**The Analysis of Stochastic Models using SPlus**

by

Paul Willard Ramsey
B.Sc., University of British Columbia, 1993

A Thesis Submitted in Partial Fulfullment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Mathematics and Statistics

We accept this thesis as conforming
to the required standard

[redacted]

*Dr. M. Lesperance, Supervisor (Department of Mathematics & Statistics)*

[redacted]

*Dr. B. Johnson, Departmental Member (Department of Mathematics & Statistics)*

[redacted]

*Dr. J. Ye, Departmental Member (Department of Mathematics & Statistics)*

[redacted]

*Dr. J. A. Giles, External Examiner (Department of Economics)*

© Paul Willard Ramsey, 1996
University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.

QA 274
R34

Supervisor: Dr. Mary Lesperance

## ABSTRACT

*The Analysis of Stochastic Processes using Splus* is a revised companion to the text *The Analysis of Stochastic Processes using GLIM*, by James K. Lindsey, making use of the SPlus statistics package in place of GLIM. The revised edition covers all the material in the first six chapters of Lindsey, including Markov processes, renewal processes, survival data, growth curves, and time series in the time domain. Additionally, analysis and model information have been elaborated in the revised edition.

Examiners:

[redacted]

*Dr. M. Lesperance, Supervisor (Department of Mathematics & Statistics)*

[redacted]

*Dr. B. Johnson, Departmental Member (Department of Mathematics & Statistics)*

[redacted]

*Dr. J. Ye, Departmental Member (Department of Mathematics & Statistics)*

[redacted]

*Dr. J. A. Giles, External Examiner (Department of Economics)*

# Contents


**1 Introduction** 2

**2 Normal Theory Models and Extensions** 4

2.1 Linear Regression . . . . . 4

2.2 Analysis of Variance . . . . . 12

2.3 Analysis of Covariance . . . . . 14

2.4 The Extension to Non-Normal Models . . . . . 16

2.5 Fitting Distributions . . . . . 19

2.6 Further SPlus Instructions . . . . . 23

2.6.1 Conventions . . . . . 23

2.6.2 Data Manipulation . . . . . 24

2.6.3 Data Entry . . . . . 24

2.6.4 Custom Functions . . . . . 24

**3 Markov Chains** 26

3.1 Binary Point Processes . . . . . 26

3.2 Multi-state Markov Chains . . . . . 33

3.3 Stationarity . . . . . 36

3.4 Reversibility and Equilibrium . . . . . 40

3.5 Random Walks . . . . . 44

3.6 The Mover-Stayer Model . . . . . 48

**4 Point and Renewal Processes** **51**

4.1 Point Processes . . . . . 51

4.2 The Poisson Process . . . . . 54

4.3 Kaplan-Meier Estimation . . . . . 57

4.4 Probability Plots . . . . . 59

4.5 Fitting a Distribution . . . . . 63

4.6 A Nonhomogeneous Point Process . . . . . 66

4.7 An Example with Periodicity . . . . . 74

**5 Survival Curves** **82**

5.1 Censored Data . . . . . 82

5.2 The Hazard Function . . . . . 83

5.3 Exponential Distribution . . . . . 84

5.4 Pareto Distribution . . . . . 87

5.5 Weibull Distribution . . . . . 89

5.6 Extreme Value . . . . . 91

5.7 Log Normal Distribution . . . . . 93

5.8 Log Logistic Distribution . . . . . 94

5.9 Gamma Distribution . . . . . 96

5.10 Inverse Gaussian Distribution . . . . . 98

5.11 Cox Proportional Hazards Model . . . . . 99

5.12 Piecewise Exponential Distribution . . . . . 101

**6 Growth Curves** **106**

6.1 Exponential Growth: Continuous Data . . . . . 106

6.2 Exponential Growth: Count Data . . . . . 111

6.3 The Logistic Growth Curve . . . . . 115

6.4 The Gomperz Growth Curve . . . . . 117

6.5 Supplement: Fitting with nls() . . . . . 119

6.5.1 Logistic . . . . . 120
6.5.2 Gomperz . . . . . 121

**7 Time Series: The Time Domain** **123**
7.1 Trends and Correlograms . . . . . 123
7.2 Autoregression and Random Walks . . . . . 128
7.3 Examination of the Distributional Assumptions . . . . . 131
7.4 Mis-specification of the Linear Model . . . . . 138
7.5 Serial Correlation in Regression Analysis . . . . . 140

**8 Conclusion** **146**

**9 Custom Functions** **149**
9.1 glmtab() . . . . . 149
9.2 tsfr() . . . . . 150
9.3 mpct() . . . . . 151
9.4 plotsurv() . . . . . 152
9.5 f.test() . . . . . 155

**10 Data Tables** **156**
10.1 Liège Divorce Data . . . . . 156
10.2 Hospital Admissions Data . . . . . 170
10.3 Recidivism Data . . . . . 172

# List of Tables


2.1 Explanatory variables and parameters used in distribution-fitting with log-linear models. . . . . 20

3.1 Number of compensable accidents per day for one shift in a section of a mine. . . . . 27

3.2 Number of patients arriving at an intensive care unit per day. . . . . 34

3.3 One step transitions for voting intentions in the 1940 US Presidential Elections . . . 37

3.4 Voting changes between Belgian elections, 1981-1985. . . . . 41

3.5 Changes in vote between the two British elections of 1974. . . . . 45

3.6 Migrant behaviour in Britain between 1966 and 1971. . . . . 48

4.1 Times in seconds between vehicles passing a point on the road. . . . . 52

4.2 Table of survivor functions and linear relationships between $S(t)$ and time for common probability distributions. . . . . 61

4.3 Days between successive coal-mining disasters (at least 10 deaths) in Great Britain . 66

6.1 Gross domestic fixed capital formation in the United Kingdom. . . . . 107

6.2 AIDS cases reported in the UK . . . . . 111

7.1 Student enrolment at Yale University . . . . . 124

7.2 Annual percentage increase in average salaries of white collar workers in Britain. . . 133

# List of Figures


2.1 Normal distributions of points about a regression line. . . . . 5

2.2 Pairs plot of data on divorce. . . . . 7

2.3 Plot of fitted and observed lengths of marriage versus husband's age. . . . . 11

2.4 Plot of fitted values for length of marriage versus husband's age for generalized linear model with gamma family. . . . . 18

2.5 Plot of estimates of $p_k$ versus year for distribution of length of marriage. . . . . 21

4.1 Cumulative sums plot of traffic data. . . . . 52

4.2 Bar chart of numbers of cars passing a point in successive 60 second intervals. . . . . 53

4.3 Plot of observed/fitted values for truncated normal model of traffic data. . . . . 57

4.4 Empirical survivor and hazard plots for traffic data. . . . . 60

4.5 Plot of observed and fitted versus time for log-normal model of traffic data. . . . . 65

4.6 Bar chart of the number of British coal disasters in successive 400 day intervals. . . 67

4.7 Cumulated number of coal mine disasters over time. . . . . 68

4.8 Empirical survivor curves of coal mine disaster data, split into two time periods. . . 71

4.9 Empirical hazard functions for coal mine disaster data. Regular and magnified plots. 72

4.10 Probability plot of empirical survival estimates against exponential distribution. . . 73

4.11 Residual plot for constant rate model of monthly hospital arrival counts. . . . . 78

4.12 Histogram of time intervals between hospital arrivals. . . . . 79

4.13 Empirical survivor curve for inter-arrival times at a hospital. . . . . 81

5.1 Empirical hazard curves for recidivism rates of inmates over two studies. . . . . 85

5.2 Estimated hazard function for recidivism data assuming an exponential survival distribution. . . . . 87

5.3 Estimated hazard function for recidivism data assuming a Pareto survival distribution. 89

5.4 Estimated hazard function for recidivism data assuming a Weibull survival distribution. 91

5.5 Estimated hazard function for recidivism data assuming an extreme value survival distribution. . . . . 92

5.6 Estimated hazard function for recidivism data assuming a log normal survival distribution. . . . . 94

5.7 Estimated hazard function for recidivism data assuming a log logistic survival distribution. . . . . 96

5.8 Hazard function for recidivism data calculated using Cox proportional hazards. . . . 101

5.9 Hazard function for recidivism data calculated using piecewise exponential model. . 104

5.10 Hazard function for recidivism data calculated using piecewise exponential model with breaks for all intervals with uncensored deaths. . . . . 105

6.1 Plot of observed and fitted values in multiplicative errors fit of GDFC growth curve. 109

6.2 Plot of observed and fitted values in additive errors fit of GDFC growth curve. . . . 110

6.3 Plot of observed and fitted values for quadratic model of growth of AIDS incidence. 113

6.4 Plot of observed and fitted values for quadratic model (4 obs removed) of growth of AIDS incidence. . . . . 114

6.5 Deviance curve for binomial models of logistic growth with varying values of $N$. . . . 116

6.6 Plot of observed and fitted values for logistic model of growth of AIDS incidence. . . 117

6.7 Deviance curve for binomial models of Gomperz growth with varying values of $N$. . 118

6.8 Plot of observed and fitted values for Gomperz model of growth of AIDS incidence. . 119

6.9 Plot of observed and fitted values for logistic model of growth of AIDS incidence. . . 120

6.10 Plot of observed and fitted values for Gomperz model of growth of AIDS incidence. . 122

7.1 Time series plot of Yale enrolment data. . . . . 124

7.2 Autocorrelation plot of Yale enrolment data. . . . . 125

7.3 Residual autocorrelation of quadratic model of Yale enrolment data. . . . . 126

7.4 Residual autocorrelation of split model of Yale enrolment data. . . . . 127

7.5 Diagnostic plot of ARIMA(1,1,0) model of Yale enrolment data. . . . . 129

7.6 Diagnostic plot of ARIMA(1,0,0) model of Yale enrolment data with a quadratic deterministic component. . . . . 132

7.7 Pairs plot of white collar wage data. . . . . 134

7.8 Normal quantile plot of residuals of a multiple linear regression on white collar wage data. . . . . 136

7.9 Score test plot of residuals of a multiple linear regression on white collar wage data. 137

7.10 Plot of fitted values versus observed for multiple linear regression on white collar wage data. . . . . 137

7.11 Autocorrelation plot of residuals for multiple linear regression on white collar wage data. . . . . 138

7.12 CUSUM plot of predicted residuals based on model fitted to first 12 points of white collar wage data. . . . . 141

7.13 Autocorrelation plot of the residuals of a three parameter model of white collar wage data. . . . . 143

# Chapter 1

# Introduction

We live in a world filled with stochastic processes. That is, we live in time, so almost everything we observe can potentially be analysed as a stochastic process. It is not surprising, therefore, that the study of stochastic (serially collected) data in statistics encompasses a huge range of statistical techniques: generalized linear models, survival processes, time series, process control, non-linear models, and more. Nor is it surprising that stochastic models are of importance to fields of study as diverse as physics, econometrics, biology or engineering.

This book can be considered a revised companion to "The Analysis of Stochastic Processes using GLIM" by James K. Lindsey [14]. It originally began as a project to convert Lindsey's GLIM macros into SPlus equivalents. The GLIM macros presented by Lindsey in his book are ingenious applications of general linear models to complex stochastic processes. However, the GLIM analysis package is not widely used in North America, while the SPlus program is.

SPlus is a substantially more developed statistical package than GLIM, incorporating standard model fitting routines for many more models than GLIM. In fact, for almost all of the models for which Lindsey had to code custom macros, SPlus already has built-in routines. In the end, the real task of the project was not converting macros, but converting analysis techniques so that they made sense within the context of existing SPlus resources.

This book takes a broad look at stochastic processes, common mathematical models used to describe them, and the statistical techniques needed to make valid inferences about the models. The models to be examined include Markov processes, renewal processes, survival distributions, growth curves, and time series (in the time domain).

The techniques presented in this book would be of interest to any researcher who wishes to use stochastic models to make statistical inferences. A grasp of probability distributions, stochastic processes [20], linear regression [22], time series [4], and generalized linear models [7, 15] is necessary for a thorough understanding of the methods in this book.

The SPlus statistics package is an object oriented data analysis engine. It combines the flexibility of an object oriented programming environment with the interactivity of an interpreted command-

line. This means that you can use pre-made SPlus functions to do data analysis and graphics on the fly, or you can create your own functions, by combining built-in routines and your own calculations. Plotting and graphical data analysis are available in SPlus through a versatile graphics window, which can also produce PostScript files for output to hardcopy. SPlus was originally designed for UNIX workstations, but is now available for most personal computer platforms as well.

Although the project originally focused on simply converting GLIM macros to SPlus, the highly developed nature of the SPlus environment made many of the original GLIM macros irrelevant. Algorithms which had to be painstakingly constructed by Lindsey using the limited GLIM language already exist in the current SPlus distribution. While some custom functions still remain, the focus of this revision is on analysing stochastic models within the SPlus environment, largely using native SPlus commands.

Topics are introduced in the following order:

- **Chapter 2**
  Focuses on the SPlus environment, with examples of simple linear models, generalized linear models, and fitting distributions.
- **Chapter 3**
  Introduces Markov processes, and applications of general linear models to modelling them.
- **Chapter 4**
  Examines point and renewal processes, with emphasis on the Poisson process. Also examined are non-homogeneous processes and probability distribution fitting.
- **Chapter 5**
  Introduces several important survival distributions, with methods for fitting them parametrically and semi-parametrically.
- **Chapter 6**
  Examines growth curves, and methods of fitting them using iterative routines and non-linear fitting.
- **Chapter 7**
  Examines time series analysis in the time domain.

Revised versions of the final three chapters of the Lindsey book may be added in the future.

# Chapter 2

# Normal Theory Models and Extensions

## 2.1 Linear Regression

One of the most widely used tools in all of statistics is *linear regression.* Linear regression is often mistakenly referred to as "least squares regression", but "least squares" refers to a deterministic process of estimation in which the best straight line is fitted through a series of points. In linear regression the interpretation is much different, although the technical calculations remain the same. Normal theory linear regression carries the assumption that the response variable has a normal or Gaussian distribution,

$$f(y;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2} \tag{2.1}$$

and that the mean of this distribution changes in some deterministic way with the values of the explanatory variables, for example,

$$E(y_i) = \mu_i = \beta_0 + \sum_j \beta_j x_{ij} \qquad i = 1,\ldots,n \tag{2.2}$$

while the variance remains constant. In other words, the regression equation specifies how the *mean* of the distribution changes for each value of the explanatory variables—individual observations are dispersed about the mean with the given variance. This principle is illustrated in Figure 2.1.

Once we see that such a model is describing changes in a normal distribution we can easily imagine extending the model to other distributions. When we extend the regression model using a distribution from the exponential family we have a *generalized linear model.* SPlus plays a key role in the analysis of generalized linear models. Both normal linear regression models and generalized linear models may be easily fitted with SPlus.

