.2^2}{1.41} \approx 0.028$$

Under the null hypothesis, this is an observed value from a $\chi^2(1)$ distribution, and H_0 should be rejected for *large* values of T. Using a *level of significance* of 5%, the cutting point for the value of T is 3.84, far from our observed value of 1.41. The conclusion is therefore that there is no (statistically significant) difference in survival chances between `letters` and `numbers`. Note, however, that this result depends on asymptotic (large sample) properties, and in this toy example, these properties are not valid.

For more detail about the underlying theory, see Appendix A.

In **R**, the log-rank test is performed by the `coxph` function in the package survival (there are other options). With the data in our example, see Table 3.3, we get:

```
> library(survival)
> ex.data <- read.table("exlr.dat", header = TRUE)
```

TABLE 3.3
Data in a simple example

group	time	event
numbers	4.0	1
numbers	2.0	0
numbers	6.0	1
numbers	1.0	1
numbers	3.5	0
letters	5.0	1
letters	3.0	1
letters	6.0	0
letters	1.0	1
letters	2.5	0

```
> fit <- coxph(Surv(time, event) ~ group, data = ex.data)
> summary(fit)
Call:
coxph(formula = Surv(time, event) ~ group, data = ex.data)
  n= 10, number of events= 6

               coef exp(coef) se(coef)     z Pr(>|z|)
groupnumbers 0.1130    1.1196   0.8231 0.137    0.891

             exp(coef) exp(-coef) lower .95 upper .95
groupnumbers      1.12     0.8931    0.2231     5.619

Concordance= 0.481  (se = 0.15 )
Rsquare= 0.002   (max possible= 0.85 )
Likelihood ratio test= 0.02  on 1 df,   p=0.8908
Wald test            = 0.02  on 1 df,   p=0.8908
Score (logrank) test = 0.02  on 1 df,   p=0.8907
```

Obviously, there is a lot of information here. We have, in fact, performed a *Cox regression* ahead of schedule! This also tells us that the simplest way of performing a logrank test in **R** is to run a Cox regression. The result of the logrank test is displayed on the last line of output. The p-value is 0.891; that is, there is no significant difference between groups whatsoever. This is not very surprising, since we are using a toy example with almost no information, only six events in total.

Let us now look at a real data example, the old age mortality data set `oldmort` in `eha`.

```
> require(eha) # Loads 'survival' as well.
> data(oldmort)
> fit <- coxph(Surv(enter, exit, event) ~ sex, data = oldmort)
> summary(fit)
Call:
coxph(formula = Surv(enter, exit, event) ~ sex, data = oldmort)
  n= 6495, number of events= 1971

              coef exp(coef) se(coef)      z Pr(>|z|)
sexfemale -0.1930    0.8245   0.0456 -4.232 2.32e-05

           exp(coef) exp(-coef) lower .95 upper .95
sexfemale    0.8245      1.213     0.754    0.9016

Concordance= 0.532  (se = 0.007 )
Rsquare= 0.003   (max possible= 0.985 )
Likelihood ratio test= 17.73  on 1 df,   p=2.545e-05
Wald test            = 17.91  on 1 df,   p=2.32e-05
Score (logrank) test = 17.96  on 1 df,   p=2.254e-05
```

As in the trivial example, the last line of the output gives the p-value of the logrank test, $p = 0.0000225$ (in "exponential notation" in the output). This is

very small, and we can conclude that the mortality difference between women and men is highly statistically significant. But how large is it, that is, what is the value of the proportionality constant? The answer to that question is found five rows up, where `sexfemale` reports the value 0.8245 for `exp(Coef)`. The interpretation of this is that female mortality is about 82.45 per cent of the male mortality, that is, 17.55 per cent lower. This is for ages above 60, because this data set contains only information in that range.

Remember that this result depends on the proportional hazards assumption. We can graphically check it as follows.

```
> ##plot(Surv(oldmort$enter, oldmort$exit, oldmort$event),
>   ##   strata = oldmort$sex, fn = "cum")
> with(oldmort, plot(Surv(enter, exit, event), strata = sex))
```

First, note that the out-commented (##) version works equally well, but it is more tedious to type, and also harder to read. The `with` function is very handy and will be used often throughout the book. Note also that the grouping factor (`sex`) is given through the argument `strata`. The result is shown in Figure 3.4. The proportionality assumption seems to be a good description from 60 to 85–

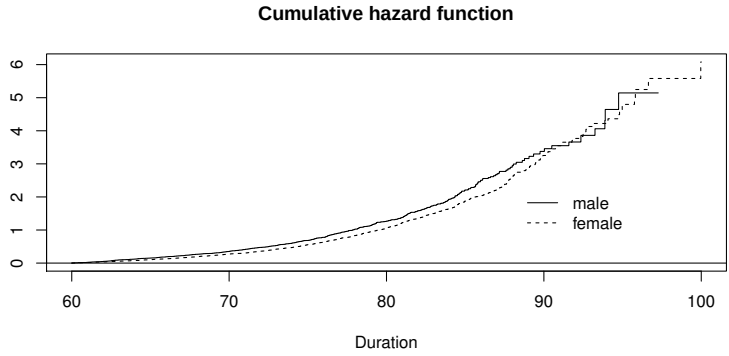


FIGURE 3.4
Old age mortality, women vs. men, cumulative hazards.

90 years of age, but it seems more doubtful in the very high ages. One reason for this may be that the high-age estimates are based on few observations (most of the individuals in the sample died earlier), so random fluctuations have a large impact in the high ages.

3.3.2 Several Samples

The result for the two-sample case is easily extended to the k-sample case. Instead of one 2×2 table per observed event time, we get one $k \times 2$ table per observed event time and we have to calculate expected and observed numbers

of events for $(k-1)$ groups at each failure time. The resulting test statistic will have $(k-1)$ degrees of freedom and still be approximately χ^2 distributed. This is illustrated with the same data set, `oldmort`, as above, but with the covariate *birthplace*, which is a `factor` with three levels, instead of *sex*.

```
> require(eha) # Loads 'survival' as well.
> data(oldmort)
> fit <- coxph(Surv(enter, exit, event) ~ birthplace,
               data = oldmort)
> summary(fit)

Call:
coxph(formula = Surv(enter, exit, event) ~ birthplace,
      data = oldmort)
  n= 6495, number of events= 1971

                    coef exp(coef) se(coef)     z Pr(>|z|)
birthplaceregion 0.0594    1.0612   0.0552 1.076   0.2819
birthplaceremote 0.1008    1.1060   0.0595 1.694   0.0903

                 exp(coef) exp(-coef) lower .95 upper .95
birthplaceregion     1.061     0.9423    0.9524     1.182
birthplaceremote     1.106     0.9041    0.9843     1.243

Concordance= 0.507  (se = 0.007 )
Rsquare= 0.001   (max possible= 0.985 )
Likelihood ratio test= 3.25  on 2 df,   p=0.1968
Wald test            = 3.28  on 2 df,   p=0.1941
Score (logrank) test = 3.28  on 2 df,   p=0.1939
```

The degree of freedom for the *score test* is now 2, equal to the number of levels in *birthplace* (`parish`, `region`, `remote`) minus one. The birthplace thus does not seem to have any great impact on old age mortality. It is, however, recommended to check the proportionality assumption graphically; see Figure 3.5.

```
> with(oldmort, plot(Surv(enter, exit, event), strata = birthplace))
```

There is obviously nothing that indicates nonproportionality in this case either.

We do not go deeper into this matter here, mainly because the logrank test generally is a special case of Cox regression, which will be described in detail later in this chapter. There are, however, two special cases of the logrank test, the *weighted* and *stratified* versions, that may be of interest on their own. The interested reader may consult the textbooks by Kalbfleisch & Prentice (2002, pp. 22–23) or Collett (2003, pp. 49–53).

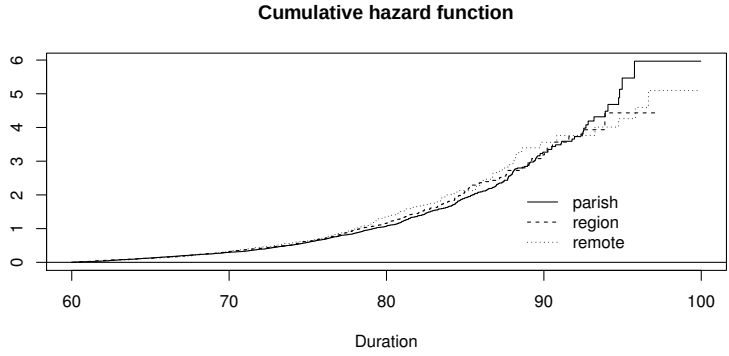


FIGURE 3.5
Old age mortality by birthplace, cumulative hazards.

3.4 Proportional Hazards in Continuous Time

The definition in Section 3.2 is valid for models in continuous time. Figure 3.6 shows the relationships between the cumulative hazards functions, the density functions, and the survival functions when the hazard functions are proportional. Note that the cumulative hazards functions are proportional by implication, with the same proportionality constant ($\psi = 2$ in this case). On the other hand, for the density and survival functions, proportionality does not hold; it is, in fact, theoretically impossible except in the trivial case that the proportionality constant is unity.

3.4.1 Proportional Hazards, Two Groups

The definition of proportionality, given in equation (3.1), can equivalently be written as

$$h_x(t) = \psi^x h_0(t), \quad t > 0,\ x = 0, 1,\ \psi > 0. \tag{3.3}$$

It is easy to see that this "trick" is equivalent to (3.1): When $x = 0$ it simply says that $h_0(t) = h_0(t)$, and when $x = 1$ it says that $h_1(t) = \psi h_0(t)$. Since $\psi > 0$, we can calculate $\beta = \log(\psi)$, and rewrite (3.3) as

$$h_x(t) = e^{\beta x} h_0(t), \quad t > 0;\ x = 0, 1;\ -\infty < \beta < \infty,$$

or with a slight change in notation,

$$h(t; x) = e^{\beta x} h_0(t), \quad t > 0;\ x = 0, 1;\ -\infty < \beta < \infty. \tag{3.4}$$

The sole idea of this rewriting is to pave the way for the introduction of

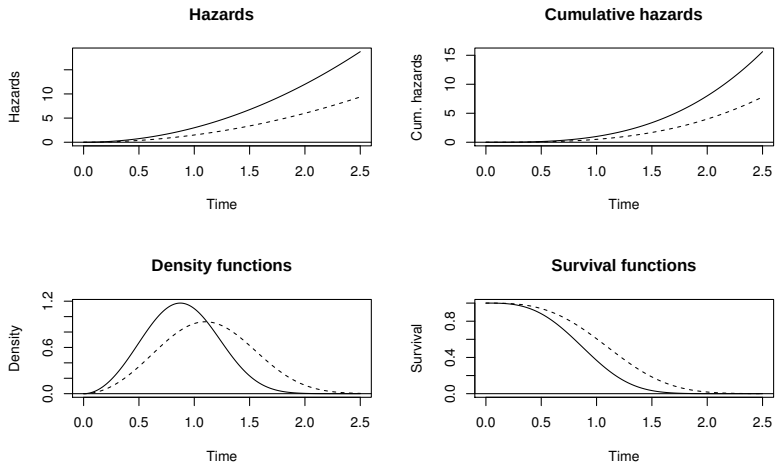


FIGURE 3.6
The effect of proportional hazards on the density and survival functions.

Cox's regression model (Cox 1972), which in its elementary form is a proportional hazards model. In fact, we can already interpret equation (3.4) as a Cox regression model with an explanatory variable x with corresponding regression coefficient (to be estimated from data) β. The covariate x is still only a dichotomous variate, but we will now show how it is possible to generalize this to a situation with explanatory variables of any form. The first step is to go from the two-sample situation to a k-sample one.

3.4.2 Proportional Hazards, More Than Two Groups

Can we generalize to $(k+1)$ groups, $k \geq 2$? Yes, by expanding the procedure in the previous subsection:

$$\begin{aligned} h_0(t) &\sim \text{group 0} \\ h_1(t) &\sim \text{group 1} \\ \ldots &\quad \ldots \\ h_k(t) &\sim \text{group k} \end{aligned}$$

The underlying model is: $h_j(t) = \psi_j h_0(t), \quad t \geq 0,\ j = 1, 2, \ldots, k$. That is, with $(k+1)$ groups, we need k proportionality constants $\psi_1, \ldots, \psi_k$ in order to define proportional hazards. Note also that in this formulation (there are others), one group is "marked" as a *reference group*, that is, to this group there is no proportionality constant attached. All relations are relative to the reference group. Note also that it essentially doesn't matter which group is

chosen as the reference. This choice doesn't change the model itself, only its representation.

With $(k+1)$ groups, we need k indicators. Let

$$\mathbf{x} = (x_1, x_2, \ldots, x_k).$$

Then

$$\begin{aligned}
\mathbf{x} = (0,0,\ldots,0) &\Rightarrow \text{in group 0}\\
\mathbf{x} = (1,0,\ldots,0) &\Rightarrow \text{in group 1}\\
\mathbf{x} = (0,1,\ldots,0) &\Rightarrow \text{in group 2}\\
\cdots &\quad \cdots\\
\mathbf{x} = (0,0,\ldots,1) &\Rightarrow \text{in group k}
\end{aligned}$$

and

$$h(t;\mathbf{x}) = h_0(t)\prod_{\ell=1}^{k}\psi_\ell^{x_\ell} = \begin{cases} h_0(t), & \mathbf{x} = (0,0,\ldots,0)\\ h_0(t)\psi_j, & x_j = 1, \quad j = 1,\ldots k \end{cases}$$

With $\psi_j = e^{\beta_j}$, $j = 1,\ldots,k$, we get

$$h(t;\mathbf{x}) = h_0(t)e^{x_1\beta_1 + x_2\beta_2 + \cdots + x_k\beta_k} = h_0(t)e^{\mathbf{x}\boldsymbol{\beta}}, \tag{3.5}$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_k)$.

3.4.3 The General Proportional Hazards Regression Model

We may now generalize (3.5) by letting the components of $\mathbf{x}_i$ take *any value*. Let data and model take the following form:

Data: $(t_{i0}, t_i, d_i, \mathbf{x}_i)$, $i = 1,\ldots,n$, where

- t_{i0} is the *left truncation time point*. If $y_{i0} = 0$ for all i, then this variable may be omitted.
- t_i is the *end time point*.
- d_i is the *"event indicator"* (1 or *TRUE* if event, else 0 or *FALSE*).
- $\mathbf{x}_i$ is a vector of *explanatory variables*.

Model:

$$h(t;\mathbf{x}_i) = h_0(t)e^{\mathbf{x}_i\boldsymbol{\beta}}, \quad t > 0. \tag{3.6}$$

This is a *regression model* where

- the *response variable* is (t_0, t, d) (we will call it a *survival object*)
- and the *explanatory variable* is $\mathbf{x}$, possibly (often) vector valued.

In equation (3.6) there are two components to estimate, the regression coefficients $\boldsymbol{\beta}$, and the *baseline hazard function* $h_0(t)$, $t > 0$. For the former task, the *partial likelihood* (Cox 1975) is used. See Appendix A for a brief summary.

3.5 Estimation of the Baseline Hazard

The usual estimator (continuous time) of the baseline cumulative hazard function is

$$\hat{H}_0(t) = \sum_{j:t_j \le t} \frac{d_j}{\sum_{\ell \in R_j} e^{\mathbf{x}_\ell \hat{\boldsymbol{\beta}}}}, \tag{3.7}$$

where d_j is the number of events at t_j. Note that if $\hat{\boldsymbol{\beta}} = 0$, this reduces to

$$\hat{H}_0(t) = \sum_{j:t_j \le t} \frac{d_j}{n_j}, \tag{3.8}$$

the Nelson–Aalen estimator. In (3.8), n_j is the size of R_j.

In the **R** package `eha`, the baseline hazard is estimated at the means of the covariates (or, more precisely, at the means of the columns of the design matrix; this makes a difference for factors).

$$\hat{H}_0(t) = \sum_{j:t_j \le t} \frac{d_j}{\sum_{\ell \in R_j} e^{(\mathbf{x}_\ell - \bar{\mathbf{x}})\hat{\boldsymbol{\beta}}}}, \tag{3.9}$$

In order to calculate the cumulative hazards function for an individual with a specific covariate vector $\mathbf{x}$, use the formula

$$\hat{H}(t; \mathbf{x}) = \hat{H}_0(t) e^{(\mathbf{x} - \bar{\mathbf{x}})\hat{\boldsymbol{\beta}}}. \tag{3.10}$$

The corresponding survival functions may be estimated by the relation

$$\hat{S}(t; \mathbf{x}) = \exp\bigl(-\hat{H}(t; \mathbf{x})\bigr)$$

It is also possible to use the terms in the sum (3.7) to build an estimator analogous to the Kaplan–Meier estimator (3.8). In practice, there is no big difference between the two methods.

3.6 Explanatory Variables

Explanatory variables, or *covariates*, may be of essentially two different types, *continuous* and *discrete*. The discrete type usually takes only a finite number of distinct values and is often called a `factor`, for example, in **R**. A special case of a factor is one that takes only two distinct values, say 0 and 1. Such a factor is called an *indicator*, because we can let the value 1 indicate the presence of a certain property and 0 denote its absence. To summarize, there is

Covariate: taking values in an *interval* (e.g., *age*, *blood pressure*).

Factor: taking a *finite* number of values (e.g., *civil status*, *occupation*).

Indicator: a factor taking *two* values (e.g., *gender*).

3.6.1 Continuous Covariates

We use the qualifier *continuous* to stress that factors are excluded, because often the term *covariate* is used as a synonym for *explanatory variable*.

Values taken by a continuous covariate are *ordered*. The *effect* on the response is by model definition ordered in the *same* or *reverse* order. On the other hand, values taken by a factor are *unordered* (but may be defined as ordered in **R**).

3.6.2 Factor Covariates

An explanatory variable that can take only a finite (usually small) number of distinct values is called a *categorical variable*. In **R** language, it is called a *factor*. Examples of such variables are `gender`, `socioeconomic status`, and `birth place`. Students of statistics have long been taught to create *dummy variables* in such situations, in the following way:

- Given a categorical variable F with $(k+1)$ levels $(f_0, f_1, f_2, \ldots f_k)$ $(k+1$ levels),
- Create k *indicator* ("dummy") variables $(I_1, I_2, \ldots I_k)$.

F	I_1	I_2	I_3	$\cdots$	I_{k-1}	I_k	Comment
f_0	0	0	0	$\cdots$	0	0	*reference*
f_1	1	0	0	$\cdots$	0	0	
f_2	0	1	0	$\cdots$	0	0	
f_3	0	0	1	$\cdots$	0	0	
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	
f_{k-1}	0	0	0	$\cdots$	1	0	
f_k	0	0	0	$\cdots$	0	1	

The level f_0 is the reference category, characterized by that all indicator variables are `zero` for an individual with this value. Generally, for the level, $f_i,\ i = 1, \ldots, k$, the indicator variable I_i is one, and the rest are zero. In other words, for a single individual, at most one indicator is one, and the rest are zero.

In **R**, there is no need to explicitly create dummy variables; it is done behind the scene by the functions `factor` and `as.factor`.

Note that a factor with *two* levels, that is, an indicator variable, can always be treated as a continuous covariate, if coded numerically (e.g., 0 and 1).

Example 10 *Infant mortality and mother's age*

Consider a demographic example, the influence of mother's age (a continuous covariate) on infant mortality. It is considered wellknown that a *young* mother means high risk for the infant, and also that *old* mother means high risk, compared to *"in-between-aged"* mothers. So the risk order is not the same (or reverse) as the age order.

One solution to this problem is to *factorize*: Let, for instance,

$$\textbf{mother's age} = \begin{cases} \text{low}, & 15 < \textbf{age} \leq 25 \\ \text{middle}, & 25 < \textbf{age} \leq 35 \\ \text{high}, & 35 < \textbf{age} \end{cases} \tag{3.11}$$

In this layout, there will be two parameters measuring the deviation from the reference category, which will be the first category by default.

In **R**, this is easily achieved with the aid of the `cut` function. It works like this:

```
> age <- rnorm(100, 30, 6)
> age.group <- cut(age, c(15, 25, 35, 48))
> summary(age.group)
(15,25] (25,35] (35,48]    NA's
     18      64      17       1
```

Note that the created intervals by default are closed to the right and open to the left. This has consequences for how observations exactly on a boundary are treated; they belong to the lower-valued interval. The argument `right` in the call to `cut` can be used switch this behaviour the other way around.

Note further that values falling below the smallest value (15 in our example) or above the largest value (48) are reported as *missing values* (`NA` in **R** terminology, `Not Available`).

For further information about the use of the `cut` function, see the help page. □

3.7 Interactions

The meaning of *interaction* between two explanatory variables is described by an example, walking through the three possible combinations: two factors, one factor and one continuous covariate, and two covariates.

Example 11 *Mortality in lung cancer*

There are four possible explanatory variables behind survival after a diagnosis of lung cancer in the example:

$$\begin{aligned}
\textbf{smoking habits} &= \begin{cases} \textit{smoker} & \\ \textit{nonsmoker} & (e^{\alpha_1}) \end{cases} \\
\textbf{tumor size} &= \begin{cases} \textit{no tumor} & \\ \textit{small tumor} & \left(e^{\beta_1}\right) \\ \textit{large tumor} & \left(e^{\beta_2}\right) \end{cases} \\
\textbf{age at diagnosis } (x_1) & \qquad (e^{\gamma_1 x_1}) \\
\textbf{tumor size in mm } (x_2) & \qquad (e^{\gamma_2 x_2})
\end{aligned}$$

The two first are factors, and the last two are continuous covariates. We go through the three distinct combinations and illustrate the meaning of interaction in each. □

3.7.1 Two Factors

The concept of interaction is easiest to explain with two factors; see Table 3.4. With no interaction, the model is called *multiplicative*. The reason is that with

TABLE 3.4 The role of interaction, two factors

		Tumor size		
		no tumor	small	large
		No interaction		
Smoking	nonsmoker	1	e^{β_1}	e^{β_2}
habits	smoker	e^{α_1}	$e^{\alpha_1+\beta_1}$	$e^{\alpha_1+\beta_2}$
		With interaction		
Smoking	nonsmoker	1	e^{β_1}	e^{β_2}
habits	smoker	e^{α_1}	$e^{\alpha_1+\beta_1+\theta_{11}}$	$e^{\alpha_1+\beta_2+\theta_{12}}$

no interactions, the (relative) effect of smoking is *the same for each level* of `tumor size`, $\exp(\alpha)$, and vice versa. The model with interaction is sometimes called *saturated*, and the reason is that each combination of the two levels has its own parameter, except for the *reference* cell, to which all other effects are compared.

In this example, two extra parameters, θ_{11} and θ_{12} measure interaction. If both are zero, we get the multiplicative model in the upper part of Table 3.4.

3.7.2 One Factor and one Continuous Covariate

With one covariate and one factor, the interpretation of *no* interaction is that the "slope" (coefficient connected to the covariate) is the same for each level of the factor, it is only the "intercept" that varies. With interaction, the slopes also varies between levels of the factor. In Table 3.5, the parameters θ_{11} and

TABLE 3.5
The role of interaction, one factor and one continuous covariate

	Tumor size		
	no tumor	small	large
	No interaction		
age at diagnosis	$e^{\gamma_1 x_1}$	$e^{\beta_1+\gamma_1 x_1}$	$e^{\beta_2+\gamma_1 x_1}$
	With interaction		
age at diagnosis	$e^{\gamma_1 x_1}$	$e^{\beta_1+(\gamma_1+\theta_{11})x_1}$	$e^{\beta_2+(\gamma_1+\theta_{12})x_1}$

θ_{12} measure the strength and direction of the interaction between the two covariates.

3.7.3 Two Continuous Covariates

With two continuous covariates, the interaction term is constructed through a new covariate which is the product of the two. The main effects have a

TABLE 3.6
The role of interaction, two continuous covariates

	Tumor size in mm
No interaction	
age at diagnosis	$e^{\gamma_1 x_1+\gamma_2 x_2}$
With interaction	
age at diagnosis	$e^{\gamma_1 x_1+\gamma_2 x_2+\theta_{12} x_1 x_2}$

specific interpretation in case of the presence of an interaction; they measure the effect of one covariate when the other covariate is zero. Note that this fact will have severe consequences if zero (0) is not included in the natural range of the covariate(s). For instance, in the examples in this book, *calendar year* (e.g., *birthdate*) is often present as a potential covariate. If that variable is used "as is" in an interaction, its reference value will come from the year 0 (zero)! This is an extreme extrapolation when studying nineteenth century mortality. The way around this is to "center" the covariate by subtracting a value in its natural range.

3.8 Interpretation of Parameter Estimates

In the proportional hazards model the parameter estimates are logarithms of risk ratios relative to the baseline hazard. The precise interpretations of the coefficients for the two types of covariates are discussed. The conclusion is that e^β has a more direct and intuitive interpretation than β itself.

3.8.1 Continuous Covariate

If x is a continuous covariate, and $h(t;x) = h_0(t)e^{\beta x}$, then

$$\frac{h(t;x+1)}{h(t;x)} = \frac{h_0(t)e^{\beta(x+1)}}{h_0(t)e^{\beta x}} = e^\beta.$$

so the risk increases with a factor e^β, when x is increased by one unit. In other words, e^β is a *relative risk* (or a *hazard ratio*, which often is preferred in certain professions).

3.8.2 Factor

For a factor covariate, in the usual coding with a reference category, e^β is the relative risk compared to the *reference* category.

3.9 Proportional Hazards in Discrete Time

In discrete time, the hazard function is, as we saw earlier, a set of (conditional) probabilities, and so its range is restricted to the interval $(0, 1)$. Therefore, the definition of proportional hazards used for continuous time is unpractical; the multiplication of a probability by a constant may easily result in a quantity larger than one.

One way of introducing proportional hazards in discrete time is to regard the discreteness as a result of grouping true continuous time data, for which the proportional hazards assumption hold. For instance, in a follow-up study of human mortality, we may only have data recorded once a year, and so life length can only be measured in years. Thus, we assume that there is a true exact life length T, but we can only observe that it falls in an interval (t_i, t_{i+1}).

Assume *continuous* proportional hazards, and a *partition* of time:

$$0 = t_0 < t_1 < t_2 < \cdots < t_k = \infty.$$

Then

$$P(t_{j-1} \leq T < t_j \mid T \geq t_{j-1};\ \mathbf{x}) = \frac{S(t_{j-1} \mid \mathbf{x}) - S(t_j \mid \mathbf{x})}{S(t_{j-1} \mid \mathbf{x})}$$
$$= 1 - \frac{S(t_j \mid \mathbf{x})}{S(t_{j-1} \mid \mathbf{x})} = 1 - \left(\frac{S_0(t_j)}{S_0(t_{j-1})}\right)^{\exp(\boldsymbol{\beta}\mathbf{x})}$$
$$= 1 - (1 - h_i)^{\exp(\boldsymbol{\beta}\mathbf{x})} \quad (3.12)$$

with $h_i = P(t_{j-1} \leq T < t_j \mid T \geq t_{j-1};\ \mathbf{x} = \mathbf{0})$, $j = 1, \ldots, k$. We take (3.12) as the *definition* of proportional hazards in discrete time.

3.9.1 Logistic Regression

It turns out that a proportional hazards model in discrete time, according to definition (3.12), is nothing else than a *logistic regression* model with the *cloglog link* (cloglog is short for "complementary log-log" or $\boldsymbol{\beta}\mathbf{x} = \log(-\log(p)))$. In order to see that, let

$$(1 - h_j) = \exp(-\exp(\alpha_j)),\ j = 1, \ldots, k$$

and

$$X_j = \begin{cases} 1, & t_{j-1} \leq T < t_j \\ 0, & \text{otherwise} \end{cases}, \quad j = 1, \ldots, k$$

Then

$$P(X_1 = 1;\ \mathbf{x}) = 1 - \exp(-\exp\left(\alpha_1 + \boldsymbol{\beta}\mathbf{x}\right))$$
$$P(X_j = 1 \mid X_1 = \cdots = X_{j-1} = 0;\ \mathbf{x}) = 1 - \exp(-\exp\left(\alpha_j + \boldsymbol{\beta}\mathbf{x}\right)),$$
$$j = 2, \ldots, k.$$

This is logistic regression with a cloglog link. Note that extra parameters $\alpha_1, \ldots, \alpha_k$ are introduced, one for each potential event age. They correspond to the baseline hazard function in continuous time, and are estimated simultaneously with the regression parameters.

3.10 Model Selection

In regression modeling, there is often several competing models for describing data. In general, there are no strict rules for "correct selection." However, for *nested* models, there are some formal guidelines. For a precise definition of this concept, see Appendix A.

3.10.1 Model Selection in General

Some general advice regarding model selection is given here.

- Remember, there are no *true* models, only some *useful* ones. This statement is due to G.E.P. Box.
- More than one model may be useful.
- Keep *important* covariates in the model.
- Avoid automatic stepwise procedures!
- If interaction effects are present, the corresponding main effects *must* be there.

3.11 Male Mortality

We utilize the male mortality data, Skellefteå 1800–1820, to illustrate the aspects of an ordinary Cox regression. Males aged 20 (exact) are sampled between 1820 and 1840 and followed to death or reaching the age of 40, when they are right censored. So, in effect, male mortality in the ages 20–40 are studied. The survival time starts counting at zero for each individual at his 21st birthdate (i.e., exactly at age 20 years).

There are two **R** functions that are handy for a quick look at a data frame, `str` and `head`. The first give a summary description of the *structure* of an **R** object, often a *data frame* :

```
> data(mort) #Loads the data frame 'mort'
> str(mort)
'data.frame':        1208 obs. of  6 variables:
 $ id       : int  1 2 3 3 4 5 5 6 7 7 ...
 $ enter    : num  0 3.48 0 13.46 0 ...
 $ exit     : num  20 17.6 13.5 20 20 ...
 $ event    : int  0 1 0 0 0 0 0 0 0 1 ...
 $ birthdate: num  1800 1800 1800 1800 1800 ...
 $ ses      : Factor w/ 2 levels "lower","upper": 2 1 2 1 1 1 2..
```

First, there is information about the data frame: It *is* a `data.frame`, with six variables measured on 1208 objects. Then each variable is individually described: name, type, and a few of the first values. The values are usually rounded to a few digits. The *Factor* line is worth noticing: It is, of course, a `factor` covariate, it takes two levels, `lower` and `upper`. Internally, the levels are coded 1, 2, respectively. Then (by default), `lower` (with internal code equal to 1) is the *reference category*. If desired, this can be changed by the `relevel` function.

```
> mort$ses2 <- relevel(mort$ses, ref = "upper")
> str(mort)
'data.frame':        1208 obs. of  7 variables:
 $ id       : int  1 2 3 3 4 5 5 6 7 7 ...
 $ enter    : num  0 3.48 0 13.46 0 ...
 $ exit     : num  20 17.6 13.5 20 20 ...
 $ event    : int  0 1 0 0 0 0 0 0 0 1 ...
 $ birthdate: num  1800 1800 1800 1800 1800 ...
 $ ses      : Factor w/ 2 levels "lower","upper": 2 1 2 1 1 1 2..
 $ ses2     : Factor w/ 2 levels "upper","lower": 1 2 1 2 2 2 1..
```

Comparing the information about the variables `ses` and `ses2`, you find that the codings, and also the reference categories, are reversed. Otherwise the variables are identical, and any analysis involving any of them as an explanatory variable would lead to identical *conclusions* (but not identical parameter estimates, see below).

The function `head` prints the first few lines (observations) of a data frame (there is also a corresponding `tail` function that prints a few of the *last* rows).

```
> head(mort)
  id  enter   exit event birthdate   ses  ses2
1  1  0.000 20.000     0  1800.010 upper upper
2  2  3.478 17.562     1  1800.015 lower lower
3  3  0.000 13.463     0  1800.031 upper upper
4  3 13.463 20.000     0  1800.031 lower lower
5  4  0.000 20.000     0  1800.064 lower lower
6  5  0.000  0.089     0  1800.084 lower lower
```

Apparently, the variables `ses` and `ses2` are identical, but we know that behind the scenes they are formally different; different coding and (therefore) different reference category. This shows up in analyses involving the two variables, one at a time.

First, the analysis is run with `ses`.

```
> res <- coxreg(Surv(enter, exit, event) ~ ses + birthdate,
                data = mort)
> res
Call:
coxreg(formula = Surv(enter, exit, event) ~ ses + birthdate,
    data = mort)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses
           lower    0.416     0          1 (reference)
           upper    0.584    -0.484      0.616     0.121     0.000
birthdate        1811.640     0.029      1.029     0.011     0.008

Events                    276
```

```
Total time at risk         17038
Max. log. likelihood      -1841.7
LR test statistic         23.1
Degrees of freedom        2
Overall p-value           9.72167e-06
```

Then the variable `ses2` is inserted instead.

```
> res2 <- coxreg(Surv(enter, exit, event) ~ ses2 + birthdate,
                 data = mort)
> res2
Call:
coxreg(formula = Surv(enter, exit, event) ~ ses2 + birthdate,
    data = mort)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses2
           upper    0.584     0         1 (reference)
           lower    0.416     0.484     1.623     0.121     0.000
birthdate        1811.640     0.029     1.029     0.011     0.008

Events                    276
Total time at risk         17038
Max. log. likelihood      -1841.7
LR test statistic         23.1
Degrees of freedom        2
Overall p-value           9.72167e-06
```

Notice the difference; it is only formal. The substantial results are completely equivalent.

3.11.1 Likelihood Ratio Test

If we judge the result by the Wald p-values, it seems as if both variables are highly significant. To be sure, it is advisable to apply the likelihood ratio test. In **R**, this is easily achieved by applying the `drop1` function on the result, `res`, of the Cox regression.

```
> drop1(res, test = "Chisq")
Single term deletions
Model:
Surv(enter, exit, event) ~ ses + birthdate
          Df    AIC     LRT   Pr(Chi)
<none>       3687.3
ses        1 3701.4 16.1095 5.978e-05
birthdate  1 3692.6  7.2752  0.006991
```

In fact, we have performed *two* likelihood ratio tests here. First, the effect of removing `ses` from the full model. The p-value for this test is very small, $p = 0.00005998$. The scientific notation, 5.998e-05, means that the decimal

point should be moved five steps to the left (that's the minus sign). The reason for this notation is essentially to save space. Then, the second LR test concerns the absence or presence of `birthdate` in the full model. This also gives a small p-value, $p = 0.006972$.

The earlier conclusion is not altered; both variables are highly significant.

3.11.2 The Estimated Baseline Cumulative Hazard Function

The estimated baseline cumulative function (see (3.7)) is best plotted by the `plot` command.

```
> plot(res)
```

The result is shown in Figure 3.7 This figure shows the baseline cumulative

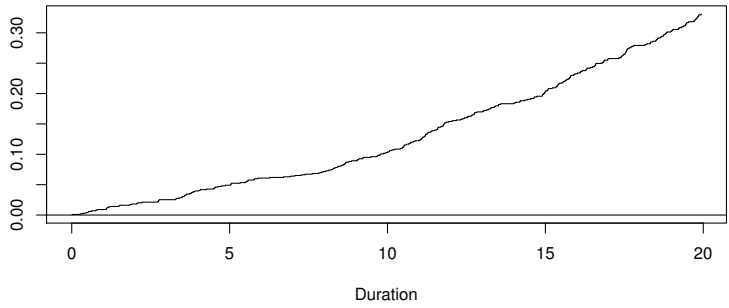


FIGURE 3.7
Estimated cumulative baseline hazard function.

hazard function for an *average* individual (i.e., with the value zero on all covariates when all covariates are *centered* around their weighted means).

3.11.3 Interaction

The introduction of the interaction effect is done as follows.

```
> res2  <- coxreg(Surv(enter, exit, event) ~ ses * birthdate,
                  data = mort)
> res2
Call:
coxreg(formula = Surv(enter, exit, event) ~ ses * birthdate,
    data = mort)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses
```

```
           lower    0.416     0         1 (reference)
           upper    0.584   -53.514     0.000    39.767     0.178
birthdate        1811.640     0.016     1.016     0.014     0.252
ses:birthdate
   upper:                     0.029     1.030     0.022     0.182

Events                    276
Total time at risk         17038
Max. log. likelihood      -1840.8
LR test statistic         24.9
Degrees of freedom        3
Overall p-value           1.64202e-05
```

Note that main effects mean nothing in isolation when involved in an interaction. For instance, if the covariate `birthdate` is centered,

```
> birthdate.c <- mort$birthdate - 1810
> res3  <- coxreg(Surv(enter, exit, event) ~ ses * birthdate.c,
                  data = mort)
> res3
Call:
coxreg(formula = Surv(enter, exit, event) ~ ses * birthdate.c,
    data = mort)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses
           lower    0.416     0         1 (reference)
           upper    0.584    -0.558     0.572     0.134     0.000
birthdate.c         1.640     0.016     1.016     0.014     0.252
ses:birthdate.c
   upper:                     0.029     1.030     0.022     0.182

Events                    276
Total time at risk         17038
Max. log. likelihood      -1840.8
LR test statistic         24.9
Degrees of freedom        3
Overall p-value           1.64202e-05
```

the main effects are different. Note the use of the function `I`; it is the *identity function*, that is, it returns its argument unaltered. The reason for using it here is that it is the (only) way to avoid a mix-up with the formula notation. The drawback with this is that the printed variable name is too long; an alternative is to first create a new variable.

```
> mort$bth1810 <- mort$birthdate - 1810
> res3  <- coxreg(Surv(enter, exit, event) ~ ses * bth1810,
                  data = mort)
> res3
Call:
coxreg(formula = Surv(enter, exit, event) ~ ses * bth1810,
```

```
       data = mort)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses
           lower    0.416     0          1 (reference)
           upper    0.584    -0.558      0.572     0.134     0.000
bth1810             1.640     0.016      1.016     0.014     0.252
ses:bth1810
   upper:                     0.029      1.030     0.022     0.182

Events                    276
Total time at risk         17038
Max. log. likelihood      -1840.8
LR test statistic         24.9
Degrees of freedom        3
Overall p-value           1.64202e-05
```

The likelihood ratio test of interaction becomes

```
> drop1(res2, test = "Chisq")
Single term deletions
Model:
Surv(enter, exit, event) ~ ses * birthdate
              Df    AIC    LRT Pr(Chi)
<none>           3687.5
ses:birthdate  1 3687.3 1.7899  0.1809
```

with non-centered `birthdate`, and

```
> drop1(res3, test = "Chisq")
Single term deletions
Model:
Surv(enter, exit, event) ~ ses * bth1810
             Df    AIC    LRT Pr(Chi)
<none>          3687.5
ses:bth1810  1 3687.3 1.7899  0.1809
```

with centered `birthdate`. The results are identical, so the interaction effect is unaffected by centering.

So, there are two things to be noted: (i) The results with centered or non-centered `birthdate` are the same, and (ii) the main effects are not tested. Such tests would be (almost) meaningless in the presence of an interaction effect.

In Figure 3.8 the effect of `birthdate` (centered at 1810) on mortality is shown for the two levels of `ses`. The two curves seem to converge over time, meaning that the socioeconomic differences in male mortality shrink over time. On the other hand mortality increases over time in both groups.

Note that the reference point is at the intersection of the year 1810 (e.g., at 1 January 1810, 00:00) and the curve corresponding to the ses category `lower` (marked by a circle in the figure).

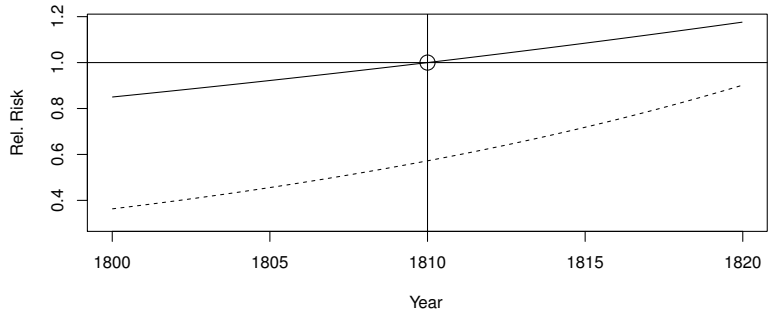


FIGURE 3.8
Effect of *birthdate* (centered at 1810) on mortality, *upper* class (upper curve) and *lower* class (lower curve). Model with interaction.

In Figure 3.9 the effect of `birthdate` (not centered) on mortality is shown for the two levels of `ses`. Here the reference point is at year 0 (which doesn't

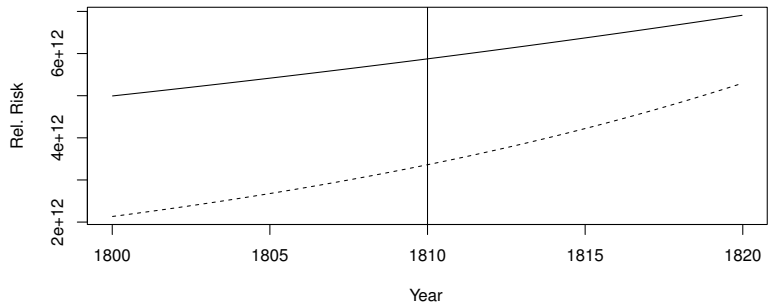


FIGURE 3.9
Effect of *birthdate* (not centered) on mortality, upper class (upper curve) and lower class (lower curve). Model with interaction.

even exist!), in the intersection with the curve corresponding to the `lower` class. That is the reason for the ridiculously large numbers on the y-axis, and the extremely large absolute value of the coefficient for the main effect of `upper`: It measures the extrapolated, hypothetical, difference between `lower` and `upper` at year zero (1 January, 00:00). This is, of course, nonsense, and

the background for the earlier statement that main effects mean nothing when corresponding interaction effects are in the model.

That said, given proper centering of continuous covariates, the main effects *do* mean something. It is the effect when the value of the other covariate is kept at zero, or at its reference category, if it is a factor. But then, anyway, it may be argued that the term *main effect* is quite misleading.

4

Poisson Regression

4.1 Introduction

Here *Poisson regression* is introduced and its connection to Cox regression is discussed. We start by defining the Poisson distribution, then discuss the connection to Cox regression and tabular lifetime data.

4.2 The Poisson Distribution

The *Poisson distribution* is used for *count data*, that is, when the result may be any positive integer $0, 1, 2, \ldots$, without upper limit. The *probability density function* (pdf) P of a random variable X following a Poisson distribution is

$$P(X = k) = \frac{\lambda^k}{k!} \exp(-\lambda), \quad \lambda > 0;\ k = 0, 1, 2, \ldots, \tag{4.1}$$

The parameter λ is both the mean and the variance of the distribution. In Figure 4.1 the pdf (4.1) is plotted for some values of λ. Note that when λ increases, the distribution looks more and more like a normal distribution.

In **R**, the Poisson distribution is represented by four functions, `dpois`, `ppois`, `qpois`, and `rpois`, representing the probability density function (pdf); the cumulative distribution function (cdf); the quantile function (the inverse of the cdf); and random number generation, respectively. See the help page for the Poisson distribution for more detail. In fact, this is the scheme present for all probability distributions available in **R**.

For example, the upper-left bar plot in Figure 4.1 is produced in **R** by

```
> barplot(dpois(0:5, lambda = 0.5), axes = FALSE,
          main = expression(paste(lambda, " = ", 0.5)))
```

Note that the function `dpois` is *vectorized*:

```
> dpois(0:5, lambda = 0.5)
[1] 0.6065306597 0.3032653299 0.0758163325 0.0126360554
[5] 0.0015795069 0.0001579507
```

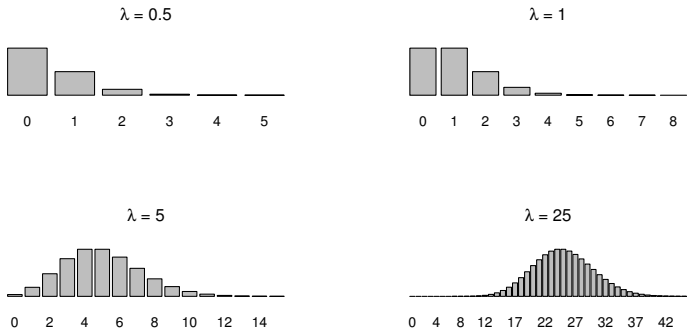


FIGURE 4.1
The *Poisson* cdf for different values of the mean, λ.

Example 12 *Marital fertility*

As an example where the Poisson distribution may be relevant, we look at the number of children born to a woman after marriage. The data frame `fert` in eha can be used to calculate the number of births per married woman i Skellefteå during the 19th century; however, this data set contains only marriages with one or more births. Let us instead count the number of births beyond one.

```
> library(eha)
> data(fert)
> f0 <- fert[fert$event == 1, ]
> kids <- tapply(f0$id, f0$id, length) - 1
> barplot(table(kids))
```

The result is shown in Figure 4.2. The question is now: Does this look like a Poisson distribution? One way of checking this is to plot the *theoretic* distribution with the same mean (the parameter λ) as the sample mean in the data.

```
> lam <- mean(kids)
> barplot(dpois(0:12, lambda = lam))
```

The result is shown in Figure 4.3. Obviously, the fertility data do not follow the Poisson distribution so well. It is, in fact, *over-dispersed* compared to the Poisson distribution. A simple way to check that is to calculate the sample mean and variance of the data. If data come from a Poisson distribution, these numbers should be equal (theoretically) or reasonably close.

```
> mean(kids)
[1] 4.548682
```

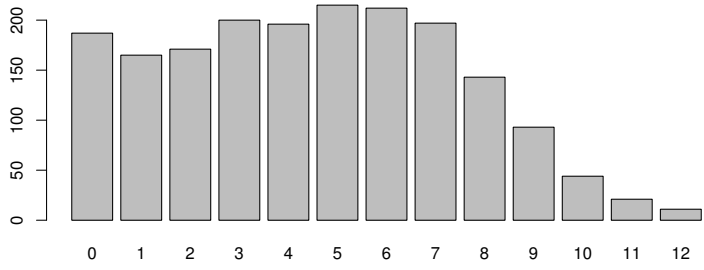


FIGURE 4.2
Number of children beyond one for married women with at least one child.

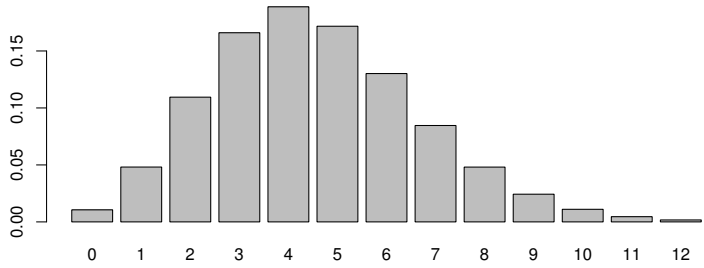


FIGURE 4.3
Theoretical *Poisson* distribution with $\lambda = 4.549$.

```
> var(kids)
[1] 8.586838
```

They are not very close, which also is obvious from comparing the graphs.

4.3 The Connection to Cox Regression

There is an interesting connection between Cox regression and Poisson regression, which can be illustrated as follows.

```
> dat <- data.frame(enter = rep(0, 4), exit = 1:4,
                    event = rep(1, 4), x = c(0, 1, 0, 1))
> dat
  enter exit event x
1     0    1     1 0
2     0    2     1 1
3     0    3     1 0
4     0    4     1 1
```

We have generated a very simple data set, `dat`. It consists of four life lengths with a covariate x. A Cox regression, where the covariate x is related to survival is given by

```
> library(eha)
> fit <- coxreg(Surv(enter, exit, event) ~ x, data = dat)
> fit
Call:
coxreg(formula = Surv(enter, exit, event) ~ x, data = dat)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
x                   0.600    -0.941     0.390     1.240     0.448

Events                    4
Total time at risk           10
Max. log. likelihood      -2.87
LR test statistic         0.62
Degrees of freedom        1
Overall p-value           0.43248
> fit$hazards
[[1]]
     [,1]      [,2]
[1,]    1 0.2246892
[2,]    2 0.3508641
[3,]    3 0.4493785
[4,]    4 1.6004852
attr(,"class")
[1] "hazdata"
```

Note that `fit$hazards` is a `list` with one component per stratum. In this case there is only one stratum, so the list has one component. This component is a matrix with two columns; the first contains the observed distinct failure times, and the second the corresponding estimates of the hazard "atoms."

Now, this can be replicated by Poisson regression! First, the data set is summarized around each observed failure time. That is, a snapshot of the risk set at each failure time is created by the function `toBinary`.

```
> library(eha)
> datB <- toBinary(dat)
> datB
```

```
    event riskset risktime x orig.row
1       1       1        1 0        1
2       0       1        1 1        2
3       0       1        1 0        3
4       0       1        1 1        4
2.1     1       2        2 1        2
3.1     0       2        2 0        3
4.1     0       2        2 1        4
3.2     1       3        3 0        3
4.2     0       3        3 1        4
4.3     1       4        4 1        4
```

Note that three new "covariates" are created; `riskset`, `risktime`, and `orig.row`. In the Poisson regression to follow, the factor `riskset` must be included as a *cluster*. Formally, this is equivalent to including it as a factor covariate, but for data set with many distinct failure times, the number of levels will be too large to be run by `glm`. The function `glmmboot` in `eha` is better suited.

```
> fit2 <- glmmboot(event ~ x, cluster = riskset,
                   family = poisson, data = datB)
> coefficients(fit2)
         x
-0.9406136
```

The parameter estimate corresponding to x is exactly the same as in the Cox regression. The baseline hazard function is estimated with the aid of the *frailty* estimates for the clustering based on `riskset`.

```
> exp(fit2$frail)
[1] 0.3596118 0.5615528 0.7192236 2.5615528
```

These numbers are the hazard atoms in the baseline hazard estimation, so they should be equal to the output from `fit$hazards` in the Cox regression. This is not the case, and this depends on the fact that `coxreg` estimates the baseline hazards at the mean of the covariate, that is, at $x = 0.5$ in our toy example. The easiest way to make the numbers comparable is by subtracting 0.5 from x in the Poisson regression.

```
> datB$x <- datB$x - fit$means
> fit2 <- glmmboot(event ~ x, cluster = riskset,
                   family = poisson, data = datB)
> exp(fit2$frail)
[1] 0.2246892 0.3508641 0.4493785 1.6004852
```

This is exactly what `coxreg` produced.

This correspondence between Poisson and Cox regression was pointed out by Søren Johansen (Johansen 1983).

4.4 The Connection to the Piecewise Constant Hazards Model

For completeness, the connection between the Poisson distribution and the *piecewise constant hazard model* is mentioned here. This is explored in detail in Section 6.2.4 (Chapter 6).

4.5 Tabular Lifetime Data

Although it is possible to use Poisson regression in place of Cox regression, the most useful application of Poisson regression in survival analysis is with tabular data.

Example 13 *Mortality in ages 61–80, Sweden 2007*

In `eha`, there is the data set `swe07`, which contains population size and number of deaths by age and sex in Sweden 2007. The age span is restricted to ages 61–80.

```
> data(swe07)
> head(swe07)
    pop deaths    sex age  log.pop
1 63483    286 female  61 11.05853
2 63770    309 female  62 11.06304
3 64182    317 female  63 11.06948
4 63097    366 female  64 11.05243
5 61671    387 female  65 11.02957
6 57793    419 female  66 10.96462
> tail(swe07)
     pop deaths  sex age  log.pop
35 31074    884 male  75 10.34413
36 29718    904 male  76 10.29951
37 29722   1062 male  77 10.29964
38 28296   1112 male  78 10.25048
39 27550   1219 male  79 10.22376
40 25448   1365 male  80 10.14439
```

The Poisson model is, where D_{ij} is number of deaths and P_{ij} population size for `age` i and `sex` j, $i = 61, \ldots, 80$; $j =$ female (0), male (1), and λ_{ij} is the corresponding *mortality*,

$$D_{ij} \sim \text{Poisson}(\lambda_{ij} P_{ij}), \quad i = 61, 62, \ldots, 80;\ j = 0, 1,$$

with

$$\begin{aligned}\lambda_{61,j}P_{61,j} &= P_{61,j}\exp(\gamma+\beta*j)\\ &= \exp(\log(P_{61,j})+\gamma+\beta*j), \quad j=0,1,\end{aligned}$$

and

$$\begin{aligned}\lambda_{ij}P_{ij} &= P_{ij}\exp(\gamma+\alpha_i+\beta*j)\\ &= \exp(\log(P_{ij})+\gamma+\alpha_i+\beta*j), \quad i=62,63,\ldots,80;\ j=0,1.\end{aligned}$$

This calculation shows that it is the log of the population sizes ($\log(P_{ij})$ that is the correct *offset* to use in the Poisson regression. First, we want `age` to be a factor (no restrictions like linearity), then the **R** function `glm` ("generalized linear model") is used to fit a Poisson regression model.

```
> swe07$age <- factor(swe07$age)
> fit <- glm(deaths ~ offset(log.pop) + sex + age, family = poisson,
             data = swe07)
> drop1(fit, test = "Chisq")
Single term deletions
Model:
deaths ~ offset(log.pop) + sex + age
       Df Deviance     AIC    LRT   Pr(Chi)
<none>        10.2   382.8
sex     1   1428.0  1798.5 1417.7 < 2.2e-16
age    19   9764.6 10099.2 9754.4 < 2.2e-16
```

The function `drop1` is used above to perform two LR tests. The results are that both variables are highly statistically significant, meaning that both are very important in describing the variation in mortality over sex and age. To know *how*, we present the parameter estimates.

```
> summary(fit)
Call:
glm(formula = deaths ~ offset(log.pop) + sex + age, family = poisson,
    data = swe07)
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.94763  -0.40528   0.00104   0.32469   1.00890

Coefficients:
            Estimate Std. Error  z value Pr(>|z|)
(Intercept) -5.41436    0.03777 -143.358  < 2e-16
sexmale      0.46500    0.01246   37.311  < 2e-16
age62        0.04018    0.05171    0.777  0.43705
age63        0.15654    0.05019    3.119  0.00181
age64        0.28218    0.04896    5.764 8.23e-09
age65        0.36374    0.04834    7.524 5.30e-14
age66        0.48712    0.04786   10.178  < 2e-16
```

```
age67        0.61210    0.04755   12.873  < 2e-16
age68        0.62987    0.04868   12.938  < 2e-16
age69        0.81418    0.04742   17.168  < 2e-16
age70        0.81781    0.04744   17.239  < 2e-16
age71        0.93264    0.04690   19.887  < 2e-16
age72        1.03066    0.04659   22.122  < 2e-16
age73        1.12376    0.04618   24.334  < 2e-16
age74        1.27287    0.04548   27.985  < 2e-16
age75        1.37810    0.04507   30.577  < 2e-16
age76        1.45245    0.04481   32.413  < 2e-16
age77        1.61461    0.04366   36.979  < 2e-16
age78        1.73189    0.04317   40.115  < 2e-16
age79        1.83958    0.04266   43.125  < 2e-16
age80        2.00326    0.04217   47.502  < 2e-16

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 10876.873  on 39  degrees of freedom
Residual deviance:    10.232  on 19  degrees of freedom
AIC: 382.81

Number of Fisher Scoring iterations: 3
```

The results tell us that males have a distinctly higher mortality than females, and that mortality steadily increases with age (no surprises!). The parameter estimates for `age` also opens up the possibility to simplify the model by assuming a linear or quadratic effect of age on mortality. We leave that possibility for later.

One question remains: Is the female advantage relatively the same over ages? To answer that question, we introduce an interaction:

```
> fit1 <- glm(deaths ~ offset(log.pop) + sex * age,
                family = poisson, data = swe07)
> drop1(fit1, test = "Chisq")
Single term deletions
Model:
deaths ~ offset(log.pop) + sex * age
        Df Deviance    AIC    LRT Pr(Chi)
<none>       0.000 410.58
sex:age 19   10.232 382.81 10.232   0.947
```

Note that the `drop1` function only tests the interaction term. This is because, as we saw before, the main effects tend to be meaningless in the presence of interaction. However, here there is no sign of interaction, and we can conclude that in a survival model with `sex` as covariate, we have *proportional hazards*! This is most easily seen in *graphs*. The first plot is based on the estimated Poisson model without interaction. Some calculations are first needed.

```
> beta <- coefficients(fit)[2]
> alpha <- coefficients(fit)[-2] # Remove sex
> alpha[2:length(alpha)] <- alpha[2:length(alpha)] + alpha[1]
> lambda.females <- exp(alpha)
> lambda.males <- exp(alpha + beta)
```

Then the plot of the hazard functions by

```
> plot(61:80, lambda.males, ylim = c(0, 0.06), xlab = "Age",
       type = "s")
> lines(61:80, lambda.females, type = "s", lty = 2)
> abline(h = 0)
```

Note the parameter `ylim`. It sets the scale on the y axis, and some trial and error may be necessary to get a good choice. The function `line` adds a curve to an already existing plot, and the function `abline` adds a straight line; the argument `h = 0` results in a *horizontal* line. See the help pages for these functions for further information. at 0. The result is shown in Figure 4.4.

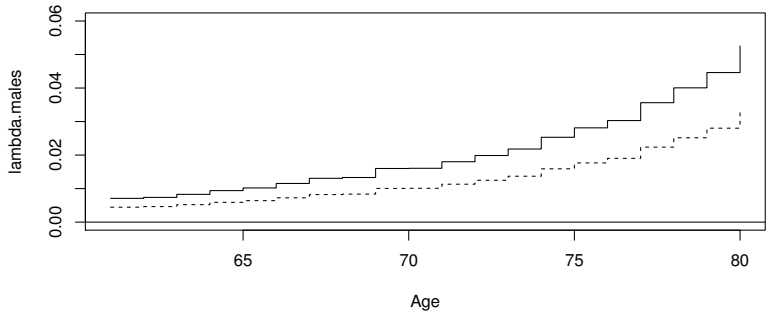


FIGURE 4.4
Model based hazard functions for males (upper) and females (lower).

We can also make a plot based only on raw data. For that it is only necessary to calculate the *occurrence-exposure rates*. An occurrence-exposure rate is simply the ratio between the number of occurrences (of some event) divided by total exposure (waiting) time. In the present example, mean population size during one year is a very good approximation of the exact total exposure time (in years).

```
> femal <- swe07[swe07$sex == "female", ]
> mal <- swe07[swe07$sex == "male", ]
> f.rate <- femal$deaths / femal$pop
> m.rate <- mal$deaths / mal$pop
```

And finally the plot of the raw death rates is created as follows.

```
> plot(61:80, m.rate, ylim = c(0, 0.06), xlab = "Age",
       ylab = "Mortality", type = "s")
> lines(61:80, f.rate, col = "red", type = "s")
> abline(h = 0)
```

with the result as shown in Figure 4.5.

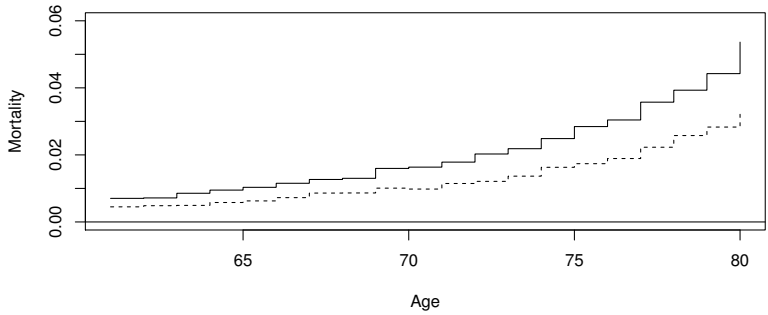


FIGURE 4.5
Hazard functions for males (upper) and females (lower) based on raw data.

The differences between Figure 4.4 and Figure 4.5 are very small, showing that the Poisson model without interaction fits the Swedish old age mortality data very well. □

5

More on Cox Regression

5.1 Introduction

Necessary and vital concepts like *time-dependent covariates*, *communal covariates*, handling of *ties*, *model checking*, and *sensitivity analysis* are introduced in this chapter.

5.2 Time-Varying Covariates

Only a special kind of time-varying covariates can be treated in **R** by the packages `eha` and `survival`, and that is so-called *piecewise constant* functions. How this is done is best described by an example.

Example 14 *Civil status*

The covariate (factor) `civil status` (called x in the **R** data frame) is an explanatory variable in a mortality study, which changes value from 0 to 1 at marriage. How should this be coded in the data file? The solution is to create *two* records (for individuals that marry), each with a *fixed* value of x:

1. Original record: $(t_0, t, d, x(s), t_0 < s \leq t)$, married at time T, $t_0 < T < t$:

$$x(s) = \begin{cases} 0, & s \leq T \\ 1, & s > T \end{cases}$$

2. First new record: $(t_0, T, 0, 0)$, *always censored*.

3. Second new record: $(T, t, d, 1)$.

The data file will contain two records as (with $T = 3$) you can see in Table 5.1. In this way, a time-varying covariate can always be handled by utilizing left truncation and right censoring. See also Figure 5.1 for an illustration. Note that this situation is formally equivalent to a situation with *two individuals*, one unmarried and one married. The first one is right censored at exactly the same time as the second one enters the study (left truncation).

TABLE 5.1
The coding of a time-varying covariate

id	enter	exit	event	civil status
23	0	3	0	0
23	3	8	1	1

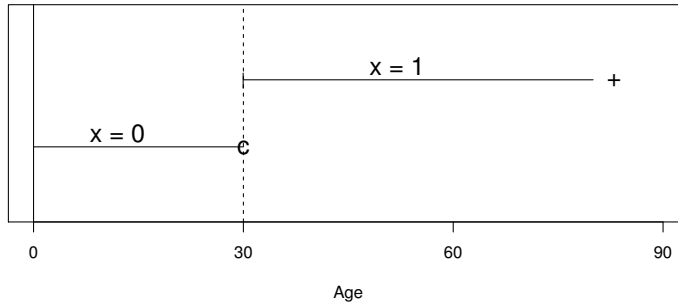


FIGURE 5.1
A time-varying covariate (`civil status`, 0 = unmarried, 1 = married). Right censored and left truncated at age 30 (at marriage).

A word of caution: Marriage status may be interpreted as an *internal* covariate, that is, the change of marriage status is individual, and may depend on health status. For instance, maybe only healthy persons get married. If so, a vital condition in Cox regression is violated; the risk of dying may only be modeled by conditions from the strict history of each individual. Generally, the use of time-dependent covariates is dangerous, and one should always think of possible reversed causality taking place when allowing for it. □

5.3 Communal covariates

Communal (external) covariates are covariates that vary in time outside any individual, and are common to all individuals at each specific *calendar* time. In the econometric literature, such a variable is often called *exogenous*. This could be viewed upon as a special case of a time-varying covariate, but without the danger of reversed causality that was discussed earlier.

Example 15 *Did food prices affect mortality in the 19th century?*

For some 19th century parishes in southern Sweden, yearly time series of food prices are available. In this example we use deviation from the log trend in annual rye prices, shown in Figure 5.2. The idea behind this choice of

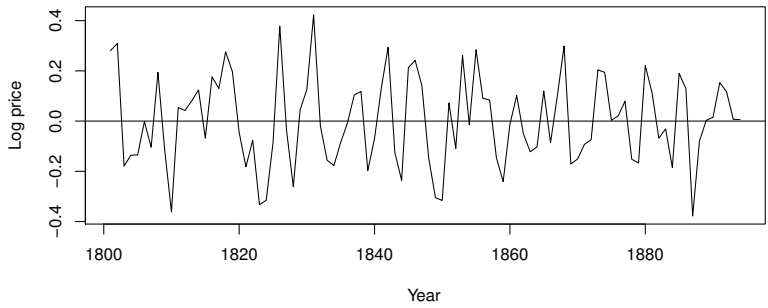


FIGURE 5.2
Log rye price deviations from trend, 19th century southern Sweden.

transformation is to focus on deviations from the "normal" as a proxy for *short-term variation in economic stress*.

```
> data(scania)
> summary(scania)
       id              enter          exit           event
 Min.   :   1.0   Min.   :50   Min.   :50.00   Min.   :0.0000
 1st Qu.: 483.5   1st Qu.:50   1st Qu.:55.37   1st Qu.:0.0000
 Median : 966.0   Median :50   Median :62.70   Median :1.0000
 Mean   : 966.0   Mean   :50   Mean   :63.97   Mean   :0.5624
 3rd Qu.:1448.5   3rd Qu.:50   3rd Qu.:72.05   3rd Qu.:1.0000
 Max.   :1931.0   Max.   :50   Max.   :80.00   Max.   :1.0000
   birthdate         sex       parish      ses        immigrant
 Min.   :1763   male  :975   1: 159   upper: 375   no : 759
 1st Qu.:1786   female:956   2: 181   lower:1556   yes:1172
 Median :1808                3: 180
 Mean   :1806                4: 258
 3rd Qu.:1825                5:1153
 Max.   :1845
> scania[scania$id == 1, ]
   id enter   exit event birthdate  sex parish   ses immigrant
29  1    50 59.242     1  1781.454 male      1 lower        no
```

Now, to put on the food prices as a time-varying covariate, use the function `make.communal`.

```
> require(eha)
> scania <- make.communal(scania, logrye[, 2, drop = FALSE],
                          start = 1801.75)
> scania[scania$id == 1, ]
    enter   exit event birthdate id  sex parish   ses immigrant
1  50.000 50.296     0  1781.454  1 male      1 lower        no
2  50.296 51.296     0  1781.454  1 male      1 lower        no
3  51.296 52.296     0  1781.454  1 male      1 lower        no
4  52.296 53.296     0  1781.454  1 male      1 lower        no
5  53.296 54.296     0  1781.454  1 male      1 lower        no
6  54.296 55.296     0  1781.454  1 male      1 lower        no
7  55.296 56.296     0  1781.454  1 male      1 lower        no
8  56.296 57.296     0  1781.454  1 male      1 lower        no
9  57.296 58.296     0  1781.454  1 male      1 lower        no
10 58.296 59.242     1  1781.454  1 male      1 lower        no
   foodprices
1       0.126
2       0.423
3      -0.019
4      -0.156
5      -0.177
6      -0.085
7      -0.007
8       0.104
9       0.118
10     -0.197
```

A new variable, `foodprices` is added, and each individual's duration is split up in one-year-long intervals (except the first and last). This is how `foodprices` is treated as a time-varying covariate.

```
> fit <- coxreg(Surv(enter, exit, event) ~ ses + sex + foodprices,
                data = scania)
> fit
Call:
coxreg(formula = Surv(enter, exit, event) ~ ses + sex + foodprices,
    data = scania)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses
           upper    0.202     0          1 (reference)
           lower    0.798    -0.030     0.970     0.075     0.686
sex
            male    0.504     0          1 (reference)
          female    0.496     0.041     1.042     0.061     0.500
foodprices          0.002     0.321     1.378     0.182     0.077

Events                    1086
Total time at risk         26979
Max. log. likelihood      -7184.1
```

```
LR test statistic         3.7
Degrees of freedom        3
Overall p-value           0.295547
```

Socioeconomic status and sex do not matter much, but food prices do; the effect is almost significant at the 5% level. □

5.4 Tied Event Times

Tied event times can in theory not occur with continuous-time data, but it is impossible to measure duration and life lengths with infinite precision. Data are always more or less rounded, and tied event times occur frequently in practice. This may cause problems (biased estimates) if they occur too frequently. There are a few ways to handle tied data, and the so-called exact method considers all possible permutations of the tied event times in each risk set. It works as shown in the following example.

Example 16 *Likelihood contribution at tied event time.*

$R_i = \{1, 2, 3\}$, 1 and 2 are events; two possible orderings:

$$\begin{aligned} L_i(\boldsymbol{\beta}) &= \frac{\psi(1)}{\psi(1)+\psi(2)+\psi(3)} \times \frac{\psi(2)}{\psi(2)+\psi(3)} \\ &+ \frac{\psi(2)}{\psi(1)+\psi(2)+\psi(3)} \times \frac{\psi(1)}{\psi(1)+\psi(3)} \\ &= \frac{\psi(1)\psi(2)}{\psi(1)+\psi(2)+\psi(3)} \left\{ \frac{1}{\psi(2)+\psi(3)} + \frac{1}{\psi(1)+\psi(3)} \right\} \end{aligned}$$

□

The main drawback of the exact method is that it easily becomes unacceptably slow, because of the huge number of permutations that may be necessary to consider.

Fortunately, there are a few excellent approximations, most notably Efron's (Efron 1977), which is the default method in most survival packages in **R**. Another common approximation, due to Breslow (Breslow 1974), is the default in some statistical software and also possible to choose in the `eha` and `survival` packages in **R**. Finally, there is no cost involved in using these approximations in the case of no ties at all; they will all give identical results.

With too much ties in the data, there is always the possibility of using discrete time methods. One simple method is to use the option *method = "ml"* in the `coxreg` function in the **R** package `eha`.

Example 17 *Birth intervals.*

We look at lengths of birth intervals between first and second births for married women in 19th century northern Sweden, a subset of the data set `fert`, available in the `eha` package. Four runs with `coxreg` are performed, with all the possible ways of treating ties.

```
> require(eha)
> data(fert)
> first <- fert[fert$parity == 1, ]
> ## Default method (not necessary to specify method = "efron"
> fit.e <- coxreg(Surv(next.ivl, event) ~ year + age, data = first,
                  method = "efron")
> ## Breslow
> fit.b <- coxreg(Surv(next.ivl, event) ~ year + age, data = first,
                  method = "breslow")
> ## The hybrid mppl:
> fit.mp <- coxreg(Surv(next.ivl, event) ~ year + age, data = first,
                   method = "mppl")
> ## True discrete:
> fit.ml <- coxreg(Surv(next.ivl, event) ~ year + age, data = first,
                   method = "ml")
```

Then we compose a table of the four results; see Table 5.2.

TABLE 5.2
Comparison of four methods of handling ties, estimated coefficients

	year	age
Efron	0.0017	-0.0390
Breslow	0.0017	-0.0390
MPPL	0.0016	-0.0390
ML	0.0016	-0.0390

There are almost no difference in results between the four methods. For the `exact` method mentioned above, the number of permutations is $\sim 7 \times 10^{192}$, which is impossible to handle. □

With the help of the function `risksets` in `eha`, it is easy to check the composition of risk sets in general and the frequency of ties in particular.

```
> rs <- risksets(Surv(first$next.ivl, first$event))
> tt <- table(rs$n.event)
> tt
  1   2   3   4   5   6   7   9
369 199 133  60  27  14   2   2
```

This output says that 369 risk sets have only one event each (no ties), 199 risk sets have exactly two events each, etc.

The object `rs` is a `list` with seven components. Two of them are `n.event`, which counts the number of events in each risk set, and `size`, which gives the size (number of individuals under observation) of each risk set. Both these vectors are ordered after the `risktimes`, which is another component of the list. For the rest of the components, see the help page for `risksets`.

It is easy to produce the *Nelson–Aalen plot* with the output from `risksets`. The code is

```
> plot(rs$risktimes, cumsum(rs$n.event / rs$size), type = "s",
       xlab = "Duration (years)", ylab = "Cumulative hazards")
> abline(h = 0)
```

and the result is shown in Figure 5.3.

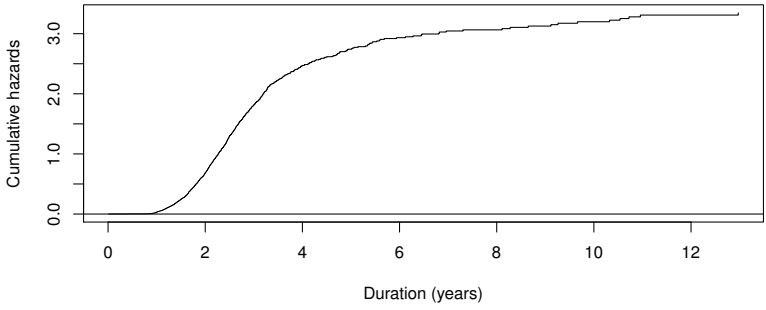


FIGURE 5.3
Nelson–Aalen plot with the aid of the function `risksets` in `eha`. Birth intervals between first and second births, 19th century Swedish data.

One version of the corresponding survival function can be constructed by

```
> sur <- exp(-cumsum(rs$n.event / rs$size))
> plot(rs$risktimes, sur, type = "s",
       xlab = "Duration (years)", ylab = "Surviving fraction")
> abline(h = 0)
```

and the result is shown in Figure 5.4

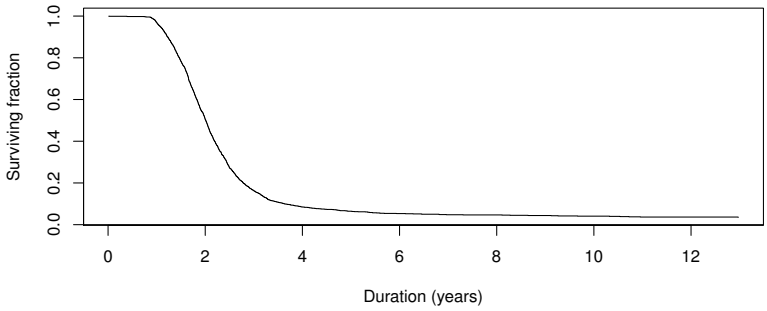


FIGURE 5.4
Survival plot with the aid of the function `risksets` in `eha`. Birth intervals between first and second births, 19th century Swedish data.

5.5 Stratification

Stratification means that data is split up in groups called *strata*, and a separate partial likelihood function is created for each stratum, but with common values on the regression parameters corresponding to the common explanatory variables. In the estimation, these partial likelihoods are multiplied together, and the product is treated as a likelihood function. There is one restriction on the parameters; they are the same across strata.

There are typically two reasons for stratification. First, if the proportionality assumption (see Section 5.8) does not hold for a factor covariate, a way out is to stratify along it. Second, a factor may have too many levels, so that it is inappropriate to treat is as an ordinary factor. This argument is similar to the one about using a frailty model (Chapter 7). Stratification is also a useful tool with matched data; see Section 9.6.

When a *factor* does not produce proportional hazards between categories, *stratify* on the categories. For tests of the proportionality assumption, see Section 5.8.

Example 18 *Birth intervals.*

For the birth interval data, we stratify on `ses` in the Cox regression:

```
> data(fert)
> fert1 <- fert[fert$parity == 1, ]
> fit <- coxreg(Surv(next.ivl, event) ~ strata(ses) +
                  age + year + prev.ivl + parish, data = fert1)
```

and plot the fit; see Figure 5.5. The nonproportionality pattern is clearly visible. □

```
> plot(fit)
```

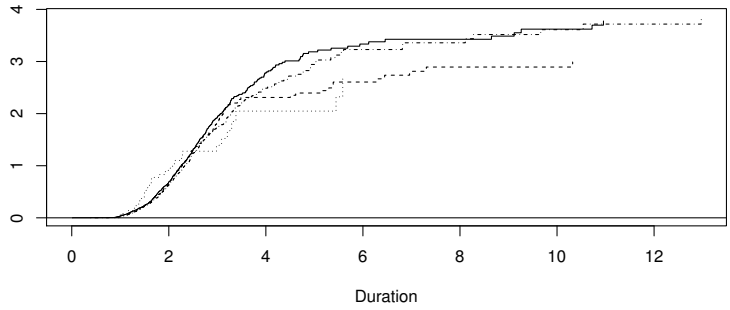


FIGURE 5.5
Cumulative hazards by socioeconomic status, birth intervals.

5.6 Sampling of Risk Sets

Some of the work by (Langholz & Borgan 1995) is implemented in *eha*. The idea behind sampling of risk sets is that with huge data sets, in each risk set there will be a few (often one) events, but many survivors. From an efficiency point of view, not much will be lost by only using a random fraction of the survivors in each risk set. The gain is, of course, computation speed and memory saving. How it is done in `eha` is shown in with an example.

Example 19 *Sampling of risk sets, male mortality.*

For the `male` mortality data set, we compare a full analysis with one where only four survivors are sampled in each risk set.

```
> data(mort)
> system.time(fit <- coxreg(Surv(enter, exit, event) ~ ses,
                            data = mort))
   user  system elapsed
  0.020   0.000   0.017
```

```
> system.time(fit.4 <- coxreg(Surv(enter, exit, event) ~ ses,
                              max.survs = 4, data = mort))

   user  system elapsed
  0.012   0.000   0.012

> fit

Call:
coxreg(formula = Surv(enter, exit, event) ~ ses, data = mort)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses
           lower    0.416     0         1 (reference)
           upper    0.584    -0.480     0.619     0.121     0.000

Events                    276
Total time at risk         17038
Max. log. likelihood      -1845.3
LR test statistic         15.8
Degrees of freedom        1
Overall p-value           7.01379e-05

> fit.4

Call:
coxreg(formula = Surv(enter, exit, event) ~ ses, data = mort,
    max.survs = 4)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
ses
           lower    0.416     0         1 (reference)
           upper    0.584    -0.501     0.606     0.135     0.000

Events                    276
Total time at risk         17038
Max. log. likelihood      -437.98
LR test statistic         13.9
Degrees of freedom        1
Overall p-value           0.00019136
```

The results are comparable, and a gain in execution time is noted. Of course, with the small data sets we work with here, the difference is of no practical importance. □

It is worth noting that the function `risksets` has an argument `max.survs`, which, when it is set, sample survivors for each risk set in the data. The output can then be used as input to `coxreg`; see the relevant help pages for more information.

5.7 Residuals

Residuals are generally the key to judgment of model fit to data. It is easy to show how it works with *linear regression.*

Example 20 *Linear regression.*

Figure 5.6 shows a scatter plot of bivariate data and a fitted straight line. The residuals are the vertical distances between the line and the points. □

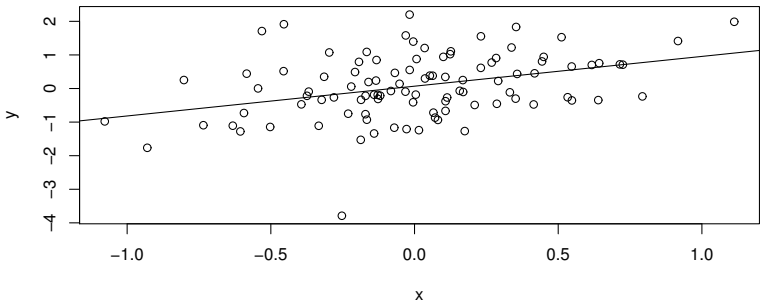


FIGURE 5.6
Residuals in linear regression.

Unfortunately, this kind of simple and intuitive graph does not exist for results from a Cox regression analysis. However, there are a few ideas how to accomplish something similar. The main idea goes as follows. If T is a survival time, and $S(t)$ the corresponding survivor function (continuous time), then it is a mathematical fact that $U = S(T)$ is uniformly distributed on the interval $(0, 1)$. Continuing, this implies that $-\log(U)$ is exponentially distributed with rate (parameter) 1.

It is thus shown that $H(T)$ (H is the cumulative hazard function for T) is exponentially distributed with rate 1. This motivates the *Cox–Snell* residuals, given a sample $T_1, \ldots, T_n$ of survival times,

$$r_{Ci} = \hat{H}(T_i; \mathbf{x}_i) = \exp(\boldsymbol{\beta}\mathbf{x}_i)\hat{H}_0(T_i), \quad i = 1, ldots, n,$$

which should behave like a censored sample from an exponential distribution with parameter 1, if the model is good. If, for some i, T_i is censored, so is the residual.