SPlus is an object oriented statistical analysis package which incorporates numerous different analysis tools within a flexible programming environment for easy customization. SPlus is primarily

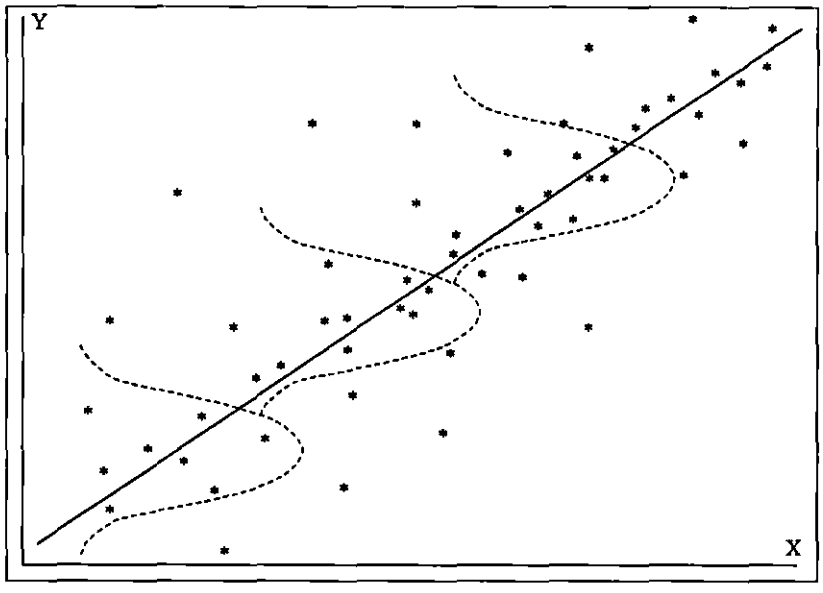



Figure 2.1: Normal distributions of points about a regression line.

command line driven, and can accept commands either interactively or within pre-written scripts. In the X Windows environment, SPlus can display graphical plots, and can produce PostScript plot files for printing. PostScript files can also be generated in text mode.

Before going further, we consider an example based on a relatively large set of 1727 observations of all people divorcing in the city of Liège. The data are given in Section 10.1. We are interested in exploring how the distribution of lengths of marriages varies with certain explanatory variables.

To start the analysis, data must first be read into SPlus. Data can be entered interactively or from an ASCII file. Data read into SPlus from ASCII files may be delimited in almost any fashion, but is usually arranged in columns and separated with tabs or white space. Missing values should be coded "NA". Because the Liège data set is large, we read the data directly from a file, using the scan() function as follows:

```
> married.data <- scan(file = "data/Chpt1-1", what = list(A = 0, HA = 0, WA = 0, LM = 0, LP =
0, C = 0), skip = 1)
```

The file parameter of the scan function specifies the ASCII file from which the data is read, the what parameter specifies the structure of the file (in this case, six columns of numerical data), and the skip parameter directs SPlus to skip the first line of the ASCII file, as it contains column headers and not data. The what parameter specifies both data type and data name. In the example above the name for the first column is set to be A and the data type is set to be numeric with 0; if A were a string the line would read A = "". The assignment operator <- places the results of the scan() function into the data list married.data. For ease of use, the assignment operator <- can be abbreviated to _. The > character at the beginning of the example above is not a part of the

command invocation; it is the default SPlus prompt symbol. Your SPlus system may use > as a prompt, or it may have a different symbol defined as the prompt.

The scan() function returns the data it reads in an object of data type list. However, when using SPlus regression tools it is much easier to handle the data if it is contained in a "data frame".

```
> married.data <- data.frame(married.data)
```

The new data frame married.data includes the six variables we are going to examine:

- A = origin of divorce application (coded as husband = 1, wife = 2, and mutual = 3)
- HA = husband's age at time of divorce
- WA = wife's age at time of divorce
- LM = length of marriage
- LP = length of divorce procedure
- C = number of children

Before we begin using quantitative analysis tools, we first plot the data and examine qualitative relationships between the variables. Within the X Windows environment, SPlus can create detailed graphic plots which are useful for preliminary data viewing and model assessment. To begin, activate a plot window using one of the following commands:

```
> X11()
> motif()
> openlook()
```

We display our data set using the pairs() function, which plots each variable individually against every other variable in the data frame.

```
> pairs(married.data)
```

We see several obvious trends in Figure 2.2. First, HA (husband's age) and WA (wife's age) show a strong linear relationship; this isn't surprising, since people tend to marry people near their own age. Second, the relationships between LM (length of marriage) and HA and between LM and WA are both triangular; this is also not surprising, since a 20 year old could only have a marriage of at most 20 years, likely much less. Additionally, the relationships between LM and HA and between LM and WA display a moderate linear trend, although one end of the plot is wider than the other, reflecting the fact that older people can have short marriages as well as long ones.

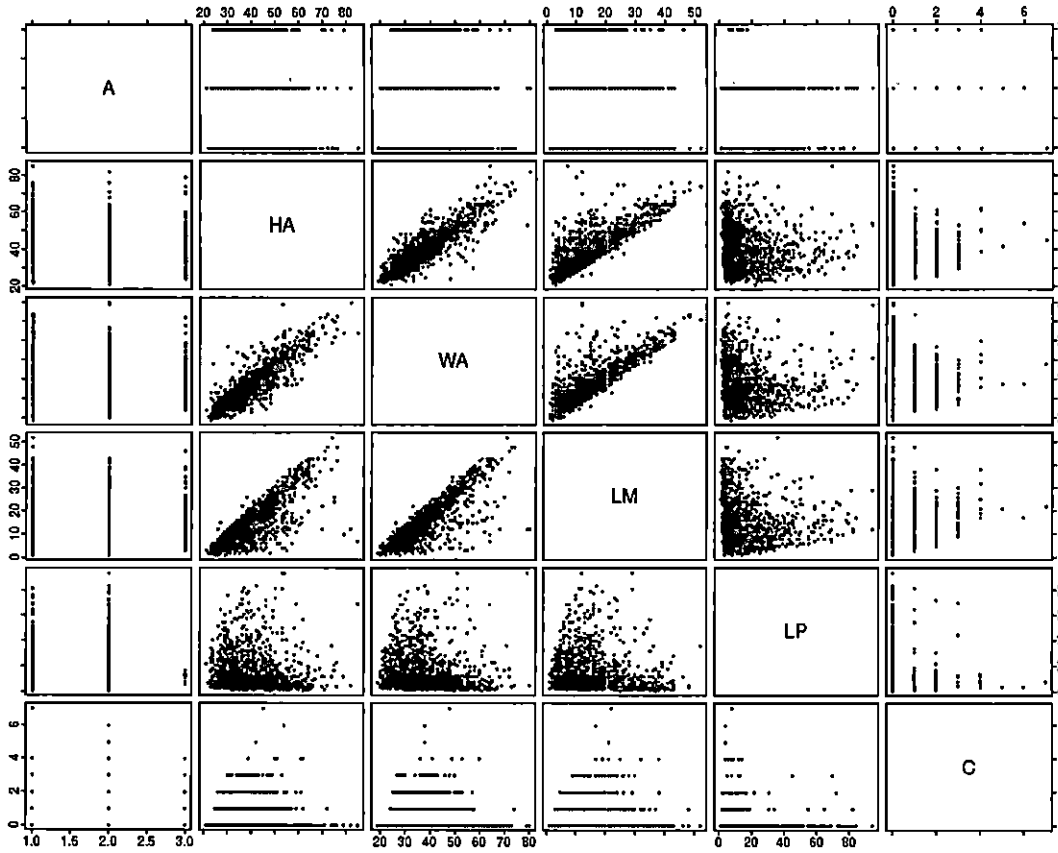



Figure 2.2: Pairs plot of data on divorce.

To fit a normal linear regression model, we make use of the lm() function. The lm() function creates a *model object* which contains all the relevant information about the fitted model, such as residuals, deviances, error terms, coefficients, and so on.

The lm() function has the following syntax:

```
> model <- lm(response ~ explanatory1 + ... + explanatoryK, data=<data object>,
weights=<weight vector>, na.action=<na.omit/na.fail>)
```

The first argument of the function describes the relationship between the response and the explanatory variables. If interactions between explanatory variables are to be modelled, they are entered into the formula as explanatory1*explanatory2. The weights parameter of the function allows the user to include a vector of weights to use in fitting the model. The data parameter of the lm() function tells SPlus from which data frame to read the data for the model; if the data parameter is absent, SPlus searches its directory path for relevant data objects. The na.action parameter tells SPlus how to deal with missing values.

The Liège data contain several missing data points, so it is possible to omit them from the model using na.omit. If not explicitly provided, the na.action parameter defaults to na.fail, which will cause lm() to abort when it encounters missing data points. This allows you to choose how to deal with missing data before continuing with your analysis.

Rather than deal with missing values in the Liège data during the model fitting process, we remove them all from the data frame prior to model fitting. Removing missing points ahead of time will allow us to compare fitted models which all have the same number of observations.

```
> married.data <- na.omit(married.data)
```

To start analysis of the Liège data, we first fit a model for LM (length of marriage) with one parameter, $\beta_0$, for the overall mean and with no explanatory variables.

$$\mu_i = \beta_0 \tag{2.3}$$

```
> married.lm1 <- lm(LM ~ 1, data = married.data)
```

When the lm() command is entered, there is no output to the screen, but the results of the model fitting process are stored in the object married.lm1. Information about the fitted model is displayed using the summary() function.

```
> summary(married.lm1)
```

```
Call: lm(formula = LM ~ 1, data = married.data)
Residuals:
    Min      1Q Median      3Q     Max
```

```
 -12.78 -6.782 -1.782 4.218 38.22

Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept) 13.7825  0.2113    65.2350  0.0000

Residual standard error: 8.678 on 1686 degrees of freedom
Multiple R-Squared: 2.612e-29
F-statistic: Inf on 0 and 1686 degrees of freedom, the p-value is NA
```

Of the 1727 original observations, 40 were previously eliminated as missing values, resulting in 1686 residual degrees of freedom. The maximum likelihood estimate of the mean is 13.78 years of marriage with a standard error of 0.2113. The residual standard error is 8.678 on 1686 degrees of freedom. It is computed as the square root of the residual deviance (residual sum of squares) divided by the degrees of freedom.

Next we fit the simple linear regression of LM (length of marriage) on HA (the age of the husband). The model is

$$\mu_i = \beta_0 + \beta_1 x_{i1} \tag{2.4}$$

where $\beta_0$ is the intercept and $\beta_1$ is the slope of the regression line. We expect a relatively strong relationship, as our initial plot of LM versus HA showed a visible linear trend. We fit the model as follows:

```
> married.lm2 <- lm(LM ~ HA, data=married.data)
> summary(married.lm2)


Call: lm(formula = LM ~ HA, data = married.data)
Residuals:
    Min      1Q Median     3Q    Max
 -40.99 -2.013 0.4552 2.796 12.52

Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept) -14.4155   0.4695   -30.7023   0.0000
         HA   0.7341   0.0118    62.0009   0.0000

Residual standard error: 4.792 on 1685 degrees of freedom
Multiple R-Squared: 0.6952
F-statistic: 3844 on 1 and 1685 degrees of freedom, the p-value is 0

Correlation of Coefficients:
   (Intercept)
HA -0.9686
```

The intercept estimate is $-14.42$ and the slope estimate is 0.7341. As expected, LM (length of marriage) increases with HA (age of the husband). We are interested in knowing if this slope is significantly different from zero. One quick way is to compare the absolute value of a parameter estimate to its standard error. If the ratio is at least two, the estimate is significantly different from zero at approximately the 5% level. Here, the ratio is much larger (62), so we may clearly reject the hypothesis that the slope is zero. Additionally, SPlus provides you with a more accurate method of evaluating significance of coefficients via the Student's t values and associated p-values. The t-test also implies that the slope is significantly different from zero.

We now complicate our model somewhat by adding a quadratic term in age of the husband.

$$\mu_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 \tag{2.5}$$

To fit this model in SPlus, we first calculate the squares of the husband's age. To isolate the variable HA from the rest of the data in the married.data data frame, and to create a new variable in the data frame, we use the $ modifier as follows:

```
> married.data$HA.squared <- married.data$HA ^ 2
```

Note that from now on we will be using _ for the assignment operator in place of <-.

```
> married.lm3_lm(LM ~ HA + HA.squared, data=married.data)
> summary(married.lm3)
```

```
Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept) -20.3658   1.5553   -13.0945   0.0000
         HA   1.0265   0.0738    13.9014   0.0000
 HA.squared  -0.0034   0.0008    -4.0113   0.0001

Residual standard error: 4.771 on 1684 degrees of freedom
Multiple R-Squared: 0.6981
F-statistic: 1947 on 2 and 1684 degrees of freedom, the p-value is 0
```

For clarity, part of the output from the summary() function has been omitted. The slope parameter for the linear effect of HA (husband's age) remains very significant, while that for the quadratic effect is also significant, as indicated by the t-value and p-value.

It is possible to plot our model with SPlus. If you are working in the X Windows environment, activate a plot window with

```
> X11() or motif() or openlook()
```

and view diagnostic plots of the model using

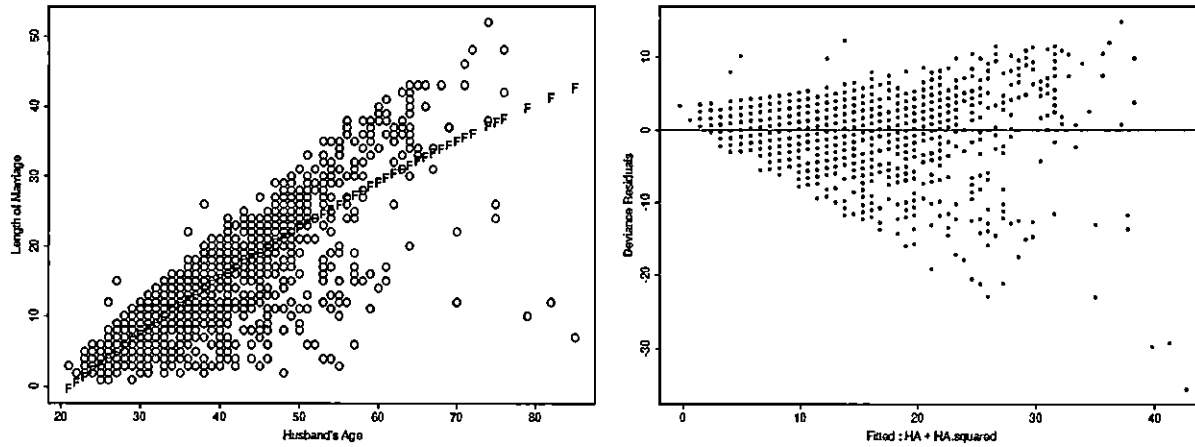



Figure 2.3: Plot of fitted and observed lengths of marriage versus husband's age.

```
> plot(married.lm3,ask=T)
```

to start an interactive plotting routine. You will be given a choice of seven different diagnostic plots to view:

```
Make a plot selection (or 0 to exit):

1: plot: All
2: plot: Residuals vs Fitted Values
3: plot: Sqrt of abs(Residuals) vs Fitted Values
4: plot: Response vs Fitted Values
5: plot: Normal QQplot of Residuals
6: plot: r-f spread plot
7: plot: Cook's Distances
Selection:
```

We can also build our own plot from scratch, as follows:

```
> matplot(married.data$HA, cbind(married.lm3$fitted, married.data$LM), pch = "FO", xlab =
"Husband's Age", ylab = "Length of Marriage")
```

While the plot of the fitted values (Figure 2.3) shows that the fitted model appears to be a reasonable approximation of the data, the plot of the residuals (drawn using option 2 of the plot menu above) shows that the assumption of constant variance is untenable. The size of the variance appears to be indexed to the size of the mean.