5.7.1 Martingale Residuals

A drawback with the Cox–Snell residuals is that they contain censored values. An attempt to correct the censored residuals leads to the so-called *martingale* residuals. The idea is simple; it builds on the "lack-of-memory" property of the exponential distribution. The expected value is 1, and by adding 1 to the *censored* residuals, they become predictions (estimates) of the corresponding uncensored values. Then finally a twist is added: Subtract 1 from *all* values and multiply by -1, which leads to to the definition of the martingale residuals.

$$r_{Mi} = \delta_i - r_{Ci}, i = 1, \ldots, n.$$

where δ_i is the indicator of event for the i:th observation. These residuals may be interpreted as the *Observed* minus the *Expected* number of events for each individual.

Direct plots of the martingale residuals tend to be less informative, especially with large data sets.

Example 21 *Plot of martingale residuals.*

The data set `kidney` from the `survival` package is used. For more information about it, read its help page in **R**. For each patient, two survival times are recorded, but only one will be used here. That is accomplished with the function `duplicated`.

```
> require(eha)
> data(kidney)
> head(kidney)
  id time status age sex disease frail
1  1    8      1  28   1   Other   2.3
2  1   16      1  28   1   Other   2.3
3  2   23      1  48   2      GN   1.9
4  2   13      0  48   2      GN   1.9
5  3   22      1  32   1   Other   1.2
6  3   28      1  32   1   Other   1.2
> k1 <- kidney[!duplicated(kidney$id), ]
> fit <- coxreg(Surv(time, status) ~ disease + age + sex,
                data = k1)
```

The *extractor function* `residuals` is used to extract the martingale residuals from the fit.

```
> plot(residuals(fit), ylab = "Residuals")
```

See Figure 5.7.

With a large data set, the plot will be hard to interpret; see Figure 5.8, which shows the martingale residuals from a fit of the *male mortality* (`mort`) data in the package `eha`. It is clearly seen that the residuals are grouped in two clusters. The positive ones are connected to the observations with an event,

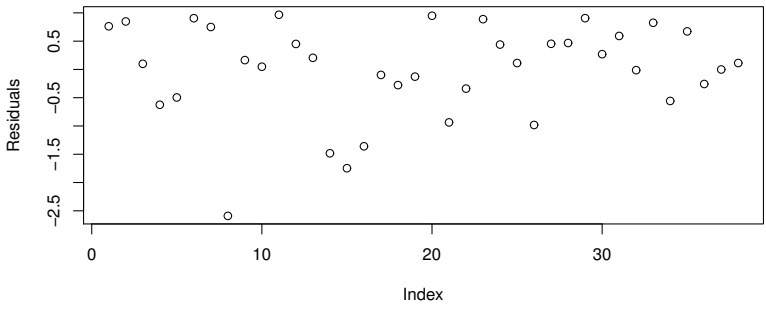


FIGURE 5.7
Martingale residuals from a fit of the kidney data.

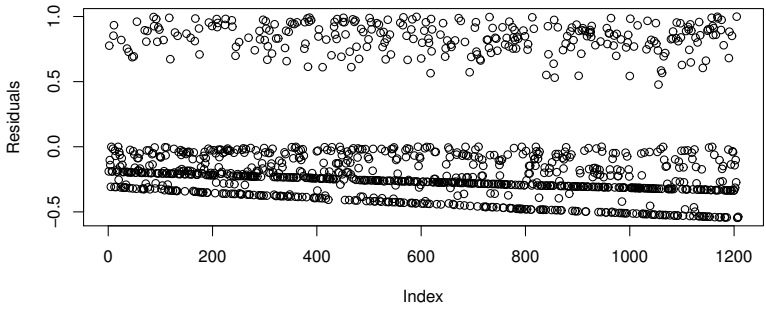


FIGURE 5.8
Martingale residuals from a fit of the male mortality data.

while the negative ones are corresponding to the censored observations. It is hard to draw any conclusions from plots like this one. However, the residuals are used in functions that evaluate goodness-of-fit. □

5.8 Checking Model Assumptions

5.8.1 Proportionality

It is extremely important to check the proportionality assumption in the proportional hazards models. We have already seen how violations of the assumption can be dealt with (Section 5.5).

We illustrate this with the `births` data set in the `eha` package.

Example 22 *Birth intervals, proportionality assumption.*

In order to keep observations reasonably independent, we concentrate on one specific birth interval per woman, the interval between the second and third births. That is, in our sample are all women with at least two births, and we monitor each woman from the day of her second delivery until the third, or if that never happens, throughout her observable fertile period, i.e, to age 50 or to loss of follow-up. The latter will result in a right-censored interval.

```
> require(eha)
> ##data(births)
> fert2 <- fert[fert$parity == 2, ]
```

Then we save the fit from a Cox regression,

```
> fit <- coxreg(Surv(next.ivl, event) ~ ses + age + year + parish,
                data = fert2)
```

and check the proportionality assumption. It is done with the function `cox.zph` in the `survival` package.

```
> prop.full <- cox.zph(fit)
> prop.full
                rho   chisq       p
sesunknown -0.02773  1.1235 0.28918
sesupper   -0.05311  4.2624 0.03896
seslower    0.04774  3.4199 0.06441
age        -0.07322  6.8550 0.00884
year       -0.00315  0.0156 0.90071
parishNOR   0.00907  0.1234 0.72538
parishSKL   0.01883  0.5333 0.46524
GLOBAL           NA 20.1604 0.00523
```

In the table above, look first at the last row, `GLOBAL`. It is the result of a test of the global null hypothesis that proportionality holds. The small p-value tells us that we have a big problem with the proportionality assumption. Looking further up, we see two possible problems, the variables `age` and `ses`. Unfortunately, `age` is a continuous variable. A categorical variable can be stratified upon, but with *age* we have to categorize first.

Let us first investigate the effect of stratifying on `ses`;

```
> fit1 <- coxreg(Surv(next.ivl, event) ~ strata(ses) +
                   age + year + parish, data = fert2)
> plot(fit1)
```

and plot the result; see Figure 5.9. The nonproportionality is visible; some

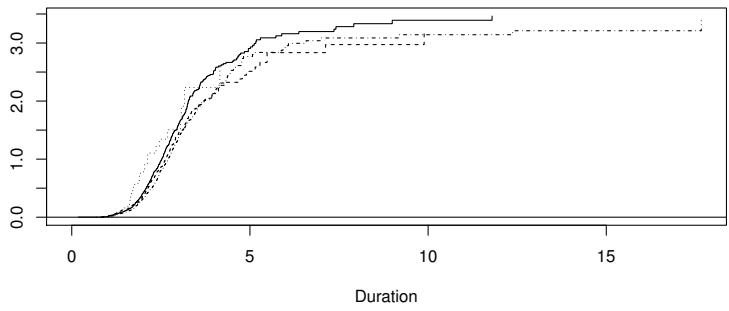


FIGURE 5.9
Stratification on socio-economic status.

curves cross. The test of proportionality of the stratified model shows that we still have a problem with `age`.

```
> fit1.zph <- cox.zph(fit1)
> print(fit1.zph)
                rho   chisq       p
age       -0.074152 6.94200 0.00842
year       0.000807 0.00101 0.97469
parishNOR  0.010598 0.16842 0.68152
parishSKL  0.021465 0.69300 0.40515
GLOBAL           NA 9.51235 0.04949
```

Since `age` is continuous covariate, we may have to categorize it. First, its distribution is checked with a histogram; see Figure 5.10. The range of `age` may reasonably be split into four equal-length intervals with the `cut` function.

```
> hist(fert2$age, main = "", xlab = "age")
```

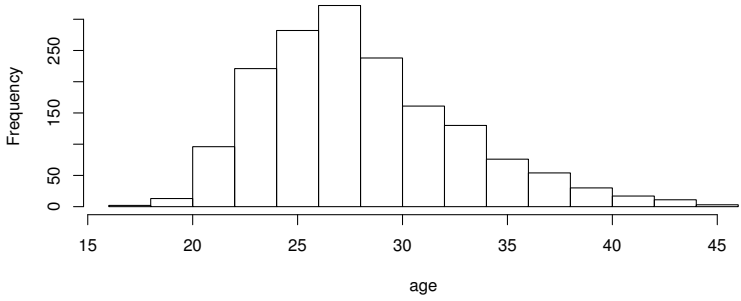


FIGURE 5.10
The distribution of `age`, birth interval data.

```
> fert2$qage <- cut(fert2$age, 4)
> fit2 <- coxreg(Surv(next.ivl, event) ~ strata(ses) +
                 qage + year + parish, data = fert2)
```

and then the proportionality assumption is tested again.

```
> fit2.zph <- cox.zph(fit2)
> fit2.zph
                     rho   chisq      p
qage(24.2,31.5]  0.00529 0.03975 0.8420
qage(31.5,38.8] -0.02639 1.01577 0.3135
qage(38.8,46]   -0.05763 5.13047 0.0235
year             0.00235 0.00879 0.9253
parishNOR        0.00501 0.03762 0.8462
parishSKL        0.01500 0.33875 0.5606
GLOBAL                NA 8.73344 0.1891
```

The result is now reasonable, with the high age group deviating slightly. This is not so strange; fertility becomes essentially zero in the age range in the upper forties, and therefore it is natural that the proportionality assumption is violated. One way of dealing with the problem, apart from stratification, would be to analyze the age groups separately. □

5.8.2 Log-Linearity

In the Cox regression model, the effect of a covariate on the hazard is *log-linear*; that is, it affects the log hazard linearly. Sometimes a covariate needs

to be transformed to fit into the pattern. The first question is then: Should the covariate be transformed in the analysis? A graphical test can be done by plotting the martingale residuals from a *null fit* against the covariate; see Figure 5.11.

```
> x <- rnorm(100)
> tid <- rexp(100, exp(1.5 * x^2))
> event <- rep(1, 100)
> fit <- coxreg(Surv(tid, event) ~ 1)
> plot(x, residuals(fit))
> lines(lowess(x, residuals(fit)))
```

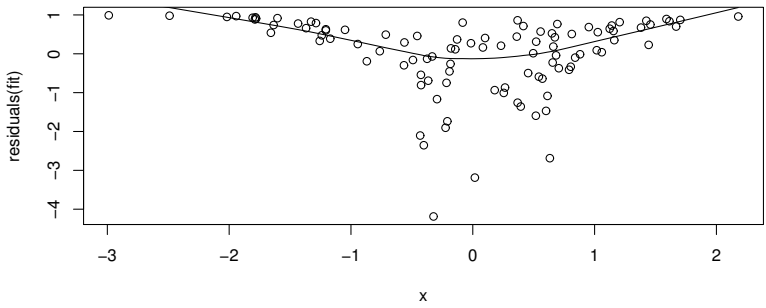


FIGURE 5.11
Plot of residuals against the explanatory variable x.

The function `lowess` fits a "smooth" curve to a scatterplot. The curvature is clearly visible; let us see what happens if we make the plot with x^2 instead of x. (Note that we now are cheating!) See Figure 5.12. This plot indicates more clearly that an increasing x^2 is associated with an increasing risk for the event to occur.

5.9 Fixed Study Period Survival

Sometimes survival up to some fixed time is of interest. For instance, in medicine five-year survival may be used as a measure of the success of a treatment. The measure is then the probability that a patient will survive five years after treatment.

This is simple to implement; it is a binary experiment, where for each patient, survival or not survival is recorded (plus covariates, such as treatment). The only complicating factor is incomplete observations, that is, right

```
> plot(x^2, residuals(fit))
> lines(lowess(x^2, residuals(fit)))
```

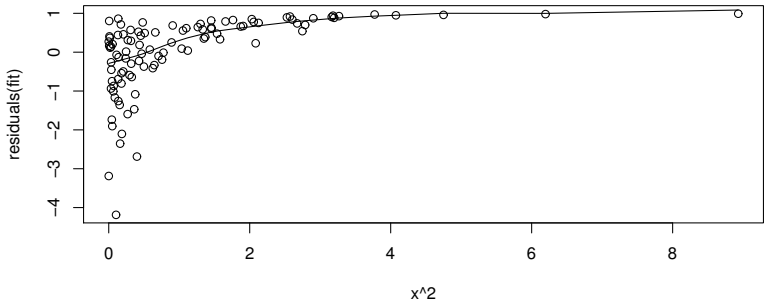


FIGURE 5.12
Plot of residuals against the explanatory variable x^2.

censoring, in which case it is recommended to use ordinary Cox regression anyway. Otherwise, logistic regression with a suitable link function works well. Of course, as mentioned earlier, the `cloglog` link is what gets you closest to the proportional hazards model, but usually it makes very little difference. See Kalbfleisch & Prentice (2002, p. 334 ff.) for slightly more detail.

5.10 Left- or Right-Censored Data

Sometimes the available data is either left or right censored; that is, for each individual i, we know survival of t_i or not, that is, the data are of the form

$$(t_i, \delta_i), \quad i = 1, \ldots n,$$

with $\delta_i = 1$ if survival (died after t_i) and $\delta_i = 0$ if not (died before t_i). It is still possible to estimate the survivor function S nonparametrically. The likelihood function is

$$L(S) = \prod_{i=1}^{n} \{S(t_i)\}^{\delta_i} \{1 - S(t_i)\}^{1-\delta_i}$$

How to maximize this under the restrictions $t_i < t_j \Rightarrow S(t_i) \geq S(t_j)$ with the *EM algorithm* is shown by (Groeneboom & Wellner 1992).

6

Parametric Models

6.1 Introduction

So far we have exclusively studied nonparametric methods for survival analysis. This is a tradition that has its roots in medical (cancer) research. In technical applications, on the other hand, parametric models dominate; of special interest is the *Weibull* model (Weibull 1951). The textbook by (Lawless 2003) is a good source for the study of parametric survival models. More technical detail about parametric distributions is found in Appendix B.

In `eha` three kinds of parametric models are available; the *proportional hazards*, the *accelerated failure time* in continuous time, and the *discrete time proportional hazards* models.

6.2 Proportional Hazards Models

A proportional hazards family of distributions is generated from one specific continuous distribution by multiplying the hazard function of that distribution by a strictly positive constant, and letting that constant vary over the full positive real line. So, if h_0 is the hazard function corresponding to the generating distribution, the family of distributions can be described by saying that h is a member of the family if

$$h_1(t) = ch_0(t) \quad \text{for some } c > 0, \text{ and all } t > 0.$$

Note that it is possible to choose any hazard function (on the positive real line) as the generating function. The resulting proportional hazards class of distributions may or may not be a well-recognized family of distributions.

In **R**, the package `eha` can fit parametric proportional hazards models. In the following subsections, the possibilities are examined.

The parametric distribution functions that can be used as the baseline distribution in the function `phreg` are the *Weibull*, *Piecewise constant hazards* (`pch`), *Lognormal*, *Loglogistic*, *Extreme value*, and the *Gompertz* distributions. The *lognormal* and *loglogistic* distributions allow for hazard functions that are

first increasing to a maximum and then decreasing, while the other distributions all have monotone hazard functions. See Figure 6.2 for a selection. The

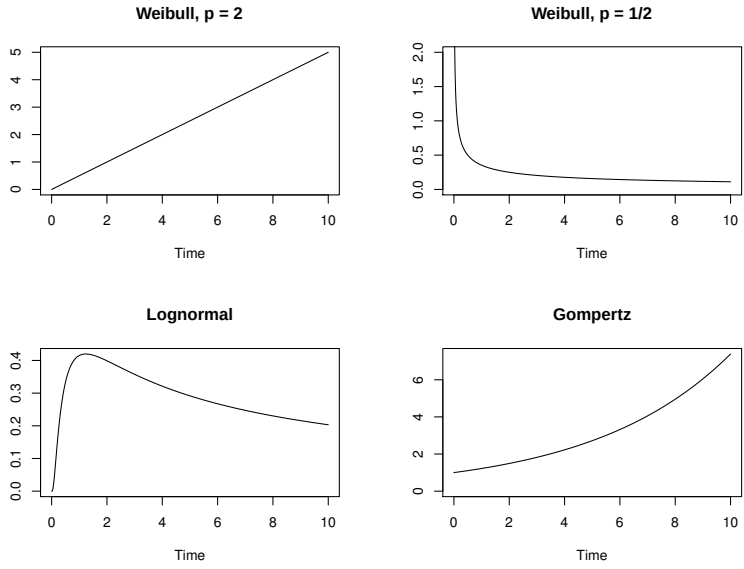


FIGURE 6.1
Selected hazard functions.

available survival distributions are described in detail in Appendix B.

In the following, we start by looking at a typical choice of survivor distribution, the *Weibull* distribution, and a rather atypical one, the *Lognormal* distribution. One reason for the popularity of the *Weibull* survivor distribution is that the survivor and hazard functions both have fairly simple closed forms, and the proportional hazards models built from a *Weibull* distribution stays within the family of *Weibull* distributions. The *Lognormal* distribution, on the other hand, does not have any of these nice properties.

After studying the *Weibull* and *Lognormal* distributions, we look at some typical data sets and find that different distributions step forward as good ones in different situations. One exception is the *Piecewise constant hazards (pch)* distribution, which is flexible enough to give a good fit to any given distribution. It is in this sense close to the nonparametric approach used in Cox regression.

6.2.1 The Weibull Model

From Appendix B.3.4 we know that the family of *Weibull* distributions may be defined by the family of hazard functions

$$h(t; p, \lambda) = \frac{p}{\lambda}\left(\frac{t}{\lambda}\right)^{p-1}, \quad t, p, \lambda > 0. \tag{6.1}$$

If we start with

$$h_1(t) = t^{p-1}, \quad p \geq 0,\ t > 0$$

for a *fixed* p, and generate a proportional hazards family from there,

$$h_c = ch_1(t), \quad c, t > 0,$$

we get

$$h_c = ct^{p-1} = \frac{p}{\lambda}\left(\frac{t}{\lambda}\right)^{p-1} \tag{6.2}$$

by setting

$$c = \frac{p}{\lambda^p},$$

which shows that for each fixed $p > 0$, a proportional hazards family is generated by varying λ in (6.2). On the other hand, if we pick two members from the family (6.2) with *different* values of p, they would not be proportional.

To summarize, the *Weibull* family of distributions is not *one* family of proportional hazards distributions, but a *collection* of families of proportional hazards. The collection is indexed by $p > 0$. It is true, though, that all the families are closed under the *Weibull* distribution.

The proportional hazards regression model with a *Weibull* baseline distribution as obtained by multiplying (6.2) by $\exp(\boldsymbol{\beta}\mathbf{x})$:

$$h(t; \mathbf{x}, \lambda, p, \boldsymbol{\beta}) = \frac{p}{\lambda}\left(\frac{t}{\lambda}\right)^{p-1} e^{\boldsymbol{\beta}\mathbf{x}}, \quad t > 0. \tag{6.3}$$

The function `phreg` in package `eha` fits models like (6.3) by default.

Example 23 *Length of birth intervals, Weibull model*

The data set `fert` in the **R** package `eha` contains birth intervals for married women in 19th century Skellefteå. It is described in detail in Chapter 1. Here, only the intervals starting with the first birth for each woman are considered. First the data are extracted and examined.

```
> require(eha)
> data(fert)
> f12 <- fert[fert$parity == 1, ]
> f12$Y <- Surv(f12$next.ivl, f12$event)
> head(f12)
```

```
   id parity age year next.ivl event prev.ivl     ses parish
2   1      1  25 1826   22.348     0    0.411  farmer    SKL
4   2      1  19 1821    1.837     1    0.304 unknown    SKL
11  3      1  24 1827    2.051     1    0.772  farmer    SKL
21  4      1  35 1838    1.782     0    6.787 unknown    SKL
23  5      1  28 1832    1.629     1    3.031  farmer    SKL
28  6      1  25 1829    1.730     1    0.819   lower    SKL
         Y
2  22.348+
4   1.837
11  2.051
21  1.782+
23  1.629
28  1.730
```

Some women never got a second child, for instance, the first woman (`id = 1`) above. Also, note the just created variable `Y`; it is printed as a vector, with some values having a trailing "+"; those are the censored observations (`event = 0`). But `Y` is really a matrix, in this case with two columns. The first column equals `next.ivl` and the second is `event`.

```
> is.matrix(f12$Y)
[1] TRUE
> dim(f12$Y)
[1] 1840    2
```

Next, the proportional hazards *Weibull* regression is fitted, and the likelihood ratio tests for an effect of each covariate is performed. Remember that the function `drop1` is the one to use for the latter task.

```
> fit.w <- phreg(Y ~ age + year + ses, data = f12)
> drop1(fit.w, test = "Chisq")
Single term deletions
Model:
Y ~ age + year + ses
       Df    AIC    LRT   Pr(Chi)
<none>    6290.1
age     1 6341.8 53.719 2.314e-13
year    1 6293.4  5.272   0.02167
ses     3 6314.2 30.066 1.337e-06
```

Three variables are included, two are extremely significant, *age*, mother's age at start of the interval, and *ses*, socioeconomic status. The variable *year* is not statistically significant.

Note: When we say that a variable is "significant," we are rejecting the hypothesis that the true parameter value attached to it is equal to zero.

Now take a look at the actual fit.

```
> fit.w
```

```
Call:
phreg(formula = Y ~ age + year + ses, data = f12)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
age                27.151    -0.039     0.961     0.005     0.000
year             1858.664     0.005     1.005     0.002     0.022
ses
          farmer    0.449     0         1           (reference)
         unknown    0.190    -0.359     0.698     0.070     0.000
           upper    0.024    -0.382     0.683     0.174     0.028
           lower    0.336    -0.097     0.908     0.056     0.085

log(scale)                    1.073     2.925     0.019     0.000
log(shape)                    0.299     1.349     0.015     0.000

Events                    1657
Total time at risk        4500.5
Max. log. likelihood       -3140
LR test statistic         74.7
Degrees of freedom        5
Overall p-value           1.09912e-14
> kof <- fit.w$coef[1]
```

The estimated coefficient for `age`, -0.039, is negative, indicating lower risk of a birth with increasing age. The effect of calendar time is positive and indicates an increase in the intensity of giving birth of the size approximately 5 per thousand. Regarding *socioeconomic status*, `farmer` wives have the highest fertility and the `upper` class the lowest when it comes to getting a second child. Finally, we see that the estimate for "log(shape)" is positive (0.299) and significantly different from zero. This means that the intensity is increasing with duration; see Figure 6.2. □

6.2.2 The Lognormal Model

An example of a more complex family of distributions with the proportional hazards property is to start with a *lognormal* distribution. The family of *lognormal* distributions is characterized by the fact that taking the natural logarithm of a random variable from the family gives a random variable from the family of *Normal* distributions. Both the hazard and the survivor functions lack closed forms. Furthermore, multiplying the hazard function by a constant not equal to 1 leads to a non-lognormal hazard function.

So, while for the *Weibull* family of distributions we found subfamilies with proportional hazards by keeping p fixed and varying λ, for the *lognormal* distribution we cannot use the same trick. Instead, by multiplying the hazard function by a constant we arrive at a three-parameter family of distributions. The lognormal proportional hazards model is possible to fit with the function `phreg`.

```
> oldpar <- par(mfrow = c(1, 2))
> plot(fit.w, fn = "haz")
> plot(fit.w, fn = "cum")
> par(oldpar)
```

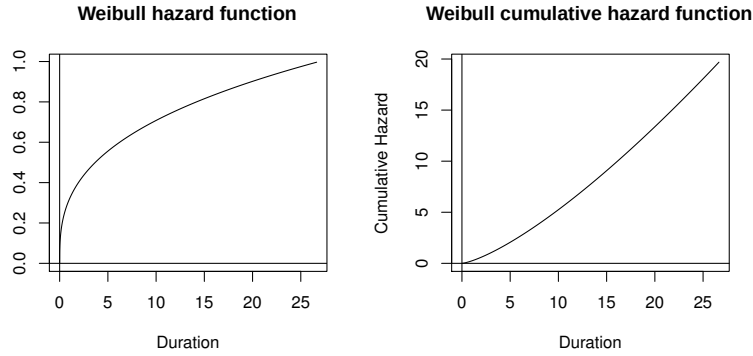


FIGURE 6.2
Estimated *Weibull* baseline distribution for length of interval between first and second birth.

Example 24 *Length of birth intervals, Lognormal model*

The data are already extracted and examined in the previous example. Next, the proportional hazards *lognormal* regression and the likelihood ratio tests for an effect of each covariate are given.

```
> fit.lognorm <- phreg(Y ~ age + year + ses, data = f12,
                       dist = "lognormal")
> drop1(fit.lognorm, test = "Chisq")
Single term deletions
Model:
Y ~ age + year + ses
       Df    AIC    LRT   Pr(Chi)
<none>    5206.0
age     1 5281.2 77.202 < 2.2e-16
year    1 5207.1  3.093 0.0786446
ses     3 5218.5 18.514 0.0003445
```

Three variables are included, and two are extremely significant. First the variable `age`, mother's age at start of the interval, and second `year`, the calendar year at the start of the interval. The variable `ses`, socioeconomic status, is also significant, but not to the extent of the previous two.

Now take a look at the actual fit.

```
> fit.lognorm
Call:
phreg(formula = Y ~ age + year + ses, data = f12, dist = "lognormal")
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
age                27.151    -0.046     0.955     0.005     0.000
year             1858.664     0.004     1.004     0.002     0.078
ses
          farmer    0.449     0         1           (reference)
         unknown    0.190    -0.282     0.754     0.069     0.000
           upper    0.024    -0.256     0.774     0.171     0.135
           lower    0.336    -0.090     0.914     0.055     0.104

log(scale)                    0.763     2.144     0.013     0.000
log(shape)                    0.604     1.830     0.018     0.000

Events                    1657
Total time at risk        4500.5
Max. log. likelihood       -2598
LR test statistic         94.7
Degrees of freedom        5
Overall p-value           0
```

The estimated coefficient for `age`, -0.046, is negative, indicating lower risk of a birth with increasing age. The effect of calendar time is positive and indicates an increase in the intensity of giving birth of the size approximately 4 per thousand. Regarding *socioeconomic status*, `farmer` wives have the highest fertility and the `unknown` and `upper` classes the lowest, when it comes to getting a second child. Finally, in Figure 6.3 it is clear that the hazard function first increases to a maximum around 3 years after the first birth and then decreases. This is quite a different picture compared to the Weibull fit in Figure 6.2! This shows the weakness with the Weibull model; it allows only for *monotone* (increasing or decreasing) hazard functions. The figure was created by the following **R** code:

```
> oldpar <- par(mfrow = c(1, 2))
> plot(fit.lognorm, fn = "haz", main = "Hazard function")
> plot(fit.lognorm, fn = "cum", main = "Cumulative hazard function")
> par(oldpar)
```

6.2.3 Comparing the Weibull and Lognormal Fits

From looking at Figures 6.2 and 6.3, it seems obvious that at least one of the fits must be less good: The *Weibull* cumulative hazards curve is convex and the *lognormal* one is concave, and the estimates of the two hazard functions are very different in shape. There are two direct ways of comparing the fits.

The first way to compare is to look at the maximized log likelihoods. For the *Weibull* fit it is -3140.04, and for the *lognormal* fit it is -2598.01. The

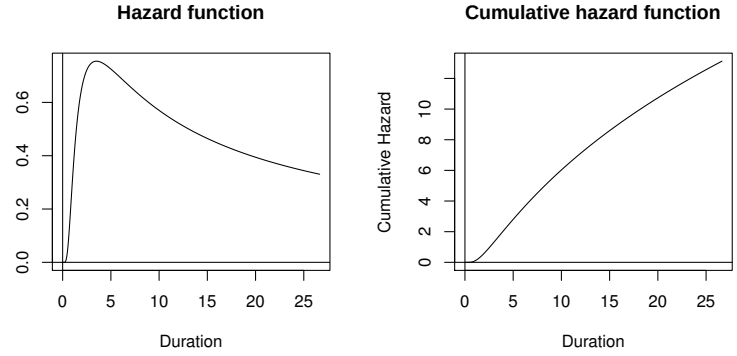


FIGURE 6.3
Estimated *lognormal* baseline distribution for length of birth intervals.

rule is that the largest value wins, so the *lognormal* model is the clear winner. Note that we do not claim to perform a formal test here; a likelihood ratio test would require that the two models we want to compare be nested, but that is not the case here. This is (here) equivalent to using the AIC as a measure of comparison, because the two models have the same number of parameters to estimate.

The second method of comparison is graphical. We can plot the cumulative hazard functions against the nonparametric estimate from a Cox regression fit, and judge which looks closer. It starts by fitting a Cox model with the same covariates:

```
> fit.cr <- coxreg(Y ~ age + year + ses, data = f12)
```

Then the function `check.dist` (from `eha`) is called. We start with the *Weibull* fit: This fit (Figure 6.4) looks very poor: One reason is that the longest waiting time for an observed event is about 12 years, while some women simply do not get more than one child, with a long waiting time ending in a censoring as a result. The time span from around 12 to 25 simply has no events, and the nonparametric estimate of the hazard function in that interval is zero. The *Weibull* distribution fit cannot cope with that.

One possibility is to try to fit a *Weibull* distribution only for the 12 first years of duration. Note below that after calling `age.window`, Y *must* be recreated! This is the danger with this "lazy" approach.

```
> f12$enter <- rep(0, NROW(f12))
> f12.cens <- age.window(f12, c(0, 12),
                         surv = c("enter", "next.ivl", "event"))
> f12.cens$Y <- Surv(f12.cens$enter, f12.cens$next.ivl,
                     f12.cens$event)
```

```
> check.dist(fit.cr, fit.w)
```

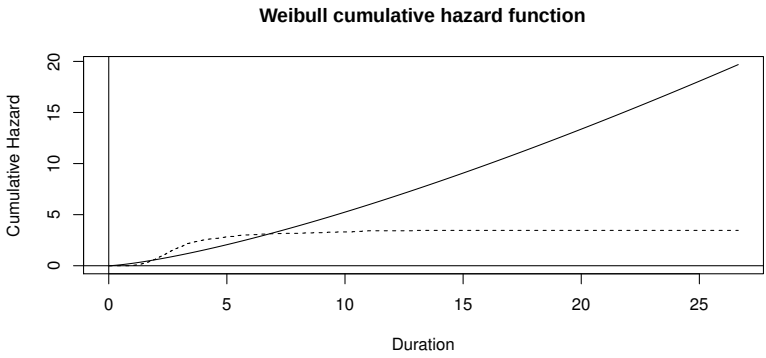


FIGURE 6.4
Check of the *Weibull* model, birth intervals.

```
> fit.wc <- phreg(Y ~ age + year + ses, data = f12.cens)
> fit.c <- coxreg(Y ~ age + year + ses, data = f12.cens)
> fit.wc
Call:
phreg(formula = Y ~ age + year + ses, data = f12.cens)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
age                27.267    -0.050     0.951     0.006     0.000
year             1858.953     0.002     1.002     0.002     0.237
ses
          farmer    0.456     0         1           (reference)
         unknown    0.182    -0.262     0.770     0.070     0.000
           upper    0.022    -0.204     0.816     0.173     0.239
           lower    0.340    -0.077     0.926     0.056     0.171

log(scale)                    1.036     2.818     0.016     0.000
log(shape)                    0.454     1.574     0.016     0.000

Events                    1656
Total time at risk          4311
Max. log. likelihood      -2927.7
LR test statistic         107
Degrees of freedom        5
Overall p-value           0
```

Compare the *Weibull* fit with what we had without cutting off durations at 12. The comparison with the nonparametric cumulative hazards function is now seen in Figure 6.5. This wasn't much better. Let's look what happens with the *lognormal* distribution.

We continue using the data censored at 12 and refit the *lognormal* model.

```
> fit.lnc <- phreg(Y ~ age + year + ses, data = f12.cens,
                   dist = "lognormal")
```

Then, in Figure 6.6 we check the fit against the nonparametric fit again. It is not very much better. In fact, this empirical distribution has features that the ordinary parametric distributions cannot cope with: Its hazard function starts off being zero for almost one year (birth intervals are rarely shorter than one year), then it grows rapidly and soon decreases towards zero again.

There is, however one parametric distribution that is flexible enough: The piecewise constant hazards distribution.

6.2.4 The Piecewise Constant Hazards (PCH) Model

The pch distribution is flexible because you can add as many parameters as you want. We will try it on the birth interval data. We start off with just four intervals, and the noncensored data set.

```
> fit.pch <- phreg(Surv(next.ivl, event) ~ age + year + ses,
                   data = f12, dist = "pch", cuts = c(4, 8, 12))
> fit.c <- coxreg(Surv(next.ivl, event) ~ age + year + ses,
                  data = f12)
> fit.pch
Call:
phreg(formula = Surv(next.ivl, event) ~ age + year + ses, data = f12,
    dist = "pch", cuts = c(4, 8, 12))


> check.dist(fit.c, fit.wc)
```

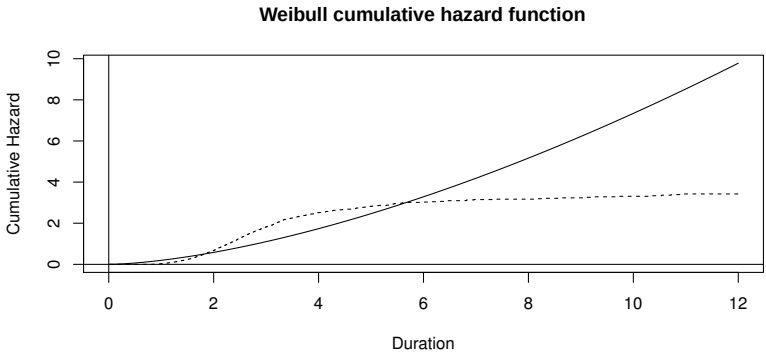


FIGURE 6.5
Check of the *Weibull* model, birth intervals censored at 12.

```
> check.dist(fit.c, fit.lnc)
```

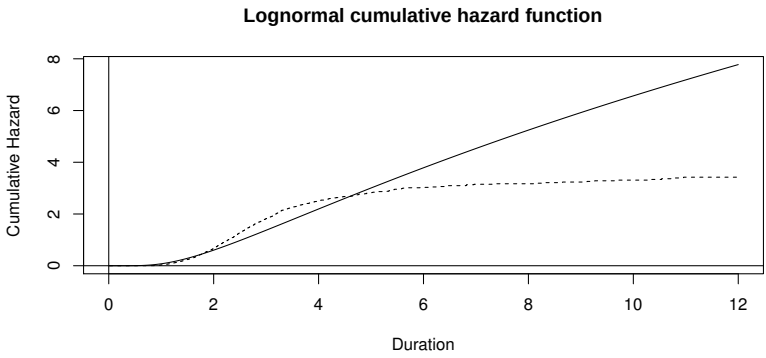


FIGURE 6.6
Check of the *lognormal* model, birth intervals censored at 12.

```
Covariate           W.mean      Coef Exp(Coef)  se(Coef)    Wald p
age                 27.151    -0.031     0.969     0.006     0.000
year              1858.664     0.001     1.001     0.002     0.622
ses
          farmer     0.449     0          1           (reference)
         unknown     0.190    -0.114     0.892     0.070     0.102
           upper     0.024    -0.033     0.968     0.173     0.849
           lower     0.336    -0.051     0.950     0.056     0.361


Events                    1657
Total time at risk        4500.5
Max. log. likelihood      -534.5
LR test statistic         39.2
Degrees of freedom        5
Overall p-value           2.15744e-07
```

Then we check the fit in Figure 6.7. This is not good enough. We need shorter intervals close to zero, so the time span is cut into one-year-long pieces:

```
> fit.pch <- phreg(Surv(next.ivl, event) ~ age + year + ses,
                   data = f12, dist = "pch", cuts = 1:13)
> fit.pch
Call:
phreg(formula = Surv(next.ivl, event) ~ age + year + ses, data = f12,
    dist = "pch", cuts = 1:13)
Covariate           W.mean      Coef Exp(Coef)  se(Coef)    Wald p
age                 27.151    -0.040     0.961     0.005     0.000
```

```
> check.dist(fit.c, fit.pch)
```

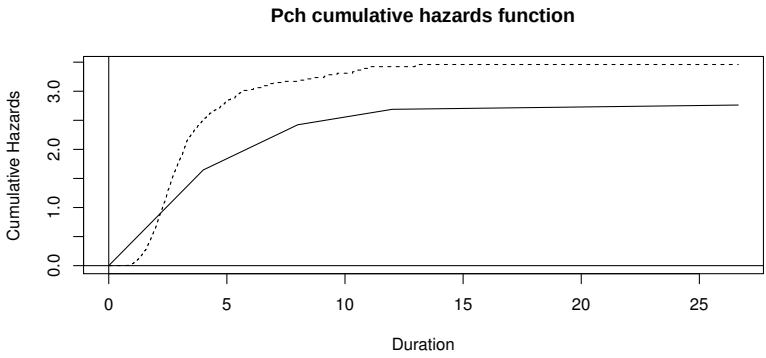


FIGURE 6.7
Check of the *pch* model, uncensored birth intervals.

```
year              1858.664     0.002     1.002     0.002     0.372
ses
          farmer     0.449         0         1           (reference)
         unknown     0.190    -0.113     0.893     0.070     0.106
           upper     0.024    -0.006     0.994     0.173     0.973
           lower     0.336    -0.070     0.932     0.056     0.212

Events                    1657
Total time at risk        4500.5
Max. log. likelihood      -2349.8
LR test statistic         67.9
Degrees of freedom        5
Overall p-value           2.83995e-13
```

This gives us 14 intervals with constant baseline hazard, that is, 14 parameters to estimate only for the baseline hazard function! But the fit is good; see Figure 6.8. In fact, it is almost perfect!

One of the advantages with parametric models is that it is easy to study and plot the hazard function, and not only the cumulative hazards function, which is the dominant tool for (graphical) analysis in the case of the nonparametric model of Cox regression. For the piecewise constant model just fitted, we get the estimated hazard function in Figure 6.9. The peak is reached during the third year of waiting, and after that the intensity of giving birth drops fast, and it becomes zero after 12 years.

What are the implications for the estimates of the regression parameters of the different choices of parametric model? Let us compare the regression

```
> check.dist(fit.c, fit.pch)
```

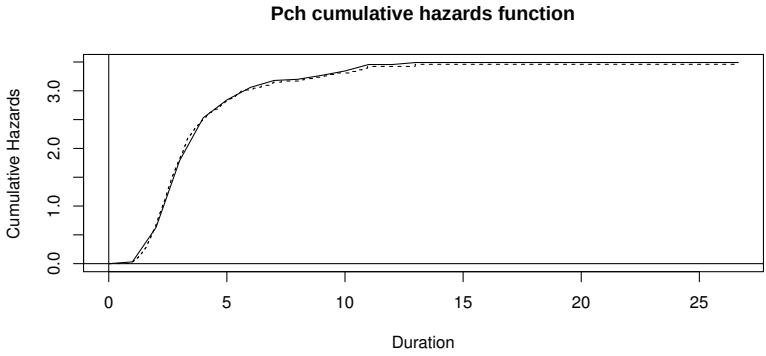


FIGURE 6.8
Check of the *pch* model with 13 constant hazard intervals, uncensored birth intervals.

```
> plot(fit.pch, fn = "haz")
```

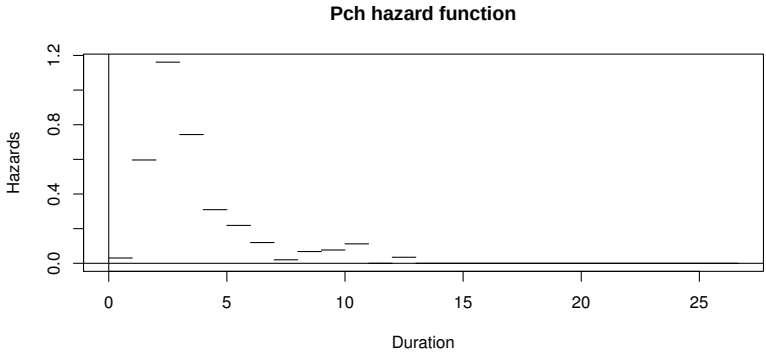


FIGURE 6.9
The estimated hazard function for the length of birth intervals, piecewise constant hazards distribution.

parameter estimates only for the three distributional assumptions, *Weibull*, *lognormal*, and *pch*. See Table 6.1 We can see that the differences are not large. However, the two assumptions closest to the "truth," the Cox model and the Pch model, are very close, while the other two have larger deviations. So an appropriate choice of parametric model seems to be somewhat important,

```
              Cox   Pch Weibull Lognormal
age         0.961 0.961   0.961     0.955
year        1.002 1.002   1.005     1.004
sesunknown  0.897 0.893   0.698     0.754
sesupper    1.018 0.994   0.683     0.774
seslower    0.930 0.932   0.908     0.914
```

contrary to "common knowledge," which says that it is not so important to have a good fit of the baseline hazard, if the only interest lies in the regression parameters.

6.2.4.1 Testing the Proportionality Assumption with the PCH Model

One advantage with the piecewise constant hazards model is that it is easy to test the assumption of proportional hazards. But it cannot be done directly with the `phreg` function. There is, however, a simple alternative. Before we show how to do it, we must show how to utilize Poisson regression when analyzing the pch model. We do it by reanalyzing the birth intervals data.

First, the data set is split up after the `cuts` in `strata`. This is done with the aid of the `survSplit` function in the `survival` package.

```
> f12 <- fert[fert$parity == 1, ]
> f12$enter <- 0 # 0 expands to a vector
> f12.split <- survSplit(f12, cut = 1:13, start = "enter",
                         end = "next.ivl", event = "event",
                         episode = "ivl")
> head(f12.split)
  id parity age year next.ivl event prev.ivl     ses parish
1  1      1  25 1826        1     0    0.411  farmer    SKL
2  2      1  19 1821        1     0    0.304 unknown    SKL
3  3      1  24 1827        1     0    0.772  farmer    SKL
4  4      1  35 1838        1     0    6.787 unknown    SKL
5  5      1  28 1832        1     0    3.031  farmer    SKL
6  6      1  25 1829        1     0    0.819   lower    SKL
  enter ivl
1     0   0
2     0   0
3     0   0
4     0   0
5     0   0
6     0   0
```

To see better what happened, we `sort` the new data frame by `id` and `enter`.

```
> f12.split <- f12.split[order(f12.split$id, f12.split$enter), ]
> head(f12.split)
```

```
     id parity age year next.ivl event prev.ivl    ses parish
1     1      1  25 1826        1     0    0.411 farmer    SKL
1841  1      1  25 1826        2     0    0.411 farmer    SKL
3681  1      1  25 1826        3     0    0.411 farmer    SKL
5521  1      1  25 1826        4     0    0.411 farmer    SKL
7361  1      1  25 1826        5     0    0.411 farmer    SKL
9201  1      1  25 1826        6     0    0.411 farmer    SKL
     enter ivl
1        0   0
1841     1   1
3681     2   2
5521     3   3
7361     4   4
9201     5   5
```

We see that a new variable, `ivl`, is created. It tells us which interval we are looking at. You may notice that the variables `enter` and `ivl` happen to be equal here; it is because our intervals start by 0, 1, 2, ..., 13, as given by `cuts` above.

In the Poisson regression to follow, we model the probability that `event = 1`. This probability will depend on the length of the studied interval, here `next.ivl - enter`, which for our data mostly equals one. The exact value we need is the logarithm of this difference, and it will be entered as an `offset` in the model.

Finally, one parameter per interval is needed, corresponding to the piecewise constant hazards. This is accomplished by including `ivl` as a `factor` in the model. Thus, we get

```
> f12.split$offs <- log(f12.split$next.ivl - f12.split$enter)
> f12.split$ivl <- as.factor(f12.split$ivl)
> fit12.pn <- glm(event ~ offset(offs) +
                    age + year + ses + ivl,
                    family = "poisson", data = f12.split)
> drop1(fit12.pn, test = "Chisq")
Single term deletions
Model:
event ~ offset(offs) + age + year + ses + ivl
       Df Deviance    AIC     LRT   Pr(Chi)
<none>      4699.7 8051.7
age     1   4756.2 8106.2   56.48 5.682e-14
year    1   4700.5 8050.5    0.80    0.3723
ses     3   4703.0 8049.0    3.29    0.3484
ivl    13   6663.7 9989.7 1964.00 < 2.2e-16
```

The piecewise cutting (`ivl`) is very statistically significant, but some caution with the interpretation is recommended: The cut generated 13 parameters, and there may be too few events in some intervals. This easily checked with the function `tapply`, which is very handy for doing calculations on subsets of

the data frame. In this case we want to sum the number of `event`s in each `ivl`:

```
> tapply(f12.split$event, f12.split$ivl, sum)
  0   1   2   3   4   5   6   7   8   9  10  11  12  13
 56 819 585 130  32  16   7   1   3   3   4   0   1   0
```

A reasonable interpretation of this is to restrict attention to the 10 first years; only one birth occurs after a longer waiting time. Then it would be reasonable to collapse the intervals in $(7, 10]$ to one interval. Thus

```
> fc <- age.window(f12.split, c(0, 11),
                   surv = c("enter", "next.ivl", "event"))
> levels(fc$ivl) <- c(0:6, rep("7-11", 7))
> tapply(fc$event, fc$ivl, sum)
   0    1    2    3    4    5    6 7-11
  56  819  585  130   32   16    7   11
```

Note two things here. First, the use of the function `age.window`, and second the `levels` function. The first makes an "age cut" in the Lexis diagram; that is, all spells are censored at (exact) age 11. The second is more intricate; factors can be tricky to handle. The problem here is that after `age.window`, the new data frame contains only individuals with values `0, 1, ..., 11` on the factor variable `ivl`, but it is still *defined* as a factor with 14 levels. The call to `levels` collapses the last seven to "`7-11`".

Now we rerun the analysis with the new data frame.

```
> fit <- glm(event ~  offset(offs) + age + year + ses + ivl,
             family = "poisson", data = fc)
> drop1(fit, test = "Chisq")
Single term deletions
Model:
event ~ offset(offs) + age + year + ses + ivl
       Df Deviance    AIC     LRT   Pr(Chi)
<none>      4696.2 8034.2
age     1   4752.6 8088.6   56.48 5.679e-14
year    1   4697.0 8033.0    0.82    0.3650
ses     3   4699.5 8031.5    3.32    0.3453
ivl     7   6492.9 9816.9 1796.78 < 2.2e-16
```

This change in categorization does not change the general conclusions about statistical significance at all.

Now let us include interactions with `ivl` in the model.

```
> fit.ia <- glm(event ~  offset(offs) + (age + year + ses) * ivl,
                family = "poisson", data = fc)
> drop1(fit.ia, test = "Chisq")
```

```
Single term deletions
Model:
event ~ offset(offs) + (age + year + ses) * ivl
         Df Deviance    AIC     LRT Pr(Chi)
<none>        4646.4 8054.4
age:ivl   7   4661.7 8055.7 15.3110 0.03221
year:ivl  7   4651.0 8045.0  4.5477 0.71497
ses:ivl  21   4676.2 8042.2 29.7964 0.09616
```

There is an apparent interaction with `age`, which is not very surprising; younger mothers will have a longer fertility period ahead than old mothers. The other interactions do not seem to be very important, so we remove them and rerun the model once again.

```
> fit.ia <- glm(event ~  offset(offs) + year + ses + age * ivl,
                family = "poisson", data = fc)
> drop1(fit.ia, test = "Chisq")
Single term deletions
Model:
event ~ offset(offs) + year + ses + age * ivl
        Df Deviance    AIC     LRT Pr(Chi)
<none>       4681.1 8033.1
year     1   4682.1 8032.1  0.9707 0.32451
ses      3   4684.1 8030.1  3.0028 0.39119
age:ivl  7   4696.2 8034.2 15.0437 0.03544
```

Let us look at the parameter estimates.

```
> summary(fit.ia)
Call:
glm(formula = event ~ offset(offs) + year + ses + age * ivl,
    family = "poisson", data = fc)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9217  -0.8446  -0.2513   0.5580   3.6734

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.720e+00  3.986e+00  -1.686 0.091839
year         2.109e-03  2.139e-03   0.986 0.324283
sesunknown  -1.051e-01  6.976e-02  -1.506 0.132006
sesupper    -2.611e-02  1.732e-01  -0.151 0.880140
seslower    -7.134e-02  5.609e-02  -1.272 0.203388
age         -2.363e-02  2.833e-02  -0.834 0.404334
ivl1         2.970e+00  7.777e-01   3.819 0.000134
ivl2         4.231e+00  7.834e-01   5.401 6.62e-08
ivl3         4.544e+00  8.657e-01   5.248 1.53e-07
ivl4         3.174e+00  1.090e+00   2.912 0.003592
ivl5         3.505e+00  1.382e+00   2.537 0.011193
```

```
ivl6          5.074e+00  2.374e+00   2.137 0.032604
ivl7-11       2.028e+00  1.877e+00   1.080 0.279928
age:ivl1     -3.914e-07  2.911e-02   0.000 0.999989
age:ivl2     -2.267e-02  2.933e-02  -0.773 0.439413
age:ivl3     -5.040e-02  3.227e-02  -1.562 0.118364
age:ivl4     -3.171e-02  3.947e-02  -0.804 0.421665
age:ivl5     -5.681e-02  5.091e-02  -1.116 0.264416
age:ivl6     -1.406e-01  9.315e-02  -1.509 0.131199
age:ivl7-11  -4.787e-02  6.993e-02  -0.685 0.493601

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 6545.4  on 5201  degrees of freedom
Residual deviance: 4681.1  on 5182  degrees of freedom
AIC: 8033.1

Number of Fisher Scoring iterations: 6
```

We can see (admittedly not clearly) a tendency of decreasing intensity in the later intervals with higher age.

This can actually also be done with `phreg` and the *Weibull* model and fixed `shape` at one (i.e., an *exponential* model).

```
> fit.exp <- phreg(Surv(enter, next.ivl, event) ~  year + ses +
                     age * ivl, dist = "weibull", shape = 1,
                     data = fc)
> drop1(fit.exp, test = "Chisq")
Single term deletions
Model:
Surv(enter, next.ivl, event) ~ year + ses + age * ivl
        Df    AIC     LRT Pr(Chi)
<none>     4630.1
year     1 4629.0  0.9707 0.32451
ses      3 4627.1  3.0028 0.39119
age:ivl  7 4631.1 15.0437 0.03544
```

The parameter estimates are now presented like this.

```
> fit.exp
Call:
phreg(formula = Surv(enter, next.ivl, event) ~ year + ses + age *
    ivl, data = fc, dist = "weibull", shape = 1)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
year             1858.968     0.002     1.002     0.002     0.324
ses
          farmer    0.456     0         1           (reference)
         unknown    0.181    -0.105     0.900     0.070     0.132
           upper    0.022    -0.026     0.974     0.173     0.880
           lower    0.340    -0.071     0.931     0.056     0.203
```

```
age                27.267    -0.024     0.977     0.028     0.404
ivl
               0    0.420     0          1           (reference)
               1    0.316     2.970     19.494     0.778     0.000
               2    0.119     4.231     68.801     0.783     0.000
               3    0.044     4.544     94.030     0.866     0.000
               4    0.027     3.174     23.909     1.090     0.004
               5    0.019     3.505     33.288     1.382     0.011
               6    0.015     5.074    159.762     2.374     0.033
            7-11    0.041     2.028      7.602     1.877     0.280
age:ivl
   :1                        -0.000     1.000     0.029     1.000
   :2                        -0.023     0.978     0.029     0.439
   :3                        -0.050     0.951     0.032     0.118
   :4                        -0.032     0.969     0.039     0.422
   :5                        -0.057     0.945     0.051     0.264
   :6                        -0.141     0.869     0.093     0.131
   :7-11                     -0.048     0.953     0.070     0.494

log(scale)                    1.601     4.956     0.052     0.000

 Shape is fixed at  1

Events                    1656
Total time at risk        4279.3
Max. log. likelihood       -2296
LR test statistic         1864
Degrees of freedom        19
Overall p-value           0
```

A lot of information, but the same content as in the output from the Poisson regression. However, here we get the exponentiated parameter estimates, in the column "*Exp(Coef)*". These numbers are *risk ratios* (or relative risks), and easier to interpret.

What should be done with the interactions? The simplest way to go to get an easy-to-interpret result is to categorize `age` and make separate analyzes, one for each category of `age`. Let us do that; first we have to decide on how to categorize: there should not be too many categories, and not too few mothers (or births) in any category. As a starting point, take three categories by cutting `age` at (exact) ages 25 and 30:

```
> fc$agegrp <- cut(fc$age, c(0, 25, 30, 100),
                   labels = c("< 25", "25-30", ">= 30"))
> table(fc$agegrp)
 < 25 25-30 >= 30
 2352  1658  1192
```

Note the use of the function `cut`. It "cuts" a continuous variable into pieces, defined by the second argument. Note that we must give lower and upper

bounds; I chose 0 and 100, respectively. They are, of course, not near real ages, but I am sure that no value falls outside the interval $(0, 100)$, and that is the important thing here. Then I can (optionally) give names to each category with the argument `labels`.

The tabulation shows that the oldest category contains relatively few mothers. Adding to that, the oldest category will probably also have fewer births. We can check that with the aid of the function `tapply`.

```
> tapply(fc$event, fc$agegrp, sum)
 < 25 25-30 >= 30
  796   567   293
```

Now, rerun the last analysis for each `agegrp` separately. For illustration only, we show how to do it for the first age group:

```
> fit.ses1 <- phreg(Surv(enter, next.ivl, event) ~  year + ses + ivl,
   dist = "weibull", shape = 1, data = fc[fc$agegrp == "< 25", ])
> fit.ses1
Call:
phreg(formula = Surv(enter, next.ivl, event) ~ year + ses + ivl,
    data = fc[fc$agegrp == "< 25", ], dist = "weibull", shape = 1)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
year             1854.228     0.001     1.001     0.003     0.838
ses
          farmer    0.461     0         1           (reference)
         unknown    0.256    -0.017     0.983     0.088     0.848
           upper    0.019    -0.269     0.764     0.308     0.382
           lower    0.264    -0.101     0.904     0.087     0.244
ivl
               0    0.433     0         1           (reference)
               1    0.326     2.895    18.076     0.196     0.000
               2    0.117     3.644    38.248     0.198     0.000
               3    0.037     3.296    27.016     0.227     0.000
               4    0.021     2.363    10.618     0.328     0.000
               5    0.016     2.082     8.024     0.401     0.000
               6    0.012     1.850     6.359     0.486     0.000
            7-11    0.039     0.882     2.415     0.450     0.050

log(scale)                    1.495     4.461     0.073     0.000

 Shape is fixed at  1

Events                    796
Total time at risk        1921.8
Max. log. likelihood      -1060.7
LR test statistic         874
Degrees of freedom        11
Overall p-value           0
```

You may try to repeat this for all age groups and check if the regression parameters for `year` and `ses` are very different. Hint: They are not.

6.2.5 Choosing the best parametric model

For modeling survival data with parametric proportional hazards models, the distributions of the function `phreg` in the package `eha` are available. We show how to select a suitable parametric model with the following illustrative example.

Example 25 *Old Age Mortality*

The data set `oldmort` in the **R** package `eha` contains life histories for people aged 60 and above in the years 1860–1880, that is, 21 years. The data come from the *Demographic Data Base* at Umeå University, Umeå, Sweden, and cover the sawmill district of Sundsvall in the middle of Sweden. This was one of the largest sawmill districts in Europe in the late 19th century. The town Sundsvall is located in the district, which also contains a rural area, where farming was the main occupation.

```
> require(eha)
> data(oldmort)
> summary(oldmort)
       id                 enter             exit
 Min.   :765000603   Min.   :60.00   Min.   : 60.00
 1st Qu.:797001170   1st Qu.:60.00   1st Qu.: 63.88
 Median :804001545   Median :60.07   Median : 68.51
 Mean   :803652514   Mean   :64.07   Mean   : 69.89
 3rd Qu.:812001564   3rd Qu.:66.88   3rd Qu.: 74.73
 Max.   :826002672   Max.   :94.51   Max.   :100.00
   event           birthdate          m.id
 Mode :logical   Min.   :1765   Min.   :     6039
 FALSE:4524      1st Qu.:1797   1st Qu.:766000610
 TRUE :1971      Median :1805   Median :775000742
 NA's :0         Mean   :1804   Mean   :771271398
                 3rd Qu.:1812   3rd Qu.:783000743
                 Max.   :1820   Max.   :802000669
                                NA's   :     3155
      f.id               sex              civ
 Min.   :     2458   male  :2884   unmarried: 557
 1st Qu.:763000610   female:3611   married  :3638
 Median :772000649                 widow    :2300
 Mean   :762726961
 3rd Qu.:780001077
 Max.   :797001468
 NA's   :     3310
     ses.50      birthplace     imr.birth           region
 middle : 233   parish:3598   Min.   : 4.348   town    : 657
```

```
unknown:2565   region:1503   1st Qu.:12.709   industry:2214
upper  :  55   remote:1394   Median :14.234   rural   :3624
farmer :1562                 Mean   :15.209
lower  :2080                 3rd Qu.:17.718
                             Max.   :31.967
```

A short explanation of the variables: `id` is an identification number. It connects records (follow-up intervals) that belong to one person. `m.id, f.id` are mother's and father's identification numbers, respectively. `enter, exit`, and `event` are the components of the *survival object*, that is, start of interval, end of interval, and an indicator of whether the interval ends with a death, respectively.

Note that by design, individuals are followed from the day they are aged 60. In order to calculate the follow-up times, we should subtract 60 from the two columns `enter` and `exit`. Otherwise, when specifying a parametric survivor distribution, it would in fact correspond to a left-truncated (at 60) distribution. However, for a Cox regression, this makes no difference.

The covariate (factor) `ses.50` is the socioeconomic status at (approximately) age 50. The largest category is `unknown`. We have chosen to include it as a level of its own, rather than treating it as missing. One reason is that we cannot assume *missing at random* here. On the contrary, a missing value strongly indicates that the person belongs to a "lower" class, or was born early.

```
> om <- oldmort
> om$Y <- Surv(om$enter - 60, om$exit - 60, om$event)
> fit.w <- phreg(Y ~ sex + civ + birthplace, data = om)
> fit.w
Call:
phreg(formula = Y ~ sex + civ + birthplace, data = om)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
sex
            male    0.406     0         1           (reference)
          female    0.594    -0.248     0.780     0.048     0.000
civ
       unmarried    0.080     0         1           (reference)
         married    0.530    -0.456     0.634     0.081     0.000
           widow    0.390    -0.209     0.811     0.079     0.008
birthplace
          parish    0.574     0         1           (reference)
          region    0.226     0.055     1.056     0.055     0.321
          remote    0.200     0.062     1.064     0.059     0.299

log(scale)                    2.900    18.170     0.014     0.000
log(shape)                    0.526     1.691     0.019     0.000

Events                     1971
Total time at risk          37824
Max. log. likelihood       -7410
```

```
LR test statistic         55.1
Degrees of freedom        5
Overall p-value           1.24244e-10
```

Note how we introduced the new variable Y; the sole purpose of this is to get more compact formulas in the modeling.

Here we applied a Weibull baseline distribution (the `default` distribution in `phreg`; by specifying nothing, the *Weibull* is chosen). Now let us repeat this with all the distributions in the `phreg` package.

```
> ln <- phreg(Y ~ sex + civ + birthplace, data = om,
              dist = "lognormal")
> ll <- phreg(Y ~ sex + civ + birthplace, data = om,
              dist = "loglogistic")
> g <- phreg(Y ~ sex + civ + birthplace, data = om,
             dist = "gompertz")
> ev <- phreg(Y ~ sex + civ + birthplace, data = om,
              dist = "ev")
```

Then we compare the maximized log-likelihoods and choose the distribution with the largest value.

```
> xx <- c(fit.w$loglik[2], ln$loglik[2], ll$loglik[2],
          g$loglik[2], ev$loglik[2])
> names(xx) <- c("w", "ln", "ll", "g", "ev")
> xx
        w        ln        ll         g        ev
-7409.954 -7928.435 -7636.601 -7273.701 -7310.142
```

The *Gompertz* (g) distribution gives the largest value of the maximized log likelihood. Let us graphically inspect the fit;

```
> fit.c <- coxreg(Y ~ sex + civ + birthplace, data = om)
> check.dist(fit.c, g)
```

See Figure 6.10. The fit is obviously great during the first 30 years (ages 60–90), but fails somewhat thereafter. One explanation of "the failure" is that after age 90, not so many persons are still at risk (alive); another is that maybe the mortality levels off at very high ages (on a very high level, of course).

The *Gompertz* hazard function is shown in Figure 6.11.

```
> plot(g, fn = "haz")
```

It is an exponentially increasing function of time; the Gompertz model usually fits old age mortality very well.

The p-values measure the significance of a group compared to the reference category, but we would also like to have an overall p-value for the covariates (factors) as such, that is, answer the question "does birthplace matter?" This can be achieved by running an analysis without `birthplace`:

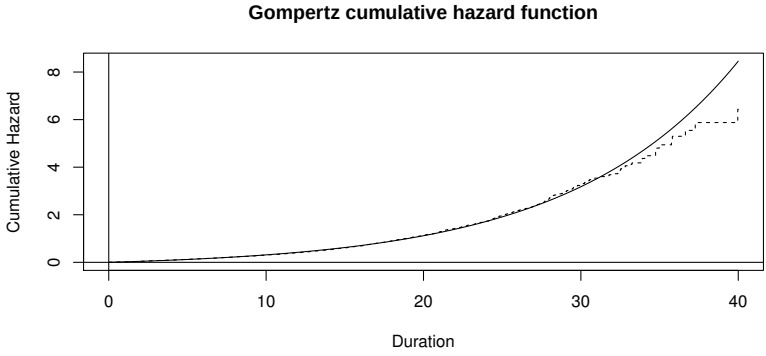


FIGURE 6.10
Graphical fit; *Gompertz* baseline versus the nonparametric (Cox regression, dashed).

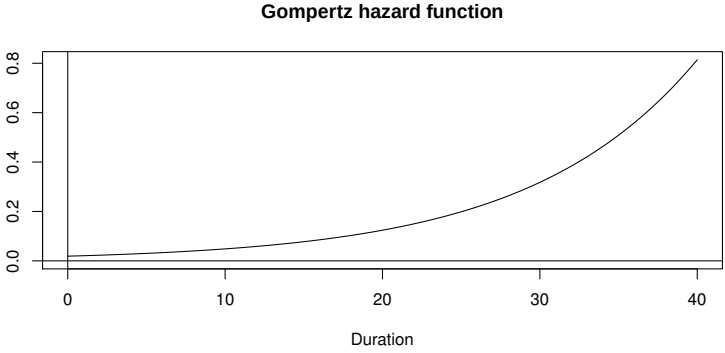


FIGURE 6.11
Estimated *Gompertz* hazard function for old age mortality data.

```
> fit.w1 <- phreg(Y ~ sex + civ, data = om)
> fit.w1
Call:
phreg(formula = Y ~ sex + civ, data = om)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
sex
            male    0.406     0         1           (reference)
          female    0.594    -0.249     0.780     0.047     0.000
civ
       unmarried    0.080     0         1           (reference)
         married    0.530    -0.455     0.634     0.081     0.000
```

```
            widow    0.390    -0.207     0.813     0.079     0.008

log(scale)                    2.901    18.188     0.014     0.000
log(shape)                    0.524     1.690     0.019     0.000

Events                     1971
Total time at risk          37824
Max. log. likelihood       -7410.8
LR test statistic          53.5
Degrees of freedom         3
Overall p-value            1.44385e-11
```

Then we compare the *max log likelihoods*: -7409.954 and -7410.763, respectively. The test statistic is 2 times the difference, 1.618. Under the null hypothesis of no `birthplace` effect on mortality, this test statistic has an approximate χ^2 distribution with 2 degrees of freedom. The degrees of freedom is the number of omitted parameters in the reduced model, two in this case. This because the factor `birthplace` has three levels.

There is a much simpler way of doing this, and that is to use the function `drop1`. As its name may suggest, it drops one variable at a time and reruns the fit, calculating the max log likelihoods and differences as above.

```
> drop1(fit.w, test = "Chisq")
Single term deletions
Model:
Y ~ sex + civ + birthplace
           Df   AIC    LRT   Pr(Chi)
<none>        14830
sex         1 14855 27.048 1.985e-07
civ         2 14866 39.989 2.073e-09
birthplace  2 14828  1.618    0.4453
```

As you can see, we do not only recover the test statistic for `birthplace`, we get the corresponding tests for all the involved covariates. Thus, it is obvious that both `sex` and `civ` have a statistically very significant effect on old age mortality, while `birthplace` does not mean much. Note that these effects are measured in the presence of the other two variables.

We can plot the baseline distribution by

```
> plot(fit.w)
```

with the result shown in Figure 6.12. We can see one advantage with parametric models here: They make it possible to estimate the hazard and density functions. This is much trickier with a semiparametric model like the Cox proportional hazards model. Another question is, of course, how well the *Weibull* model fits the data. One way to graphically check this is to fit a Cox regression model and compare the two cumulative hazards plots. This is done by using the function `check.dist`:

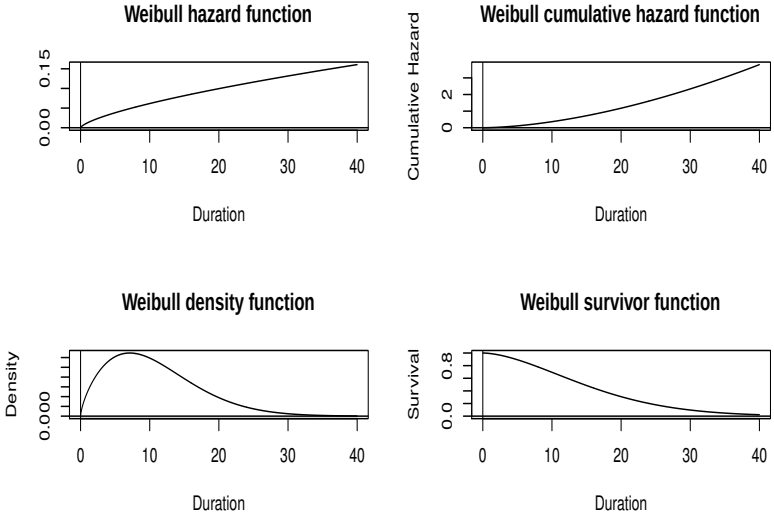


FIGURE 6.12
Baseline distribution of remaining life at 60 for an "average" person. *Weibull* model.

```
> fit <- coxreg(Y ~ sex + civ + birthplace, data = om)
> check.dist(fit, fit.w)
```

The result is shown in Figure 6.13. Obviously, the fit is not good; the *Weibull*

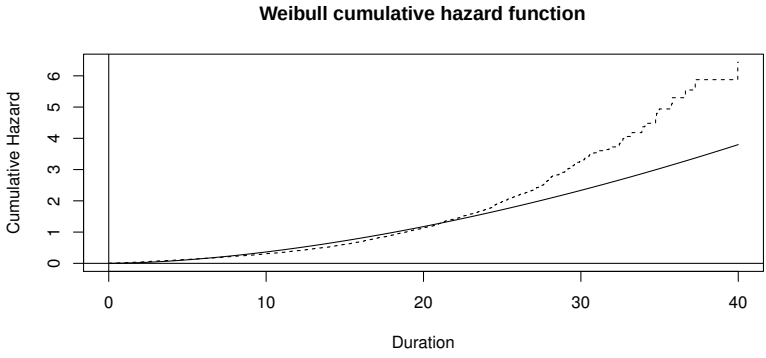


FIGURE 6.13
Check of a *Weibull* fit. The solid line is the cumulative hazards function from the *Weibull* fit, and the dashed line is the fit from a Cox regression.

model cannot capture the fast rise of the hazard by age. An exponentially increasing hazard function may be needed, so let us try the *Gompertz* distribution:

```
> fit.g <- phreg(Y ~ sex + civ + birthplace, data = om,
                 dist = "gompertz")
> fit.g

Call:
phreg(formula = Y ~ sex + civ + birthplace, data = om,
      dist = "gompertz")
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
sex
            male    0.406     0         1           (reference)
          female    0.594    -0.246     0.782     0.047     0.000
civ
       unmarried    0.080     0         1           (reference)
         married    0.530    -0.405     0.667     0.081     0.000
           widow    0.390    -0.265     0.767     0.079     0.001
birthplace
          parish    0.574     0         1           (reference)
          region    0.226     0.065     1.067     0.055     0.240
          remote    0.200     0.085     1.089     0.059     0.150

log(scale)                    2.364    10.631     0.032     0.000
log(shape)                   -3.968     0.019     0.045     0.000

Events                    1971
Total time at risk         37824
Max. log. likelihood      -7273.7
LR test statistic         45.5
Degrees of freedom        5
Overall p-value           1.14164e-08
```

One sign of a much better fit is the larger value of the maximized log likelihood, -7273.701 versus -7409.954 for the *Weibull* fit. The comparison to the Cox regression fit is given by

```
> check.dist(fit, fit.g)
```

with the result shown in Figure 6.14. This is a much better fit. The deviation that starts above 30 (corresponding to the age 90) is no big issue; in that range few people are still alive and the random variation takes over. Generally, it seems as if the *Gompertz* distribution fits old age mortality well.

The plots of the baseline *Gompertz* distribution are shown in Figure 6.15.

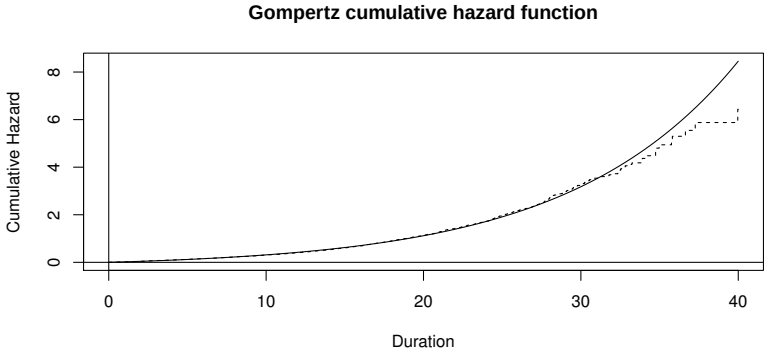


FIGURE 6.14
Check of a *Gompertz* fit. The solid line is the cumulative hazards function from the *Gompertz* fit, and the dashed line is the fit from a Cox regression.

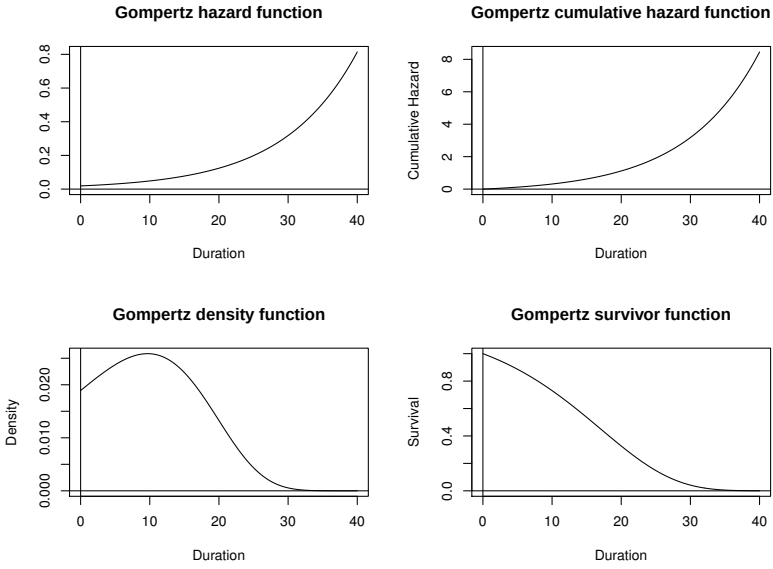


FIGURE 6.15
Baseline distribution of remaining life at 60 for an "average" person. *Gompertz* model.

6.3 Accelerated Failure Time Models

The accelerated failure time (AFT) model is best described through relations between survivor functions. For instance, comparing two groups:

Group 0: $P(T \geq t) = S_0(t)$ (control group)

Group 1: $P(T \geq t) = S_0(\phi t)$ (treatment group)

The model says that treatment *accelerates* failure time by the factor ϕ. If $\phi < 1$, treatment is good (prolongs life), otherwise bad. Another interpretation is that the *median* life length is *multiplied by* $1/\phi$ by treatment.

In Figure 6.16 the difference between the accelerated failure time and the proportional hazards models concerning the hazard functions is illustrated. The AFT hazard is not only multiplied by 2, it is also shifted to the left; the

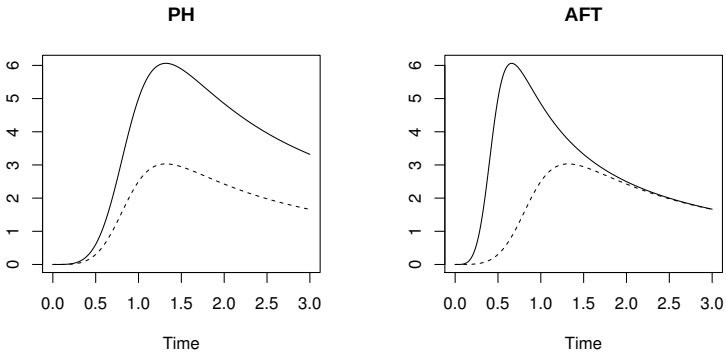


FIGURE 6.16
Proportional hazards (left) and accelerated failure time model (right). The baseline distribution is Loglogistic with shape 5 (dashed).

process is accelerated. Note how the hazards in the AFT case converges as time increases. This is usually a sign of the suitability of an AFT model.

6.3.1 The AFT Regression Model

If T has survivor function $S(t)$ and $T_c = T/c$, then T_c has survivor function $S(ct)$. Then, if $Y = \log(T)$ and $Y_c = \log(T_c)$, the following relation holds:

$$Y_c = Y - log(c).$$

With $Y = \epsilon$, $Y_c = Y$, and $\log(c) = -\boldsymbol{\beta}\mathbf{x}$ this can be written in familiar form:

$$Y = \boldsymbol{\beta}\mathbf{x} + \epsilon,$$

that is, an ordinary linear regression model for the log survival times. In the absence of right censoring and left truncation, this model can be estimated by least squares. However, the presence of these forms of incomplete data makes it necessary to rely on maximum likelihood methods. In **R**, there is the

function `aftreg` in the package `eha` and the function `survreg` in the package `survival` that perform the task of fitting AFT models.

Besides differing parametrizations, the main difference between `aftreg` and `survreg` is that the latter does not allow for left truncated data. One reason for this is that left truncation is a much harder problem to deal with in AFT models than in proportional hazards models.

Here we describe the implementation in `aftreg`. A detailed description of it can be found in Appendix B. The model is built around *scale-shape* families of distributions:

$$S^\star\big\{t,(\lambda,p)\big\} = S_0\left\{\left(\frac{t}{\lambda}\right)^p\right\}, \quad t>0;\ \lambda,p>0, \tag{6.4}$$

where S_0 is a fixed distribution. For instance, the exponential with parameter 1:

$$S_0(t) = e^{-t}, \quad t>0,$$

generates, through (6.4), the *Weibull* family (6.1) of distributions.

With time-constant covariate vector $\mathbf{x}$, the AFT model is

$$S(t;(\lambda,p,\boldsymbol{\beta}),\mathbf{x}) = S^\star\big\{t\exp(\boldsymbol{\beta}\mathbf{x}),(\lambda,p)\big\} = S_0\left\{\left(\frac{t\exp(\boldsymbol{\beta}\mathbf{x})}{\lambda}\right)^p\right\}, \quad t>0,$$

and this leads to the following description of the hazard function:

$$h(t;(\lambda,p,\boldsymbol{\beta}),\mathbf{x}) = \exp(\boldsymbol{\beta}\mathbf{x})h^\star\big(t\exp(\boldsymbol{\beta}\mathbf{x})\big) = \\ \exp(\boldsymbol{\beta}\mathbf{x})\frac{p}{\lambda}\left(\frac{t\exp(\boldsymbol{\beta}\mathbf{x})}{\lambda}\right)^{p-1} h_0\left\{\left(\frac{t\exp(\boldsymbol{\beta}\mathbf{x})}{\lambda}\right)^p\right\}, \quad t>0,$$

where $\mathbf{x} = (x_1,\ldots,x_p)$ is a vector of covariates, and $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_p)$ is the corresponding vector of regression coefficients.

6.3.2 Different Parametrizations

In **R**, there are two functions that can estimate AFT regression models, the function `aftreg` in the package `eha`, and the function `survreg` in the package `survival`. In the case of no time-varying covariates and no left truncation, they fit the same models (given a common baseline distribution), but use different parametrizations.

6.3.3 AFT Models in R

We repeat the examples from the proportional hazards section, but with AFT models instead.

Example 26 *Old age mortality*

For a description of this data set, see above. Here we fit an AFT model with the *Weibull* distribution. This should be compared to the proportional hazards model with the *Weibull* distribution, see earlier in this chapter.

```
> fit.w1 <- aftreg(Y ~ sex + civ + birthplace, data = om)
> fit.w1
Call:
aftreg(formula = Y ~ sex + civ + birthplace, data = om)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
sex
            male    0.406     0         1           (reference)
          female    0.594    -0.147     0.863     0.028     0.000
civ
       unmarried    0.080     0         1           (reference)
         married    0.530    -0.270     0.764     0.049     0.000
           widow    0.390    -0.124     0.884     0.046     0.008
birthplace
          parish    0.574     0         1           (reference)
          region    0.226     0.032     1.033     0.033     0.321
          remote    0.200     0.036     1.037     0.035     0.299

log(scale)                    2.639    13.993     0.049     0.000
log(shape)                    0.526     1.691     0.019     0.000

Events                    1971
Total time at risk         37824
Max. log. likelihood       -7410
LR test statistic         55.1
Degrees of freedom        5
Overall p-value           1.24228e-10
```

Note that the "Max. log. likelihood" is exactly the same, but the regression parameter estimates differ. The explanation for this is that (i) for the *Weibull* distribution, the AFT and the PH models are the same, and (ii) the only problem is that different parametrizations are used. The p-values and the signs of the parameter estimates should be the same. □

6.4 Proportional Hazards or AFT Model?

The problem of choosing between a proportional hazards and an accelerated failure time model (everything else equal) can be solved by comparing the AIC of the models. Since the numbers of parameters are equal in the two cases, this amounts to comparing the maximized likelihoods. For instance, in the case with *old age mortality*:

Let us see what happens with the *Gompertz* AFT model: Exactly the same procedure as with the *Weibull* distrbution, but we have to specify the *Gompertz* distribution in the call (remember, the *Weibull* distribution is the default choice, both for `phreg` and `aftreg`).

```
> fit.g1 <- aftreg(Y ~ sex + civ + birthplace, data = om,
                   dist = "gompertz")
> fit.g1
Call:
aftreg(formula = Y ~ sex + civ + birthplace, data = om,
       dist = "gompertz")
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
sex
            male    0.406     0         1           (reference)
          female    0.594    -0.081     0.922     0.020     0.000
civ
       unmarried    0.080     0         1           (reference)
         married    0.530    -0.152     0.859     0.034     0.000
           widow    0.390    -0.100     0.905     0.031     0.001
birthplace
          parish    0.574     0         1           (reference)
          region    0.226     0.022     1.022     0.023     0.340
          remote    0.200     0.038     1.039     0.025     0.132

log(scale)                    2.222     9.224     0.046     0.000
log(shape)                   -3.961     0.019     0.045     0.000

Events                    1971
Total time at risk         37824
Max. log. likelihood       -7280
LR test statistic         33
Degrees of freedom        5
Overall p-value           3.81949e-06
```

Comparing the corresponding result for the proportional hazards and the AFT models with the *Gompertz* distribution, we find that the maximized log likelihood in the former case is -7279.973, compared to -7273.701 for the latter. This indicates that the proportional hazards model fit is better. Note, however, that we cannot formally test the proportional hazards hypothesis; the two models are not nested.