The same series of analyses could be carried out for the age of the wife. These give very similar results. Finally, we can combine the two models to obtain a four variable multiple linear regression which is quadratic in both ages;

$$\mu_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2} + \beta_4 x_{i2}^2 \tag{2.6}$$

or

$$E(\mathbf{LM}) = \beta_0 + \beta_1 \mathbf{HA} + \beta_2 \mathbf{HA}^2 + \beta_3 \mathbf{WA} + \beta_4 \mathbf{WA}^2$$

The SPlus instructions are

```
> married.data$WA.squared_married.data$WA ^ 2
> married.lm4_lm(LM ~ HA + HA.squared + WA + WA.squared, data = married.data)
> summary(married.lm4)
```

This gives

```
Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept) -21.9657   1.4617   -15.0280   0.0000
         HA   0.6050   0.1042     5.8074   0.0000
 HA.squared  -0.0031   0.0012    -2.6240   0.0088
         WA   0.5201   0.1018     5.1087   0.0000
 WA.squared  -0.0010   0.0012    -0.8146   0.4154

Residual standard error: 4.364 on 1682 degrees of freedom
Multiple R-Squared: 0.7477
F-statistic: 1246 on 4 and 1682 degrees of freedom, the p-value is 0
```

In this model, the quadratic effect of wife's age is not significant and could be eliminated.

We have completed our introduction to linear regression. To terminate the session and exit from SPlus, we use the instruction:

```
> q()
```

## 2.2 Analysis of Variance

Not all of the variables in our data set are continuous like the ages of the two ex-spouses. Some are categorical variables, such as A, the origin of the application for divorce (husband = 1, wife = 2, or mutual = 3). We consider a one-way analysis of variance:

$$\mu_{ij} = \mu + \alpha_j \qquad j = 1, 2, 3 \tag{2.7}$$

where $\mu$ represents a grand mean, and $\alpha_j$ the effect of the $j$'th application type. The procedure for fitting the model is the same as above for a regression model. However, if we immediately proceed in this way, with A (the application origin) coded 1, 2, 3, we will have a regression with A treated as a continuous explanatory variable, not an analysis of variance. We must let SPlus know that A is a categorical variable, or factor, by transforming it with the factor() function.

```
> married.data$A_factor(married.data$A)
```

As it stands, our model has four parameters—$\mu$, $\alpha_1$, $\alpha_2$, and $\alpha_3$—one more parameter than the number of categories. We must add a constraint. This may be done in a number of ways, all of which are mathematically equivalent, but not all of which are as easily interpretable. By default, SPlus uses a *Helmert contrast* as the final constraint in analysis of variance. We use the contrasts parameter of lm() to force SPlus to use a "simpler" constraint in which $\alpha_1 = 0$ and the other $\alpha_j$ represent comparisons to the first category.

```
> married.aov1_lm(LM ~ A, data = married.data, contrasts = list(A = contr.treatment(3)))
```

The contr.treatment() function returns a contrasts matrix. A list() is used to include the contrasts matrix in the calculation because the matrix has to be associated with a specific factor variable, in this case A.

```
> summary(married.aov1)
```

```
Coefficients:
             Value Std. Error  t value Pr(>|t|)
(Intercept) 16.6417   0.3853  43.1949   0.0000
         A2 -2.8529   0.5194  -5.4924   0.0000
         A3 -5.0788   0.5132  -9.8968   0.0000

Residual standard error: 8.441 on 1684 degrees of freedom
Multiple R-Squared: 0.05497
F-statistic: 48.97 on 2 and 1684 degrees of freedom, the p-value is 0
```

Although the F-test is significant and both parameters for the variable, A, are significantly different from zero, we note that the residual standard error is larger than in the regression models of the first section. As might be expected, age "explains" the length of marriage much better than does the type of applicant. SPlus has calculated the parameter values for the qualitative variables in relation to the first category. Thus, the average length of marriage when the husband is the applicant is 16.64 years. It is 2.85 years less when the wife is the applicant, and 5.08 years less when application is by mutual consent.

An alternative, but equivalent, way of placing a constraint on our model is to have

$$\sum \alpha_j = 0$$

This is known as the conventional constraint and provides us with comparisons around the mean instead of with respect to one privileged category, the first, as was the case in the preceding model. This new constraint must also be explicitly specified to SPlus in the lm() function.

```
> married.aov2_lm(LM ~ A, data=married.data, contrasts=list(A=contr.sum(3)))
```

The contr.sum() function above returns a contrasts matrix corresponding to the constraint $\sum \alpha_j = 0$.

```
> summary(married.aov2)
```

```
Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)  13.9978    0.2068  67.7027   0.0000
         A1   2.6439    0.3037   8.7060   0.0000
         A2  -0.2090    0.2885  -0.7246   0.4688

Residual standard error: 8.441 on 1684 degrees of freedom
Multiple R-Squared: 0.05497
F-statistic: 48.97 on 2 and 1684 degrees of freedom, the p-value is 0
```

We note immediately that the residual deviance in this fit is identical to the residual deviance when we used contr.treatment(). This should not be surprising since we are fitting the same model, only with different constraints. We next note that the differences between types of application are the same in the two cases. For example, in the first model the contrast between the first two categories of application was 2.85; in the second model, it is $2.64 - (-0.21) = 2.85$.

With these constraints on the model, the first line of the display gives the grand mean over all application types. The parameter values for A1 and A2 are the differences from the grand mean for these two categories. Since the sum for the three categories is, by definition, zero, the parameter estimate for the third category is $-(2.64 - 0.21) = -2.43$. Thus, marriages where the wife applied for divorce are about average in length (the standard error indicates it to be non-significantly different from zero), while those with the husband applying are about 2.64 years longer and those with mutual applications are the 2.43 years shorter.

## 2.3 Analysis of Covariance

Once we can fit linear regression and analysis of variance with SPlus, analysis of covariance is simple: combine the two. We fit a parallel line analysis of covariance:

$$\mu_{ij} = \alpha_j + \sum_k \beta_k x_{ijk} \qquad j = 1, 2, 3 \tag{2.8}$$

or

$$E(\mathbf{LM}) = \mathbf{A}_j + \beta_1 \mathbf{HA} + \beta_2 \mathbf{HA}^2 + \beta_3 \mathbf{WA} + \beta_4 \mathbf{WA}^2 \qquad j = 1, 2, 3 \tag{2.9}$$

In this model, expected length of marriage (LM) depends on the ex-spouses' ages (HA and WA) in the same way within each type of application (A). In other words, the slope is the same for all application types, while the intercept is free to change (A indexed by j).

Before fitting the model, we specify a global option to ensure that all factor variables in SPlus linear models are returned using the *treatment contrast*. Global options are set using the options() function, as follows:

```
> options(contrasts="contr.treatment")
> married.aoc1_lm(LM ~ HA + HA.squared + WA + WA.squared + A, data = married.data)
```

which gives

```
Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept) -21.5639    1.4603  -14.7666   0.0000
         HA   0.5731    0.1042    5.4983   0.0000
 HA.squared  -0.0028    0.0012   -2.3963   0.0167
         WA   0.5679    0.1022    5.5562   0.0000
 WA.squared  -0.0016    0.0012   -1.3003   0.1937
         A2  -0.3586    0.2722   -1.3175   0.1879
         A3  -1.0101    0.2748   -3.6754   0.0002

Residual standard error: 4.348 on 1680 degrees of freedom
Multiple R-Squared: 0.7498
F-statistic: 839.1 on 6 and 1680 degrees of freedom, the p-value is 0
```

We see that the residual standard error has been somewhat reduced from the multiple linear regression case and that the mean differences in length of marriage are less than in the analysis of variance model. The quadratic parameter for wife's age is still non-significant. The difference in length of marriage for husband and wife applications appears non-significant when ages are taken into account, while mutual consent is different from the individual applications.

We may also fit an analysis of covariance model where the slopes are not constrained to be parallel:

$$\mu_{ij} = \alpha_j + \sum_k \beta_{jk} x_{ijk} \tag{2.10}$$

or

$$E(\mathbf{LM}) = \mathbf{A}_j + \beta_{1j} \mathbf{HA} + \beta_{2j} \mathbf{HA}^2 + \beta_{3j} \mathbf{WA} + \beta_{4j} \mathbf{WA}^2 \qquad j = 1, 2, 3 \tag{2.11}$$

Now, the slope can be different for each application type, as well as the intercept. In other words, we have an interaction between the ages (HA and WA) and the type of application (A)—length of

marriage (LM) does not vary with age (HA and WA) in the same way for all types of application (A). We add the interaction terms to the model.

```
> married.aoc2_lm(LM ~ HA + HA.squared + WA + WA.squared + A + HA*A + WA*A +
WA.squared*A + HA.squared*A, data=married.data)
```

to obtain

```
Coefficients:
                Value Std. Error  t value Pr(>|t|)
 (Intercept) -20.5495   2.3643    -8.6915   0.0000
          HA   1.1735   0.1787     6.5686   0.0000
  HA.squared  -0.0092   0.0019    -4.9187   0.0000
          WA  -0.1958   0.1727    -1.1339   0.2570
  WA.squared   0.0079   0.0019     4.0525   0.0001
          A2  -2.2208   3.4385    -0.6459   0.5185
          A3  -5.2909   3.7095    -1.4263   0.1540
        HAA2  -1.3553   0.2501    -5.4196   0.0000
        HAA3  -0.1352   0.2681    -0.5041   0.6142
        WAA2   1.6519   0.2342     7.0542   0.0000
        WAA3   0.4904   0.2847     1.7226   0.0851
WA.squaredA2  -0.0208   0.0027    -7.7696   0.0000
WA.squaredA3  -0.0062   0.0035    -1.7651   0.0777
HA.squaredA2   0.0152   0.0027     5.5361   0.0000
HA.squaredA3   0.0005   0.0031     0.1531   0.8783

Residual standard error: 4.239 on 1672 degrees of freedom
Multiple R-Squared: 0.7634
F-statistic: 385.2 on 14 and 1672 degrees of freedom, the p-value is 0
```

Again the residual standard error is smaller. However, judging from the parameter standard errors, only the interactions with the second application type appear to be significantly different from zero. The relationship between length of marriage and age is the same for husband and mutual applications, but differs for wife's applications.

## 2.4 The Extension to Non-Normal Models

As stated in the first section of this chapter, the construction of models with error distributions other than normal can be relatively simple with SPlus. Specifically, five other distributions may be very easily applied: the binomial, Poisson, inverse gaussian, quasi-likelihood and gamma distributions.

Here, the gamma distribution,

$$f(y;\theta,\phi) = \frac{\theta^{\phi} y^{\phi-1} e^{-\theta y}}{\Gamma(\phi)} \tag{2.12}$$

may interest us, since this is one of the distributions often used to describe survival times. Length of marriage may be considered to be the survival time of the marriage.

The main modification we must make to our analysis procedure of the previous sections is to use the glm() function to fit our model in place of the lm() function. The glm() function allows us to specify a link between the mean of the response and the explanatory variables, and an error distribution which is not necessarily gaussian.

glm(response ~ explanatory1 + ...+ explanatoryK, family = family(link=link), data=<data.frame>, weights=<weight vector>, na.action=na.fail)

We use the family parameter of the glm() function to specify a gamma distribution in place of the default gaussian. Note that an exponential distribution is a special case of the gamma, when $\phi = 1$. In SPlus, the distribution is specified to the glm() function through the use of a "family function". Family functions also allow you to specify the link to use between the response and the explanatory variables. To fit the model with a gamma distribution we invoke glm() as follows

```
> married.glm1_glm(LM ~ HA + HA.squared, data = married.data, Gamma(link = inverse))
```

Note that in the call of the Gamma() function we have specified link=inverse. This is not strictly necessary, since the Gamma() function will default to an inverse link if the parameter is not supplied. To view the result of the model fitting use

```
> summary(married.glm1)
```

```
Coefficients:
                  Value   Std. Error   t value
(Intercept)  0.4173470787 8.939966e-03  46.68329
         HA -0.0127432284 3.769889e-04 -33.80266
 HA.squared  0.0001043519 3.873194e-06  26.94207

(Dispersion Parameter for gamma family taken to be 0.1085201)
    Null Deviance: 675.4591 on 1686 degrees of freedom
Residual Deviance: 237.2879 on 1684 degrees of freedom
```

If we look at the graph of fitted values in Figure 2.4, we see that it is substantially different from that for the normal theory model.

```
> plot(married.data$HA, fitted(married.glm1), main="Plot of Fitted Values of LM vs HA",
xlab="HA", ylab="Fitted LM")
```

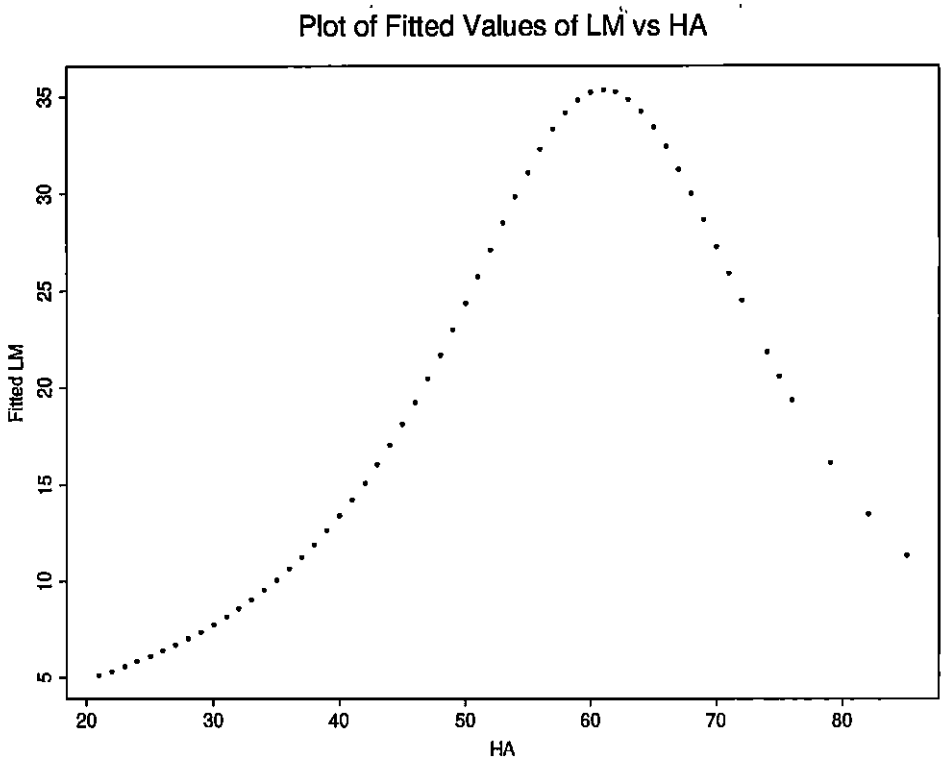



Figure 2.4: Plot of fitted values for length of marriage versus husband's age for generalized linear model with gamma family.