6.5 Discrete Time Models

There are two ways of looking at discrete duration data; either time is truly discrete, i.e, the number of trials until an event occurs, or an approximation

due to rounding of continuous time data. In a sense, all data are discrete, because it is impossible to measure anything on a continuous scale with infinite precision, but from a practical point of view it is reasonable to say that data is discrete when tied events occur embarrassingly often.

When working with register data, time is often measured in years, which makes it necessary and convenient to work with discrete models. A typical data format with register data is the so-called *wide* format, where there is one record (row) per individual, and measurements for many years. We have so far only worked with the *long* format. The data sets created by `survSplit` are in long format; there is one record per individual and age category. The **R** workhorse in switching back and forth between the long and wide formats is the function `reshape`. It may look confusing at first, but if data follow some simple rules, it is quite easy to use `reshape`.

The function `reshape` is typically used with *longitudinal data*, where there are several measurements at different time points for each individual. If the data for one individual is registered within one record (row), we say that data are in wide format, and if there is one record (row) per time (several records per individual), data are in long format. Using wide format, the rule is that time-varying variable names must end in a numeric value indicating at which time the measurement was taken. For instance, if the variable `civ` (civil status) is noted at times 1, 2, and 3, there must be variables named `civ.1, civ.2, civ.3`, respectively. It is optional to use any *separator* between the base name (`civ`) and the time, but it should be one character or empty. The "." is what `reshape` expects by default, so using that form simplifies coding somewhat.

We start by creating an example data set as an illustration. This is accomplished by starting off with the data set `oldmort` in `eha` and "trimming" it.

```
> data(oldmort)
> om <- oldmort[oldmort$enter == 60, ]
> om <- age.window(om, c(60, 70))
> om$m.id <- om$f.id <- om$imr.birth <- om$birthplace <- NULL
> om$birthdate <- om$ses.50 <- NULL
> om1 <- survSplit(om, cut = 61:69, start = "enter", end = "exit",
                   event = "event", episode = "agegrp")
> om1$agegrp <- factor(om1$agegrp, labels = 60:69)
> om1 <- om1[order(om1$id, om1$enter), ]
> head(om1)
            id enter   exit event    sex   civ   region agegrp
1    800000625    60 61.000     0   male widow    rural     60
3224 800000625    61 62.000     0   male widow    rural     61
6447 800000625    62 63.000     0   male widow    rural     62
9670 800000625    63 63.413     0   male widow    rural     63
2    800000631    60 61.000     0 female widow industry     60
3225 800000631    61 62.000     0 female widow industry     61
```

We may change the row numbers and recode the `id` so they are easier to read.

```
> rownames(om1) <- 1:NROW(om1)
> om1$id <- as.numeric(as.factor(om1$id))
> head(om1)
  id enter   exit event    sex   civ   region agegrp
1  1    60 61.000     0   male widow    rural     60
2  1    61 62.000     0   male widow    rural     61
3  1    62 63.000     0   male widow    rural     62
4  1    63 63.413     0   male widow    rural     63
5  2    60 61.000     0 female widow industry     60
6  2    61 62.000     0 female widow industry     61
```

This is the long format; each individual has as many records as "presence ages." For instance, person No. 1 has four records, for the ages 60–63. The maximum possible No. of records for one individual is 10. We can check the distribution of No. of records per person by using the function `tapply`:

```
> recs <- tapply(om1$id, om1$id, length)
> table(recs)
recs
  1   2   3   4   5   6   7   8   9  10
400 397 351 315 307 250 192 208 146 657
```

It is easier to get to grips with the distribution with a graph, in this case a *barplot*; see Figure 6.17.

```
> barplot(table(recs))
```

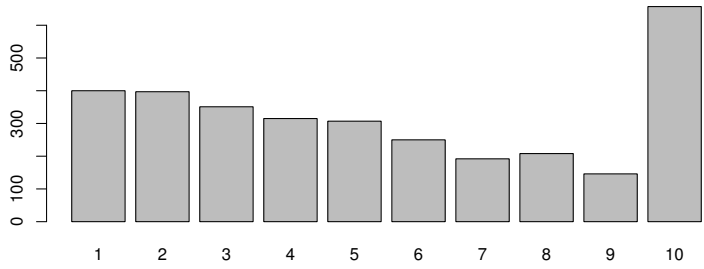


FIGURE 6.17
Barplot of the number of records per person.

Now, let us turn `om1` into a data frame in `wide` format. This is done with the function `reshape`. First, we remove the redundant variables `enter` and `exit`.

```
> om1$exit <- om1$enter <- NULL
> om2 <- reshape(om1, v.names = c("event", "civ", "region"),
                  idvar = "id", direction = "wide",
                  timevar = "agegrp")
> names(om2)
 [1] "id"        "sex"       "event.60"  "civ.60"    "region.60"
 [6] "event.61"  "civ.61"    "region.61" "event.62"  "civ.62"
[11] "region.62" "event.63"  "civ.63"    "region.63" "event.64"
[16] "civ.64"    "region.64" "event.65"  "civ.65"    "region.65"
[21] "event.66"  "civ.66"    "region.66" "event.67"  "civ.67"
[26] "region.67" "event.68"  "civ.68"    "region.68" "event.69"
[31] "civ.69"    "region.69"
```

Here there are two time-fixed variables, `id` and `sex`, and three time-varying variables, `event`, `civ`, and `region`. The time-varying variables have a suffix of the type `.xx`, where `xx` varies from 60 to 69.

This is how data in wide format usually show up; the suffix may start with something else than `.`, but it must be a single character, or nothing. The real problem is how to switch from wide format to long, because our survival analysis tools want it that way. The solution is to use `reshape` again, but other specifications.

```
> om3 <- reshape(om2, direction = "long", idvar = "id",
                  varying = 3:32)
> head(om3)
     id    sex time event     civ   region
1.60  1   male   60     0   widow    rural
2.60  2 female   60     0   widow industry
3.60  3   male   60     0 married     town
4.60  4   male   60     0 married     town
5.60  5 female   60     0 married     town
6.60  6 female   60     0 married     town
```

There is a new variable `time` created, which goes from 60 to 69, one step for each of the ages. We would like to have the file sorted primarily by `id` and secondarily by time.

```
> om3 <- om3[order(om3$id, om3$time), ]
> om3[1:11, ]
     id    sex time event   civ   region
1.60  1   male   60     0 widow    rural
1.61  1   male   61     0 widow    rural
1.62  1   male   62     0 widow    rural
1.63  1   male   63     0 widow    rural
1.64  1   male   64    NA  <NA>     <NA>
1.65  1   male   65    NA  <NA>     <NA>
1.66  1   male   66    NA  <NA>     <NA>
1.67  1   male   67    NA  <NA>     <NA>
```

```
1.68  1   male   68    NA  <NA>     <NA>
1.69  1   male   69    NA  <NA>     <NA>
2.60  2 female   60     0 widow industry
```

Note that all individuals got 10 records here, even those who only are observed for fewer years. Individual No. 1 is only observed for the ages 60–63, and the next six records are redundant; they will not be used in an analysis if kept, so it is from a practical point of view a good idea to remove them.

```
> NROW(om3)
[1] 32230
> om3 <- om3[!is.na(om3$event), ]
> NROW(om3)
[1] 17434
```

The data frame shrunk to almost half of what it was originally. First, let us summarize the data.

```
> summary(om3)
       id           sex             time           event
 Min.   :   1   male  : 7045   Min.   :60.00   Min.   :0.00000
 1st Qu.: 655   female:10389   1st Qu.:61.00   1st Qu.:0.00000
 Median :1267                  Median :63.00   Median :0.00000
 Mean   :1328                  Mean   :63.15   Mean   :0.02587
 3rd Qu.:1948                  3rd Qu.:65.00   3rd Qu.:0.00000
 Max.   :3223                  Max.   :69.00   Max.   :1.00000
        civ             region
 unmarried: 1565   town    :2485
 married  :11380   industry:5344
 widow    : 4489   rural   :9605
```

The key variables in the discrete time analysis are `event` and `time`. For the baseline hazard, we need one parameter per value of `time`, so it is practical to transform the continuous variable `time` to a factor.

```
> om3$time <- as.factor(om3$time)
> summary(om3)
       id           sex             time          event
 Min.   :   1   male  : 7045   60     :3223   Min.   :0.00000
 1st Qu.: 655   female:10389   61     :2823   1st Qu.:0.00000
 Median :1267                  62     :2426   Median :0.00000
 Mean   :1328                  63     :2075   Mean   :0.02587
 3rd Qu.:1948                  64     :1760   3rd Qu.:0.00000
 Max.   :3223                  65     :1453   Max.   :1.00000
                               (Other):3674
        civ             region
 unmarried: 1565   town    :2485
 married  :11380   industry:5344
 widow    : 4489   rural   :9605
```

The summary now produces a frequency table for `time`.

For a given time point and a given individual, the response is whether an event has occurred or not; that is, it is modeled as a *Bernoulli* outcome, which is a special case of the *binomial* distribution. The discrete time analysis may now be performed in several ways. It is most straightforward to run a logistic regression with `event` as response through the basic `glm` function with `family = binomial(link=cloglog)`. The so-called *cloglog* link is used in order to preserve the proportional hazards property.

```
> fit.glm <- glm(event ~ sex + civ + region + time,
                 family = binomial(link = cloglog), data = om3)
> summary(fit.glm)

Call:
glm(formula = event ~ sex + civ + region + time,
    family = binomial(link = cloglog), data = om3)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.4263  -0.2435  -0.2181  -0.1938   2.9503

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -3.42133    0.21167 -16.164  < 2e-16
sexfemale      -0.36332    0.09927  -3.660 0.000252
civmarried     -0.33683    0.15601  -2.159 0.030847
civwidow       -0.19008    0.16810  -1.131 0.258175
regionindustry  0.10784    0.14443   0.747 0.455264
regionrural    -0.22423    0.14034  -1.598 0.110080
time61          0.04895    0.18505   0.265 0.791366
time62          0.46005    0.17400   2.644 0.008195
time63          0.05586    0.20200   0.277 0.782134
time64          0.46323    0.18883   2.453 0.014162
time65          0.31749    0.20850   1.523 0.127816
time66          0.45582    0.21225   2.148 0.031751
time67          0.91511    0.19551   4.681 2.86e-06
time68          0.68549    0.22575   3.036 0.002394
time69          0.60539    0.24904   2.431 0.015064

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4186.8  on 17433  degrees of freedom
Residual deviance: 4125.2  on 17419  degrees of freedom
AIC: 4155.2

Number of Fisher Scoring iterations: 7
```

This output is not so pleasant, but we can anyway see that females (as usual) have lower mortality than males, that the married are better off than the unmarried, and that regional differences maybe are not so large. To get a

better understanding of the statistical significance of the findings, we run `drop1` on the fit.

```
> drop1(fit.glm, test = "Chisq")
Single term deletions
Model:
event ~ sex + civ + region + time
       Df Deviance    AIC    LRT   Pr(Chi)
<none>      4125.2 4155.2
sex     1   4138.5 4166.5 13.302 0.0002651
civ     2   4130.2 4156.2  4.978 0.0829956
region  2   4135.8 4161.8 10.526 0.0051810
time    9   4161.2 4173.2 36.005 3.956e-05
```

Mildly surprisingly, `civil status` is not that statistically significant, but `region` (and the other variables) is. The strong significance of the time variable is, of course, expected; mortality is expected to increase with higher age.

An equivalent way, with a nicer printed output, is to use the function `glmmboot` in the package `eha`.

```
> fit.boot <- glmmboot(event ~ sex + civ + region, cluster = time,
                       family = binomial(link = cloglog),
                       data = om3)
> fit.boot
Call:  glmmboot(formula = event ~ sex + civ + region,
       family = binomial(link = cloglog),
       data = om3, cluster = time)
                  coef se(coef)       z Pr(>|z|)
sexfemale      -0.3633  0.09929 -3.6593 0.000253
civmarried     -0.3368  0.15605 -2.1585 0.030900
civwidow       -0.1901  0.16813 -1.1305 0.258000
regionindustry  0.1078  0.14448  0.7464 0.455000
regionrural    -0.2242  0.14037 -1.5974 0.110000

Residual deviance: 4273 on 17419 degrees of freedom    AIC: 4303
```

The parameter estimates corresponding to `time` are contained in the variable `fit$frail`. They need to be transformed to get the "baseline hazards."

```
> haz <- plogis(fit.boot$frail)
> haz
 [1] 0.03163547 0.03317001 0.04920603 0.03339226 0.04935523
 [6] 0.04294934 0.04900860 0.07542353 0.06089140 0.05646886
```

A plot of the hazard function is shown in Figure 6.18. By some data manipulation, we can also use `eha` for the analysis. For that to succeed, we need intervals as responses, and the way to accomplish that is to add two variables, *exit* and *enter*. The latter must be *slightly* smaller than the former:

```
> barplot(haz)
```

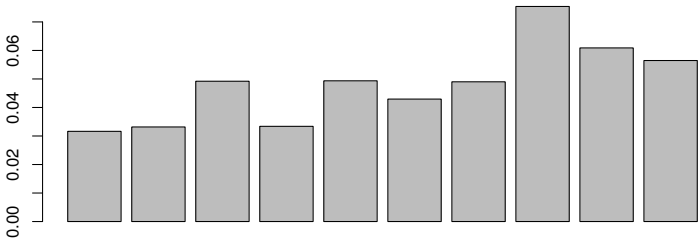


FIGURE 6.18
Baseline hazards, old age mortality.

```
> om3$exit <- as.numeric(as.character(om3$time))
> om3$enter <- om3$exit - 0.5
> fit.ML <- coxreg(Surv(enter, exit, event) ~ sex + civ + region,
                   method = "ml", data = om3)
> fit.ML
Call:
coxreg(formula = Surv(enter, exit, event) ~ sex + civ + region,
    data = om3, method = "ml")
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
sex
            male    0.404     0         1 (reference)
          female    0.596    -0.363     0.695     0.099     0.000
civ
       unmarried    0.090     0         1 (reference)
         married    0.653    -0.337     0.714     0.156     0.031
           widow    0.257    -0.190     0.827     0.168     0.258
region
            town    0.143     0         1 (reference)
        industry    0.307     0.108     1.114     0.144     0.455
           rural    0.551    -0.224     0.799     0.140     0.110

Events                    451
Total time at risk          8717
Max. log. likelihood      -2062.6
LR test statistic         27.2
Degrees of freedom        5
Overall p-value           5.14285e-05
```

Plots of the cumulative hazards and the survival function are easily produced; see Figures 6.19 and 6.20.

```
> plot(fit.ML, fn = "cum", xlim = c(60, 70))
```

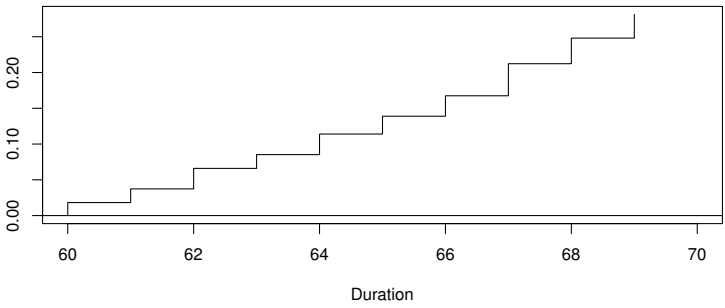


FIGURE 6.19
The cumulative hazards, from the coxreg fit.

```
> plot(fit.ML, fn = "surv", xlim = c(60, 70))
```

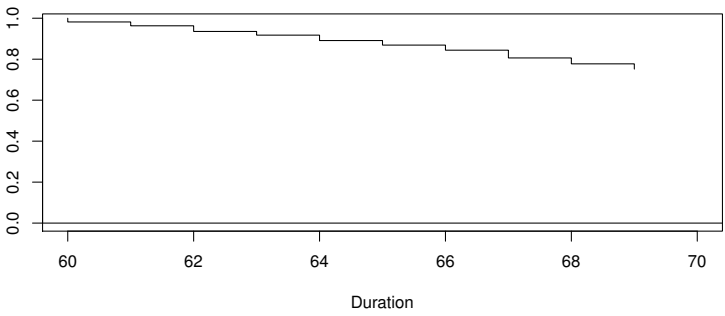


FIGURE 6.20
The survival function, from the coxreg fit.

Finally, the proportional hazards assumption can be tested in the discrete time framework by creating an interaction between `time` and the covariates in question. This is only possible by using `glm`.

```
> fit2.glm <- glm(event ~ (sex + civ + region) * time,
                  family = binomial(link = cloglog),
                  data = om3)
> drop1(fit2.glm, test = "Chisq")
Single term deletions
Model:
event ~ (sex + civ + region) * time
            Df Deviance    AIC    LRT Pr(Chi)
<none>           4077.1 4197.1
sex:time     9   4088.2 4190.2 11.116  0.2679
civ:time    18   4099.5 4183.5 22.425  0.2137
region:time 18   4093.7 4177.7 16.600  0.5508
```

There is no sign of nonproportionality.

This page intentionally left blank

7

Multivariate Survival Models

7.1 Introduction

Sometimes, survival data come in clusters, and multivariate, or *frailty*, models are appropriate to use.

Ever since the paper by Vaupel, Manton & Stallard (1979), the concept of frailty has spread in even wider circles of the research community. Although their primary purpose was to show various consequences of admitting individual frailties ("individuals age faster than cohorts," due to the selection effect), the effect was that people started to implement their frailty model in Cox regression models.

7.1.1 An Introductory Example

Let us assume that in a follow-up study, the cohort is not homogeneous but instead consists of two equally sized groups with differing hazard rates. Assume further that we have no indication of which group an individual belongs to, and that members of both groups follow an exponential life length distribution:

$$\begin{aligned} h_1(t) &= \lambda_1 \\ h_2(t) &= \lambda_2 \end{aligned} \qquad t > 0.$$

This implies that the corresponding survival functions S_1 and S_2 are

$$\begin{aligned} S_1(t) &= e^{-\lambda_1 t} \\ S_2(t) &= e^{-\lambda_2 t} \end{aligned} \qquad t > 0,$$

and a randomly chosen individual will follow the "population mortality" S, which is a *mixture* of the two distributions:

$$S(t) = \frac{1}{2}S_1(t) + \frac{1}{2}S_2(t), \quad t > 0.$$

Let us calculate the hazard function for this mixture. We start by finding the *density function* f:

$$f(t) = -\frac{dS(x)}{dx} = \frac{1}{2}\left(\lambda_1 e^{-\lambda_1 t} + \lambda_2 e^{-\lambda_2 t}\right), \quad t > 0.$$

Then, by the definition of h, we get

$$h(t) = \frac{f(t)}{S(t)} = \omega(t)\lambda_1 + \big(1 - \omega(t)\big)\lambda_2, \quad t > 0, \tag{7.1}$$

with

$$\omega(t) = \frac{e^{-\lambda_1 t}}{e^{-\lambda_1 t} + e^{-\lambda_2 t}}.$$

It is easy to see that

$$\omega(t) \to \begin{cases} 0, & \lambda_1 > \lambda_2 \\ \frac{1}{2}, & \lambda_1 = \lambda_2 \\ 1, & \lambda_1 < \lambda_2 \end{cases}, \quad \text{as } t \to \infty,$$

implying that

$$h(t) \to \min(\lambda_1, \lambda_2), \quad t \to \infty,$$

see Figure 7.1. The important point here is that it is *impossible* to tell from

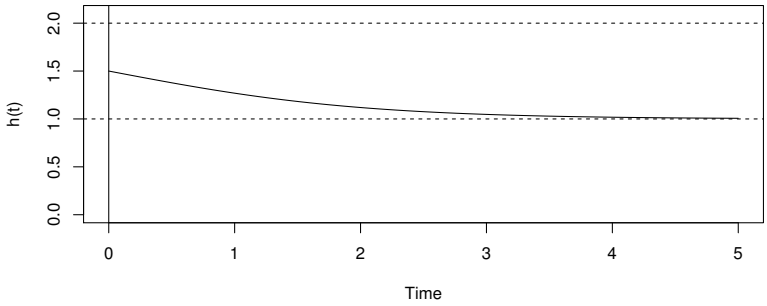


FIGURE 7.1
"Population hazard function" (solid line). The dashed lines are the hazard functions of each group, $\lambda_1 = 1$, $\lambda_2 = 2$.

data alone whether the population is homogeneous, with all individuals following the same hazard function (7.1), or if it in fact consists of two groups, each following a constant hazard rate. Therefore, individual frailty models (like $h_i(t) = Z_i h(t), \quad i = 1, \ldots, n$, where Z_i is the "frailty" for individual No. i, and $Z_1, \ldots, Z_n$ are independent and identically distributed (iid)) are less useful.

A heuristic explanation to all this is the dynamics of the problem: we follow a population (cohort) over time, and the *composition* of it changes over time. The weaker individuals die first, and the proportion stronger will steadily grow as time goes by.

Another terminology is to distinguish between *individual* and *population* hazards. In Figure 7.1 the solid line is the population hazard, and the dashed lines represent the two kinds of individual hazards present. Of course, in a truly homogeneous population, these two concepts coincide.

7.2 Frailty Models

Frailty models in survival analysis correspond to *hierarchical* models in linear or generalized linear models. They are also called *mixed effects models*. A general theory, with emphasis on using **R**, of mixed effects models can be found in (Pinheiro & Bates 2000).

7.2.1 The Simple Frailty Model

Vaupel et al. (1979) described an individual frailty model,

$$h(t; \mathbf{x}, Z) = h_0(t) Z e^{\boldsymbol{\beta}\mathbf{x}}, \quad t > 0,$$

where Z is assumed to be drawn independently for each individual. Hazard rates for "random survivors" are not proportional, but converging (to each other) if the frailty distribution has finite variance. Thus, the problem may be less pronounced in AFT than in PH regression. However, as indicated in the introductory example, with individual frailty the identification problems are large, and such models are best avoided.

7.2.2 The Shared Frailty Model

Frailty models work best when there is a natural grouping of the data, so that observations from the same group are dependent (*share* the same *frailty*), while two individual survival times from different groups can be regarded as independent. Such a model may be described as

$$h_i(t; \mathbf{x}) = h_{i0}(t) e^{\boldsymbol{\beta}\mathbf{x}}, \quad i = 1, \ldots, s;\ t > 0, \tag{7.2}$$

which simply is a stratified Cox regression model. By assuming

$$h_{i0}(t) = Z_i h_0(t), \quad i = 1, \ldots s;\ t > 0, \tag{7.3}$$

the traditional multivariate frailty model emerges. Here it is assumed that $Z_1, \ldots, Z_s$ are independent and identically distributed (*iid*), usually with a lognormal distribution. From (7.2) and (7.3) we get, with $U_i = \log(Z_i)$,

$$h_i(t; \mathbf{x}) = h_0(t) e^{\boldsymbol{\beta}\mathbf{x} + U_i}, \quad i = 1, \ldots, s;\ t > 0.$$

In this formulation, $U_1, \ldots, U_s$ are *iid* normal with mean zero and unknown variance σ^2. Another popular choice of distribution for the Z:s is the gamma distribution.

In **R**, the package `coxme` (Therneau 2011) fits frailty models. We look at the fertility data set in the **R** package `eha`.

```
> require(eha)
> data(fert)
> head(fert)
  id parity age year next.ivl event prev.ivl     ses parish
1  1      0  24 1825    0.411     1       NA  farmer    SKL
2  1      1  25 1826   22.348     0    0.411  farmer    SKL
3  2      0  18 1821    0.304     1       NA unknown    SKL
4  2      1  19 1821    1.837     1    0.304 unknown    SKL
5  2      2  21 1823    2.546     1    1.837 unknown    SKL
6  2      3  23 1826    2.541     1    2.546 unknown    SKL
```

It seems natural to assume that the lengths of birth intervals vary with mother, so we try a frailty model with `id` as the grouping variable. Also notice that the first interval for each woman is measured from marriage (only married women are included in this data set) to first birth, so we will start by removing them. They are characterized by `parity` being 0.

```
> fe <- fert[fert$parity != 0, ]
> require(coxme)
> fit <- coxme(Surv(next.ivl, event) ~ age + ses + parity +
                (1 | id), data = fe)
> fit
Cox mixed-effects model fit by maximum likelihood
  Data: fe
  events, n = 8458, 10312
  Iterations= 15 65
                    NULL Integrated    Fitted
Log-likelihood -71308.31  -69905.17 -68211.87
                    Chisq      df p     AIC      BIC
Integrated loglik 2806.28    6.00 0 2794.28  2752.02
 Penalized loglik 6192.88 1211.67 0 3769.54 -4764.08

Model:  Surv(next.ivl, event) ~ age + ses + parity + (1 | id)
Fixed coefficients
                  coef exp(coef)    se(coef)      z       p
age        -0.07960314 0.9234828 0.004165514 -19.11 0.00000
sesunknown -0.07785406 0.9250994 0.060674528  -1.28 0.20000
sesupper    0.10090487 1.1061714 0.157986684   0.64 0.52000
seslower   -0.17029644 0.8434148 0.049672825  -3.43 0.00061
parity     -0.11410191 0.8921670 0.010136840 -11.26 0.00000

Random effects
 Group Variable  Std Dev   Variance
 id    Intercept 0.7518637 0.5652990
```

The estimates of the fixed effects have the same interpretation as in ordinary Cox regression. The question is whether the results point to the significance of including frailty terms. In the last line of the output we get the estimate of the frailty variance, $\sigma^2 = 0.565$, but no p-value for the test of the null hypothesis $H_0 : \sigma = 0$. One explanation of this is that ordinary asymptotic theory does not hold for parameter values at the boundary of the parameter space.

One way to get a feeling for the impact of the frailty effect is to fit the same model but without frailty, that is, the term `(1 | id)`.

```
> fit0 <- coxreg(Surv(next.ivl, event) ~ age + ses + parity,
                 data = fe)
> fit0
Call:
coxreg(formula = Surv(next.ivl, event) ~ age + ses + parity,
    data = fe)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
age                33.865    -0.080     0.923     0.003     0.000
ses
          farmer    0.487     0          1 (reference)
         unknown    0.183    -0.085     0.919     0.030     0.005
           upper    0.018     0.147     1.158     0.083     0.077
           lower    0.312    -0.083     0.920     0.026     0.001
parity              4.434     0.013     1.013     0.007     0.068

Events                    8458
Total time at risk        -21348
Max. log. likelihood      -70391
LR test statistic         1834
Degrees of freedom        5
Overall p-value           0
```

We can compare the two "max log likelihoods," in the frailty model the "Integrated" value -69905.17, and in the fixed effects case -70391.12. The difference is so large (485.94) that we safely can reject the hypothesis that the frailty model is not needed. As an "informal" test, you could take twice that difference and treat it as a χ^2 statistic with 1 degree of freedom, calculate a p-value and take as real p-value one half of that (all this because ordinary asymptotic theory does not hold for parameter values on the boundary of the parameter space!). This gives an approximation of the true p-value that is not too bad.

As a final example, let us look at old age mortality in the **R** package `eha`. This example also shows a dangerous situation that is too easy to overlook. It has nothing to do with frailty, but with a problem caused by *missing data*.

Example 27 *Old age mortality for siblings*

Take a look at the variables in `oldmort`:

```
> data(oldmort)
> head(oldmort)
```

```
         id  enter   exit event birthdate m.id f.id    sex
1 765000603 94.510 95.813  TRUE  1765.490   NA   NA female
2 765000669 94.266 95.756  TRUE  1765.734   NA   NA female
3 768000648 91.093 91.947  TRUE  1768.907   NA   NA female
4 770000562 89.009 89.593  TRUE  1770.991   NA   NA female
5 770000707 89.998 90.211  TRUE  1770.002   NA   NA female
6 771000617 88.429 89.762  TRUE  1771.571   NA   NA female
        civ  ses.50 birthplace imr.birth   region
1     widow unknown     remote  22.20000    rural
2 unmarried unknown     parish  17.71845 industry
3     widow unknown     parish  12.70903    rural
4     widow unknown     parish  16.90544 industry
5     widow  middle     region  11.97183    rural
6     widow unknown     parish  13.08594    rural
```

The variable `m.id` is *mother's id*. This means that siblings will have the same value on that variable, and we can check whether we find a "sibling effect" in the sense that siblings tend to have a similar risk of dying.

```
> fit <- coxme(Surv(enter, exit, event) ~ sex + civ + (1 | m.id),
                data = oldmort)
> fit
Cox mixed-effects model fit by maximum likelihood
  Data: oldmort
  events, n = 888, 3340 (3155 observations deleted due to missing)
  Iterations= 16 84
                     NULL Integrated    Fitted
Log-likelihood -5702.662   -5693.43 -5646.252

                   Chisq    df          p   AIC     BIC
Integrated loglik  18.46  4.00 1.0012e-03 10.46   -8.69
 Penalized loglik 112.82 49.23 6.7155e-07 14.36 -221.39

Model:  Surv(enter, exit, event) ~ sex + civ + (1 | m.id)
Fixed coefficients
                coef exp(coef)   se(coef)     z       p
sexfemale  -0.2026325 0.8165783 0.07190559 -2.82 0.00480
civmarried -0.4653648 0.6279060 0.12144548 -3.83 0.00013
civwidow   -0.3305040 0.7185615 0.11964284 -2.76 0.00570

Random effects
 Group Variable  Std Dev    Variance
 m.id  Intercept 0.23719204 0.05626007
```

Now, compare with the corresponding fixed effects model:

```
> fit0 <- coxreg(Surv(enter, exit, event) ~ sex + civ,
                 data = oldmort)
> fit0
Call:
```

```
coxreg(formula = Surv(enter, exit, event) ~ sex + civ,
       data = oldmort)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
sex
            male    0.406     0         1 (reference)
          female    0.594    -0.243     0.784     0.047     0.000
civ
       unmarried    0.080     0         1 (reference)
         married    0.530    -0.397     0.672     0.081     0.000
           widow    0.390    -0.261     0.770     0.079     0.001

Events                    1971
Total time at risk         37824
Max. log. likelihood      -13558
LR test statistic         41.2
Degrees of freedom        3
Overall p-value           5.83367e-09
```

Note that we have now got a very much smaller value of the maximized log likelihood, -13557.98 compared to -5693.43! Something is wrong, and the big problem is that the two analyses were performed on different data sets. How is that possible, when we used `oldmort` on both occasions? The variable `m.id` has a lot of missing values, almost 50% are missing (NA), and the standard treatment of NA:s in **R** is to simply remove each record that contains an NA on any of the variables in the analysis. So, in the first case, the frailty model, a lot of records are removed before analysis, but not in the second. To be able to compare the models, we must remove all records with `m.id = NA` from the second analysis.

```
> olm <- oldmort[!is.na(oldmort$m.id), ]
> fit0 <- coxreg(Surv(enter, exit, event) ~ sex + civ,
                 data = olm)
> fit0
Call:
coxreg(formula = Surv(enter, exit, event) ~ sex + civ, data = olm)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
sex
            male    0.418     0         1 (reference)
          female    0.582    -0.196     0.822     0.070     0.005
civ
       unmarried    0.076     0         1 (reference)
         married    0.555    -0.443     0.642     0.118     0.000
           widow    0.369    -0.310     0.733     0.116     0.007

Events                    888
Total time at risk         19855
Max. log. likelihood      -5693.8
LR test statistic         17.7
```

```
Degrees of freedom        3
Overall p-value           0.000504597
```

This is another story! We now got very similar values of the maximized log likelihoods, -5693.81 compared to -5693.43! The conclusion is that in this case, there is no frailty effect whatsoever.

One lesson to learn from this example is that you have to be very cautious when a data set contains missing values. Some functions, like `drop1`, give a warning when a situation like this is detected, but especially when comparisons are made in more than on step, it is too easy to miss the dangers.

Also note the warning that is printed in the results of `coxme`: `3163 observations deleted due to missingness`. This is a warning that should be taken seriously. □

7.3 Parametric Frailty Models

It is possible utilize the connection between Poisson regression and the piecewise constant proportional hazards model discussed in Chapter 6 to fit parametric frailty models. We look at the fertility data again. The standard analysis without frailty effects is

```
> fit0 <- phreg(Surv(next.ivl, event) ~ parity + ses,
                dist = "pch", cuts = 1:13, data = fe)
> drop1(fit0, test = "Chisq")

Single term deletions
Model:
Surv(next.ivl, event) ~ parity + ses
       Df   AIC    LRT   Pr(Chi)
<none>    29044
parity  1 29852 809.62 < 2.2e-16
ses     3 29104  65.93  3.17e-14

> fit0

Call:
phreg(formula = Surv(next.ivl, event) ~ parity + ses, data = fe,
    dist = "pch", cuts = 1:13)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
parity              4.242    -0.131     0.878     0.005     0.000
ses
          farmer    0.491     0         1           (reference)
         unknown    0.186     0.094     1.099     0.029     0.001
           upper    0.018     0.086     1.089     0.083     0.302
           lower    0.305    -0.150     0.860     0.026     0.000
```

```
Events                    8458
Total time at risk         29806
Max. log. likelihood      -14518
LR test statistic         838
Degrees of freedom        4
Overall p-value           0
```

For testing the presence of a random effect over women, an alternative is to use `glmmML` in `eha`.

```
> fe13 <- survSplit(fe, end = "next.ivl", event = "event",
                    cut = 1:13, episode = "years",
                    start = "start") # 1
> fe13$years <- as.factor(fe13$years) # 2
> fe13$offs <- log(fe13$next.ivl - fe13$start) # 3
> fit1 <- glmmML(event ~ parity + ses + years + offset(offs),
                 cluster = id, family = poisson, method = "ghq",
                 data = fe13, n.points = 9)
> fit1
Call:  glmmML(formula = event ~ parity + ses + years + offset(offs),
              family = poisson, data = fe13, cluster = id,
              method = "ghq", n.points = 9)
                coef se(coef)        z Pr(>|z|)
(Intercept) -3.72891 0.090077 -41.3970 0.00e+00
parity      -0.27362 0.006041 -45.2906 0.00e+00
sesunknown   0.08574 0.065331   1.3124 1.89e-01
sesupper    -0.03306 0.170860  -0.1935 8.47e-01
seslower    -0.26173 0.054002  -4.8468 1.25e-06
years1       3.16821 0.085245  37.1659 0.00e+00
years2       4.61527 0.085842  53.7649 0.00e+00
years3       4.67617 0.091603  51.0483 0.00e+00
years4       4.22567 0.102653  41.1646 0.00e+00
years5       3.56732 0.127991  27.8717 0.00e+00
years6       2.99978 0.172067  17.4338 0.00e+00
years7       2.53474 0.232450  10.9045 0.00e+00
years8       1.62728 0.388357   4.1902 2.79e-05
years9       1.96967 0.370240   5.3200 1.04e-07
years10      2.30379 0.347516   6.6293 3.37e-11
years11      1.42206 0.586882   2.4231 1.54e-02
years12      1.95708 0.510188   3.8360 1.25e-04
years13      0.34816 0.584738   0.5954 5.52e-01

Scale parameter in mixing distribution:  0.8451 gaussian
Std. Error:                              0.0242

        LR p-value for H_0: sigma = 0:  1.2e-277

Residual deviance: 27770 on 34699 degrees of freedom
AIC: 27810
```

Note the use of the function `survSplit` (# 1), the transformation to `factor` for the slices of time (`years`) created by `survSplit` (# 2), and the creation of an offset (# 3). This is described in detail in Chapter 6.

The conclusion is very much the same as with the nonparametric (`coxme`) approach: The clustering effect of *mother* is very strong and must be taken into account in the analysis of birth intervals. The nonparametric approach is easier to use and recommended.

7.4 Stratification

A simple way to eliminate the effect of clustering is to *stratify* on the clusters. In the birth intervals example, it would mean that intervals are only compared to other birth intervals from the same mother. The drawback with a stratified analysis is that it is not possible to estimate the effect of covariates that are constant within clusters. In the birth intervals case, it is probable that *ses*, socioeconomic status, would vary little within families. On the other hand, the effect of *birth order* or *mother's age* would be suitable to analyze in a stratified setting.

```
> fit2 <- coxreg(Surv(next.ivl, event) ~ parity + prev.ivl +
                  strata(id), data = fe)
> fit2
Call:
coxreg(formula = Surv(next.ivl, event) ~ parity + prev.ivl +
    strata(id), data = fe)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
parity              4.434    -0.329     0.720     0.008     0.000
prev.ivl            2.578    -0.054     0.948     0.016     0.001

Events                    8458
Total time at risk        -21348
Max. log. likelihood      -9557.9
LR test statistic         2989
Degrees of freedom        2
Overall p-value           0
```

Contrast this result with an unstratified analysis.

```
> fit3 <- coxreg(Surv(next.ivl, event) ~ parity + prev.ivl,
                  data = fe)
> fit3
Call:
coxreg(formula = Surv(next.ivl, event) ~ parity + prev.ivl,
       data = fe)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
```

```
parity              4.434    -0.084     0.920     0.005     0.000
prev.ivl            2.578    -0.319     0.727     0.011     0.000

Events                    8458
Total time at risk        -21348
Max. log. likelihood      -70391
LR test statistic         1835
Degrees of freedom        2
Overall p-value           0
```

Note how the effect of *parity* is diminished when aggregating comparison over all women, while the effect of the length of the previous interval is enlarged. Try to explain why this result is expected!

Example 28 *Matched data*

Under certain circumstances, it is actually possible to estimate an effect of a covariate that is constant within strata, but only if it is interacted with a covariate that is not constant within strata. See Example 30 in Chapter 9.

This page intentionally left blank

8

Competing Risks Models

8.1 Introduction

Classical competing risks models will be discussed, as well as their modern interpretations. The basic problem is that we want to consider more than one type of event, but where exactly one will occur. For instance, consider different causes of death.

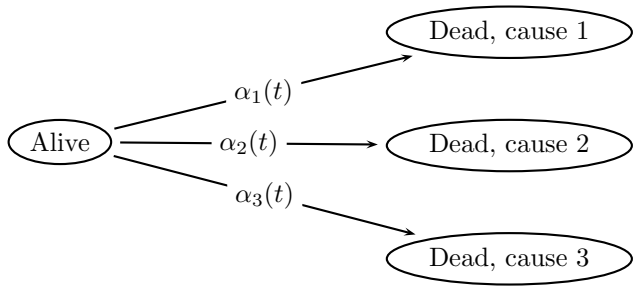


FIGURE 8.1
Competing risks: causes of death.

The first problem is: How can the intensities

$$\big((\alpha_1(t), \alpha_2(t), \alpha_3(t)\big), \quad t > 0$$

be nonparametrically estimated? It turns out that this is quite simple. The trick is to take one cause at at time and estimate its intensity as if the rest of the causes (events) are censorings. The real problem starts when these intensities are turned into probabilities.

8.2 Some Mathematics

First we need some strict definitions, so that the correct questions are asked, and the correct answers are obtained.