Although the first part of the curve is fairly similar, it reaches a peak at about 60 years and then goes back down for older people. If we look again at a graph of observed values (Figure 2.3), we see that this also may be a reasonable representation of the data. In part, the change in form of the regression line is due to the change of distribution. But the regression model has also changed. It is now

$$1/\mu_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 \tag{2.13}$$

SPlus uses a different way of linking the expected response to the regression equation for each family. For the normal distribution, it was the identity, but for the gamma it defaults to the reciprocal. A link parameter in the family function allows this default to be changed:

```
> gamma(link=<inverse/identity/logit/probit/etc...>)
```

We could redo the preceding analysis using an identity link with the gamma family.

## 2.5 Fitting Distributions

In the study of generalized linear models, the shape of the distribution is very important. If enough observations are available to form a frequency table, various distributions can be fitted to the data.

Suppose that data arising from a continuous distribution, $f(.)$, can be grouped into intervals

$$[y_1 - \frac{\Delta y_1}{2}, y_1 + \frac{\Delta y_1}{2}), [y_2 - \frac{\Delta y_2}{2}, y_2 + \frac{\Delta y_2}{2}), \ldots, [y_p - \frac{\Delta y_p}{2}, y_p + \frac{\Delta y_p}{2})$$

with centers $y_1, y_2, \ldots, y_p$. Let $c_k$ be the number of observations which fall within the interval $I_k = y_k \pm \frac{\Delta y_k}{2}$. The probability $p_k$ that an observation from $f(.)$ falls within $I_k$ is

$$p_k = \int_{y_k - \frac{\Delta y_k}{2}}^{y_k + \frac{\Delta y_k}{2}} f(y;\theta)\, dy \cong f(y_k;\theta)\Delta y_k$$

If $f(.)$ is a member of the exponential family, we have

$$p_k \cong \exp\left[\sum_{j=1}^{m} t_j(y_k)\theta_j + c(\theta) + d_2(y_k)\right]$$

where $d_2(.)$ depends on $y_k$ and on the width of the grouping intervals. See Lindsey [13] for details. We can then model the counts $c_1, c_2, \ldots, c_p$ as arising from a log-linear model where $\log p_k$ is modelled as

$$\sum_{j=1}^{m} t_j(y_k)\theta_j + c(\theta) + d_2(y_k)$$

Table 2.1 gives a list of some distributions which can be fitted this way. Lindsey [13] argues that the terms in the last column, labelled offset, may be included in the model as constants simply

| Distribution | $\theta_0$ | $\theta_1$ | $t_1$ | $\theta_2$ | $t_2$ | Offset |
|---|---|---|---|---|---|---|
| Exponential | $-\log\mu$ | $-\frac{1}{\mu}$ | $y_k$ | | | $\log\Delta y_k$ |
| Pareto ($y > c$) | $\log\theta + \theta\log c$ | $-(\theta+1)$ | $\log y_k$ | | | $\log\Delta y_k$ |
| Normal | $-\frac{1}{2}\log 2\pi\sigma^2 - \frac{\mu^2}{2\sigma^2}$ | $\frac{\mu}{\sigma^2}$ | $y_k$ | $-\frac{1}{2\sigma^2}$ | $y_k^2$ | $\log\Delta y_k$ |
| Log-Normal | $-\frac{1}{2}\log 2\pi\sigma^2 - \frac{\mu^2}{2\sigma^2}$ | $\frac{\mu}{\sigma^2}$ | $\log y_k$ | $-\frac{1}{2\sigma^2}$ | $\log^2 y_k$ | $-\log y_k + \log\Delta y_k$ |
| Inverse-Normal | $-\frac{1}{2}\log 2\pi\sigma^2 - \frac{\mu^2}{2\sigma^2}$ | $\frac{\mu}{\sigma^2}$ | $\frac{1}{y_k}$ | $-\frac{1}{2\sigma^2}$ | $\frac{1}{y_k^2}$ | $-2\log y_k + \log\Delta y_k$ |
| Gamma | $\phi\log\theta - \log\Gamma(\phi)$ | $-\theta$ | $y_k$ | $\phi - 1$ | $\log y_k$ | $\log\Delta y_k$ |
| Poisson | $-\mu$ | $\log\mu$ | $y_k$ | | | $-\log y_k! + \log\Delta y_k$ |
| Inverse Gaussian | $-\frac{1}{2}\log 2\pi\phi - \frac{\sqrt{-2\theta}}{\phi}$ | $\frac{\theta}{\phi}$ | $y_k$ | $-\frac{1}{2\phi}$ | $\frac{1}{y_k}$ | $-\frac{3}{2}\log y_k + \log\Delta y_k$ |

Table 2.1: Explanatory variables and parameters used in distribution-fitting with log-linear models.

added to the linear model. The offset includes the terms $\Delta y_k$ which may be allowed to vary within a certain objective range without greatly modifying the results. He states that if the data are grouped so the $\Delta y_k$ is not the unit of measurement, then $\boldsymbol{\theta}$ will not be accurately estimated.

Technically, to fit the grouped model, we should include grouping intervals for all possible values of the data—the $p_j$ must sum to one. In practice, we can only include a finite subset, therefore we exclude intervals with very small probability [13].

We fit these distributions to our divorce data, LM, ignoring the other explanatory variables in the data set. We first create a table for frequencies of length of marriage, using the tabulate() function.

```
> married.tab_tabulate(married.data$LM, nbin=55)
> married.tab

[1] 3 18 59 87 82 90 91 109 94 83 101 91 93 61 68 56 61 40 43
[20] 40 28 22 39 34 14 22 14 15 12 17 10 11 12 7 8 9 9 9
[39] 5 3 3 4 6 0 0 1 0 2 0 0 0 1 0 0 0
```

The nbin parameter tells tabulate() to compile the counts for lengths of marriage ranging from 1 to 55. For example, there are three marriages in the data set of length 1, so the first entry in the count vector is 3. Similarly, there are 18 marriages in the data set of length two. In Figure 2.5 we plot estimates of $p_k$ versus year to get some idea of the shape of the distribution of length of marriage. Note that a histogram of marriage lengths would serve the same purpose. The distribution is skewed to the right.

```
> plot(married.tab / sum(married.tab), ylab="Estimates of Pk")
```

Now we calculate the six functions of our variable we need from the table above and fit six distributions. To fit a log-linear model, we fit a generalized linear model with Poisson family and the default log link. Because the unit of measurement for the length of marriage is one year the offset, $-\log\Delta y_k = -\log 1 = 0$, is equal to zero for the distributions we are fitting.

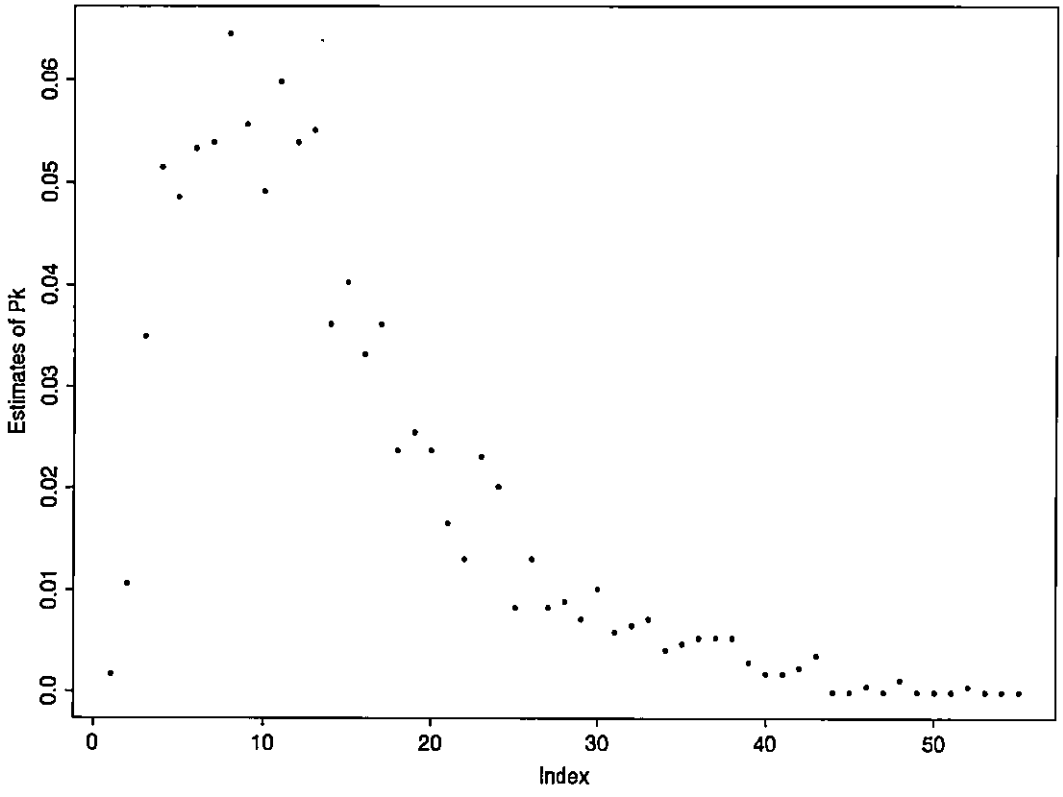



Figure 2.5: Plot of estimates of $p_k$ versus year for distribution of length of marriage.

> married.tab.data_data.frame(counts = married.tab, year = 1:55, lyear = log(1:55), year2 = 1:55*1:55, lyear2 = log(1:55)*log(1:55), iyear = 1/1:55, iyear2 = 1/(1:55*1:55))

### Exponential

> glm(counts ~ year, poisson(), data = married.tab.data)

```
Coefficients:
 (Intercept)        year
    4.809853 -0.06814945

Degrees of Freedom: 55 Total; 53 Residual
Residual Deviance: 608.6212
```

### Pareto

> glm(counts ~ lyear, poisson(), data = married.tab.data)

```
Coefficients:
 (Intercept)       lyear
     5.01514 -0.5708234

Degrees of Freedom: 55 Total; 53 Residual
Residual Deviance: 1412.802
```

### Normal

> glm(counts ~ year + year2, poisson(), data = married.tab.data)

```
Coefficients:
 (Intercept)       year       year2
    3.991895 0.06905249 -0.00368188

Degrees of Freedom: 55 Total; 52 Residual
Residual Deviance: 280.1573
```

### Log-Normal

> glm(counts ~ lyear + lyear2 + offset(-log(year)), poisson(), data = married.tab.data)

```
Coefficients:
 (Intercept)    lyear    lyear2
   0.4648555 5.276411 -1.080342

Degrees of Freedom: 55 Total; 52 Residual
Residual Deviance: 91.21889
```

**Inverse Normal**

```
> glm(counts ~ iyear + iyear2 + offset(-2*log(year)), poisson(), data = married.tab.data)

Coefficients:
 (Intercept)    iyear    iyear2
    9.935202 -9.71668 -2.458135

Degrees of Freedom: 55 Total; 52 Residual
Residual Deviance: 301.105
```

**Gamma**

```
> glm(counts ~ year + lyear, poisson(), data = married.tab.data)

Coefficients:
 (Intercept)       year    lyear
    2.699113 -0.1895884 1.621409

Degrees of Freedom: 55 Total; 52 Residual
Residual Deviance: 95.0112
```

The results of the log linear modelling are consistent with the graph in Figure 2.5. We see that the log normal and gamma distributions fit much better than any of the others, with the former slightly better. Note however, that this does not imply that one of these distributions is still suitable in some subgroups of the sample, i.e. when explanatory variables are introduced. The form of the distributions in a regression model may be different from that when all individuals are grouped together. This can be tested, if enough observations are available, by creating cross-classified tables of frequencies and fitting the above models with the explanatory variables included.

## 2.6 Further SPlus Instructions

### 2.6.1 Conventions

In order to retrieve simple estimates of factor levels, throughout the rest of the book all factor variables are fitted using the *treatment contrast.* All analyses may be assumed to have started with

the directive options(contrasts="contr.treatment"). Similarly, all analyses assume that there is an active graphics window available for plot output.

### 2.6.2 Data Manipulation

SPlus allows you to manipulate your data and retrieve subsets of data vectors using simple conditions. For example, x[1] returns the first entry in the data vector x. x[1:10] returns the first ten entries in the data vector. x[x>5] returns all entries of x such that x is greater than five. Select members of a vector can also be removed from data. x[-1] returns x without the first entry, for example.

Functional conditions can also be used to select data from a vector. For example, x[!is.na(x)] returns all values of x which are not coded as missing values. Similarly, x[-length(x)] returns x without the last entry.

Data can also be concatenated, using the c() function. If x equals $(1, 1)$ and y equals $(2, 2)$, then c(x,y) equals $(1, 1, 2, 2)$.

### 2.6.3 Data Entry

If data are not too voluminous, they may simply be typed directly in, by invoking scan() without any arguments.

```
> datavector_scan()
```

### 2.6.4 Custom Functions

Custom routines can be coded into SPlus by creating your own functions. SPlus is an object oriented language, so you can create your own functions by aggregating existing functions. In fact, most existing system functions are themselves written in the S language and can be displayed by invoking them without parentheses.

```
> mean

function(x, trim = 0, na.rm = F)
{
        if(na.rm)
                x <- x[!is.na(x)]
        else if(any(is.na(x)))
                return(NA)
        if(mode(x) == "complex") {
                if(trim > 0)
```

```
                stop("trimming not allowed")
        return(sum(x)/length(x))
    }
    x <- as.double(x)
    if(trim > 0) {
        if(trim >= 0.5)
            return(median(x))
        n <- length(x)
        i1 <- floor(trim * n) + 1
        i2 <- n - i1 + 1
        x <- sort(x, unique(c(i1, i2)))[i1:i2]
    }
    sum(x)/length(x)
}
```

To create our own simple function, we do the following:

```
> test.function_function(x)
+ {
+ y_mean(x)
+ y+2
+ }
```

We expect our custom function to return the mean of its input vector plus two, so let's test it out.

```
> test.function(c(1,5,1,5,1,5,1,5))

[1] 5
```

The test vector we have entered into our function clearly has a mean of 3, so we expect test.function() to return a value of 5, which it does.

# Chapter 3

# Markov Chains

## 3.1 Binary Point Processes

A *stochastic process* is a collection of random variables, indexed in some manner with the passage of time.

$$\{X(t), t \in T\}$$

So for every $t$ in $T$, $X(t)$ is a random variable. In the terminology of stochastic processes, $X(t)$ is the *state* of the process, and $t$ is the *index* of the process.

The set of all states $\boldsymbol{X}$ such that for all $t \in T$, $X(t) \in \boldsymbol{X}$ is called the *state space* of the process. In this chapter we deal exclusively with stochastic processes in which both $t$ and $X(t)$ take on discrete integer values only. In later chapters, we study processes in which one or both of $t$ and $X(t)$ are continuous variables.

This chapter is devoted to the analysis of *Markov chains*. A Markov chain has a discrete state space, $X(t) \in \{1, 2, 3, \ldots\}$, occupying only one state at a time. Although some Markov chains have a continuous time index, we will consider *discrete time Markov chains* in this section. So, we say that a Markov process is in state $i$ at time $t$ when $X_t = i$. The probability of going from state $i$ to state $j$ in one time interval is

$$P\{X_{t+1} = j | X_t = i\} = P_{ij}$$

Transition probabilities in a Markov chain are governed by the *Markov property*: the conditional distribution of future state $X_{t+1}$, given past and present states $X_t, X_{t-1}, \ldots, X_0$, is only dependent on the present state $X_t$. Expressed mathematically, the Markov property is

$$P\{X_{t+1} = j | X_t = i, X_{t-1} = i_{t-1}, \ldots, X_0 = i_0\} = P\{X_{t+1} = j | X_t = i\} = P_{ij}$$

A binary point process is a Markov process in which the state space is restricted to a pair of values—such that $X(t)$ is a binary random variable—and the time index is discrete.

$$X(t) \in \boldsymbol{X} \equiv \{1, 2\}$$

1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 1 0 0 2 0 1 2 5 2 1 0 0
1 0 1 1 0 3 0 1 0 0 6 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 3 0 1 0 0 0 1 0 0 0 0 2 0
0 0 3 0 1 1 1 1 1 0 0 0 0 0 0 0 3 0 2 0 1 0 2 0 3 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 1 0 2 1 1 0 0 0 0 0 0 0 4 1 0 0 0 0 0 0 0 1 0 0 2 0 2 1 1 0 0 1 0 2 0 0 0 1
0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 2 0 0 1 0 1 0 3 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1

Table 3.1: Number of compensable accidents per day for one shift in a section of a mine. (Barnett and Lewis, 1984, p.195)

$$t \in T \equiv \{0, 1, 2, 3, 4, 5, \ldots\}$$

Consider an example (Table 3.1) concerning accidents in a mine. Although we have information on the number of accidents per day, we ignore this and concentrate on presence or absence of an accident on a given day. Thus, we have two states in our state space: a state where at least one accident occurred in a day of mining and a state where no accidents occurred in a day of mining.

We read the data into SPlus and create a new vector with the binary values zero (for an accident-free day) and one (for an accident day). Next, we create a new vector lagged by one time period and eliminate the first observation from the original vector. We cross-tabulate the two variables to obtain a frequency table and display it.

```
> markov.data_scan("data/Chpt2-1")
> markov.data_ifelse(markov.data > 0, 1, 0)
> table(markov.data[1:222], markov.data[2:223])
```

| | | Lag 0 | |
|---|---|---|---|
| | | **0** | **1** |
| Lag 1 | **0** | 115 | 46 |
| | **1** | 46 | 15 |

We see that 115 times, two no-accident days succeed one another, while 15 times two accident days follow one another. For 46 times each, a no-accident day comes after one with accidents or vice versa. The probability of transition from an accident free day to one with an accident is estimated as $46/161 = 0.286$ and in the opposite direction $46/61 = 0.754$.

We are interested in determining whether the state of the process is dependent on the state immediately previous; that is, whether the category variable indicating current state is independent of the lagged variable. To make this determination we use a log-linear *independence model* of the expected cell counts in our frequency table,

$$\log e_{ij} = \log nP_{ij} = \mu + \alpha_i + \beta_j \tag{3.1}$$

where $e_{ij}$ are the expected cell counts, $\alpha_i$ and $\beta_j$ are factors indicating the state of the process before and after a transition respectively, and $n$ is the total number of events observed. Including

$\alpha_i$ in the model forces the expected row totals to match the observed row totals when the model is fitted, and including $\beta_j$ forces the expected column totals to match the observed column totals. In this way, we impose an independence model, where $P_{ij} = P_{i.}P_{.j}$. (See for example Agresti [2]).

To fit the model in SPlus, we use the glm() function, specifying the Poisson family. Unfortunately, the glm() function does not accept multiple columns in the response variable. In order to fit the frequency table, we have to turn it into a single column, and expand the factor vectors to fit. Instead of a table, we will be passing the data to glm() in a form somewhat like this:

```
Response      lag0      lag1
  115           0         0
   46           1         0
   46           0         1
   15           1         1
```

To create a correctly formatted set of columns which glm() can handle, we use a custom function, given in Section 9.1, called glmtab(). If you look closely at glmtab(), you will notice that it is simply a variation on the table() function, providing an alternate form of output.

```
> markov.tab_glmtab(markov.0lag = markov.data[2:223], markov.1lag = markov.data[1:222])
> print(markov.tab)

  Counts markov.0lag markov.1lag
1    115           0           0
2     46           1           0
3     46           0           1
4     15           1           1
```

Now we can fit the log-linear independence model.

```
> options(contrasts = "contr.treatment")
> markov.glm1_glm(Counts ~ markov.0lag + markov.1lag, family=Poisson(), data = markov.tab)
> summary(markov.glm1)

Coefficients:
                 Value Std. Error   t value
(Intercept)  3.7896008 0.08244870  45.96314
markov.0lag -0.4852653 0.07517409  -6.45522
markov.1lag -0.4852653 0.07517409  -6.45522

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 93.77185 on 3 degrees of freedom
Residual Deviance: 0.3570362 on 1 degrees of freedom
```

We assess the fit of the independence model using the likelihood ratio statistic: a comparison with the saturated model

$$\log nP_{ij} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} \tag{3.2}$$

which has one more parameter than the independence model, and a residual deviance of zero. The likelihood ratio statistic in this case is the residual deviance of the independence model minus the residual deviance of the saturated model, and has an asymptotic $\chi^2_1$ distribution. The p-value is computed as $1 - P(\chi^2_{(1)} \geq 0.357) = 0.55$ below.

```
> 1-pchisq(0.3570362,1)

[1] 0.5501569
```

The likelihood ratio test shows that there is no evidence against the independence model; therefore, there is no evidence that odds of an accident occurring in the mine on a given day are dependent on whether an accident occurred in the mine one day previous.

However, it is possible that an event at time $t$ depends on events further in the past. Consider a model which incorporates dependence on events two days previous, a *second order Markov chain.* We create a variable shifted two days and cross tabulate a three-way table.

```
> markov.tab_glmtab(markov.0lag = markov.data[3:223], markov.1lag = markov.data[2:222],
markov.2lag = markov.data[1:221])
> print(markov.tab)
```

| | Counts | markov.0lag | markov.1lag | markov.2lag |
|---|---|---|---|---|
| 1 | 87 | 0 | 0 | 0 |
| 2 | 28 | 1 | 0 | 0 |
| 3 | 38 | 0 | 1 | 0 |
| 4 | 7 | 1 | 1 | 0 |
| 5 | 28 | 0 | 0 | 1 |
| 6 | 18 | 1 | 0 | 1 |
| 7 | 7 | 0 | 1 | 1 |
| 8 | 8 | 1 | 1 | 1 |

We fit the model where the present event does not depend on events at the two previous time periods. All margins (main effects) and all interactions among explanatory variables (here, the previous two time periods) must be included in a log linear model for independence. By including the main effects, we fix all the main marginal totals in the contingency table equal to the observed totals.

The inclusion of the interaction effect between the two lag terms forces the two-way fitted marginal totals to be equal to the observed totals. This is required because, conditional on the state in lags 1 and 2, we must move to either state 0 or 1, ie. $p_{ij0} + p_{ij1} = 1$.

```
> markov.glm2_glm(Counts ~ markov.0lag + (markov.1lag * markov.2lag), data = markov.tab, family
= Poisson())
> summary(markov.glm2)
```

```
Coefficients:
                              Value Std. Error     t value
            (Intercept)  4.4219432  0.1020821  43.3175053
            markov.0lag -0.9643000  0.1504771  -6.4082853
            markov.1lag -0.9382696  0.1758347  -5.3360887
            markov.2lag -0.9162907  0.1744556  -5.2522849
markov.1lag:markov.2lag -0.1823216  0.3454287  -0.5278124

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 151.6666 on 7 degrees of freedom
Residual Deviance: 11.53655 on 3 degrees of freedom
```

```
> 1-pchisq(11.53655,3)
```

```
[1] 0.009151722
```

This time, the chi-squared value is significant, suggesting evidence against the independence model, so that some time dependence must be present. We add dependence on the previous day by incorporating the interaction term markov.0lag:markov.1lag.

```
> markov.glm3_glm(Counts ~ (markov.0lag * markov.1lag) + (markov.1lag * markov.2lag), data =
markov.tab, family = Poisson())
> summary(markov.glm3)
```

```
Coefficients:
                              Value Std. Error    t value
            (Intercept)  4.4084599  0.1057361  41.693042
            markov.0lag -0.9162907  0.1744557  -5.252284
            markov.1lag -0.8894795  0.1973775  -4.506488
            markov.2lag -0.9162907  0.1744557  -5.252284
markov.0lag:markov.1lag -0.1823216  0.3454244  -0.527819
markov.1lag:markov.2lag -0.1823216  0.3454244  -0.527819

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 151.6666 on 7 degrees of freedom
Residual Deviance: 11.25389 on 2 degrees of freedom
```

```
> 1-pchisq(11.25389,2)
```

```
[1] 0.003599555
```

As expected, the chi-square changes very little, showing once again that there is no dependence on the immediately preceding day. Now we add dependence two days back:

```
> markov.glm4_glm(Counts  (markov.0lag * markov.1lag) + (markov.0lag * markov.2lag) +
(markov.1lag * markov.2lag), data = markov.tab, family = Poisson())
> summary(markov.glm4)

Coefficients:
                              Value Std. Error     t value
            (Intercept)  4.4888926  0.1049072  42.7891694
            markov.0lag -1.2316735  0.2127682  -5.7888047
            markov.1lag -0.9060073  0.1904797  -4.7564518
            markov.2lag -1.2316735  0.2127682  -5.7888047
markov.0lag:markov.1lag -0.1514564  0.3526732  -0.4294525
markov.0lag:markov.2lag  0.9713264  0.3221019   3.0155877
markov.1lag:markov.2lag -0.1514564  0.3526732  -0.4294525

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 151.6666 on 7 degrees of freedom
Residual Deviance: 2.27852 on 1 degrees of freedom

> 1-pchisq(2.27852,1)

[1] 0.131177
```

Here the change in chi-square is significant and there is no evidence against the two-day dependence model (p-value = 0.13). The model suggests that if an accident occurs in the mine, there is an increased probability that another accident will occur two days later. We have a second-order Markov chain.

As far as the data permit, there not being too many zeroes in the table, we could continue to test for dependence further back in time by creating larger tables. However, for our data, the essential dependence has already been determined: the two day period.

We now look at another way in which all the above analyses can be carried out: directly on the observed vector of binary data, without constructing a table. The binary response can be analyzed as a *logistic model*

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{3.3}$$

where $\pi_i$ is the probability of an accident day and $\{x_1, x_2\}$ are the lag one and lag two vectors, respectively. For a general discussion of the relationship between $I \times J \times 2$ tables and logistic models, see Agresti [2].

```
> summary(glm(markov.data[2:223] ~ markov.data[1:222], family = binomial(link=logit)))

Coefficients:
                       Value Std. Error    t value
      (Intercept) -0.9162906  0.1744289 -5.2530882
markov.data[1:222] -0.2042989  0.3445517 -0.5929412

(Dispersion Parameter for Binomial family taken to be 1)
    Null Deviance: 261.0499 on 221 degrees of freedom
Residual Deviance: 260.6929 on 220 degrees of freedom
```

Here we see that the difference in residual deviance between the simple model and the model with one lag term is is 0.357 which corresponds to a probability of 0.55. This suggests that the indicator for accidents on a given day is not significantly related to the accident indicator for the previous day.

```
> summary(glm(markov.data[3:223] ~ markov.data[2:222] + markov.data[1:221], family =
binomial(link=logit)))

Coefficients:
                       Value Std. Error    t value
      (Intercept) -1.2316654  0.2124867 -5.7964349
markov.data[2:222] -0.1514497  0.3522421 -0.4299592
markov.data[1:221]  0.9713167  0.3219393  3.0170800

(Dispersion Parameter for Binomial family taken to be 1)
    Null Deviance: 260.4057 on 220 degrees of freedom
Residual Deviance: 251.1476 on 218 degrees of freedom

> 260.6929 - 251.1476

[1] 9.5453

> 1 - pchisq(9.5453,1)

[1] 0.002004621
```

The change in deviance of 9.5453 between the model with a one day lag and the model with a two day lag corresponds to a p-value of 0.002, which confirms our previous analysis using log-linear models. Also note that the two parameter values, lag one (-0.1514) and lag two (0.9713) are similar to those for markov.0lag:markov.1lag and markov.0lag:markov.2lag in the final Poisson model, giving the dependence one and two steps back. Notice that only the second parameter value in each case is significantly different from zero.

```
> summary(glm(markov.data[4:223] ~ markov.data[3:222] + markov.data[2:221] +
markov.data[1:220], family = binomial(link=logit)))

Coefficients:
                      Value Std. Error    t value
      (Intercept) -1.3420280  0.2321129 -5.7817891
markov.data[3:222] -0.2764446  0.3655953 -0.7561493
markov.data[2:221]  1.0255390  0.3254371  3.1512666
markov.data[1:220]  0.4612950  0.3465393  1.3311480

(Dispersion Parameter for Binomial family taken to be 1)
    Null Deviance: 259.758 on 219 degrees of freedom
Residual Deviance: 248.2462 on 216 degrees of freedom
```

We did not fit the three-step model using the Poisson family. However, here we see from the t-value for the lag three term that the third step is not necessary.

This second approach to binary point processes is especially useful if some explanatory variables are available for each point. For example, if we had the day of the week coded, we could test to see if accidents were more frequent on weekends or during the week.

## 3.2 Multi-state Markov Chains

The natural extension of the model of the previous section is to the case where several states are possible, instead of just two. The essential procedure for the analysis with SPlus, using log linear models, remains unchanged. In the previous section, the different states could have been the numbers of accidents per day. Instead of reanalysing that table as a multi-state model, consider a new data set on the number of patients arriving at a hospital (Table 3.2).

When this table is analysed as a binary process, with the methods of the previous section, we do not find any dependence over time. Let us see if anything changes when we take into account the different numbers of patients arriving each day. Since there is only one day with four patients entering, we shall take the fourth and last state as being three or more entries. Our four possible events are zero, one, two, and three or more patients during the day.

```
> hospital.data_scan("data/Chpt2-2")
> hospital.data_ifelse(hospital.data > 3, 3, hospital.data)
> hospital.1lag_hospital.data[1:410]
> hospital.0lag_hospital.data[2:411]
> table(hospital.1lag, hospital.0lag)
```

0 0 0 2 0 0 0 1 0 0 1 0 0 0 0 1 0 2 0 2 1 0 0 0 2 0 0 1 1 0 1 0 0 0 2 0 1 0 0 0 0 0 1 1 1 0 0 1
0 1 0 3 0 0 0 2 0 1 0 0 2 0 0 0 1 0 0 2 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1
0 0 0 1 0 1 2 0 1 0 1 1 1 0 1 2 1 1 0 0 1 0 1 0 2 0 1 0 1 0 0 0 0 2 0 1 2 0 0 1 0 2 0 0 2 1 0 2
2 1 1 0 1 1 0 2 0 0 0 0 0 1 1 1 0 1 3 0 0 0 0 0 0 0 1 1 0 1 1 0 0 2 0 1 0 0 1 1 2 0 0 0 3 0 0 0 1
0 3 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 2 1 0 1 0 1 1 0 1 0 0 1 0 1 2 0 0 1 1
0 0 1 1 2 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 3 0 1 1 0 0 1 0 0 0 0 1 1 2 0 0 2 1 1 1 1 2 0 1 1 0 0 3
1 1 1 1 0 0 1 2 1 0 1 1 0 1 2 1 1 4 0 2 0 2 1 1 0 1 1 1 2 1 1 1 0 0 2 1 1 2 2 2 0 0 1 0 0 1 0 1
1 0 1 1 3 0 0 2 1 1 0 2 1 0 1 0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 0 0 0 0 2 1 1 0 0 0 2 1 0 0 0 0 1 3 1
1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 2

Table 3.2: Number of patients arriving at an intensive care unit per day, February 1963 - March, 1964 (Cox and Lewis, 1966, pp.254)

| | | Lag 0 | | | |
|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** |
| Lag 1 | **0** | 112 | 75 | 25 | 4 |
| | **1** | 76 | 53 | 12 | 5 |
| | **2** | 20 | 16 | 3 | 0 |
| | **3** | 7 | 2 | 0 | 0 |

We then simply fit the independence model as in the previous section.

```
> hospital.m_glmtab(hospital.1lag = hospital.1lag, hospital.0lag = hospital.0lag)
> hospital.glm1_glm(Counts ~ hospital.0lag + hospital.1lag, family = Poisson(), data = hospital.m)
> anova(hospital.glm1, test = "Chisq")

Analysis of Deviance Table
Poisson model
Response: Counts

Terms added sequentially (first to last)
             Df Deviance Resid. Df Resid. Dev Pr(Chi)
        NULL                    15   616.8887
hospital.0lag  3 302.7604         12   314.1283       0
hospital.1lag  3 306.1538          9     7.9745       0
```

There is no evidence of dependence between successive days. The residual deviance is 7.97 on 9 degrees of freedom, indicating that the independence model provides a good fit. Note that the output here is slightly different—we have printed out the *analysis of deviance table* using anova(). Because so many parameters are estimated in the model, and we do not need to see all the estimates at this stage, using anova() allows us to view only the relevent information about variable effects.