The *cause-specific* cumulative hazard functions are

$$\Gamma_k(t) = \int_0^t \alpha_k(s)ds, \quad t > 0, \quad k = 1, 2, 3.$$

The *total mortality* is

$$\lambda(t) = \sum_{k=1}^{3} \alpha_k(t) \quad t > 0$$

$$\Lambda(t) = \sum_{k=1}^{3} \Gamma_k(t), \quad t > 0$$

and *total survival* is

$$S(t) = \exp\{-\Lambda(t)\}$$

So far, this is not controversial. But asking for "cause-specific survivor functions" is.

8.3 Estimation

The quantities $\Gamma_k,\ k = 1, 2, 3$, Λ, and S can be estimated in the usual way. Γ_1, the *cumulative hazard function for cause 1* is estimated by regarding all other causes (2 and 3) as censorings. The total survivor function S is estimated by the method of Kaplan–Meier, regarding all causes as the same cause (just death).

Is it meaningful to estimate (calculate) $S_k(t) = \exp\{-\Gamma_k(t)\},\ k = 1, 2, 3$? The answer is "No." The reason is that it is difficult to define what these probabilities mean in the presence of other causes of death. For instance, what would happen if one cause was eradicated?

8.4 Meaningful Probabilities

It *is* meaningful to estimate (and calculate)

$$P_k(t) = \int_0^t S(s)\alpha_k(s)ds, \quad t > 0, \quad k = 1, 2, 3,$$

the probability to die from cause k before time t. Note that

$$S(t) + P_1(t) + P_2(t) + P_3(t) = 1 \texttt{ for all } t > 0.$$

Now, *estimation* is straightforward with the following estimators:

$$\hat{P}_k(t) = \sum_{i:t_i \leq t} \hat{S}(t_i-)\frac{d_i^{(k)}}{n_i}, \quad k = 1, 2, 3,$$

and $\hat{S}(t),\ t > 0$ is the ordinary Kaplan–Meier estimator for total mortality. See the function `comp` in Section 8.6 for how to write code for this estimation.

8.5 Regression

It is also possible to include covariates in the estimating procedure.

$$P_k(t) = 1 - \exp\left\{-\Gamma_k(t)\exp\left(X\beta^{(k)}\right)\right\}, \quad t > 0, \quad k = 1, 2, 3.$$

These equations are estimated separately. In **R**, this is done with the package `cmprsk` (Fine & Gray 1999, Gray 2011).

Example 29 Mortality and Emigration

People born in Skellefteå in northern Sweden around the year 1800 are followed over time from 15 years of age to outmigration or death. Right censoring occurs for those who are still present on January 1, 1870. At exit the cause is noted; see Figure 8.2.

The data at hand looks like this:

```
> load("zz.RData")
> head(zz)
        Pnr      Mpnr      Fpnr birthdate enter     exit event
1 801000758 775000541 771000493  1801.023    15 39.47241     2
2 801000759 758000309 753000242  1801.028    15 50.00000     0
3 801000760 770000458 765000452  1801.031    15 50.00000     0
4 801000763 777000494 775000404  1801.039    15 26.10868     2
5 801000768 780000572 774000453  1801.061    15 16.43292     2
6 801000770 774000442 776000421  1801.072    15 48.59237     1
  parity  ab
1      3 leg
2      4 leg
3     10 leg
4      3 leg
5      2 leg
6      2 leg
```

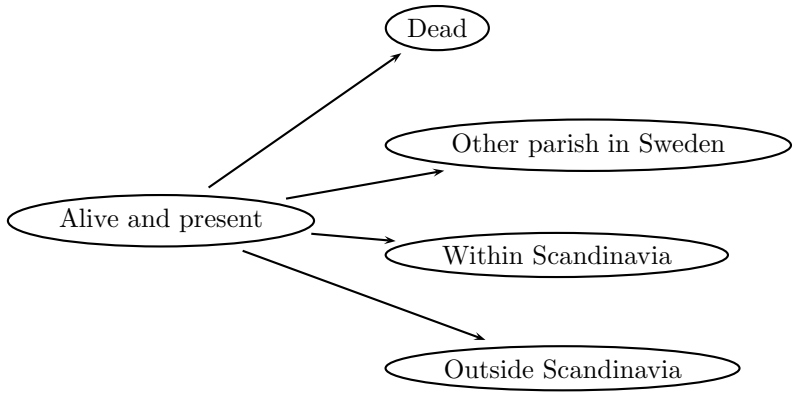


FIGURE 8.2
Death and emigration, Skellefteå, 1800–1870.

```
> summary(zz[, -(1:3)])
   birthdate         enter          exit           event
 Min.   :1801   Min.   :15   Min.   :15.00   Min.   :0.0000
 1st Qu.:1835   1st Qu.:15   1st Qu.:20.47   1st Qu.:0.0000
 Median :1856   Median :15   Median :28.34   Median :0.0000
 Mean   :1852   Mean   :15   Mean   :31.82   Mean   :0.7138
 3rd Qu.:1872   3rd Qu.:15   3rd Qu.:44.82   3rd Qu.:2.0000
 Max.   :1886   Max.   :15   Max.   :50.00   Max.   :4.0000
    parity           ab
 Min.   : 1.000   leg  :16423
 1st Qu.: 2.000   illeg:  850
 Median : 3.000
 Mean   : 3.916
 3rd Qu.: 6.000
 Max.   :19.000
```

The variable *event* above is treated as a continuous variable, which is not very useful, so we make a table instead.

```
> table(zz$event)
    0     1     2     3     4
10551  2102  4097    59   464
```

The variable *event* is coded as shown in Table 8.1.

Without covariates we get the picture shown in Figure 8.3.

```
> source("comp.R")
> comp(zz$enter, zz$exit, zz$event, start.age = 15)
```

TABLE 8.1 Competing risks in Skellefteå

Code	Meaning	Frequency
0	Censored	10 543
1	`Dead`	2 102
3	*Moved* to other parish in Sweden	4 097
4	*Moved* to other Scandinavian country	59
5	*Moved* outside the Scandinavian countries	464

The code for the function `comp` is displayed at the end of this chapter. It is not necessary to understand before reading on. It may be useful as a template for analyzing competing risks. And with covariates and `cmprsk`; the syntax to

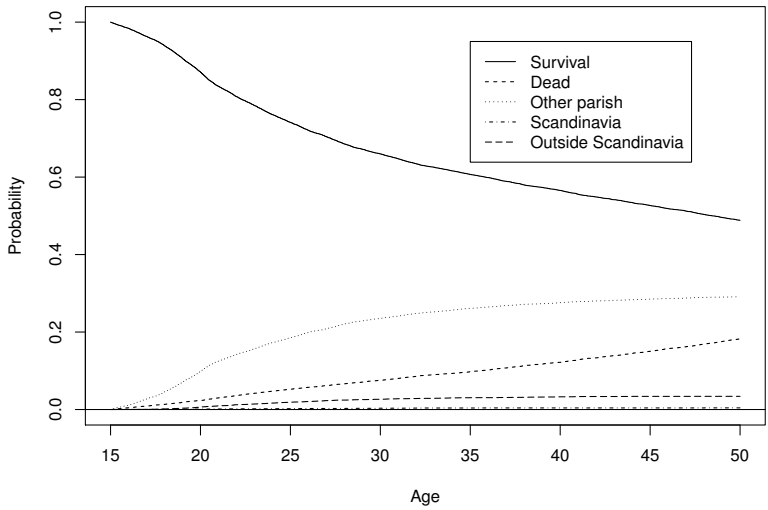


FIGURE 8.3
Cause-specific exit probabilities; migration from Skellefteå to various other places.

get it done is a little bit nonstandard. Here, *moving to other parish in Sweden* is analysed (`failcode = 2`).

```
> library(cmprsk)
> xx <- model.matrix(~ -1 + parity, zz)
> fit <- crr(zz$exit, zz$event, xx, failcode = 3)
> fit
convergence:  TRUE
coefficients:
```

```
 parity
0.03927
standard errors:
[1] 0.04489
two-sided p-values:
parity
  0.38
```

Note that this is *not* Cox regression (but close!). The corresponding Cox regression is

```
> coxreg(Surv(enter, exit, event == 3) ~ parity, data = zz)
Call:
coxreg(formula = Surv(enter, exit, event == 3) ~ parity, data = zz)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
parity              3.890     0.045     1.046     0.048     0.355

Events                    59
Total time at risk        290569
Max. log. likelihood      -548.83
LR test statistic         0.83
Degrees of freedom        1
Overall p-value           0.363362
```

□

8.6 R Code for Competing Risks

The code that produced Figure 8.3 is shown here.

```
> comp
function (enter, exit, event, start.age = 0)
{
    require(eha)
    n <- max(event)
    rs.tot <- risksets(Surv(enter, exit, event > 0.5))
    haz.tot <- rs.tot$n.events/rs.tot$size
    n.times <- length(haz.tot) + 1
    S <- numeric(n.times)
    S[1] <- 1
    for (i in 2:n.times) S[i] <- S[i - 1] * (1 - haz.tot[i - 1])
    haz <- matrix(0, nrow = n, ncol = length(haz.tot))
    P <- matrix(0, nrow = n, ncol = length(haz.tot) + 1)
    for (row in 1:n) {
        rs <- risksets(Surv(enter, exit, event == row))
```

```
        haz.row <- rs$n.events/rs$size
        tmp <- 0
        cols <- which(rs.tot$risktimes %in% rs$risktimes)
        haz[row, cols] <- haz.row
        P[row, 2:NCOL(P)] <- cumsum(S[1:(n.times - 1)] * haz[row,
            ])
    }
    plot(c(start.age, rs.tot$risktimes), S, ylim = c(0, 1),
         xlim = c(start.age,
         max(rs.tot$risktimes)), xlab = "Age", type = "s",
         ylab = "Probability")
    for (i in 1:n) lines(c(start.age, rs.tot$risktimes), P[i,
        ], lty = i + 1, type = "s")
    abline(h = 0)
    legend(35, 0.95, lty = 1:(n + 1), legend = c("Survival",
        "Dead", "Other parish", "Scandinavia",
        "Outside Scandinavia"))
    invisible(list(P = P, S = S))
}
```

This page intentionally left blank

9

Causality and Matching

9.1 Introduction

Causality is a concept that statisticians and statistical science traditionally shy away from. Recently, however, many successful attempts have been made to include the concept of causality in the statistical theory and vocabulary. A good review of the topic, from a modern event history analysis point of view, is given in Aalen et al. (2008), Chapter 9. Some of their interesting ideas are presented here.

The traditional standpoint among statisticians was that "we deal with association and correlation, not causality"; see Pearl (2000) for a discussion. An exception was the clinical trial, and other situations, where *randomization* could be used as a tool. However, during the last couple of decades, there has been an increasing interest in the possibility of making causal statements even without randomization, that is, in *observational* studies (Rubin 1974, Robins 1986).

Matching is a statistical technique, that has an old history without an explicit connection to causality. However, as we will see, matching is a very important tool in the modern treatment of causality.

Unfortunately, the models for event history analysis presented here are not implemented in **R** or, to my knowledge, in any other publicly available software. One exception is *matched data analysis*, which, except the matching itself, can be performed with ordinary stratified Cox regression.

9.2 Philosophical Aspects of Causality

The concept of causality has a long history in philosophy; see for instance (Aalen et al. 2008) for a concise review. A fundamental question, with a possibly unexpected answer, is "Does everything that happens have a cause?" According to (Zeilinger 2005), the answer is "no."

> The discovery that individual events are irreducibly random is probably one of the most significant findings of the twentieth cen-

> tury. Before this, one could find comfort in the assumption that random events only seem random because of our ignorance. ...
>
> But for the individual event in quantum physics, not only do we not know the cause, there is no cause.

Of course, this statement must not necessarily be taken literally, but it indicates that nothing is to be taken as granted.

9.3 Causal Inference

According to (Aalen et al. 2008), there are three major schools in statistical causality: (i) graphical models, (ii) predictive causality, and (iii) counterfactual causality. They also introduce a new concept, *dynamic path analysis*, which can be seen as a merging of (i) and (ii), with the addition that *time* enters explicitly the models.

9.3.1 Graphical Models

Graphical models have a long history, emanating from (Wright 1921), who introduced *path analysis*. The idea was to show by writing diagrams how variables influence one another. Graphical models have had a revival during the last decades with very active research; see (Pearl 2000) and (Lauritzen 1996). A major drawback for event history analysis purposes is, according to (Aalen et al. 2008), that *time* is not explicitly taken into account. The idea is that causality evolves in time; That is, a *cause* must precede an *effect*.

9.3.2 Predictive causality

The concept of predictive causality is based on *stochastic processes*, and that a cause must *precede* an effect in time. This may seem obvious, but very often you see it not clearly stated. This leads sometimes to confusion, for instance, to questions like "What is the cause, and what is the effect?"

One early example is *Granger causality* (Granger 1969) in time series analysis. Another early example with more relevance to event history analysis is the concept of *local dependence*. It was introduced by Tore Schweder (Schweder 1970).

Local dependency is illustrated in Figure 9.1. A and B are events, and the superscript (c) indicates their complements, that is, they have not (yet) occurred if superscripted. This model is used in the matched data example later in this chapter. There A stands for *mother dead* and B means *infant dead*. The mother and her newborn (alive) infant are followed from the birth to the death of the infant (but at most a one-year follow-up). During this

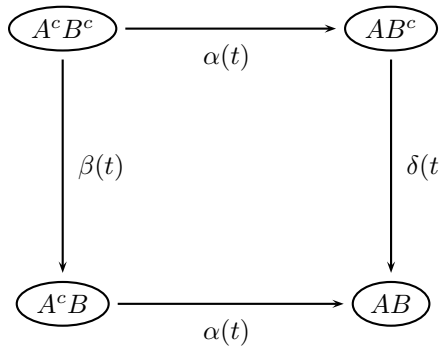


FIGURE 9.1
Local dependence.

follow-up, both mother and infant are observed and the eventual death of the mother is reported. The question is whether the mother's death influences the survival chances of the infant (it does!).

In Figure 9.1: If $\beta(t) \neq \delta(t)$, then B is *locally dependent* on A, but A is *locally independent* on B: The vertical transition intensities are different, which means that the intensity of B happening is influenced by A happening or not. On the other hand, the horizontal transitions are equal, meaning that the intensity of A happening is not influenced by B happening or not. In our example, this means that mother's death influences the survival chances of the infant, but mother's survival chances are unaffected by the eventual death of her infant (maybe not probable in the real world).

9.3.3 Counterfactuals

In situations where interest lies in estimating a *treatment effect*, the idea of *counterfactual outcomes* is an essential ingredient in the causal inference theory advocated by Rubin (Rubin 1974) and Robins (Robins 1986). An extensive introduction to the field is given by Hernán & Robins (2012).

Suppose we have a sample of individuals, some treated and some not treated, and we wish to estimate a marginal treatment effect in the sample at hand. If the sample is the result of randomization, that is, individuals are randomly allocated to treatment or not treatment (placebo), then there are in principle no problems. If, on the other hand, the sample is self-allocated to treatment or placebo (an observational study), then the risk of *confounders* destroying the analysis is overwhelming. A confounder is a variable that is correlated both with treatment and effect, eventually causing biased effect estimates.

The theory of counterfactuals tries to solve this dilemma by allowing each individual to be its own control. More precisely, for each individual, two hypothetical outcomes are defined; the outcome if treated and the outcome if

not treated. Let us call them Y_1 and Y_0, respectively. They are counterfactual (counter to fact), because none of them can be observed. However, since an individual cannot be both treated and untreated, in the real data, each individual has exactly one observed outcome Y. If the individual was treated, then $Y = Y_0$, otherwise $Y = Y_1$. The individual treatment effect is $Y_1 - Y_0$, but this quantity is not possible to observe, so how do we proceed?

The Rubin school fixes balance in the data by *matching*, while the Robins school advocates *inverse probability weighting* . It is possible to apply both these methods to event history research problems (Hernán, Brumback & Robins 2002, Hernán, Cole, Margolick, Cohen & Robins 2005), but unfortunately there is no publicly available software for performing these kinds of analyses, partly with the exception of matching, of which an example is given later in this chapter. However, with the programming power of **R**, it is fairly straightforward to write our own functions for specific problems. This is, however, out of the scope of this presentation.

The whole theory based on counterfactuals relies on the assumption that *there are no unmeasured confounders*. Unfortunately, this assumption is completely untestable, and even worse, it never holds in practice. Does this mean that the whole theory of causal inference is worthless? No, it does not, because the derived procedures are often sound even without the attribute "causal." But claiming causality is an exaggeration in most circumstances. What we can hope for is the elimination of the effects of the *measured* confounders.

9.4 Aalen's Additive Hazards Model

In certain applications it may be reasonable to assume that risk factors acts additively rather than multiplicatively on hazards. Aalen's additive hazards model (Aalen 1989, Aalen 1993) takes care of that.

For comparison, recall that the *proportional hazards* model may be written

$$h(t \mid \mathbf{x}_i) = h_0(t) r\big(\boldsymbol{\beta}, \mathbf{x}_i(t)\big), \quad t > 0,$$

where $r(\boldsymbol{\beta}, \mathbf{x}_i)$ is a *relative risk* function. In *Cox regression*, we have: $r(\boldsymbol{\beta}, \mathbf{x}_i) = \exp(\boldsymbol{\beta}^T \mathbf{x}_i)$,

The additive hazards model is given by

$$h(t \mid \mathbf{x}_i) = h_0(t) + \beta_1(t) x_{i1}(t) + \cdots + \beta_p(t) x_{ip}(t), \quad t > 0,$$

where $h_0(t)$ is the *baseline hazard*, and $\boldsymbol{\beta}(t) = (\beta_0(t), \ldots, \beta_p(t))$ is a (multivariate) nonparametric *regression function.*

Note that $h(t, \mathbf{x}_i)$ may be negative if some coefficients or parameters are negative. In contrast to the Cox regression model, there is no automatic protection against this.

The function `aareg` in the `survival` fits the additive model.

```
> require(eha)
> data(oldmort)
> fit <- aareg(Surv(enter-60, exit-60, event) ~ sex, data = oldmort)
> fit
Call:
aareg(formula = Surv(enter - 60, exit - 60, event) ~ sex,
      data = oldmort)
  n= 6495
    1805 out of 1806 unique event times used

             slope      coef se(coef)     z         p
Intercept  0.1400  0.000753 2.58e-05 29.20 9.82e-188
sexfemale -0.0281 -0.000136 3.26e-05 -4.16  3.12e-05

Chisq=17.34 on 1 df, p=3.1e-05; test weights=aalen
```

Obviously, *sex* is important; females have a lower mortality. Plots of the time-varying cumulative intercept and regression coefficient are given by

```
> oldpar <- par(mfrow = c(1, 2))
> plot(fit)
> par(oldpar)
```

See Figure 9.2, where 95% confidence limits are added around the fitted time-varying cumulative coefficients. Also note the use of the function `par`; the first call to it sets the plotting area to "one row and two columns" and saves the old `par`-setting in *oldpar*. Then the plotting area is restored to what it was earlier.

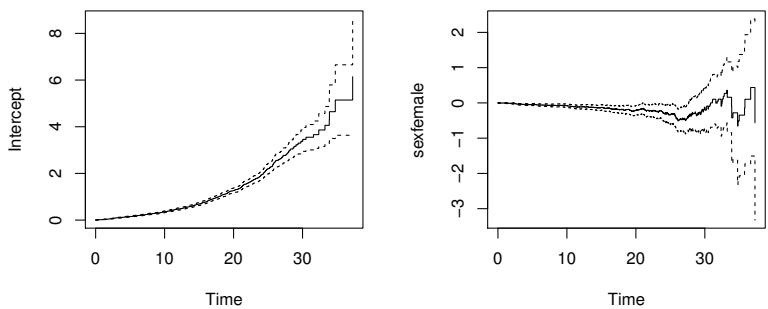


FIGURE 9.2
Cumulative intercept (left) and cumulative regression coefficient (right).

9.5 Dynamic Path Analysis

The term *dynamic path analysis* is coined by Odd Aalen and coworkers (Aalen et al. 2008). It is an extension, by explicitly introducing time, of *path analysis* described by (Wright 1921).

The inclusion of time in the model implies that there are one path analysis at each time point; see Figure 9.3.

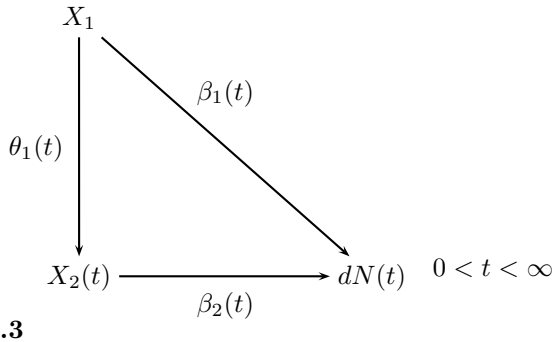


FIGURE 9.3
Dynamic path analysis; X_2 is an intermediate covariate, while X_1 is measured at baseline ($t = 0$). $dN(t)$ is the number of events at t.

The *structural equations*

$$dN(t) = \big(\beta_0(t) + \beta_1(t)X_1 + \beta_2(t)X_2(t)\big)dt + dM(t)$$
$$X_2(t) = \theta_0(t) + \theta_1(t)X_1 + \epsilon(t)$$

are estimated by ordinary least squares (linear regression) at each t with $dN(t) > 0$. Then the second equation is inserted in the first:

$$dN(t) = \{\beta_0(t) + \beta_2(t)\theta_0(t) + \big(\beta_1(t) + \theta_1(t)\beta_2(t)\big)X_1 + \beta_2(t)\epsilon(t)\}dt + dM(t)$$

and the *total treatment effect* is $\big(\beta_1(t) + \theta_1(t)\beta_2(t)\big)dt$, so it can be split into two parts according to

$$\begin{aligned}\text{total effect} = &\ \text{direct effect} + \ \text{indirect effect}\\ = &\ \beta_1(t)dt + \theta_1(t)\beta_2(t)dt\end{aligned}$$

Some book-keeping is necessary in order to write an **R** function for this dynamic model. Of great help is the function `risksets` in `eha`; it keeps track of the composition of the riskset over time, which is exactly what is needed for implementing the dynamic path analysis.

9.6 Matching

Matching is a way of eliminating the effect of confounders and therefore important to discuss in connection with causality. One reason for matching is the wish to follow the causal inference paradigm with counterfactuals. Another reason is that the exposure we want to study is very rare in the population. It will then be inefficient to take a simple random sample from the population; in order to get enough cases in the sample, the sample size needs to be very large. On the other hand, today's register-based research tends to analyze whole populations, and the limitations in terms of computer memory and processing power are more or less gone. Nevertheless, small, properly matched data sets may contain more information about the specific question at hand than a standard regression analysis on the whole register!

When creating a matched data set, we have to decide how many controls we want per case. In the true counterfactual paradigm in "causal inference," it is common practice to choose one control per case, to "fill in" the unobservable in the pair of counterfactuals. We first look at the case with matched pairs, then a case study with two controls per case.

9.6.1 Paired Data

Certain kinds of observations come naturally in pairs. The most obvious situation is data from twin studies. Many countries keep registers of twins, and the advantage with twin data is that it is possible to control for genetic variation; monozygotic twins have identical genes. Humans have pairs of many details: arms, legs, eyes, ears, etc. This can be utilized in medical studies concerning comparison of treatments, the two in a pair simply get one treatment each.

In a survival analysis context, pairs are followed over time and it is noted who first experiencesthe event of interest. In each pair, one is treated and the other is a control, but otherwise they are considered more or less identical. Right censoring may make it impossible to decide who in the pair experienced the event first. Such pairs are simply discarded in the analysis.

The model is

$$h_i(t;x) = h_{0i}(t)e^{\beta x}, \tag{9.1}$$

where x is treatment (0-1) and i is the pair No. Note that each pair has its own baseline hazard function, which means that the proportional hazards assumption is only required within pairs. This is a special form of stratified analysis in that each stratum only contains two individuals, a case and its control. If we denote by T_{i1} the life length of the case in pair No. i and T_{i0} the life length of the corresponding control, we get

$$P(T_{1i} < T_{0i}) = \frac{e^{\beta}}{1+e^{\beta}}, \quad i = 1, \ldots, n,$$

and the result of the analysis is simply a study of binomial data; how many times did the case die before the control? We can estimate this probability, and also test the null hypothesis that it is equal to one half, which corresponds to $\beta = 0$ in (9.1).

9.6.2 More than One Control

The situation with more than one control per case is as simple as the paired data case to handle. Simply stratify, with one case and its controls per stratum. It is also possible to have a varying number of controls per case.

As an example where two controls per case were used, let us see how a study of maternal death and its effect on infant mortality was performed (Broström 1987).

Example 30 *Maternal death and infant mortality.*

This is a study on historical data from northern Sweden, 1820–1895 (Broström 1987). The simple question asked was: How much does the death risk increase for an infant that loses her mother? More precisely, by a maternal death we mean that a mother dies within one year after the birth of her child. In this particular study, only first births were studied. The total number of such births was 5,641 and of these 35 resulted in a maternal death (with the infant still alive). Instead of analyzing the full data set, it was decided to use matching. To each case of maternal death, two controls were selected in the following way. At each time an infant changed status from `mother alive` to `mother dead`, two controls are selected without replacement from the subset of the current risk set, where all infants have `mother alive` and not already used as controls and with correct matching characteristics. If a control changes status to case (its mother dies), it is immediately censored as a control and starts as a case with two controls linked to it. However, this situation never occurred in the study.

Let us load the data into **R** and look at it.

```
> require(eha)
> data(infants)
> head(infants)
  stratum enter exit event mother age  sex      parish   civst
1       1    55  365     0   dead  26  boy Nedertornea married
2       1    55  365     0  alive  26  boy Nedertornea married
3       1    55  365     0  alive  26  boy Nedertornea married
4       2    13   76     1   dead  23 girl Nedertornea married
5       2    13  365     0  alive  23 girl Nedertornea married
6       2    13  365     0  alive  23 girl Nedertornea married
     ses year
1 farmer 1877
2 farmer 1870
3 farmer 1882
```

```
4  other 1847
5  other 1847
6  other 1848
```

Here, we see the two first triplets in the data set, which consists of 35 triplets, or 105 individuals. Infant No. 1 (the first row) is a case, his mother died when he was 55 days old. At that moment, two controls were selected, that is, two boys 55 days of age, and with the same characteristics as the case. The matching was not completely successful; we had to look a few years back and forth in calendar time (covariate `year`). Note also that in this triplet all infants survived one year of age, so there is no information of risk differences in it. It will be automatically removed in the analysis.

The second triplet, on the other hand, will be used, because the case dies at age 76 days, while both controls survive one year of age. This is information suggesting that cases have higher mortality than controls after the fatal event of a maternal death.

The matched data analysis is performed by stratifying on triplets (the variable `stratum`).

```
> fit <- coxreg(Surv(enter, exit, event) ~ mother + strata(stratum),
   data = infants)
> fit
Call:
coxreg(formula = Surv(enter, exit, event) ~ mother + strata(stratum),
    data = infants)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
mother
           alive    0.763     0         1 (reference)
            dead    0.237     2.605    13.534     0.757     0.001

Events                    21
Total time at risk         21616
Max. log. likelihood      -10.815
LR test statistic         19.6
Degrees of freedom        1
Overall p-value           9.31744e-06
```

The result is that the mother's death increases the death risk of the infant by a factor 13.5! It is statistically very significant, but the number of infant deaths (21) is very small, so the p-value may be unreliable. A note in passing: The "Overall p-value" is the likelihood ratio test p-value for the whole model versus a model with no covariates. Since in this case there is only one regression parameter to estimate, the "Wald p" and the "Overall p-value" corresponds to the same null hypothesis, "mother's death has no effect on her infant's survival chances." The part "e-06" means "move the decimal point six steps to the left," giving the value $0.00000931744 \approx 0.00001$. Normally, this is the p-value to trust; see the discussion of the *Hauck–Donner effect* in Appendix A.

In a stratified analysis, it is normally not possible to include covariates that

are constant within strata. However, here it is possible to estimate the *interaction* between a stratum-constant covariate and exposure, that is, mother's death. However, it is important *not* to include the main effect corresponding to the stratum-constant covariate! This is a rare exception to the rule that when an interaction term is present, the corresponding main effects must be included in the model.

We investigate whether the effect of mother's death is modified by her age by first calculating an interaction term (*mage*):

```
> infants$mage <- ifelse(infants$mother == "dead", infants$age, 0)
```

Note the use of the function `ifelse`: It takes three arguments; the first is a logical expression (resulting in a *logical* vector, with values `TRUE` and `FALSE`. Note that in this example, the length is the same as the length of *mother* (in `infants`). For each component that is `TRUE`, the result is given by the second argument, and for each component that is `FALSE` the value is given by the third argument.

Including the created covariate in the analysis gives

```
> fit <- coxreg(Surv(enter, exit, event) ~ mother + mage +
                strata(stratum), data = infants)
> fit
Call:
coxreg(formula = Surv(enter, exit, event) ~ mother + mage +
       strata(stratum), data = infants)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
mother
           alive    0.763     0         1 (reference)
            dead    0.237     2.916    18.467     4.860     0.548
mage                6.684    -0.012     0.988     0.188     0.948

Events                    21
Total time at risk         21616
Max. log. likelihood      -10.813
LR test statistic         19.6
Degrees of freedom        2
Overall p-value           5.40649e-05
```

What happened here? The effect of the mother's age is even larger than in the case without *mage*, but the statistical significance has vanished altogether. There are two problems with one solution here. First, due to the inclusion of the interaction, the (main) effect of mother's death is now measured at the mother's age equal to 0 (zero!), which of course is completely nonsensical, and second, the construction makes the two covariates strongly correlated: When *mother* is `alive` *mage* is zero, and when *mother* is `dead`, *mage* takes a rather large value. This is a case of *collinearity*, however, not a very severe case, because it is very easy to get rid of.

The solution to both problems is to *center* the mother's age (*age*)! Recreate the variables and run the analysis again:

```
> infants$age <- infants$age - mean(infants$age)
> infants$mage <- ifelse(infants$mother == "dead", infants$age, 0)
> fit1 <- coxreg(Surv(enter, exit, event) ~ mother + mage +
                 strata(stratum), data = infants)
> fit1
Call:
coxreg(formula = Surv(enter, exit, event) ~ mother + mage +
       strata(stratum), data = infants)
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
mother
           alive    0.763     0         1 (reference)
            dead    0.237     2.586    13.274     0.809     0.001
mage                0.280    -0.012     0.988     0.188     0.948

Events                    21
Total time at risk         21616
Max. log. likelihood      -10.813
LR test statistic         19.6
Degrees of freedom        2
Overall p-value           5.40649e-05
```

The two fits are equivalent (look at the Max. log. likelihoods), but with different parametrizations. The second, with a centered mother's age, is preferred, because the two covariates are close to uncorrelated there.

A subject-matter conclusion in any case is that the mother's age does not affect the effect of her death on the survival chances of her infant. □

A general advice: Always center continuous covariates before the analysis! This is especially important in the presence of models with interaction terms, and the advice is valid for all kinds of regression analyses, not only Cox regression.

9.7 Conclusion

Finally, some thoughts about causality, and the techniques in use for "causal inference." As mentioned above, in order to claim causality, one must show (or assume) that there are "no unmeasured confounders." Unfortunately, this is impossible to prove or show from data alone, but even worse is the fact that in practice, at least in demographic and epidemiological applications, there are *always* unmeasured confounders present. However, with this in mind, note that

- Causal *thinking* is important.
- Counterfactual reasoning and marginal models yield little insight into

"how it works," but it is a way of reasoning around research problems that helps sorting out thoughts.

  - Joint modeling is the alternative.

- Creation of *pseudo-populations* through weighting and matching may limit the understanding of how things work.

  - Analyze the process as it presents itself, so that it is easier to generalize findings.

Read more about this in Aalen et al. (2008).

A

Basic Statistical Concepts

A.1 Introduction

The statistical concepts that are important for understanding what is going on in this book are gathered here, but briefly treated. The interested reader who wants a deeper understanding of statistical concepts should have no problems in finding suitable text books. There are, for instance, some recent texts that teaches statistics and how to use it in **R** (Dalgaard 2008).

A.2 Statistical Inference

Statistical inference is the science that help us draw conclusions about real-world phenomena by observing and analyzing samples from them. The theory rests on probability theory and the concept of *random* sampling. The statistical analysis never gives absolute truths, but only statements coupled to certain measures of their validity. These measures are almost always *probability statements*.

The crucial concept is that of a *model*, despite the fact that the present trend in statistical inference is toward nonparametric statistics. It is often stated that with today's huge data registers, statistical models are unnecessary, but nothing could be more wrong.

The important idea in a statistical model is the concept of a *parameter*. It is often confused with its estimator from data. For instance, when we talk about *mortality* in a population, it is a hypothetical concept that is different from the ratio between the observed number of deaths and the population size (or any other measure based on data). The latter is an *estimate* (at best) of the former. The whole idea about statistical inference is to extract information about a *population parameter* from observing data.

A.2.1 Point Estimation

The case in *point estimation* is to find the best guess (in some sense) of a population parameter from data. That is, we try to find the best single value that is closest to the true, but unknown, value of the population parameter.

Of course, a point estimator is useless if it is not connected to some measure of its uncertainty. That takes us to the concept of *interval estimation*.

A.2.2 Interval Estimation

The philosophy behind *interval estimation* is that a guess on a single value of the unknown population parameter is useless without an accompanying measure of the uncertainty of that guess. A *confidence interval* is an interval, in which we say that the true value of the population parameter lies with a certain probability (often 95%).

A.2.3 Hypothesis Testing

We are often interested in a specific value of a parameter, and in regression problems this value is almost always *zero* (0). The reason is that regression parameters measure *effects*, and to test for an effect is then equivalent to testing that the corresponding parameter has value zero.

There is a link between interval estimation and hypothesis testing: To test the hypothesis that a parameter value is zero can be done through constructing a confidence interval for the parameter. The test rule is then: If the interval does not cover *zero*, reject the hypothesis, otherwise do not.

A.2.3.1 The Log-Rank Test

The general hypothesis testing theory behind the log-rank test builds on the *hyper-geometric distribution*. The calculations under the null hypothesis of no difference in survival chances between the two groups are performed *conditional on both margins*. In Table A.1, if the margins are fixed, there is only one degree of freedom left; for a given value of (say) d_1, the three values d_2, $(n_1 - d_1)$, and $(n_2 - d_2)$ are determined. Utilizing the fact that, under the

TABLE A.1
The general table at one event time

Group	Deaths	Survivors	Total
I	d_1	$n_1 - d_1$	n_1
II	d_2	$n_2 - d_2$	n_2
Total	d	$n - d$	n

null, d_1 is hyper-geometrically distributed, results in the following algorithm for calculating a test statistic as follows:

1. Observe $O = d_1$
2. Calculate the expected value E of O (under the null):

$$E = d\frac{n_1}{n}.$$

3. Calculate the variance V of O (under the null):

$$V = \frac{(n-d)dn_1n_2}{n^2(n-1)}.$$

4. Repeat 1 – 3 for all tables and aggregate according to Equation (A.1).

The log rank test statistic T is

$$T = \frac{\sum_{i=1}^{k}(O_i - E_i)}{\sqrt{\sum_{i=1}^{k} V_i}} \tag{A.1}$$

Note carefully that this procedure is *not* equivalent to aggregating all tables of raw data!

Properties of the log rank test;

- The test statistic T^2 is approximately distributed as $\chi^2(1)$.
- It is available in most statistical software.
- It can be generalized to comparisons of more than two groups.
- For s groups, the test statistic is approximately $\chi^2(s-1)$.
- The test has *high power* against alternatives with *proportional hazards*, but can be weak against nonproportional alternatives.

A.3 Asymptotic theory

A.3.1 Partial likelihood

Here is a very brief summary of the asymptotics concerning the partial likelihood. Once defined, it turns out that you may treat it as an ordinary likelihood function (Andersen et al. 1993). The setup is as follows.

Let $t_{(1)}, t_{(2)}, \ldots, t_{(k)}$ the ordered observed event times and let $R_i = R(t_{(i)})$ be the risk set at $t_{(i)}$, $i = 1, \ldots, k$, see Chapter 2, Equation (2.4). At $t_{(i)}$, *condition* with respect to the composition of R_i and that one event occurred (for tied events, a correction is necessary).

Then the contribution to the partial likelihood from $t_{(i)}$ is

$$L_i(\boldsymbol{\beta}) = P(\text{No. } m_i \text{ dies} \mid \text{one event occur}, R_i) = \frac{h_0(t_{(i)})\exp(\boldsymbol{\beta}\mathbf{x}_{m_i})}{\sum_{\ell\in R_i} h_0(t_{(i)})\exp(\boldsymbol{\beta}\mathbf{x}_\ell)} = \frac{\exp(\boldsymbol{\beta}\mathbf{x}_{m_i})}{\sum_{\ell\in R_i}\exp(\boldsymbol{\beta}\mathbf{x}_\ell)}$$

and the full partial likelihood is

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{k} L_i(\boldsymbol{\beta}) = \prod_{i=1}^{k} \frac{\exp(\boldsymbol{\beta}\mathbf{x}_{m_i})}{\sum_{\ell\in R_i}\exp(\boldsymbol{\beta}\mathbf{x}_\ell)}$$

This is where the doubt about the partial likelihood comes in; the conditional probabilities multiplied together do not have a proper interpretation as a conditional probability. Nevertheless, it is prudent to proceed as if the expression really is a likelihood function. The log partial likelihood becomes

$$\log\big(L(\boldsymbol{\beta})\big) = \sum_{i=1}^{k}\left\{\boldsymbol{\beta}\mathbf{x}_{m_i} - \log\left(\sum_{\ell\in R_i}\exp(\boldsymbol{\beta}\mathbf{x}_\ell)\right)\right\}, \tag{A.2}$$

and the components of the *score vector* are

$$\frac{\partial}{\partial\beta_j}\log L(\boldsymbol{\beta}) = \sum_{i=1}^{k}\mathbf{x}_{m_i j} - \sum_{i=1}^{k}\frac{\sum_{\ell\in R_i} x_{\ell j}\exp(\boldsymbol{\beta}\mathbf{x}_\ell)}{\sum_{\ell\in R_i}\exp(\boldsymbol{\beta}\mathbf{x}_\ell)}, \quad j = 1,\ldots,s. \tag{A.3}$$

The *maximum partial likelihood (MPL)* estimator of $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}$, is found by setting (A.3) equal to zero and solve for $\boldsymbol{\beta}$.

For inference, we need to calculate the inverse of minus the *Hessian*, evaluated at $\hat{\boldsymbol{\beta}}$. This gives the estimated *covariance matrix*. The Hessian is the matrix of the second partial derivatives. The expectation of minus the Hessian is called the *information matrix*. The *observed* information matrix is

$$\hat{I}(\hat{\boldsymbol{\beta}})_{j,m} = -\frac{\partial^2\log L(\boldsymbol{\beta})}{\partial\beta_j\partial\beta_m}\,|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}}$$

and asymptotic theory says that

$$\hat{\boldsymbol{\beta}} \sim N(\boldsymbol{\beta}, \hat{I}^{-1}(\hat{\boldsymbol{\beta}}))$$

This is to say that $\hat{\boldsymbol{\beta}}$ is asymptotically unbiased and normally distributed with the given covariance matrix (or the limit of it). Further, $\hat{\boldsymbol{\beta}}$ is a consistent estimator of $\boldsymbol{\beta}$. This is used for testing, confidence intervals, variable selection.

Note that these are only *asymptotic* results, that is, useful in *large to medium sized samples*. In small samples, *bootstrapping* is a possibility. This option is available in the **R** package eha.

Here a warning is in order: Tests based on standard errors (*Wald* tests) may be *highly unreliable*, as in all *nonlinear* regression (Hauck & Donner 1977). A better alternative is the *likelihood ratio test*.

A.4 Model Selection

In regression models, there is often several competing models for describing data. In general, there are no strict rules for "correct selection." However, for *nested* models, there are some formal guidelines. For a precise definition of this concept, see Appendix A.

A.4.1 Nested Models

The meaning of *nesting* of models is best described by an example.

Example 31 *Two competing models*

- $\mathcal{M}_2: \ h(t;(x_1,x_2)) = h_0(t)\exp(\beta_1 x_1 + \beta_2 x_2)$
- $\mathcal{M}_1: \ h(t;(x_1,x_2)) = h_0(t)\exp(\beta_1 x_1)$: x_2 has no effect.

Thus, the model $\mathcal{M}_1$ is a special case of $\mathcal{M}_2$ ($\beta_2 = 0$). We say that $\mathcal{M}_1$ is *nested* in $\mathcal{M}_2$. Now, *assume* that $\mathcal{M}_2$ is *true*. Then, testing the hypothesis $H_0:$ $\mathcal{M}_1$ is true (as well) is the same as testing the hypothesis $H_0;\ \beta_2 = 0$.

The formal theory for and procedure for performing the likelihood ratio test (LRT) can be summarized as follows:

1. Maximize $\log L(\beta_1,\beta_2)$ under $\mathcal{M}_2$; gives $\log L(\hat{\beta}_1,\hat{\beta}_2)$.
2. Maximize $\log L(\beta_1,\beta_2)$ under $\mathcal{M}_1$, that is, maximize $\log L(\beta_1,0)$; gives $\log L(\beta_1^*,0)$.
3. Calculate the test statistic
$$T = 2\big(\log L(\hat{\beta}_1,\hat{\beta}_2) - \log L(\beta_1^*,0)\big)$$
4. Under H_0, T has a χ^2 (chi-square) distribution with d degrees of freedom: $T \sim \chi^2(d)$, where d is the difference in numbers of parameters in the two competing models, in this case, $2-1=1$.
5. Reject H_0 if T is large enough. Exactly how much that is depends on the level of significance; if it is α, choose the limit t_d equal to the $100(1-\alpha)$ percentile of the $\chi^2(d)$ distribution.

This result is a *large sample approximations.*

The *Wald test* is theoretically performed as follows:

1. Maximize $\log L(\beta_1,\beta_2)$ under $\mathcal{M}_2$; this gives $\log L(\hat{\beta}_1,\hat{\beta}_2)$, and $\hat{\beta}_2$, $\text{se}(\hat{\beta}_2)$.

2. Calculate the test statistic

$$T_W = \frac{\hat{\beta}_2}{\text{se}(\hat{\beta}_2)}$$

3. *Under* H_0, T_W has a *standard normal* distribution: $T_W \sim N(0, 1)$.

4. Reject H_0 if the absolute value of T_W is larger than 1.96 on a significance level of 5%.

This is a *large sample approximation*, with the advantage that it is automatically available in all software. In comparison to the LRT, one model less has to be fitted. This saves time and efforts, unfortunately on the expense of accuracy, because it may occasionally give *nonsensic results*. This phenomenon is known as the *Hauck-Donner effect* (Hauck & Donner 1977). □

B

Survival Distributions

B.1 Introduction

The survival distributions we discuss here are all available as functions in **R**. The most basic survival distribution is the *Exponential distribution*, not because it is useful as is in demographic applications (it isn't), but because it is a reference distribution, a special case of other distributions, and a building block in many useful models.

The piecewise constant hazards, Gamma, and the Weibull distributions are all generalizations of the exponential.

B.2 Relevant Distributions in R

In **R**, there are several families of distributions available. The ones that are relevant in survival analysis are characterized by having positive support. For each family of distributions, four functions are available; a *density* function (name prefix `d`), a *cumulative distribution* function (name prefix `p`), a *quantile* function (name prefix `q`), and a *random number* generator (name prefix `r`). In the package `eha`, the two functions with prefix 'h' and 'H' are added for some distributions. For instance, for the *Weibull* distribution there are `hweibull` and `Hweibull`, where the first is the hazard function and the second is the cumulative hazards function.

In Section B.2.1, *The Exponential distribution*, we show exactly what these functions are and how to use them. It should be noted that in base **R**, there are no survival functions, hazard functions, or cumulative hazards functions. Some are available in the package `eha` and others. However, they can easily be derived from the given `p-` and `d-` functions, as shown in Section B.2.1.

B.2.1 The Exponential Distribution

The exponential distribution is characterized by having a constant hazard rate λ. It has density function

$$f(t;\lambda) = \lambda e^{-\lambda x}, \quad \lambda > 0,\ t > 0,$$

and survival function

$$S(t;\lambda) = e^{-\lambda x}, \quad \lambda > 0,\ t > 0.$$

The mean is $1/\lambda$ (which, unfortunately, also is the *scale* parameter), and the variance is $1/\lambda^2$. The exponential distribution is represented in **R** by the functions `pexp`, `dexp`, `qexp`, and `rexp`, in order the cumulative distribution function, the density function, the quantile function (which is the inverse to the cumulative distribution function), and the random number generator function.

In Figure B.1, the relevant functions are plotted for an exponential distribution with $\lambda = 1$ (the default value). It is created as follows.

```
> require(eha)
> x <- seq(0, 4, length = 1000)
> oldpar <- par(mfrow = c(2, 2))
> plot(x, dexp(x), type = "l",
       main = "Density", ylab = "")
> plot(x, pexp(x, lower.tail = FALSE), type = "l",
       main = "Survival", ylab = "")
> plot(x, hweibull(x, shape = 1), type = "l",
       main = "Hazard", ylab = "")
> plot(x, Hweibull(x, shape = 1), type = "l",
       main = "Cumulative hazards", ylab = "")
> par(oldpar)
```

Note that there are no functions `hexp` or `Hexp`, maybe because they are too simple, see Figure B.1. The fact that the exponential distribution is special case of the Weibull distribution (`shape = 1`) is utilized. The exponential distribution is characterized by the fact that it *lacks memory*. In other words, items whose life lengths follow an exponential distribution do not age; no matter how old they are, if they are alive they are as good as new. This concept is not useful when it comes to human lives, but the life lengths of electronic components are often modeled by the exponential distribution in reliability theory.

B.2.2 The Piecewise Constant Hazard Distribution

If the exponential distribution is not useful in describing human lives, it may be so for short segments of life. At least it will be a good approximation if the segment is short enough.

This is the idea behind the *piecewise constant hazards* distribution, called

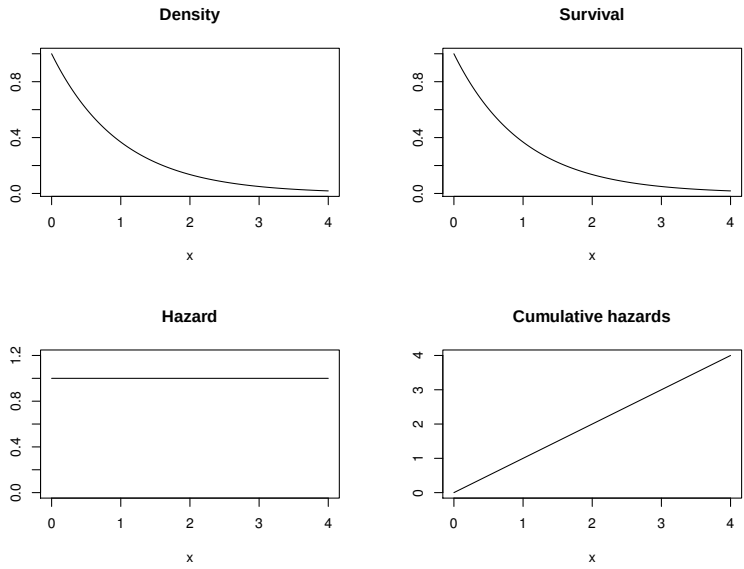


FIGURE B.1
The exponential distribution with scale parameter 1.

`pch` in `eha`. Its definition involves a partition of time (age) axis, and one positive constant (the hazard level) corresponding to each interval. Note that the last interval will be open, with infinite length; only a finite number of cut points are allowed. The definition of the hazard function h becomes, with the cuts denoted $\mathbf{t} = (t_1, < \cdots < t_n)$ and the levels denoted $\mathbf{h} = (h_1, \ldots, h_{n+1})$:

$$h(t; \mathbf{t}, \mathbf{h}) = \begin{cases} h_1 & t \leq t_1, \\ h_i & t_{i-1} < t \leq t_i,\ i = 2, \ldots, n, \\ h_{n+1} & t_n < t. \end{cases} \tag{B.1}$$

In this definition, the number of levels must be exactly one more than the number of cut points. The relevant functions are shown in Figure B.2, created as follows:

```
> cuts <- c(1, 2, 3)
> levels <- c(1, 2, 1, 2)
> par(mfrow = c(2, 2))
> plot(x, dpch(x, cuts = cuts, levels = levels),
       type = "l", main = "Density")
> plot(x, ppch(x, cuts = cuts, levels = levels, lower.tail = FALSE),
       type = "l", main = "Survival")
> plot(x, hpch(x, cuts = cuts, levels = levels),
       type = "l", main = "Hazard", ylab = "", ylim = c(0, 2.2))
```

```
> plot(x, Hpch(x, cuts = cuts, levels = levels),
       type = "l", main = "Cumulative hazards")
```

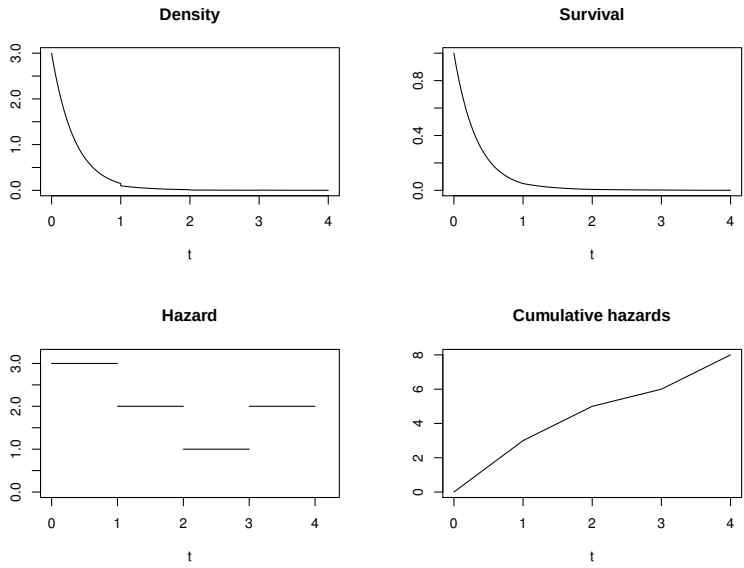


FIGURE B.2
Piecewise constant hazards distribution.

Note that, despite the fact that the hazard function is *not* continuous, the other functions are. They are not differentiable at the cut points, though.

The piecewise constant distribution is very flexible. It can be made arbitrarily close to any continuous distribution by increasing the number of cut points and choose the levels appropriately. Parametric proportional hazards modeling with the `pch` distribution is a serious competitor to the Cox regression model, especially with large data sets.

B.2.3 The Weibull Distribution

The *Weibull* distribution is a very popular parametric model for survival data, described in detail by Waloddi Weibull (Weibull 1951), known earlier. It is one of the so-called extreme-value distributions, and as such very useful in reliability theory. It is becoming popular in demographic applications, but in mortality studies it is wise to avoid it for old age mortality (the hazard grows too slow) and mortality in ages 0–15 years of age (U-shaped hazards, which the Weibull model doesn't allow).

The hazard function of a Weibull distribution is defined by

$$h\big(t;(p,\lambda)\big) = \frac{p}{\lambda}\left(\frac{t}{\lambda}\right)^{p-1}, \quad t,p,\lambda > 0.$$

where p is a *shape* parameter and λ is a *scale* parameter. When $p = 1$, this reduces to $h(t;(1,\lambda) = 1/\lambda$, which is the exponential distribution with rate $1/\lambda$. Compare to the definition of the exponential distribution and note that there λ is the *rate* and here it is a *scale* parameter, which is the inverted value of the rate.

B.2.3.1 Graphical Test of the Weibull Distribution

As mentioned above, the Weibull distribution has a long history in reliability theory. Early on simple graphical tests were constructed for judging if a real data set could be adequately described by the Weibull distribution. The so-called *Weibull paper* was invented. Starting with the definition of the *cumulative* hazards function H,

$$H\big(t;(p,\lambda)\big) = \left(\frac{t}{\lambda}\right)^{p},$$

by taking logarithms of both sides, we get

$$\ln H\big(t;(p,\lambda)\big) = p\ln(t) - p\ln(\lambda)$$

with $x = \ln t$ and $y = \ln H(t)$, this is a straight line. So by plotting the Nelson–Aalen estimator of the cumulative hazards function against log time, it is possible to graphically check the Weibull and exponential assumptions.

Example 32 *Male mortality and the Weibull distribution*

Is it reasonable to assume that the survival data in the data set `mort` can be modeled by the Weibull distribution? We construct the *Weibull* plot as follows.

```
> data(mort)
> with(mort, plot(Surv(enter, exit, event), fn = "loglog"))
```

The result is shown in Figure B.3

Except for the early disturbance, linearity is not too far away (remember that the log transform magnifies the picture close to zero; exp(-1.5) is only 0.22 years). Is the slope close to 1? Probably not, the different scales on the axis makes it hard to judge exactly. We can estimate the parameters with the `phreg` function.

```
> fit <- phreg(Surv(enter, exit, event) ~ 1, data = mort)
> fit
```

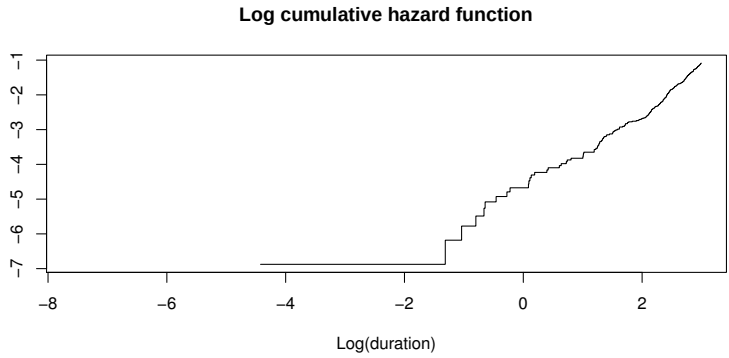


FIGURE B.3
Weibull plot of male mortality data.

```
Call:
phreg(formula = Surv(enter, exit, event) ~ 1, data = mort)
Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
log(scale)                    3.785    44.015     0.066     0.000
log(shape)                    0.332     1.393     0.058     0.000

Events                    276
Total time at risk         17038
Max. log. likelihood      -1399.3
```

The estimate of the shape parameter is 1.393, and it is significantly different from 1 ($p = 0.000$), so we can firmly reject the hypothesis that data come from an exponential distribution.

B.2.4 The Lognormal Distribution

The lognormal distribution is connected to the normal distribution through the exponential function: If X is normally distributed, then $Y = \exp(X)$ is lognormally distributed. Conversely, if Y is lognormally distributed, then $X = \log(Y)$ is normally distributed.

The lognormal distribution has the interesting property that the hazard function is first increasing, then decreasing, in contrast to the Weibull distribution which only allows for monotone (increasing or decreasing) hazard functions.

The **R** functions are named `(dpqrhH)lnorm`.

B.2.5 The Loglogistic Distribution

The loglogistic distribution is very close to the lognormal, but have heavier tail to the right. Its advantage over the lognormal is that the hazard function has closed form. It is given by

$$h(t;(p,\lambda)) = \frac{\frac{p}{\lambda}(\frac{t}{\lambda})^{p-1}}{1+(\frac{t}{\lambda})^{p}}, \quad t,p,\lambda > 0.$$

With shape $p = 2$ and scale $\lambda = 1$, its appearance is shown in Figure B.4. It

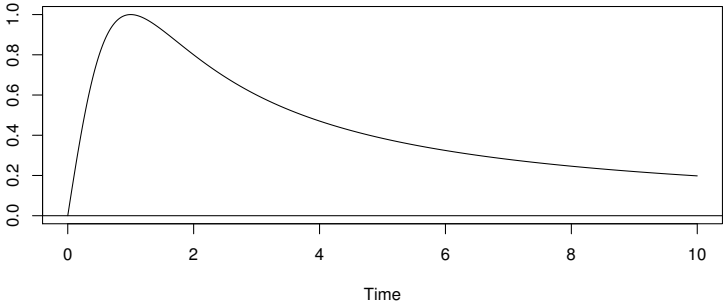


FIGURE B.4
Loglogistic hazard function with shape 2 and scale 1.

is produced by the code

```
> x <- seq(0, 10, length = 1000)
> plot(x, hllogis(x, shape = 2, scale = 1),
        type = "l", ylab = "", xlab = "Time")
> abline(h = 0)
```

The **R** functions are named `(dpqrhH)llogis`.

B.2.6 The Gompertz Distribution

The Gompertz distribution is useful for modeling old age mortality. The hazard function is exponentially increasing.

$$h(t;(p,\lambda)) = p\exp\left(\frac{t}{\lambda}\right), \quad t,p,\lambda > 0.$$

It was suggested by Gompertz (1825).

The **R** functions are named `(dpqrhH)gamma`.

B.2.7 The Gompertz–Makeham Distribution

The Gompertz distribution was generalized by (Makeham 1860). The generalization consists of adding a positive constant to the Gompertz hazard function,

$$h(t;(\alpha,p,\lambda)) = \alpha + p\exp\left(\frac{t}{\lambda}\right), \quad t,\alpha,p,\lambda > 0.$$

It is more difficult to work with than the other distributions described here. It is not one of the possible choices in the function `phreg` in `eha`.

The **R** functions are named `(dpqrhH)makeham`.

B.2.8 The Gamma Distribution

The Gamma distribution is another generalization of the exponential distribution. It is popular in modeling shared frailty, see Hougaard (2000). It is not one of the possible distributions for the functions `phreg` and `survreg`, and it is not considered further.

The **R** functions are named `(dpqr)gamma`.

B.3 Parametric Proportional Hazards and Accelerated Failure Time Models

B.3.1 Introduction

There is a need for software for analyzing parametric proportional hazards (PH) and accelerated failure time (AFT) data, which are right or interval censored and left truncated. The package `eha` gives one solution to this problem with a specific class of families of survival distributions, but at the time of this writing, interval censoring is not implemented.

The implementation in `eha` starts with the class of *scale/shape* families of distributions, to be explained later. The idea is that each family of distributions is generated by shape-scale transformations from any given survival distribution, see Section B.3.3.

B.3.2 The Proportional Hazards Model

We define proportional hazards models in terms of an expansion of a given survival function S_0,

$$s_{\boldsymbol{\theta}}(t;\mathbf{z}) = \{S_0(g(t,\boldsymbol{\theta}))\}^{\exp(\mathbf{z}\boldsymbol{\beta})}, \tag{B.2}$$

where $\boldsymbol{\theta}$ is a parameter vector used in modeling the baseline distribution, $\boldsymbol{\beta}$ is the vector of regression parameters, and g is a positive function, which helps

defining a parametric family of baseline survival functions through

$$S(t;\boldsymbol{\theta}) = S_0\big(g(t,\boldsymbol{\theta})\big), \quad t > 0, \quad \boldsymbol{\theta} \in \boldsymbol{\Theta}. \tag{B.3}$$

With f_0 and h_0 defined as the density and hazard functions corresponding to S_0, respectively, the density function corresponding to S is

$$\begin{aligned} f(t;\boldsymbol{\theta}) &= -\frac{\partial}{\partial t} S(t,\boldsymbol{\theta}) \\ &= -\frac{\partial}{\partial t} S_0(g(t,\boldsymbol{\theta})) \\ &= g_t(t,\boldsymbol{\theta}) f_0(g(t,\boldsymbol{\theta})), \end{aligned}$$

where

$$g_t(t,\boldsymbol{\theta}) = \frac{\partial}{\partial t} g(t,\boldsymbol{\theta}).$$

Correspondingly, the hazard function is

$$\begin{aligned} h(t;\boldsymbol{\theta}) &= \frac{f(t;\boldsymbol{\theta})}{S(t;\boldsymbol{\theta})} \\ &= g_t(t,\boldsymbol{\theta}) h_0(g(t,\boldsymbol{\theta})). \end{aligned} \tag{B.4}$$

So, the proportional hazards model is

$$\begin{aligned} \lambda_{\boldsymbol{\theta}}(t;\mathbf{z}) &= h(t;\boldsymbol{\theta}) \exp(\mathbf{z}\boldsymbol{\beta}) \\ &= g_t(t,\boldsymbol{\theta}) h_0(g(t,\boldsymbol{\theta})) \exp(\mathbf{z}\boldsymbol{\beta}), \end{aligned} \tag{B.5}$$

corresponding to (B.2).

B.3.2.1 Data and the Likelihood Function

Given left-truncated and right- or interval-censored data $(s_i, t_i, u_i, d_i, \mathbf{z}_i)$, $i = 1, \ldots, n$ and the model (B.5), the likelihood function becomes

$$\begin{aligned} L\big((\boldsymbol{\theta},\boldsymbol{\beta}); (\mathbf{s},\mathbf{t},\mathbf{u},\mathbf{d}), \mathbf{Z}\big) = \prod_{i=1}^{n} \big\{ &\big(h(t_i;\boldsymbol{\theta}) \exp(\mathbf{z}_i\boldsymbol{\beta})\big)^{I_{\{d_i=1\}}} \\ &\times \big(S(t_i;\boldsymbol{\theta})^{\exp(\mathbf{z}_i\boldsymbol{\beta})}\big)^{I_{\{d_i\neq 2\}}} \\ &\times \big(S(t_i;\boldsymbol{\theta})^{\exp(\mathbf{z}_i\boldsymbol{\beta})} - S(u_i;\boldsymbol{\theta})^{\exp(\mathbf{z}_i\boldsymbol{\beta})}\big)^{I_{\{d_i=2\}}} \\ &\times S(s_i;\boldsymbol{\theta})^{-\exp(\mathbf{z}_i\boldsymbol{\beta})} \big\} \end{aligned} \tag{B.6}$$

Here, for $i = 1, \ldots, n$, $s_i < t_i \leq u_i$ are the left truncation and exit intervals, respectively, $d_i = 0$ means that $t_i = u_i$ and right censoring at u_i, $d_i = 1$ means that $t_i = u_i$ and an event at u_i, and $d_i = 2$ means that $t_i < u_i$ and an event occurs in the interval (t_i, u_i) (interval censoring) and $\mathbf{z}_i = (z_{i1}, \ldots, z_{ip})$ is a vector of explanatory variables with corresponding parameter vector $\boldsymbol{\beta} =$

$(\beta_1, \ldots, \beta_p)$, $i = 1, \ldots, n$. Note that interval censoring is not yet implemented in `eha`.

From (B.6) we now get the log likelihood and the *score vector* in a straightforward but tedious manner. All the details can be found in the *vignette* of `eha`.

B.3.3 The Shape-Scale Families

From (B.2) we get a *shape-scale* family of distributions by choosing $\boldsymbol{\theta} = (p, \lambda)$ and

$$g(t, (p, \lambda)) = \left(\frac{t}{\lambda}\right)^p, \quad t \geq 0; \quad p, \lambda > 0.$$

However, for reasons of efficient numerical optimization and normality of parameter estimates, we use the parametrization $p = \exp(\gamma)$ and $\lambda = \exp(\alpha)$, thus redefining g to

$$g(t, (\gamma, \alpha)) = \left(\frac{t}{\exp(\alpha)}\right)^{\exp(\gamma)}, \quad t \geq 0; \quad -\infty < \gamma, \alpha < \infty. \tag{B.7}$$

For the calculation of the score and Hessian of the log likelihood function, some partial derivatives of g are needed. See the *vignette* of the `eha` package. It is part of the distribution of the package.

B.3.3.1 The Weibull Family of Distributions

The Weibull family of distributions is obtained by

$$S_0(t) = \exp(-t), \quad t \geq 0,$$

leading to

$$f_0(t) = \exp(-t), \quad t \geq 0,$$

and

$$h_0(t) = 1, \quad t \geq 0.$$

We need some first- and second-order derivatives of f and h, and they are particularly simple in this case, for h they are both zero, and for f we get

$$f_0'(t) = -\exp(-t), \quad t \geq 0.$$

B.3.3.2 The EV family of distributions

The EV (Extreme Value) family of distributions is obtained by setting

$$h_0(t) = \exp(t), \quad t \geq 0,$$

leading to

$$S_0(t) = \exp\{-(\exp(t) - 1)\}, \quad t \geq 0,$$

The rest of the necessary functions are easily derived from this.

B.3.3.3 The Gompertz Family of Distributions

The Gompertz family of distributions is given by

$$h(t) = \tau \exp(t/\lambda), \quad t \geq 0; \quad \tau, \lambda > 0.$$

This family is not directly possible to generate from the described shape-scale models, so it is treated separately by direct maximum likelihood.

B.3.3.4 Other Families of Distributions

Included in the *eha* package are the lognormal and the loglogistic distributions as well.

B.3.4 The Accelerated Failure Time Model

The presentation here assumes time-fixed covariates. The function `aftreg` in `eha` allows for time-varying covariates and left truncation, which requires an assumption of constancy and knowledge of covariate values during the time of nonobservation. More can be found in the online documentation of `aftreg`.

In the description of this family of models, we generate a scape-scale family of distributions as defined by the Equations (B.3) and (B.7). We get

$$\begin{aligned} S(t;(\gamma,\alpha)) &= S_0\big(g(t,(\gamma,\alpha))\big) \\ &= S_0\left(\left\{\frac{t}{\exp(\alpha)}\right\}^{\exp(\gamma)}\right), \quad t > 0, \quad -\infty < \gamma, \alpha < \infty. \end{aligned} \tag{B.8}$$

Given a vector $\mathbf{z} = (z_1, \ldots, z_p)$ of explanatory variables and a vector of corresponding regression coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, the AFT regression model is defined by

$$\begin{aligned} S(t;(\gamma,\alpha,\boldsymbol{\beta})) &= S_0\big(g(t\exp(\mathbf{z}\boldsymbol{\beta}),(\gamma,\alpha))\big) \\ &= S_0\left(\left\{\frac{t\exp(\mathbf{z}\boldsymbol{\beta})}{\exp(\alpha)}\right\}^{\exp(\gamma)}\right) \\ &= S_0\left(\left\{\frac{t}{\exp(\alpha-\mathbf{z}\boldsymbol{\beta})}\right\}^{\exp(\gamma)}\right) \\ &= S_0\big(g(t,(\gamma,\alpha-\mathbf{z}\boldsymbol{\beta}))\big), \quad t > 0. \end{aligned} \tag{B.9}$$

So, by defining $\boldsymbol{\theta} = (\gamma, \alpha - \mathbf{z}\boldsymbol{\beta})$, we are back in the framework of Section B.3.2. We get

$$f(t;\boldsymbol{\theta}) = g_t(t,\boldsymbol{\theta}) f_0(g(t,\boldsymbol{\theta}))$$

and

$$h(t;\boldsymbol{\theta}) = g_t(t,\boldsymbol{\theta}) h_0(g(t,\boldsymbol{\theta})), \tag{B.10}$$

defining the AFT model generated by the survival function S_0 and corresponding density f_0 and hazard h_0.

B.3.4.1 Data and the Likelihood Function

Given left-truncated and right- or interval-censored data $(s_i, t_i, u_i, d_i, \mathbf{z}_i)$, $i = 1, \ldots, n$ and the model (B.10), the likelihood function becomes

$$\begin{aligned} L\big((\gamma, \alpha, \boldsymbol{\beta}); (\mathbf{s}, \mathbf{t}, \mathbf{d}), \mathbf{Z}\big) = \prod_{i=1}^{n} \big\{ & h(t_i; \boldsymbol{\theta}_i)^{I_{\{d_i=1\}}} \\ & \times S(t_i; \boldsymbol{\theta}_i)^{I_{\{i\neq 2\}}} \\ & \times \big(S(t_i; \boldsymbol{\theta}_i) - S(u_i; \boldsymbol{\theta}_i)\big)^{I_{\{d_i=2\}}} \\ & \times S(s_i; \boldsymbol{\theta}_i)^{-1} \big\} \end{aligned} \tag{B.11}$$

Here, for $i = 1, \ldots, n$, $s_i < t_i \leq u_i$ are the left truncation and exit intervals, respectively, $d_i = 0$ means that $t_i = u_i$ and right censoring at t_i, $d_i = 1$ means that $t_i = u_i$ and an event at t_i, and $d_i = 2$ means that $t_i < u_i$ and an event occurs in the interval (t_i, u_i) (interval censoring), and $\mathbf{z}_i = (z_{i1}, \ldots, z_{ip})$ is a vector of explanatory variables with corresponding parameter vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, $i = 1, \ldots, n$.

From (B.11) we now get the log likelihood and the score vector in a straightforward but tedious manner. See the *vignette* of the **R** package `eha`.

C

A Brief Introduction to R

There are already many excellent sources of information available for those who want to learn more about **R**. A visit to the **R** home page, `http:\\www.r-project.org`, is highly recommendable. Look under *Documentation*. When **R** is installed on a computer, it is easy to access the documentation that comes with the distribution; start **R** and open the html help (`?help.start`). A good start for a beginner is also the book *Introductory Statistics with R* by Dalgaard (2008). The book on *S Programming* by Venables & Ripley (2000) is recommended for the more advanced studies of the topic that its title implies.

C.1 R in General

In this section we investigate how to work with **R** in general, without special reference to event history analysis, which will be treated separately in a later section in this chapter.

C.1.1 R Objects

R is an object-oriented programming language, which have certain implications for terminology. So is for instance everything i **R** *objects*. We will not draw too much on this fact, but it may be useful to know.

C.1.2 Expressions and Assignments

Commands are either *expressions* or *assignments*. Commands are separated by newline (normally) or semicolon. The hash mark (#) marks the rest of the line as a *comment*.

```
> 1 + exp(2.3) # exp is the exponential function
[1] 10.97418
> x <- 1 + exp(2.3)
> x
[1] 10.97418
```

Note that the **R** *prompt* is >. If the previous expression is *incomplete*, it changes to +.

The first line above is an expression. It is evaluated as `return` is pressed, and the result is normally printed. The second line is an assignment; the result is stored in x and not printed. By typing x, its content is normally printed.

The assignment symbol is " <- ", which consists of two consecutive key strokes, "<" followed by "-". It is strongly recommended to enclose the assignment symbol (and any arithmetic operator) between spaces. It is *forbidden* to separate the two symbols by a space! You will then get comparison with something negative:

```
> x <- 3
> x < - 3
[1] FALSE
```

C.1.3 Objects

Everything in **R** is an *object* (object-oriented programming). All objects have a *mode* and a *length*. Data objects have modes *numeric*, *complex*, *logical*, or *character*, and language objects have modes *function*, *call*, *expression*, *name*, etc. All data objects are *vectors*; there are *no scalars* in **R**.

Vectors contain elements of the same kind; a *list* can be thought of as a vector where each element can be anything, even a list, and the elements can be completely different.

There are more properties of objects worth mentioning, but we save that for the situations where we need them.

C.1.4 Vectors and Matrices

There are five basic types of vectors: `logical`, `integer`, `double`, `complex`, and `character`. The function `c` (for *concatenate*) creates vectors:

```
> dat <- c(2.0, 3.4, 5.6)
> cols <- c("red", "green", "blue")
```

The elements of a vector may be *named*:

```
> names(dat) <- c("Sam", "John", "Kate")
> names(dat)
[1] "Sam"  "John" "Kate"
> dat
 Sam John Kate
 2.0  3.4  5.6
> dat["John"]
John
 3.4
```

Note the first two lines; `names` may either *display* (second row) or *replace* (first row).

The function *matrix* creates a matrix (surprise!):

```
> Y <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3)
> Y
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> dim(Y) <- c(3, 2)
> Y
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> as.vector(Y)
[1] 1 2 3 4 5 6
> mode(Y)
[1] "numeric"
> attributes(Y)
$dim
[1] 3 2
```

It is also possible to create vectors by the functions `numeric`, `character`, and `logical`.

```
> x <- logical(7)
> x
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> x <- numeric(7)
> x
[1] 0 0 0 0 0 0 0
> x <- character(7)
> x
[1] "" "" "" "" "" "" ""
```

These functions are useful for *allocating* storage (memory), which later can be filled. This is better practise than letting a vector grow sequentially.

The functions `nrow` and `ncol` extracts the number of rows and columns, respectively, of a matrix.

An *array* is an extension of a matrix to more than two dimensions.

C.1.5 Lists

Family records may be produced as a list:

```
> fam <- list(FamilyID = 1, man = "John", wife = "Kate",
              children = c("Sam", "Bart"))
> fam
$FamilyID
[1] 1
$man
[1] "John"

$wife
[1] "Kate"

$children
[1] "Sam"  "Bart"
> fam$children
[1] "Sam"  "Bart"
```

C.1.6 Data Frames

A *data frame* is a special case of a `list`. It consists of variables of the same length, but not necessarily of the same type. Data frames are created by either `data.frame` or `read.table`, where the latter reads data from an ASCII file on disk.

```
> dat <- data.frame(name = c("Homer", "Flanders", "Skinner"),
                    income = c(1000, 23000, 85000))
> dat
      name income
1    Homer   1000
2 Flanders  23000
3  Skinner  85000
```

A data frame has *row names* (here: 1, 2, 3) and variable (column) names (here: `name`, `income`). The data frame is the normal unit for data storage and statistical analysis in **R**.

C.1.7 Factors

Categorical variables are conveniently stored as `factors` in **R**.

```
> country <- factor(c("se", "no", "uk", "uk", "us"))
> country
[1] se no uk uk us
Levels: no se uk us
> print.default(country)
[1] 2 1 3 3 4
```

Factors are internally coded by integers (1, 2, ...). The levels are by default sorted into alphabetical order. Can be changed:

```
> country <- factor(c("se", "no", "uk", "uk", "us"),
                    levels = c("uk", "us", "se", "no"))
> country
[1] se no uk uk us
Levels: uk us se no
```

The first level often have a special meaning ("reference category") in statistical analyses in **R**. The function `relevel` can be used to change the reference category for a given factor.

C.1.8 Operators

Arithmetic operations are performed on vectors, element by element. The common operators are `+ - * / ^` , where `^` is exponentiation. For instance,

```
> x <- c(1, 2, 3)
> x^2
[1] 1 4 9
> y <- 2 * x
> y / x
[1] 2 2 2
```

C.1.9 Recycling

The last examples above were examples of *recycling*; If two vectors of different lengths occur in an expression, the shorter one is recycled (repeated) as many times as necessary. For instance,

```
> x <- c(1, 2, 3)
> y <- c(2, 3)
> x / y
[1] 0.5000000 0.6666667 1.5000000
Warning message:
In x/y : longer object length is not a multiple of shorter object
length
```

so the actual operation performed is

```
> c(1, 2, 3) / c(2, 3, 2) # Truncated to equal length
[1] 0.5000000 0.6666667 1.5000000
```

It is most common with the shorter vector being of length one, that is, a scalar multiplication. If the length of the longer vector is not a multiple of the length of the shorter vector, a *warning* is issued; see output above. This is often a sign of a mistake in the code.

TABLE C.1 Precedence of operators, from highest to lowest

$ [[	List extraction of elements of a list.
[	Vector extraction.
^	Exponentiation.
-	Unary minus.
:	Sequence generation
%% %/% %*%	Special operators.
* /	Multiplication and division.
+ -	Addition and subtraction.
< > <= >= == !=	Comparison.
!	Logical negation.
& | && ||	Logical operators.
~	Formula.
<-	Assignment.

C.1.10 Precedence

Multiplication and division is performed before addition and subtraction, as in pure mathematics. The (almost) full set of rules is displayed in Table C.1.

It is, however, highly recommended to use parentheses often rather than relying on remembering those rules. Compare, for instance,

```
> 1:3 + 1
[1] 2 3 4
> 1:(3 + 1)
[1] 1 2 3 4
```

For descriptions of specific operators, consult the help system.

C.2 Some Standard R Functions

1. `round`, `floor`, `ceiling`: Rounds to nearest integer, downwards, and upwards, respectively.

```
> round(2.5) # Round to even
[1] 2
> floor(2.5)
[1] 2
> ceiling(2.5)
[1] 3
```

2. %/% and %% for integer divide and modulo reduction.

```
> 5 %/% 2
[1] 2
> 5 %% 2
[1] 1
```

3. Mathematical functions: `abs`, `sign`, `log`, `exp`, etc.

4. `sum`, `prod`, `cumsum`, `cumprod` for sums and products.

5. `min`, `max`, `cummin`, `cummax` for extreme values.

6. `pmin`, `pmax` parallel min and max.

```
> x <- c(1, 2, 3)
> y <- c(2, 1, 4)
> max(x, y)
[1] 4
> pmax(x, y)
[1] 2 2 4
```

7. `sort`, `order` for sorting and ordering. Especially useful is `order` for rearranging a data frame according to a specific ordering of one variable:

```
> require(eha)
> data(mort)
> mt <- mort[order(mort$id, -mort$enter), ]
> last <- mt[!duplicated(mt$id), ]
```

   First, `mort` is sorted after *id*, and within *id* decreasingly after *enter* (notice the minus sign in the formula!). Then all rows with duplicated *id* are removed, only the *first* appearance is kept. In this way we get a data frame with exactly one row per individual, the row that corresponds to the individual's last (in age or calendar time). See the next item!

8. `duplicated` and `unique` for marking duplicated elements in a vector and removing duplicates, respectively.

```
> x <- c(1,2,1)
> duplicated(x)
[1] FALSE FALSE  TRUE
> unique(x)
[1] 1 2
> x[!duplicated(x)]
[1] 1 2
```

C.2.1 Sequences

The operator `:` is used to generate sequences of numbers.

```
> 1:5
[1] 1 2 3 4 5
> 5:1
[1] 5 4 3 2 1
> -1:5
[1] -1  0  1  2  3  4  5
> -(1:5)
[1] -1 -2 -3 -4 -5
> -1:-5
[1] -1 -2 -3 -4 -5
```

Don't trust that you remember rules of precedence; use parentheses!

There is also the functions `seq` and `rep`, see the help pages!

C.2.2 Logical expression

Logical expressions can take only two distinct values, `TRUE` and `FALSE` (note uppercase; **R** acknowledges the difference between lower case and upper case, unlike, e.g., `Windows`).

Logical vectors may be used in ordinary arithmetic, when they are *coerced* into numeric vectors with `FALSE` equal to zero and `TRUE` equal to one.

C.2.3 Indexing

Indexing is a powerful feature in **R**. It comes in several flavors.

A logical vector Must be of the same length as the vector it is applied to.

```
> x <- c(-1:4)
> x
[1] -1  0  1  2  3  4
> x[x > 0]
[1] 1 2 3 4
```

Positive integers Selects the corresponding values.

Negative integers Rejects the corresponding values.

Character strings For named vectors; select the elements with the given names.

Empty Selects all values. Often used to select rows/columns from a matrix,

```
> x <- matrix(c(1,2,3,4), nrow = 2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> x[, 1]
[1] 1 2
> x[, 1, drop = FALSE]
     [,1]
[1,]    1
[2,]    2
```

Note that a dimension is lost when one row/column is selected. This can be overridden by the argument `drop = FALSE` (*don't drop dimension(s)*).

C.2.4 Vectors and Matrices

A *matrix* is a vector with a `dim` attribute.

```
> x <- 1:4
> x
[1] 1 2 3 4
> dim(x) <- c(2, 2)
> dim(x)
[1] 2 2
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> t(x)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Note that matrices are stored *column-wise*. The function `t` gives the transpose of a matrix.

An example of matrix multiplication.