We next construct the table for dependence over the previous two time periods. Since we already have some zeroes in our table, the corresponding parameters in the new table will not be estimable. First we fit the independence model to the three-way table.

```
> hospital.m_glmtab(hospital.0lag = hospital.data[3:411],hospital.1lag =
hospital.data[2:410],hospital.2lag = hospital.data[1:409])

> hospital.glm2_glm(Counts ~ hospital.0lag + (hospital.1lag * hospital.2lag), family = Poisson(), data
= hospital.m)
> anova(hospital.glm2, test = "Chisq")


Analysis of Deviance Table
Poisson model
Response: Counts

Terms added sequentially (first to last)
                          Df Deviance Resid. Df Resid. Dev   Pr(Chi)
                        NULL                    63   947.2529
              hospital.0lag  3 301.2811        60   645.9719 0.0000000
              hospital.1lag  3 304.6651        57   341.3067 0.0000000
              hospital.2lag  3 304.6651        54    36.6416 0.0000000
hospital.1lag:hospital.2lag  9   7.6451        45    28.9965 0.5702702

> 1-pchisq(28.9965,45)

[1] 0.9693111
```

Again, the independence model appears to fit satisfactorily. The residual deviance is 28.9 on 45 degrees of freedom, indicating a satisfactory fit. There is no evidence of dependence at lag one or at lag two.

When we attempt to fit the model for dependence two steps back, the extra zeros cause the glm() algorithm to end without reaching convergence.

```
> hospital.glm3_glm(Counts ~ (hospital.0lag * hospital.1lag) + (hospital.0lag * hospital.2lag) +
(hospital.1lag * hospital.2lag), family = Poisson(), data = hospital.m)


Warning messages:
  linear convergence not obtained in  10  iterations. in: glm.fitter(x = X, y =
        Y, w = w, start = start, offset = offset, family = family, ....
```

However, we can force glm() to continue iterating longer and obtain estimates for the lag one and lag two effects, by increasing the maxit parameter above the default value of 10.

```
> hospital.glm3_glm(Counts ~ (hospital.0lag * hospital.1lag) + (hospital.0lag * hospital.2lag) +
(hospital.1lag * hospital.2lag), family = Poisson(), data = hospital.m, maxit = 20)
> anova(hospital.glm3, test = "Chisq")

Analysis of Deviance Table
Poisson model
Response: Counts

Terms added sequentially (first to last)
                              Df Deviance Resid. Df Resid. Dev   Pr(Chi)
                    NULL                      63   947.2529
               hospital.0lag  3 301.2811         60   645.9719 0.0000000
               hospital.1lag  3 304.6651         57   341.3067 0.0000000
               hospital.2lag  3 304.6651         54    36.6416 0.0000000
hospital.0lag:hospital.1lag  9   7.9983         45    28.6433 0.5343127
hospital.0lag:hospital.2lag  9   5.3049         36    23.3383 0.8069600
hospital.1lag:hospital.2lag  9   7.5307         27    15.8077 0.5820423
```

We conclude that there is no evidence of any relationship between the numbers of patients entering on successive days, whether one or two steps back. Given the large number of zeroes, we cannot test for larger steps.

## 3.3 Stationarity

In observations of the type just considered, it is often useful to determine if the same pattern of change occurs in each period. Suppose that individual responses at a given time point depend only on those of the immediately preceding point, the hypothesis of a first order Markov chain. Then, the probability of an event in any given category depends only on the event for the immediately preceding time point. Therefore, there is a square transition matrix of probabilities.

If the rows are the categories at the previous time point and the columns are the present categories, then the row probabilities sum to one. Post-multiplying the vector of frequencies of individuals in the different categories (the marginal frequencies) at a given time point, $t-1$, by this matrix will give the vector of expected frequencies for the next time period, $t$:

$$\boldsymbol{n}_t^T = \boldsymbol{n}_{t-1}^T \boldsymbol{P} \tag{3.4}$$

The matrix $\boldsymbol{P}$ represents the pattern of change; if it is the same over each time period $t$, we have stationarity. In this section, we test the stationarity of a first-order Markov chain, assuming the first order hypothesis to be true.

A common application of Markov chains is to voting behaviour. Here, we consider successive monthly expressions of intention to vote in the 1940 US presidential elections for Erie County (Table

| | | Party | | |
|---|---|---|---|---|
| | | Republican | Democrat | Undecided |
| | | | June | |
| **Month** | **Party** | | | |
| | Republican | 125 | 5 | 16 |
| May | Democrat | 7 | 106 | 15 |
| | Undecided | 11 | 18 | 142 |
| | | | July | |
| | Republican | 124 | 3 | 16 |
| June | Democrat | 6 | 109 | 14 |
| | Undecided | 22 | 9 | 142 |
| | | | August | |
| | Republican | 146 | 2 | 4 |
| July | Democrat | 6 | 111 | 4 |
| | Undecided | 40 | 36 | 96 |
| | | | September | |
| | Republican | 184 | 1 | 7 |
| August | Democrat | 4 | 140 | 5 |
| | Undecided | 10 | 12 | 82 |
| | | | October | |
| | Republican | 192 | 1 | 5 |
| September | Democrat | 2 | 146 | 5 |
| | Undecided | 11 | 12 | 71 |

Table 3.3: One step transitions for voting intentions in the 1940 US Presidential Elections, Erie County. (Goodman, 1962)

3.3). The data consist of a series of five two-way tables, yielding a three-way table over the five time periods. Note that observations over time have been made on a large number of individuals, whereas, in the previous sections, only one subject (the mine or hospital) was observed over time.

We have three variables: the voting intention at the beginning of any period (t1), the voting intention at the end of any period (t2), and the five time periods (time). The test for stationarity is a test of independence between time period and intention to vote at the end of the period; this relationship is omitted from the model. Thus, the model will contain: the three sets of main effects parameters (t1, t2, and time); the relationships between intentions at the beginning and end of a period (t1 × t2); and, the relationships between intentions at the beginning of a period and the period itself (time × t1).

We read in the data, and then use the custom function mpct() (Section 9.3) to calculate the transition matrix. Because the calculations made by mpct() are predicated on an assumption of stationarity, the transition matrix generated will only have meaning if the process is stationary.

```
> elect.data_data.frame(scan("data/Chpt2-3", what = list(counts = 0, t1 = "", t2 = "", time =
""")))
> mpct(counts, t1, t2, time, data = elect.data)

First Order Markov Chain
Estimated Stationary Transition Probabilities

          [,1]       [,2]       [,3]
[1,] 0.92779783 0.01444043 0.05776173
[2,] 0.03676471 0.90000000 0.06323529
[3,] 0.13165266 0.12184874 0.74649860
```

To test for stationarity, we use the following log-linear model

$$\log nP_{ij} = \mu + \mathbf{t1}_i + \mathbf{t2}_j + \mathbf{time}_k + (\mathbf{t1} \times \mathbf{t2})_{ij} + (\mathbf{t1} \times \mathbf{time})_{ik} \tag{3.5}$$

The t1, t2 and t1 × t2 terms serve to fix the terms describing cells in the individual transition tables, aggregated over all time periods. The time and t1 × time terms further restrict the model, fixing the estimated row totals to be equal to the observed totals. This leaves the t2 × time term unrestricted, which is a measure of how much the state of the process after a transition varies with the time index of the transition: if the model is stationary there should be no variation at all.

```
> anova(glm(counts ~ (t1 * t2) + (t1 * factor(time)), family = poisson(), data = elect.data), test =
"Chisq")

Poisson model
Response: counts

Terms added sequentially (first to last)
                 Df Deviance Resid. Df Resid. Dev   Pr(Chi)
          NULL                    44   2995.355
            t1  2   16.673        42   2978.682 0.0002396
            t2  2   48.893        40   2929.788 0.0000000
   factor(time)  4    0.000        36   2929.788 1.0000000
         t1:t2  4 2758.261        32    171.528 0.0000000
t1:factor(time)  8   70.021        24    101.507 0.0000000
```

We see that the hypothesis of stationarity is decisively rejected. The residual deviance of 101.507 on 24 degrees of freedom indicates a p-value of zero for the model.

One important piece of information which was not included in the initial model for stationarity is that the Democratic convention was held during the third period. If we reconstruct separate tables for the first two time periods and for the last two and apply the stationarity test separately to each we may get a different result.

```
> mpct(counts, t1, t2, time, data = elect.data[1:18,])
> anova(glm(counts ~ (t1 * t2) + (t1 * factor(time)), family = poisson(), data = elect.data[1:18,]),
test = "Chisq")


First Order Markov Chain
Estimated Stationary Transition Probabilities

           [,1]       [,2]      [,3]
[1,] 0.86159170 0.02768166 0.1107266
[2,] 0.05058366 0.83657588 0.1128405
[3,] 0.09593023 0.07848837 0.8255814

Analysis of Deviance Table
Poisson model
Response: counts

Terms added sequentially (first to last)
                Df Deviance Resid. Df Resid. Dev   Pr(Chi)
          NULL                    17   1021.642
            t1  2  12.9369        15   1008.705 0.0015517
            t2  2  15.2472        13    993.458 0.0004888
   factor(time)  1   0.0000        12    993.458 0.9999988
         t1:t2  4 985.9994         8      7.459 0.0000000
t1:factor(time)  2   0.0467         6      7.412 0.9769393

> mpct(counts, t1, t2, time, data = elect.data[28:45,])
> anova(glm(counts ~ (t1 * t2) + (t1 * factor(time)), family = poisson(), data = elect.data[28:45,]),
test="Chisq")


First Order Markov Chain
Estimated Stationary Transition Probabilities

           [,1]        [,2]       [,3]
[1,] 0.96410256 0.005128205 0.03076923
[2,] 0.01986755 0.947019868 0.03311258
[3,] 0.10606061 0.121212121 0.77272727

Analysis of Deviance Table
Poisson model
Response: counts

Terms added sequentially (first to last)
                Df Deviance Resid. Df Resid. Dev   Pr(Chi)
```

```
                 NULL                    17   1468.807
                   t1  2   64.002        15   1404.806 0.0000000
                   t2  2   93.608        13   1311.198 0.0000000
      factor(time)     1    0.000        12   1311.198 0.9999326
              t1:t2    4 1309.048         8      2.150 0.0000000
t1:factor(time)        2    0.651         6      1.499 0.7223256
```

In each separate table, stationarity is no longer rejected. For the May-July process, the residual deviance is 7.412 on 6 degrees of freedom (p-value of 0.2844191) and for the August-October process, the residual deviance is 1.499 on 6 degrees of freedom (p-value of 0.959561).

We note, as expected, that the diagonal transition probabilities are considerably smaller before the convention than after. In the first table (May-July), 86% of those intending to vote for the Republicans and 84% of those for the Democrats do not change their minds over a months period, whereas, after the convention (August-October), the percents are 96% and 95% respectively.

Remember, however, that this analysis supposes that intentions at one point in time only depend on intentions one month before. This is the hypothesis of a first-order Markov chain, which we have not tested for these data.

## 3.4 Reversibility and Equilibrium

In a Markov chain, if the probability of transition between states is the same in each direction, we have what is known as reversibility. In terms of the analysis of contingency tables using log-linear models, this is called *quasi-symmetry*. In the same way, if the margins are not changing over time, the process described by the Markov chain is in the equilibrium state. In terms of contingency tables, this is *marginal homogeneity*. If we combine the two, we have a completely symmetrical table.

In what follows, we shall assume the hypothesis of a first order, stationary Markov chain with transition matrix $\boldsymbol{P} = [\pi_{ij}]$. In any case, in a square, two-dimensional contingency table, we have no information to test these hypotheses.

A system showing both reversibility and equilibrium will have a completely symmetrical table in which the probabilities in opposing cells across the diagonal are equal:

$$\pi_{ik} = \pi_{ki} \tag{3.6}$$

The corresponding log linear model is

$$\log n\pi_{ik} = \alpha_{ik} \qquad \text{where} \qquad \alpha_{ik} = \alpha_{ki} \tag{3.7}$$

The factor variable used to fit this model is a special symmetric matrix, which can be created by hand, or by using the custom function sym.matrix:

| | 1985 | | | | | |
|---|---|---|---|---|---|---|
| **1981** | PS | PRL | PSC | Ecolo | PCB | BB |
| Socialist (PS) | 281 | 12 | 5 | 6 | 1 | 2 |
| Liberal (PRL) | 14 | 164 | 10 | 0 | 0 | 1 |
| Social-Christian (PSC) | 9 | 13 | 121 | 1 | 0 | 0 |
| Ecology (Ecolo) | 16 | 4 | 8 | 50 | 2 | 0 |
| Communist (PCB) | 4 | 1 | 1 | 0 | 14 | 0 |
| Blank Ballot (BB) | 4 | 6 | 1 | 1 | 0 | 11 |

Table 3.4: Voting changes between Belgian elections, 1981-1985. (R.Doutrelepont)

```
function(size)
{
        if(!is.numeric(size))
                stop("size is not numeric")
        if(length(size) > 1)
                stop("size is a scalar")
        sym <- matrix(0, ncol = size, nrow = size)
        lower <- row(sym) > col(sym)
        sym <- cumsum(lower) * lower
        sym <- sym + t(sym) + diag(max(sym) + 1:size)
        sym
}
```

For example, sym.matrix(4) returns

```
     [,1] [,2] [,3] [,4]
[1,]    7    1    2    3
[2,]    1    8    4    5
[3,]    2    4    9    6
[4,]    3    5    6   10
```

We fit this model to data on how voters, interviewed outside the polling station in the October 1985 Belgian election, stated they had just voted and how they had voted the previous election (Table 3.4). A symmetry model implies that all parties received the same number of votes in each election, and that the number of people changing from any party "A" to party "B" will be equal to the number changing from party "B" to party "A".