```
> x <- matrix(c(1,2,3,4, 5, 6), nrow = 2)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> y <- matrix(c(1,2,3,4, 5, 6), ncol = 2)
> y
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> y %*% x
     [,1] [,2] [,3]
[1,]    9   19   29
[2,]   12   26   40
[3,]   15   33   51
> x %*% y
     [,1] [,2]
[1,]   22   49
[2,]   28   64
```

Dimensions must match according to ordinary mathematical rules.

C.2.5 Conditional Execution

Conditional execution uses the `if` statement (but see also `switch`). Conditional constructs are typically used inside *functions*.

```
> x <- 3
> if (x > 3) {
       cat("x is larger than 3\n")
   }else{
       cat("x is smaller than or equal to 3\n")
   }
x is smaller than or equal to 3
```

See also the functions `cat` and `print`.

C.2.6 Loops

Loops are typically only used inside *functions*.

1. A `for` loop makes an expression to be iterated as a variable assumes all values in a sequence.

```
> x <- 1
> for (i in 1:4){
       x <- x + 1
       cat("i = ", i, ", x = ", x, "\n")
   }
i =  1 , x =  2
i =  2 , x =  3
i =  3 , x =  4
i =  4 , x =  5
```

```
> x
[1] 5
```

2. The `while` loop iterates as long as a condition is `TRUE`.

```
> done <- FALSE
> i <- 0
> while (!done & (i < 5)){
       i <- i + 1
       done <- (i > 3)
   }
> if (!done) cat("Did not converge\n")
```

3. The `repeat` loop iterates indefinitely or until a `break` statement is reached.

```
> i <- 0
> done <- FALSE
> repeat{
       i <- i + 1
       done <- (i > 3)
       if (done) break
   }
> i
[1] 4
```

C.2.7 Vectorizing

In **R**, there is a family of functions that operates directly on vectors. See the documentation of `apply`, `tapply`, `lapply`, and `sapply`.

C.3 Writing Functions

Functions are created by assignment using the keyword `function`.

```
> fun <- function(x) exp(x)
> fun(1)
[1] 2.718282
```

This function, `fun`, is the exponential function, obviously. Usually, the function body consists of several statements.

```
> fun2 <- function(x, maxit){
       if (any(x <= 0)) error("Only valid for positive x")
       i <- 0
       sum <- 0
       for (i in 0:maxit){
           sum <- sum + x^i / gamma(i + 1)
       }
       sum
   }
> fun2(1, 5)
[1] 2.716667
> fun2(1:5, 5)
[1]  2.716667  7.266667 18.400000 42.866667 91.416667
```

This function, `fun2`, uses the Taylor expansion of $\exp(x)$ for calculation. This is a very crude variant, and it is only good for positive x. It throws an error if fed by a negative value. Note that it is *vectorized*, that is, its first argument may be a real-valued vector. Also note the use of the function `any`.

In serious function writing, it is very unconvenient to define the function at the command prompt. It is much better to write it in a good editor and then `source` it into the **R** session as needed. If it will be used extensively in the future, the way to go is to write a `package`. This is covered in *Writing R Extensions*, available from the help system. Our favourite editor, and the one used in examples in these notes, is `emacs`, which is free and available on all platforms. Make sure you use ESS (*Emacs Speaks Statistics*) together with emacs. Emacs and ESS are easily found by searching the web.

Functions can be defined inside the body of another function. In this way, it is hidden for other functions, which is important when considering scoping rules. This is a feature that is not available in FORTRAN (77) or `C`. Suppose that the function `inside` is defined inside the function `head`.

1. The function `inside` is not visible outside `head` and cannot be used there.

2. A call to `inside` will use the function defined in `head`, and not some function outside with the same name.

The *return* value of a function is the value of the last expression in the function body.

C.3.1 Calling Conventions

The arguments to a function are either *specified* or *unspecified* (or both). Unspecified arguments are shown as "..." in the function definition. See, for instance, the functions `c` and `min`. The "..." may be replaced by any number of arguments in a call to the function.

The *formal* arguments are those specified in the function definition, and

the *actual* arguments are those given in the actual call to the function. The rules by which formal and actual arguments are *matched* are

1. Any actual argument given as `name = value`, where `name` exactly matches a formal argument, is matched.
2. Arguments specified by `name = value`, but no exact match, are then considered. If a perfect *partial* match is found, it is accepted.
3. The remaining unnamed arguments are then matched *in sequence* (*positional* matching).
4. Remaining unmatced actual arguments are part of the ... argument if there is one. Otherwise, an error occurs.
5. Formal arguments need not be matched.

C.3.2 The Argument "..."

The purpose of the `...` argument is usually to pass arguments to a function call inside the actual function. An example with the `plot` function, see Figure C.1:

```
> source("dots.R")
> dots
function (x, ...)
{
    plot(x, ...)
}
```

Note that "dot arguments" must be named in the function call.

C.3.3 Writing Functions

Below is a function that calculates the maximum likelihood estimate of the parameter in the exponential distribution, given a censored sample.

```
> source("mlexp.R")
> mlexp
function (x, d)
{
    n <- length(x)
    if (any(x <= 0))
        stop("This function needs positive x")
    if (length(d) != n)
        stop("Lengths of x and d do not match")
    d <- as.numeric(d != 0)
    sumx <- sum(x)
    sumd <- sum(d)
```

```
> dots(rnorm(100), main = "Normal variate")
```

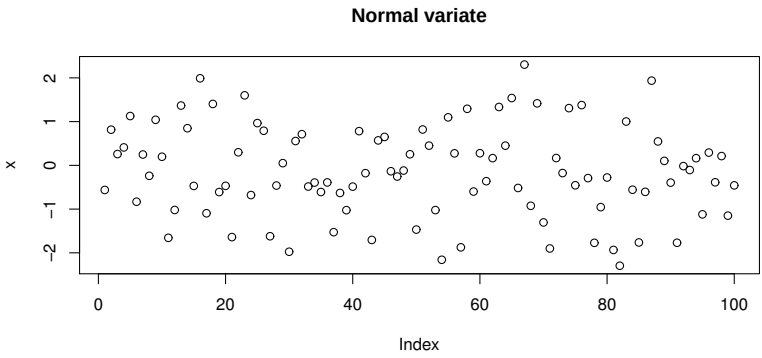


FIGURE C.1
Example of the use of "..." in function calls.

```
    loglik <- function(alpha) {
        sumd * alpha - sumx * exp(alpha)
    }
    dloglik <- function(alpha) {
        sumd - sumx * exp(alpha)
    }
    alpha <- 0
    res <- optim(alpha, fn = loglik, gr = dloglik, method = "BFGS",
        control = list(fnscale = -1))
    res
}
```

First note that the arguments of the function, `x` and `d`, do not have any default values. Both arguments must be matched in the actual call to the function. Second, note the check of sanity in the beginning of the function and the calls to `stop` in case of an error. For less severe mistakes it is possible to give a `warning`, which does not break the execution of the function.

```
> exit <- rexp(100, 1)
> event <- as.numeric(exit < 2)
> exit <- pmax(exit, 2)
> res <- mlexp(exit, event)
> res
$par
[1] -0.8450675
$value
[1] -166.0561
```

```
$counts
function gradient
      20        6

$convergence
[1] 0

$message
NULL
> unlist(res)
            par           value counts.function counts.gradient
     -0.8450675    -166.0560743      20.0000000       6.0000000
    convergence
      0.0000000
```

C.3.4 Lazy Evaluation

When a function is called, the arguments are parsed, but not evaluated until it is needed in the function. Specifically, if the function does not use the argument, it is never evaluated, and the variable need not even exist. This is in contrast to rules in C and FORTRAN.

```
> fu <- function(x, y) 2 * x
> fu(3)
[1] 6
> fu(3, y = zz)
[1] 6
> exists("zz")
[1] FALSE
```

C.3.5 Recursion

Functions are allowed to be *recursive*, that is, a function in **R** is allowed to call itself. As an example, the function `gsort`, defined below, does sorting by finding the smallest value in a vector to be sorted, and then calls itself on the remaining values after the smallest is removed.

```
> source("gsort.R")
> gsort
function (x)
{
    n <- length(x)
    if (n == 1)
        return(x)
    sorted <- numeric(n)
```

```
    smallest <- which.min(x)
    sorted[1] <- x[smallest]
    sorted[2:n] <- gsort(x[-smallest])
    sorted
}
> gsort(rnorm(10))
 [1] -0.3147130 -0.1752400  0.1704180  0.2291849  0.4321751
 [6]  0.4782381  1.2201831  1.2799731  1.3885228  1.3996963
```

Use this technique with extreme caution; it is extremely memory-intensive, and may be very slow. On the plus side is that it allows very neat algorithms.

More compact code:

```
> source("gcsort.R")
> gcsort
function (x)
{
    if (length(x) == 1)
        return(x)
    smallest <- which.min(x)
    c(x[smallest], gsort(x[-smallest]))
}
```

C.3.6 Vectorized Functions

Most mathematical functions in **R** are *vectorized*, meaning that if the argument to the function is a vector, then the result is a vector of the same length. Some functions apply the recycling rule to vectors shorter than the longest:

```
> sqrt(c(9, 4, 25))
[1] 3 2 5
> pnorm(1, mean = 0, sd = c(1,2,3))
[1] 0.8413447 0.6914625 0.6305587
```

When writing own functions, the question of making it vectorized should always be considered. Often, it is automatically vectorized, like

```
> fun <- function(x) 3 * x - 2
> fun(c(1,2,3))
[1] 1 4 7
```

but not always. Under some circumstances, it may be necessary to explicitly make the calculations separately for each element in a vector.

C.3.7 Scoping Rules

Within a function in **R**, objects are found in the following order:

1. It is a *locally* defined variable.
2. It is in the argument list.
3. It is defined in the environment where the function is defined.

```
> source("scop.R")
> scop
function (x)
{
    y <- x^2
    z <- 2
    fun <- function(p) {
        z <- 0
        m <- exp(p * y) + x - z
        m
    }
    fun(z)
}
> scop(2)
[1] 2982.958
```

Note that a function will eventually look for the variable in the work space. This is often *not* the intention!

Tip: Test functions in a *clean workspace*. For instance, start R with the flag `-vanilla`.

C.4 Graphics

The workhorses in standard graphics handling in **R** are the functions `plot` and `par`. Read their respective documentations carefully; you can manipulate your plots almost without limitation!

See the code and Figure C.2 for the output.

```
> source("plot2.R")
> plot2
function ()
{
    x <- rnorm(100)
    y <- x + rnorm(100, sd = 0.1)
    oldpar <- par(mfrow = c(1, 2))
```

```
    on.exit(par(oldpar))
    plot(x, y, main = "Scatterplot of x and y")
    hist(y, main = "Histogram of y", probability = TRUE)
}
```

```
> plot2()
```

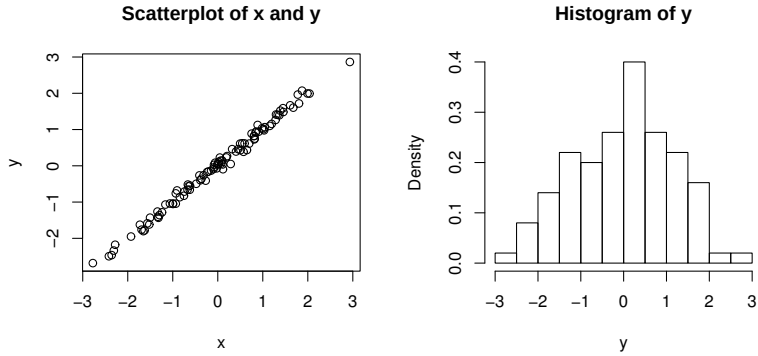


FIGURE C.2
Output from the function `plot2`.

C.5 Probability Functions

In **R** several of the common distributions are represented by (at least) four functions each, one for *density* (prefix `d`), one for *probability* (prefix `p`), one for *quantile* (prefix `q`), and one for *random number generation* (prefix `r`). For a list of available functions and more detailed information about them, see Chapter 8 in *An Introduction to R*. See also the online help system, for instance,

```
> ?pnorm
```

Some special packages contain more distribtions, for instance, the `eha` package (extreme-value, Gompertz, piecewise constant hazard). See Appendix B for more detail.

C.5.1 Some Useful R Functions

C.5.1.1 Matching

There are some useful functions in **R** doing *set operations*. For exact definitions of these functions, see the documentation.

match `match(x, y)` returns a vector of the same length as `x` with the place of the first occurrence of each element in `x` in `y`. If there is mo match for a particular element in `x`, `NA` is returned (can be changed). Example

```
> x <- c(1,3,2,5)
> y <- c(3, 2, 4, 1, 3)
> match(x, y)
```

```
[1]  4  1  2 NA
```

The `match` function is useful picking information in one data frame and put it in another (or the same). As an example, look at the data frame `oldmort`.

```
> require(eha)
> data(oldmort)
> names(oldmort)
```

```
 [1] "id"         "enter"      "exit"       "event"
 [5] "birthdate"  "m.id"       "f.id"       "sex"
 [9] "civ"        "ses.50"     "birthplace" "imr.birth"
[13] "region"
```

Note the variables *id* (id of current record) and *m.id* (id of the mother to the current). Some of the mothers are also subjects in this file, and if we in an analysis want to use mother's *birthplace* as a covariate, we can get it from her record (if present; there will be many missing mothers). This is accomplished with `match`; creating a new variable *m.birthplace* (mother's birthplace) is as simple as

```
> oldmort$m.birthplace <- with(oldmort,
                               birthplace[match(m.id, id)])
```

The tricky part here is to get the order of the arguments to `match` right. Always check the result on a few records!

%in% Simple version of `match`. Returns a logical vector of the same length as `x`, see below.

```
> x %in% y
```

```
[1]  TRUE  TRUE  TRUE FALSE
```

A simple and efficient way of selecting subsets of a certain kind, For instance, to select all cases with parity 1, 2, or 3, use

```
> data(fert)
> f13 <- fert[fert$parity %in% 1:3, ]
```

This is the same as

```
> f13 <- fert[(fert$parity >= 1) & (fert$parity <= 3), ]
```

Note that the parantheses in the last line are strictly unnecessary, but their presence increases readability and avoids stupid mistakes about precedence between operators. Use parentheses often!

subset Selects subsets of vectors, arrays, and data frames. See the help page.

tapply Applies a function to all subsets of a data frame (it is more general than that; see the help page). Together with `match` it is useful for creating new summary variables for clusters and sticking the values to each individual.

If we for all rows in the `fert` data frame want to add the age at first birth for the corresponding mother, we can do this as follows.

```
> data(fert)
> indx <- tapply(fert$id, fert$id)
> min.age <- tapply(fert$age, fert$id, min)
> fert$min.age <- min.age[indx]
```

Check this for yourself! And read the help pages for `match` and `tapply` carefully.

C.5.1.2 General utility functions

pmax, pmin Calculates the parallel max and min, respectively.

```
> x <- c(2, 3, 1, 7)
> y <- c(1, 2, 3)
> pmin(x, y)
[1] 1 2 1 1
> min(x, y)
[1] 1
```

ifelse A vectorized version of `if`.

```
> x <- rexp(6)
> d <- ifelse(x > 1, 0, 1)
> x
```

```
[1] 1.2775732 0.5135248 0.4508808 0.9722598 2.1630381 0.3899085
> d
[1] 0 1 1 1 0 1
```

C.6 Help in R

Besides all documentation that can be found on CRAN (`http://cran.r-project.org`), there is an extensive help system online in **R**. To start the general help system, type

```
> help.start()
```

at the prompt (or, find a corresponding menu item on a GUI system). Then a help window opens up with access to several FAQs, and all the packages that are installed.

For immediate help on a specific function you know the name of, type

```
> ?coxreg
```

If you only need to see the syntax and input arguments, use the function `args`:

```
> args(coxreg)
function (formula = formula(data), data = parent.frame(), weights,
    subset, t.offset, na.action = getOption("na.action"),
    init = NULL, method = c("efron", "breslow", "mppl", "ml"),
    control = list(eps = 1e-08, maxiter = 25, trace = FALSE),
    singular.ok = TRUE, model = FALSE, center = TRUE, x = FALSE,
    y = TRUE, boot = FALSE, efrac = 0, geometric = FALSE, rs = NULL,
    frailty = NULL, max.survs = NULL)
NULL
```

C.7 Functions in `eha` and `survival`

Here some useful functions in `eha` are listed. In most cases the description is very sparse, and the reader is recommended to consult the help pages when more detail is wanted.

aftreg Fits accelerated failure time (AFT) models.

age.window Makes a "horizontal cut" in the Lexis diagram, that is, it selects a subset of a data set based on age. The data frame must contain three variables that can be input to the `Surv` function. The default names are *enter*, *exit*, and *event*. As an example,

```
> require(eha)
> data(oldmort)
> mort6070 <- age.window(oldmort, c(60, 70))
```

limits the study of old age mortality to the (exact) ages between 60 and 70, that is. from 60.000 up to and including 69.999 years of age.

For *rectangular* cuts in the Lexiz diagram, use both `age.window` and `cal.window` in succession. The order is unimportant.

cal.window Makes a "vertical cut" in the Lexis diagram, that is, it selects a subset of a data set based on *calendar time*. As an example,

```
> require(eha)
> data(oldmort)
> mort18601870 <- cal.window(oldmort, c(1860, 1870))
```

limits the study of old age mortality to the time period between 1860 and 1870, that is, from 1 January 1860 up to and including 31 December 1869.

For *rectangular* cuts in the Lexiz diagram, use both `age.window` and `cal.window` in succession. The order is unimportant.

cox.zph Tests the proportionality assumption on fits from `coxreg` and `coxph`.

coxreg Fits Cox proportional hazards model. It is a wrapper for `coxph` in case of simpler Cox regression models.

glmmML Fits clustered binary and count data regression models.

phreg Fits parametric proportional hazards models.

risksets Finds the members of each riskset at all event times.

C.7.1 Checking the Integrity of Survival Data

In data sources collected through reading and coding old printed handwritten sources, many opportunities for making errors occur. Therefore, it is important to have tools for detecting logical errors like people dying before they got married, of living at two places at the same time. In the sources used in this book, such errors occur in almost all data retrievals. It is important to say that this fact is not a sign of a bad job in the transfer of data from old churchbooks to digital data files. The case is almost always that the errors are

present in the original source, and in the registration process no corrections of "obvious" mistakes by the priests are allowed; the digital files are supposed to be as close as possible to the original sources. These are the rules at the Demographic Database (DDB), Umeå University.

This means that the researcher has a responsibility to check her data for logical (and other) errors, and truthfully report how she handles these errors. In my own experience with data from the DDB, the relative frequency of errors varies between 1 and 5 per cent. In most cases it would not do much harm to simply delete erroneous records, but often the errors are "trivial," and it seems easy to guess what the truth should be.

Example 33 *Old Age Mortality*

As an example of the kind of errors discussed above, we take a look at the original version of `oldmort` in `eha`. For survival data in "interval" form (enter, exit], it is important that enter is smaller than exit. For individuals with more than one record, the intervals are not allowed to overlap, and at most one of them is allowed to end with an `event`. Note that we are talking about survival data, so there are no repeated events "by design."

The function `check.surv` in `eha` tries to do these kinds of checks and report back all individuals ("id") that do not follow the rules.

```
> load("olm.rda")
> require(eha)
> errs <- with(olm, check.surv(enter, exit, event, id))
> errs
 [1] "785000980" "787000493" "790000498" "791001135" "792001121"
 [6] "794001225" "794001364" "794001398" "796000646" "797001175"
[11] "797001217" "798001203" "800001130" "801000743" "801001113"
[16] "801001210" "801001212" "802001136" "802001155" "802001202"
[21] "803000747" "804000717" "804001272" "804001354" "804001355"
[26] "804001532" "805001442" "806001143" "807000736" "807001214"
[31] "808000663" "810000704" "811000705" "812001454" "812001863"
[36] "813000613" "814000799" "815000885" "816000894" "817000949"
[41] "819001046"
> no.err <- length(errs)
```

So, 41 individuals have bad records. Let us look at some of them.

```
> badrecs <- olm[olm$id %in% errs, ]
> dim(badrecs)
[1] 105  13
> badrecs[badrecs$id == errs[1], ]
           id  enter   exit event birthdate      m.id      f.id
283 785000980 74.427 75.917 FALSE  1785.573 743000229 747000387
284 785000980 75.525 87.917 FALSE  1785.573 743000229 747000387
285 785000980 87.917 88.306  TRUE  1785.573 743000229 747000387
       sex     civ ses.50 birthplace imr.birth region
```

```
    283 female married farmer     parish   12.4183  rural
    284 female   widow farmer     parish   12.4183  rural
    285 female   widow farmer     parish   12.4183  rural
```

The error here is that the second interval $(75.526, 87.927]$ starts before the first interval has ended. We also see that this is a married woman who became a widow in the second record.

In this case, I think that a simple correction saves the situation: Let the second interval begin where the first ends. The motivation for this choice is that the first interval apparently ends with the death of the woman's husband. Death dates are usually reliable, so I believe more in them than in other dates.

Furthermore, the exact choice here will have very little impact on the analyses we are interested in. The length of the overlap is less than half a year, and this person will eventually have the wrong civil status for at most that amount of time.

It turns out that all "bad" individuals are of this type. Therefore, we "clean" the data set by running it through this function:

```
ger <- function(){
    require(eha)
    load("olm.rda")
    out <- check.surv(olm$enter, olm$exit, olm$event, olm$id)
    ko <- olm[olm$id %in% out, ]
    ## ko is all records belonging to "bad individuals"
    ## Sort ko, by id, then by interval start, then by event
    ko <- ko[order(ko$id, ko$enter, -ko$event), ]
    for (i in ko$id){ # Go thru all 'bad individuals'
        slup <- ko[ko$id == i, , drop = FALSE]
        ## Pick out their records
        n <- NROW(slup)
        if (n > 1){ # Change them
            for (j in (2:n)){
                slup$enter[j] <- slup$exit[j-1]
            }
            ## And put them back:
            ko[ko$id == i, ] <- slup
        }
    }
    ## Finally, put 'all bad' back into place:
    om <- olm
    om[om$id %in% out, ] <- ko
    ## Keep only records that start before they end:
    om <- om[om$enter < om$exit, ]
    om
}
```

Read the comments, lines (or part of lines) starting with #. The output of this

function, `om`, is the present data file `oldmort` in `eha`, with the errors corrected as indicated above.

Running the function.

```
> source("ger.R")
> om <- ger()
> left <- with(om, check.surv(enter, exit, event, id))
> left
[1] "794001225"
```

Still one bad. Let us take a look at that individual.

```
> om[om$id == left, ]
            id  enter   exit event birthdate m.id f.id  sex
1168 794001225 65.162 68.650 FALSE  1794.838   NA   NA male
1169 794001225 68.650 85.162 FALSE  1794.838   NA   NA male
1171 794001225 78.652 85.162 FALSE  1794.838   NA   NA male
       civ ses.50 birthplace imr.birth region
1168 widow farmer     parish  12.70903  rural
1169 widow farmer     region  12.70903  rural
1171 widow farmer     parish  12.70903  rural
```

It seems as if this error (second an third records overlap) should have been fixed by our function `ger`. However, there may have been a record removed, so we have to look at the original data file:

```
> olm[olm$id == left, ]
            id  enter   exit event birthdate m.id f.id  sex
1168 794001225 65.162 68.650 FALSE  1794.838   NA   NA male
1169 794001225 65.162 85.162 FALSE  1794.838   NA   NA male
1170 794001225 68.650 78.652 FALSE  1794.838   NA   NA male
1171 794001225 78.652 85.162 FALSE  1794.838   NA   NA male
       civ ses.50 birthplace imr.birth region
1168 widow farmer     parish  12.70903  rural
1169 widow farmer     region  12.70903  rural
1170 widow farmer     parish  12.70903  rural
1171 widow farmer     parish  12.70903  rural
```

Yes, after correction, the third record was removed. The matter can now be fixed through removing the last record in `om[om$id == left, ]` :

```
> who <- which(om$id == left)
> who
[1] 1166 1167 1168
> om <- om[-who[length(who)], ]
> om[om$id == left, ]
            id  enter   exit event birthdate m.id f.id  sex
1168 794001225 65.162 68.650 FALSE  1794.838   NA   NA male
1169 794001225 68.650 85.162 FALSE  1794.838   NA   NA male
```

```
          civ ses.50 birthplace imr.birth region
1168 widow farmer     parish  12.70903  rural
1169 widow farmer     region  12.70903  rural
```

And we are done. □

C.8 Reading Data into R

The first thing you have to do in a statistical analysis (in **R**) is to get data into the computing environment. Data may come from different kind of sources, and **R** has the capacity to read from many kinds of sources, like data files from other statistical programs, spreadsheet type (e.g., *Excel*) data. This is done with the aid of the package `foreign`.

If anything else fails, it is almost always possible to get data written in ASCII format (e.g., *.csv* files). These can be read into **R** with the function `read.table`, see the next section.

C.8.1 Reading Data from ASCII Files

Ordinary text (ASCII) files in tabular form can be read into **R** with the function `read.table`. For instance, with a file looking this:

```
enter exit event
    1    5     1
    2    7     1
    1    5     1
    0    6     0
```

we read it into R with

```
> mydata <- read.table("my.dat", header = TRUE)
```

Note the argument `header`. It can take two values, `TRUE` or `FALSE`, with `FALSE` being the default value. If `header = TRUE`, then the first row of the data file is interpreted as *variable names*.

There are a few optional arguments to `read.table` that are important to consider. The first is `dec`, which gives the character used as a decimal point. The default is ".", but in some locales "," (e.g., Swedish) is commonly used in output from other software than **R**. The second is `sep`, which gives the field separator character. The default value is "", which means "white space" (tabs, spaces, newlines, etc.). See the help page for all the possibilities, including the functions `read.csv`, `read.csv2`, etc., which are variations of `read.table` with other default values.

The result should always be checked after reading. Do that by printing a few rows

```
> head(mydata)
  enter exit event
1     1    5     1
2     2    7     1
3     1    5     1
4     0    6     0
```

and/or by summarizing the columns

```
> summary(mydata)
     enter            exit          event
 Min.   :0.00   Min.   :5.00   Min.   :0.00
 1st Qu.:0.75   1st Qu.:5.00   1st Qu.:0.75
 Median :1.00   Median :5.50   Median :1.00
 Mean   :1.00   Mean   :5.75   Mean   :0.75
 3rd Qu.:1.25   3rd Qu.:6.25   3rd Qu.:1.00
 Max.   :2.00   Max.   :7.00   Max.   :1.00
```

The `str` function is also helpful:

```
> str(mydata)
'data.frame':        4 obs. of  3 variables:
 $ enter: int  1 2 1 0
 $ exit : int  5 7 5 6
 $ event: int  1 1 1 0
```

For each variable, its name and *type* is given, together with the first few values. In this example, it is all the values, and we can also see that **R** interprets all the values as `integer`. This is fine, even though at least *enter* and *exit* probably are supposed to be real-valued. In fact, **R** is not so strict with numerical types (as opposed to `C` and FORTRAN). Frequent *coercing* takes place, and the *mode* `numeric` contains the types `integer` and `double`.

Now this tends to be confusing, but the bottom line is that you, as a user, need not worry at all. Except in one place: When you are writing an **R** function that calls compiled `C` or FORTRAN code, but that is far beyond the scope of this book. Interested readers are recommended to read Venables & Ripley (2000).

C.8.2 Reading Foreign Data Files

Data file from MINITAB, `SAS`, `SPSS`, `Stata`, etc., can be read with a suitable function from the `foreign` package. Consult its help pages for more information.

The package `Hmisc` (Harrell Jr. 2011) contains some useful functions for reading data from foreign programs. They are usually wrappers for selected functions in the `foreign` package, but with convenient default behaviour.

This page intentionally left blank

D

Survival Packages in R

D.1 Introduction

The basic package for survival analysis in **R** is the `survival` package (Therneau & Grambsch 2000). It is one of the so-called *recommended packages* in **R**, which means that it is automatically installed when **R** itself is installed. You must, however, `load` it in a running **R** environment before you can use it.

There are a few other **R** packages devoted to survival and event history analysis. Besides `eha`, more or less the theme of this book, there are `timereg` and `cmprsk`. For a detailed explanation of how to use these packages, see their documentations in **R**.

D.2 eha

The package `eha` is written and maintained by the author of this book. It has a long history as a stand-alone program (written in FORTRAN, `Turbo Pascal`, and `C` during different time periods) in the "pre **R**" era. When I was aware of the existence of the **R** environment (in the mid-nineties), it was an easy decision to convert it into an **R** package.

Today, the function `coxreg` in `eha` rests to a large part on the function `coxph` in the `survival` package, but it has some features of its own, notably

Discrete time Cox regression With the option `method = 'ml'` a discrete-time Cox regression is performed with a discrete hazard atom at each observed event time. This is equivalent to a logistic regression with the `cloglog` link.

Sampling of risk sets The *weird bootstrap* (Andersen et al. 1993) is implemented in `coxreg`. It is activated by setting the argument `boot` equal to the desired number of bootstrap replicates.

Time-dependent case weights

Other features of the `eha` package are listed below.

Parametric proportional hazards models The common parametric models in other packages are of the AFT type. While there is a function in `eha` for these models (`aftreg`), the function `phreg` fits proportional hazards parametric models. Especially worth mentioning is the implementation of the *piecewise constant hazards* (`pch`) model.

Lexis diagram cuts With the aid of the two functions `age.window` and `cal.window` it is easy to make vertical and horizontal cuts in the Lexis diagram. The function `survSplit`

Tools for communal covariates The main tool is the function `make.communal`, that takes an external time series (think *weather*, *economy*, *epidemics*, etc.) and turns it into a time-dependent covariate.

For a presentation of the most important functions in `eha`, see Appendix C.

D.3 survival

The `survival` package is a recommended one, and it does not need a separate installation. It contains all the basic features that a package on survival analysis should have, and more. The main important functions are listed here:

coxph This is the main function for Cox regression. It has a number of features; time-dependent variables and strata, multiple events per subject, jackknife-type variance estimators for clustered data, and frailty models. Allows left-truncated data. Fast and reliable numerical algorithms.

survfit Takes care of the presentation and "afterwork" of a fit to a proportional hazards model or a accelerated failure time model, including plotting and printing.

survreg Fits parametric accelerated failure time models. Allows right and interval censoring, but not left truncation.

cox.zph For testing of the proportionality assumption of fit from a call to `coxph`.

aareg Fits Aalen's additive hazards model (Aalen 1989, Aalen 1993) to survival regression data.

D.4 Other Packages

D.4.1 coxme

The package `coxme` (Therneau 2011) analyzes frailty models in Cox regression. Its author, Terry Therneau, is the author of the `survival` package, which also can fit frailty models. According to the author, `coxme` is the preferred package for frailty models.

D.4.2 timereg

The `timereg` package is developed by Martinussen & Scheike (2006). A key feature of the package (and the cited book) is extensions of the Cox model, especially models with time-varying effects of covariates. Aalen's additive hazards model is in focus. Resampling is frequently used for the calculation of p-values.

Recently, the package has been promoted for being able to analyze competing risks models (Scheike & Zhang 2011).

D.4.3 cmprsk

This is a competing risks package (Gray 2011), based on work by Gray (1988) and Fine & Gray (1999).

This page intentionally left blank

Bibliography

Aalen, O. (1978). Nonparametric inference for a family of counting processes, *Annals of Statistics* **6**: 701–726.

Aalen, O. (1989). A linear regression model for the analysis of life times, *Statistics in Medicine* **8**: 907–925.

Aalen, O. (1993). Further results on the non-parametric linear model in survival analysis, *Statistics in Medicine* **12**: 1569–1588.

Aalen, O., Borgan, Ø. & Gjessing, H. (2008). *Survival and Event History Analysis: A Process Point of View*, Springer, New York.

Allison, P. (1984). *Event History Analysis*, Sage University Paper series on Quantitative Applications in the Social Sciences, No. 46, Beverly Hills, CA.

Allison, P. (1995). *Survival Analysis Using the SAS System: A Practical Guide*, BBU Press, SAS Institute, Cary, NC.

Andersen, P., Borgan, Ø., Gill, R. & Keiding, N. (1993). *Statistical Models Based on Counting Processes*, Springer-Verlag, Berlin.

Breslow, N. (1974). Covariance analysis of censored survival data, *Biometrika* **30**: 89–99.

Broström, G. (1987). The influence of mother's mortality on infant mortality: A case study in matched data survival analysis, *Scandinavian Journal of Statistics* **14**: 113–123.

Broström, G. (2002). Cox regression: Ties without tears, *Communications in Statistics: Theory and Methods* **31**: 285–297.

Broström, G. (2012). *eha: Event History Analysis*. R package version 2.0-7.

Broström, G. & Lindkvist, M. (2008). Partial partial likelihood, *Communications in Statistics: Simulation and Computation* **37**: 679–686.

Collett, D. (2003). *Modelling Survival Data in Medical Research*, Second edn, Chapman & Hall/CRC.

Cox, D. (1972). Regression models and life tables, *Journal of the Royal Statistical Society Series B (with discussion)* **34**: 187–220.

Cox, D. (1975). Partial likelihood, *Biometrika* **62**: 269–276.

Cox, D. & Oakes, D. (1984). *Analysis of Survival Data*, Chapman & Hall, London.

Dalgaard, P. (2008). *Introductory Statistics with R*, Second edn, Springer, Berlin.

Efron, B. (1977). Efficiency of Cox's likelihood function for censored data, *Journal of the American Statistical Association* **72**: 557–565.

Elandt-Johnson, R. & Johnson, N. (1999). *Survival Models and Data Analysis*, Wiley Classics Library edn, John Wiley & Sons, New York.

Fine, J. & Gray, R. (1999). A proportional hazards model for the subdistribution of a competing risk, *Journal of the American Statistical Association* **94**: 496–509.

Gompertz, B. (1825). On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies, *Philosophical Transactions of the Royal Society of London* **115**: 513–585.

Granger, C. (1969). Investigating causal relations by econometric models and cross-spectral methods, *Econometrica* **37**: 424–438.

Gray, B. (2011). *cmprsk: Subdistribution Analysis of Competing Risks*. R package version 2.2-2.
URL: *http://CRAN.R-project.org/package=cmprsk*

Gray, R. (1988). A class of K-sample tests for comparing the cumulative incidence of a competing risk, *Journal of the American Statistical Association* **94**: 496–509.

Groeneboom, P. & Wellner, J. (1992). *Nonparametric Maximum Likelihood Estimatorsfor Interval Censoring and Deconvolution*, Birkhäuser, Boston.

Harrell Jr., F. (2011). *Hmisc: Harrell Miscellaneous*. R package version 3.9-0.
URL: *http://CRAN.R-project.org/package=Hmisc*

Hauck, W. & Donner, A. (1977). Wald's test as applied to hypotheses in logit analysis, *Journal of the American Statistical Association* **72**: 851–853.

Hernán, M., Brumback, B. & Robins, J. (2002). Estimating the causal effect of zidovudine on cd4 count with a marginal structural modelfor repeated measures, *Statistics in Medicine* **21**: 1689–1709.

Hernán, M., Cole, S., Margolick, J., Cohen, M. & Robins, J. (2005). Structural accelerated failure time models for survival analysis in studies with time-varying treatments, *Pharmacoepidemiology and Drug Safety* **14**: 477–491.

Hernán, M. & Robins, J. (2012). *Causal Inference*, Chapman & Hall/CRC, London.

Hougaard, P. (2000). *Analysis of Multivariate Survival Data*, Springer, Berlin.

Johansen, S. (1983). An extension of Cox's regression model, *International Statistical Review* **51**: 165–174.

Kalbfleisch, J. & Prentice, R. (1980). *The Statistical Analysis of Failure Time Data*, Wiley, Hoboken, N.J.

Kalbfleisch, J. & Prentice, R. (2002). *The Statistical Analysis of Failure Time Data*, Second edn, Wiley, Hoboken, N.J.

Klein, J. & Moeschberger, M. (2003). *Survival Analysis. Techniques for Censored and Truncated Data*, Springer-Verlag, New York.

Langholz, B. & Borgan, Ø. (1995). Counter-matching: A stratified nested case-control sampling method, *Biometrika* **82**: 69–79.

Lauritzen, S. (1996). *Graphical Models*, Oxford Statistical Science Series No. 17, Oxford University Press, Oxford, UK.

Lawless, J. (2003). *Statistical Models and Methods for Lifetime Data*, Second edn, John Wiley & Sons, Hoboken, N.J.

Makeham, W. (1860). On the law of mortality and the construction of annuity tables, *Journal of the Institute of Actuaries and Assurance Magazine* **8**: 301–310.

Martinussen, T. . & Scheike, T. (2006). *Dynamic Regression Models for Survival Data*, Springer-Verlag, New York.

Nelson, W. (1972). Theory and applications of hazard plotting for censored failure data, *Technometrics* **14**: 945–965.

Parmar, M. & Machin, D. (1995). *Survival Analysis: A Practical Approach*, John Wiley & Sons, Chichester.

Pearl, J. (2000). *Causality: Models, Reasoning and Inference*, Cambridge University Press, New York.

Pinheiro, J. & Bates, D. (2000). *Mixed-Effects Models in S and S-Plus*, Springer-Verlag, New York.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL: *http://www.R-project.org*

Robins, J. (1986). A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect, *Mathematical Modeling* **7**: 1393–1512.

Rubin, D. (1974). Estimating causal effects of treatments in randomized and non-randomized studies, *Journal of Educational Psychology* **66**: 688–701.

Scheike, T. & Zhang, M.-J. (2011). Analyzing competing risk data using the R timereg package, *Journal of Statistical Software* **38**(2): 1–15.
URL: *http://www.jstatsoft.org/v38/i02/*

Schweder, T. (1970). Composable Markov processes, *Journal of Applied Probability* **7**: 400–410.

Silverman, B. (1986). *Density Estimation*, Chapman & Hall, London.

Therneau, T. (2011). *coxme: Mixed Effects Cox Models.* R package version 2.2-1.
URL: *http://CRAN.R-project.org/package=coxme*

Therneau, T. & Grambsch, P. (2000). *Modeling Survival Data: Extending the Cox Model*, Springer-Verlag, New York.

Therneau, T. & original Splus to R port by T. Lumley (2011). *survival: Survival Analysis, Including Penalised Likelihood.* R package version 2.36-10.
URL: *http://CRAN.R-project.org/package=survival*

Vaupel, J., Manton, K. & Stallard, E. (1979). The impact of heterogeneity in individual frailty on the dynamics of mortality, *Demography* **16**: 439–454.

Venables, W. & Ripley, B. (2000). *S Programming*, Springer-Verlag, New York.

Weibull, W. (1951). A statistical distribution function of wide applicability, *Journal of Applied Mechanics, Transactions ASME* **18**: 293–297.

Wright, S. (1921). Correlation and causation, *Journal of Agricultural Research* **20**: 557–585.

Zeilinger, A. (2005). The message of the quantum, *Nature* **438**: 743.

With an emphasis on social science applications, **Event History Analysis with R** presents an introduction to survival and event history analysis using real-life examples. Keeping mathematical details to a minimum, the book covers key topics, including both discrete and continuous time data, parametric proportional hazards, and accelerated failure times.

Features

- Introduces parametric proportional hazards models with baseline distributions like the Weibull, Gompertz, Lognormal, and Piecewise constant hazard distributions, in addition to traditional Cox regression
- Presents mathematical details as well as technical material in an appendix
- Includes real examples with applications in demography, econometrics, and epidemiology
- Provides a dedicated R package, eha, containing special treatments, including making cuts in the Lexis diagram, creating communal covariates, and creating period statistics

A much-needed primer, **Event History Analysis with R** is a didactically excellent resource for students and practitioners of applied event history and survival analysis.


CRC Press
Taylor & Francis Group
an **informa** business
www.crcpress.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
711 Third Avenue
New York, NY 10017
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

K11534