We load the data matrix as a single column of numbers. Because SPlus does matrix-to-vector and vector-to-matrix transformations by columns, unless otherwise specified, it is easier to analyze matrices if they are initially read into a single vector column by column.

We have a 36 item vector, corresponding to a 6 by 6 contingency table. Using the sym.matrix() function we construct a matrix to test the hypothesis of symmetry.

```
> elect.chan_scan("data/Chpt2-4", what = list(counts = 0, before = "", after = ""), skip = 1)
> elect.chan$counts

 [1] 281  12   5   6   1   2  14 164  10   0
[11]   0   1   9  13 121   1   0   0  16   4
[21]   8  50   2   0   4   1   1   0  14   0
[31]   4   6   1   1   0  11

> elect.sym_sym.matrix(6)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   16    1    2    3    4    5
[2,]    1   17    6    7    8    9
[3,]    2    6   18   10   11   12
[4,]    3    7   10   19   13   14
[5,]    4    8   11   13   20   15
[6,]    5    9   12   14   15   21

> elect.sym_factor(as.vector(elect.sym))
> elect.glm_glm(elect.chan$counts ~ elect.sym, family = poisson())
> anova(elect.glm, test = "Chisq")

Analysis of Deviance Table
Poisson model
Response: elect.chan$counts

Terms added sequentially (first to last)
          Df Deviance Resid. Df Resid. Dev Pr(Chi)
     NULL                    35   2365.665
elect.sym 20 2332.611        15     33.054       0

> 1-pchisq(33.054,15)

[1] 0.004613806
```

The residual deviance is large (33.054, 15 degrees of freedom, p-value of 0.0046), indicating that the symmetry model is not appropriate. Since the model is rejected, the probability of changing vote in either direction between each pair of parties is not the same. The model includes both the hypotheses of reversibility and equilibrium; one or both of them is being rejected.

Another method of fitting the symmetry model (Agresti [2], p.382) is to transpose the two-way contingency table, and place it on top of the original, to create a three-way table, then fit the model

$$\log nP_{ij} = \alpha_{ijk} \qquad \text{where} \qquad \alpha_{ij1} = \alpha_{ij2}$$

We now move on and consider the hypotheses of reversibility and equilibrium separately.

A weaker hypothesis than symmetry is that of reversibility or quasi-symmetry; the table would be symmetric if it were not for the distorting effect of the marginal totals. In other words, the system is not in equilibrium, since total numbers of votes for each party has changed between the elections. We simply add the two mean parameters to the model in equation 3.7.

$$\log n\pi_{ik} = \mu + \beta_i + \alpha_{ik} \qquad \text{where} \qquad \alpha_{ik} = \alpha_{ki} \tag{3.8}$$

```
> elect.glm2_update(elect.glm, . ~ . + elect.chan$before)
> anova(elect.glm2)

Analysis of Deviance Table
Poisson model
Response: elect.chan$counts

Terms added sequentially (first to last)
                  Df Deviance Resid. Df Resid. Dev
             NULL                    35   2365.665
        elect.sym 20 2332.611        15     33.054
elect.chan$before  5   22.953        10     10.101
```

This model fits the data very well; the residual deviance of 10.101 on 10 degrees of freedom gives a p-value of 0.432. The probability of shifting in either direction between each pair of parties is the same after taking into account the overall change in voting behaviour between the two elections. Note however that the use of the asymptotic $\chi^2$ distribution of the goodness-of-fit statistic (residual reviance) is suspect when there are cells with small expected frequencies. We print the estimated expected cell frequencies, called *fitted values* in SPlus, to check cell size and to compare the model fit with the observed data. Even if the use of the $\chi^2$ distribution in inadvisable in calculating the p-value, the residual deviance itself can still be used as a device for model comparison.

```
round(matrix(fitted(elect.glm2), ncol = 6, dimnames = list(elect.chan$before[1:6],
elect.chan$before[1:6])), 2)

       PS     PS     PS     PS    PS    PS
PS 281.00  12.71   7.68 17.95  3.74  4.92
PS  13.29 164.00  12.88  3.29  0.76  5.78
PS   6.32  10.12 121.00  7.06  0.71  0.79
```

```
PS   4.05   0.71   1.94 50.00  0.80  0.51
PS   1.26   0.24   0.29  1.20 14.00  0.00
PS   1.08   1.22   0.21  0.49  0.00 11.00
```

By examining the fitted values, we see that change between Socialist and Liberal is most probable and between Communist and blank ballot least (there are none). The fitted values compare favourably with the observed values given in Table 3.4. Also, there are many cells with small expected frequencies, rendering the p-values somewhat meaningless, as explained above.

The equilibrium model, or marginal homogeneity, is closely related to the previous two models. Suppose that the marginal totals are symmetric but the body of the table is not. The distribution of votes at the two elections is identical but the probability of shift between each pair of parties is not the same in both directions. Marginal homogeneity plus quasi-symmetry equals symmetry (Agresti [2], pp.359, 392).

$$\textit{Quasi-Symmetry} + \textit{Marginal Homogeneity} = \textit{Symmetry}$$

It follows that when quasi-symmetry holds, marginal homogeneity is equivalent to symmetry. Therefore, we can test the adequacy of the marginal homogeneity model by subtracting the residual deviance of the quasi-symmetry model from the residual deviance of the symmetry model. If the marginal homogeneity model holds given quasi-symmetry then this statistic has a $\chi^2$ distribution with $I-1$ degrees of freedom where $I$ is the dimension of the contingency table. However, note that since the symmetry model was rejected, and the quasi-symmetery model accepted, our test should also reject the addition of marginal homogeneity; that is, quasi-symmetry and marginal homogeneity together imply symmetry, but we have already rejected symmetry in a separate test.

The residual deviance from our symmetry model is 33.054, and the residual deviance from the quasi-symmetry model is 10.101, so our test statistic for marginal homogeneity in the six by six contingency table is 22.953 with 5 degrees of freedom. This yields a p-value of 0.0003446217, so the null hypothesis of marginal homogeneity is decisively rejected.

Standard log-linear models cannot be used to fit a marginal homogeneity model. However, Agresti [2] (p.501) has published a technique which allows you to fit a marginal homogeneity model using Poisson error and the *identity link*.

## 3.5 Random Walks

The model for a random walk on a line allows successive moves in either direction with probabilities $p$ and $q$ respectively ($p+q<1$ if the possibility of no move is allowed). This may be reformulated as a Markov chain where different points on the line are different states of the process. If the line is unbounded, the Markov chain will have an infinite number of states. We shall be concerned with random walks on a finite line segment with reflecting boundaries, so that the finite number of

| | October | | |
|---|---|---|---|
| **February** | Conservative | Liberal | Labour |
| Conservative | 170 | 20 | 3 |
| Liberal | 22 | 70 | 28 |
| Labour | 6 | 12 | 227 |

Table 3.5: Changes in vote between the two British elections of 1974. (Fingleton, 1984, p.131)

states corresponds to the categories of the contingency table. Random walk models yield diagonal symmetry in the square table of transition probabilities.

We apply this model to the two 1974 British elections (Table 3.5). To do this, we must assume that the parties may be ordered and that changing vote by one step in either direction (i.e. $p = q$) on the scale has a different probability than that for two steps, and so on, for greater distances. The factor variable we will use to model the symmetric nature of the random walk is:

```
> elect.britain.sym_matrix(0, nrow = 3, ncol = 3)
> elect.britain.sym_abs(row(elect.britain.sym) - col(elect.britain.sym)) + 1
> print(elect.britain.sym)

     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    1    2
[3,]    3    2    1
```

The model defined by this factor and by fixing the row and column totals of the estimates is the *symmetric minor diagonal model.*

```
> elect.britain.sym_factor(as.vector(elect.britain.sym))
> elect.britain_data.frame(scan("data/Chpt2-5", what = list(counts=0, before = "", after = ""), skip
= 1))
> summary(glm(counts ~ elect.britain.sym + before + after, family = Poisson(), data = elect.britain))

Coefficients:
                      Value Std. Error      t value
      (Intercept)  5.14820867 0.07517775  68.48048133
elect.britain.sym2 -1.77335871 0.13054230 -13.58455242
elect.britain.sym3 -3.78449206 0.33697959 -11.23062679
      beforeLabour  0.24957850 0.18967814   1.31580003
     beforeLiberal -0.61118400 0.16549086  -3.69315868
       afterLabour  0.01776692 0.19342930   0.09185229
```

```
      afterLiberal -0.28852943 0.16631784  -1.73480751

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 728.1753 on 8 degrees of freedom
Residual Deviance: 4.048641 on 2 degrees of freedom
```

This model fits the data very well. The parameter estimates of elect.britain.sym (0.000, -1.773, -3.784) indicate that the probability of changing vote decreases steeply with distance between the parties, in the order presented in the table, as would be expected from the random walk model.

If the equal probability random walk model did not fit, we could relax it and return to the original model ($p \neq q$) taking steps with different probabilities in each direction, the asymmetric minor diagonal model, with a new factor variable.

```
> elect.britain.adl_matrix(0, nrow = 3, ncol = 3)
> elect.britain.adl_row(elect.britain.adl) - col(elect.britain.adl) + 3
> print(elect.britain.adl)

     [,1] [,2] [,3]
[1,]    3    2    1
[2,]    4    3    2
[3,]    5    4    3
```

As expected, for these data, the model fits well, since the preceding one did.

```
> elect.britain.adl_factor(as.vector(elect.britain.adl))
> summary(glm(counts ~ elect.britain.adl + before + after, family = Poisson(), data = elect.britain))

Coefficients: (1 not defined because of singularities)
                      Value Std. Error    t value
       (Intercept)  1.1837115  0.4846915  2.4421959
elect.britain.adl2  1.7857211  0.4799210  3.7208650
elect.britain.adl3  3.9667967  0.4955160  8.0053859
elect.britain.adl4  2.5794081  0.6317857  4.0827264
elect.britain.adl5  0.2596665  0.8846231  0.2935335
      beforeLabour -0.3447657  0.2791183 -1.2351954
     beforeLiberal -0.9020129  0.1411728 -6.3894266
       afterLabour  0.6080480  0.2612645  2.3273270
      afterLiberal         NA         NA         NA

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 728.1753 on 8 degrees of freedom
Residual Deviance: 1.403156 on 1 degrees of freedom
```

With only three parties, not all parameters can be estimated, hence the missing value for afterLiberal.

Another possibility is to combine the random walk or minor diagonals model with symmetry i.e. with reversibility and equilibrium. When this model fits, it indicates that we would have symmetry if it were not for the unequal probabilities of larger and smaller steps. In this case, a full factor variable is not necessary because of the effect of the symmetry variable in the model. It is sufficient to define an extra factor, elect.britain.md:

```
> elect.britain.md_matrix(0, nrow = 3, ncol = 3)
> elect.britain.md_factor(elect.britain.adl*as.vector(col(elect.britain.adl) > row(elect.britain.adl)) + 1)
> print(elect.britain.md)

     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    1    1    3
[3,]    1    1    1

> elect.britain.symmat_factor(as.vector(sym.matrix(3)))
> summary(glm(counts ~ elect.britain.symmat + elect.britain.md, family = Poisson(), data =
elect.britain))

Coefficients:
                            Value Std. Error     t value
          (Intercept)  3.20215136  0.1801176  17.7781121
elect.britain.symmat2 -2.10353907  0.6047939  -3.4781088
elect.britain.symmat3 -0.04879008  0.2208844  -0.2208851
elect.britain.symmat4  1.93364708  0.1957670   9.8772889
elect.britain.symmat5  1.04634388  0.2161668   4.8404473
elect.britain.symmat6  2.22279866  0.1919574  11.5796452
    elect.britain.md2  0.69314718  0.7071068   0.9802581
    elect.britain.md3 -0.34484023  0.2240989  -1.5387860

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 728.1753 on 8 degrees of freedom
Residual Deviance: 4.275911 on 1 degrees of freedom
```

As in Section 3.4 above, 'symmat' is the symmetry variable; here, 'md' is the new minor diagonal variable. The fit is reasonably good, but we retain the symmetric minor diagonals model, since it fits better and has fewer parameters.

| | | 1971 | | |
|---|---|---|---|---|
| **1966** | Central Clydesdale | Urban Lancashire and Yorkshire | West Midlands | Greater London |
| Central Clydesdale | 118 | 12 | 7 | 23 |
| Urban Lanc. & York. | 14 | 2127 | 86 | 130 |
| West Midlands | 8 | 69 | 2548 | 107 |
| Greater London | 12 | 110 | 88 | 7712 |

Table 3.6: Migrant behaviour in Britain between 1966 and 1971.(Fingleton, 1984, p.142)

## 3.6 The Mover-Stayer Model

A modified Markov chain is frequently applied in mobility studies. A *mobility table* is a square two-dimensional table with the same categorical variable observed at two points in time. A common example, used here, is migrant behaviour (Table 3.6). Others include mobility among social classes or professions and voting behaviour data, such as that in the previous sections.

We need to test if position at the second point in time depends on that at the first point in time. This is the standard model of independence. However, here the problem is that too many individuals *do not change position* between the two time points to make an accurate determination of independence. In other words, too many observations appear on the diagonal.

The simple solution is to eliminate these diagonal elements and test for quasi-independence. More theoretically, this approach assumes that the diagonal contains two types of individuals, the movers, who might have moved, but did not happen to in the observed time interval, and the stayers who never change. Hence, the name of the model: the *mover-stayer model.* The movers, then, form a Markov chain, but we have only directly observed those that actually moved in a given period; the others are inextricably mixed up with the stayers.

From Table 3.6 note that the geographical locations are ordered from north to south of Britain. We also observe the exceptionally high values on the diagonal. We set up the data. The first model fitted is the usual one for independence:

```
> elect.msdata_data.frame(scan("data/Chpt2-6", what = list(counts = 0, before = "", after = ""),
skip = 3))
> elect.glm_glm(counts ~ before + after, family = Poisson(), data = elect.msdata)
> anova(elect.glm, test = "Chisq")

Analysis of Deviance Table
Poisson model
Response: counts

Terms added sequentially (first to last)
```

```
       Df Deviance Resid. Df Resid. Dev Pr(Chi)
  NULL                    15   40740.63
before  3 10345.88        12   30394.75       0
 after  3 10510.63         9   19884.12       0
```

As would be expected, this model is definitely not acceptable. We compare the fitted and observed frequencies and note the very large underestimation of all diagonal cells.

```
> t(matrix(fitted(elect.glm), ncol = 4))
> t(matrix(elect.msdata$counts, ncol = 4))
```

```
       [,1]     [,2]     [,3]     [,4]          [,1] [,2] [,3] [,4]
[1,]  1.846   27.203   31.531   91.431    [1,]  118   14    8   12
[2,] 28.158  414.814  480.812 1394.214    [2,]   12 2127   69  110
[3,] 33.151  488.364  566.063 1641.419    [3,]    7   86 2548   88
[4,] 96.843 1426.619 1653.595 4794.942    [4,]   23  130  107 7712
```

To refit the data with a *quasi-independence model* we remove the diagonal elements by providing a weighting vector with zeros on the diagonal.

```
> elect.diag0_matrix(1, ncol = 4, nrow = 4)
> diag(elect.diag0)_0
> elect.diag0_as.vector(elect.diag0)
> elect.glm_update(elect.glm, weights = elect.diag0)
> anova(elect.glm)
```

```
Analysis of Deviance Table
Poisson model
Response: counts

Terms added sequentially (first to last)
       Df Deviance Resid. Df Resid. Dev
  NULL                    15   486.2092
before  7 167.1885         8   319.0207
 after  3 314.6541         5     4.3666
```

The mover-stayer model fits very well. For the movers, those individuals who tend to migrate, new place of residence does not depend on original residence, a rather surprising result, given the varying distances.

```
> round(t(matrix(fitted(elect.glm), ncol = 4)),2)
```

```
      [,1]   [,2]   [,3]   [,4]
[1,]  1.58  11.83   8.89  13.27
[2,] 12.75  95.18  71.50 106.73
[3,] 10.75  80.25  60.28  89.99
[4,] 18.48 137.91 103.60 154.64
```

The fitted values for the diagonal (1.6, 95.2, 60.3, 154.6) are estimates of the numbers in each category who did not happen to move in the period under observation. The number of stayers is obtained by subtracting these values from the observed diagonal values (116.4, 2031.8, 2487.7, 7557.4). We see that 96.2% (12193.3/13171) of the population is estimated as not being susceptible to migration. The most important conclusions concern the small proportion of movers in the population and the independence of arrival point from origin for these movers.

# Chapter 4

# Point and Renewal Processes

## 4.1 Point Processes

In the previous chapter, we saw that one way to analyse a series of events in time is to condition on what events happened before. Each event occurs in a given constant period of time, assumed small enough so that only one event, however defined, can occur in each period. Series of events in time can also be studied in additional ways.

If we group the data into longer periods, we can look at the *rate* of occurrence of events. However, when we group data we lose information on the actual times between successive events, the *inter-arrival times*. Often, the inter-arrival times can be assumed to be independently and identically distributed. In such a case, we have a *renewal process*. The term "renewal process" comes from industry, where certain machines or parts must be replaced or renewed at varying intervals of time, but models for renewal processes have much wider application.

If the inter-arrival times are independently and identically distributed, then the rate of occurrence of events must be constant over time. If the event rate were changing, the inter-arrival times would not be independent, but would depend on time. Thus, before analysing our data as a renewal process we must verify that no time trend exists in the data.

As a concrete example, we consider the times in seconds between vehicles passing a point on a road, given in Table 4.1. One of the easiest ways to look for changes in trend is to plot the cumulated number of events against time. We read the data and make the plot.

```
> point.data_scan("data/Chpt3-1")
> plot(cumsum(point.data) / mean(point.data), seq(point.data), xlab = "Cumulative Sums
(Scaled)", ylab = "Data Index")
> abline(0,1)
```

The cumulative sums in Figure 4.1 are scaled so that a straight line with slope 1 can be used to assess the linearity of the data. The abline() function is used to draw in the comparison line with

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2.8 | 3.4 | 1.4 | 14.5 | 1.9 | 2.8 | 2.3 | 15.3 | 1.8 | 9.5 | 2.5 | 9.4 | 1.1 | 88.6 | 1.6 |
| 1.9 | 1.5 | 22.7 | 2.6 | 12.9 | 16.2 | 1.9 | 20.3 | 36.8 | 40.1 | 70.5 | 2.0 | 8.0 | 2.1 | 3.2 |
| 1.7 | 56.5 | 23.7 | 2.4 | 21.4 | 5.1 | 7.9 | 20.1 | 14.9 | 5.6 | 51.7 | 87.1 | 1.2 | 2.7 | 1.0 |
| 1.5 | 1.3 | 24.7 | 72.6 | 119.8 | 1.2 | 6.9 | 3.9 | 1.6 | 3.0 | 1.8 | 44.8 | 5.0 | 3.9 | 125.3 |
| 22.8 | 1.9 | 15.9 | 6.0 | 20.6 | 12.9 | 3.9 | 13.0 | 6.9 | 2.5 | 12.3 | 5.7 | 11.3 | 2.5 | 1.6 |
| 7.6 | 2.3 | 6.1 | 2.1 | 34.7 | 15.4 | 4.6 | 55.7 | 2.2 | 6.0 | 1.8 | 1.9 | 1.8 | 42.0 | 9.3 |
| 91.7 | 2.4 | 30.6 | 1.2 | 8.8 | 6.6 | 49.8 | 58.1 | 1.9 | 2.9 | 0.5 | 1.2 | 31.0 | 11.9 | 0.8 |
| 1.2 | 0.8 | 4.7 | 8.3 | 7.3 | 8.8 | 1.8 | 3.1 | 0.8 | 34.1 | 3.0 | 2.6 | 3.7 | 41.3 | 29.7 |
| 17.6 | 1.9 | 13.8 | 40.2 | 10.1 | 11.9 | 11.0 | 0.2 | | | | | | | |

Table 4.1: Times in seconds between vehicles passing a point on the road. (Cox and Lewis, 1966, p. 197)

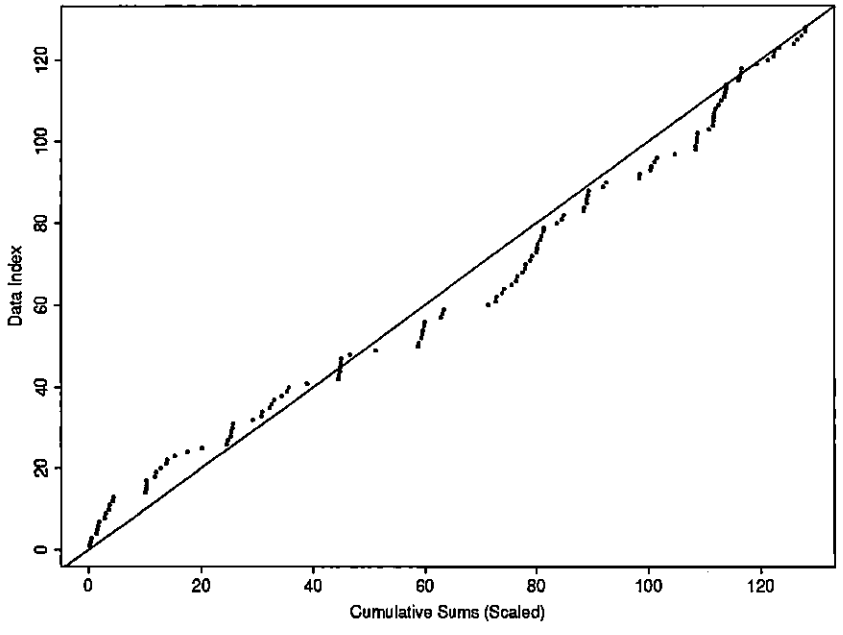



Figure 4.1: Cumulative sums plot of traffic data.

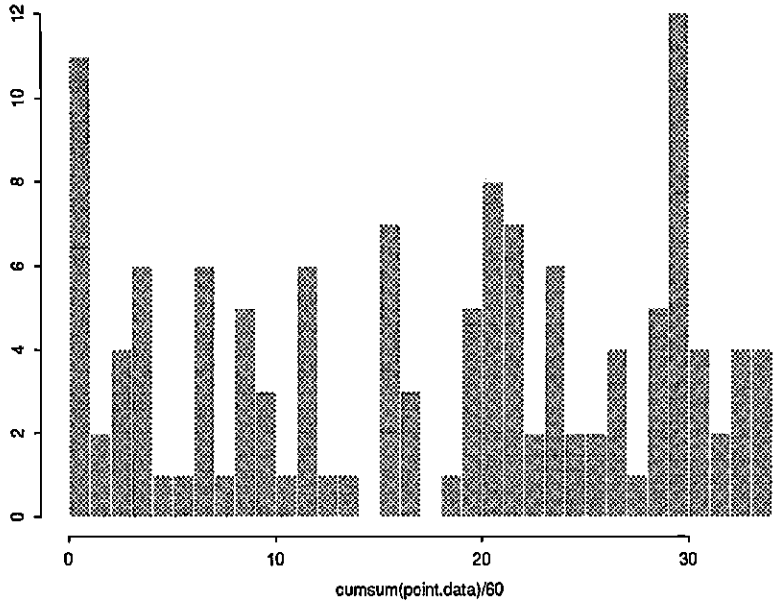



Figure 4.2: Bar chart of numbers of cars passing a point in successive 60 second intervals.

intercept 0 and slope 1. The plotted line is reasonably straight, showing no indication of a change in the rate of events.

Since we do not yet know what form the distribution of these inter-arrival times might take, we group them into 60 second time periods before testing for a trend. Although this appears similar to the procedure that must have been used to obtain the data of Tables 3.1 and 3.2, here we group them more coarsely, avoiding creating periods with zero frequency.

For the table, we must calculate how many vehicles pass in each successive minute. For example, in the first minute, 11 vehicles go by. The process of enumeration can by automated using the tsfr() function, given in Section 9.2, which produces a bar chart of the observed frequencies (Figure 4.2) per successive 60 second intervals.

```
> point.hist_tsfr(point.data, 60)
```

We set up the data frame to contain the frequencies (counts), linear (t) and quadratic (t2) terms in time (in minutes). Then we fit log-linear models for the counts, where

$$\log E(\mathsf{counts}) = \beta_0 + \beta_1 \mathsf{t}$$

and

$$\log E(\mathsf{counts}) = \beta_0 + \beta_1 \mathsf{t} + \beta_2 \mathsf{t2}$$

We first fit the log-linear model for a linear time trend.

```
> point.dataframe_data.frame(counts = point.hist$counts, t = point.hist$breaks[-1], t2 =
point.hist$breaks[-1] ^ 2)
> summary(glm(counts ~ t, family = poisson(), data = point.dataframe))

Coefficients:
                          Value  Std. Error   t value
            (Intercept) 1.253832733 0.183773904 6.8226920
                      t 0.004059685 0.009009701 0.4505904

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 77.56221 on 33 degrees of freedom
Residual Deviance: 77.35925 on 32 degrees of freedom

> summary(glm(counts ~ t + t2, family = poisson(), data = point.dataframe))

Coefficients:
                     Value   Std. Error   t value
    (Intercept)  1.667013342 0.2617852481  6.367866
              t -0.066236759 0.0356694914 -1.856958
             t2  0.001992426 0.0009836318  2.025581

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 77.56221 on 33 degrees of freedom
Residual Deviance: 73.36373 on 31 degrees of freedom
```

Neither model fits well, since both have large residual deviances compared with the residual degrees of freedom. There is no evidence of a linear trend in these data, since the change in deviance when the linear time term was added ($77.562 - 77.359 = 0.203$ with 1 df) is not significant. There is only slight evidence of a quadratic trend. We conclude that the rate of cars passing by is constant over time, and that our process is therefore homogeneous.

## 4.2 The Poisson Process

If the distribution of the inter-interval times in a renewal process is exponential, something which we shall study directly in the next sections, then the process is called a *Poisson process* [8]. The conditions for a Poisson process with rate $\rho$ are:

1. the probability of an event in $(t, t + \Delta t)$ is $\rho\Delta t + o(\Delta t)$;
2. the occurrence of events in $(t, t + \Delta t)$ is independent of what happens before $t$;
3. the probability of more than one event in $(t, t + \Delta t)$ is $o(\Delta t)$.

Then the number of events in $(0,t)$, $N_t$, has a Poisson distribution with mean $\rho t$:

$$Pr(N_t = y) = \frac{(\rho t)^y e^{-\rho t}}{y!} \tag{4.1}$$

Using the data of the previous section, we investigate the adequacy of the Poisson distribution for modelling the frequencies of occurrence in one minute intervals (counts). To get a preliminary measure of the appropriateness of a Poisson distribution, we calculate the Poisson *index of dispersion*,

$$D = \frac{\sum(X_i - \bar{X})^2}{\bar{X}}$$

which has an asymptotic $\chi^2_{(n-1)}$ distribution.

```
sum((point.hist$counts - mean(point.hist$counts)) ^ 2)/mean(point.hist$counts)
```

```
[1] 77.0625
```

```
1-pchisq(77.0625,length(point.hist$counts)-1)
```

```
[1] 2.205192e-05
```

The data fail the index of dispersion test for the Poisson distribution, with a p-value near zero, but we continue to fit a Poisson distribution for illustrative purposes. We tabulate the distribution of the frequency of occurence.

```
> point.poisson_tabulate(point.hist$counts+1)
> print(point.poisson)
```

| $y_i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $F_i$ | 2 | 8 | 5 | 2 | 5 | 3 | 4 | 2 | 1 | 0 | 0 | 1 | 1 |

We added 1 to the counts vector because the tabulate() function only recognizes values greater than zero, and our vector of counts contained two zero elements.

Let $F_i$ be the number of one minute intervals which have $y_i$ cars passing a point. If the counts have a Poisson distribution, the log-linear model describing them will follow a regression which is the logarithm of Equation 4.1:

$$\log Pr(\text{counts} = y_i) = y_i \log \mu - \mu - \log y_i! \cong \log \frac{F_i}{N} \qquad i = 1, \ldots, k \tag{4.2}$$

Here $\frac{F_i}{N}$ estimates $Pr(\text{counts} = y_i)$, $\mu = \rho t$, and $N = \sum F_i$ is the total number of observations. The fitting of such distributions was discussed in section 2.5 for duration and survival data. The same procedure can be used here.

To assess the fit of the Poisson process to the counts vector we simply fit the linear term in frequency of occurrence variable, $F_i$, with offset $-\log y_i!$. We also fit a log-linear model with a quadratic term. Note that the use of a Poisson error distribution in SPlus has nothing to do with a test for a Poisson process; in SPlus, Poisson error is used for *all* multinomial log-linear models.

In calculating our offset, we use a roundabout method which avoids taking the factorial. Computing the factorial for large integers is computationally expensive, and risks over-running the registers of the computer.

$$-\log y! = -\log \prod_{i \in I} i = -\sum_{i \in I} \log i \qquad I = \{1, 1, 2, 3, 4, \ldots, y\}$$

```
> point.t_seq(point.poisson)-1
> point.off_c(1,point.t[-1])
> point.off_-cumsum(log(point.off))
> point.dataframe_data.frame(F = point.poisson, T = point.t, T2 = point.t ^ 2, offset = point.off)
> summary(glm(F ~ T + offset(offset), family = poisson(), data = point.dataframe))

Coefficients:
                Value Std. Error    t value
(Intercept) -0.2384504 0.37439105  -0.636902
          T  1.3257003 0.08839728  14.997071

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 177.194 on 14 degrees of freedom
Residual Deviance: 25.82187 on 13 degrees of freedom

> point.glm_glm(F ~ T + T2, family = poisson(), data = point.dataframe)
> summary(point.glm)

Coefficients:
                  Value Std. Error    t value
 (Intercept)  1.46479298  0.3594756  4.0748048
           T  0.08600530  0.1686249  0.5100392
          T2 -0.02734609  0.0162567 -1.6821433

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 35.81452 on 14 degrees of freedom
Residual Deviance: 12.05877 on 12 degrees of freedom
```

The linear model is rejected in favour of a quadratic one, showing that a Poisson process does not adequately describe these data. If the Poisson model had been acceptable, our estimate of the rate would have been $\hat{\mu} = e^{1.326} = 3.765$ cars per minute. We plot the observed and predicted values for our quadratic model, which assumes a truncated normal distribution (see Section 2.5).

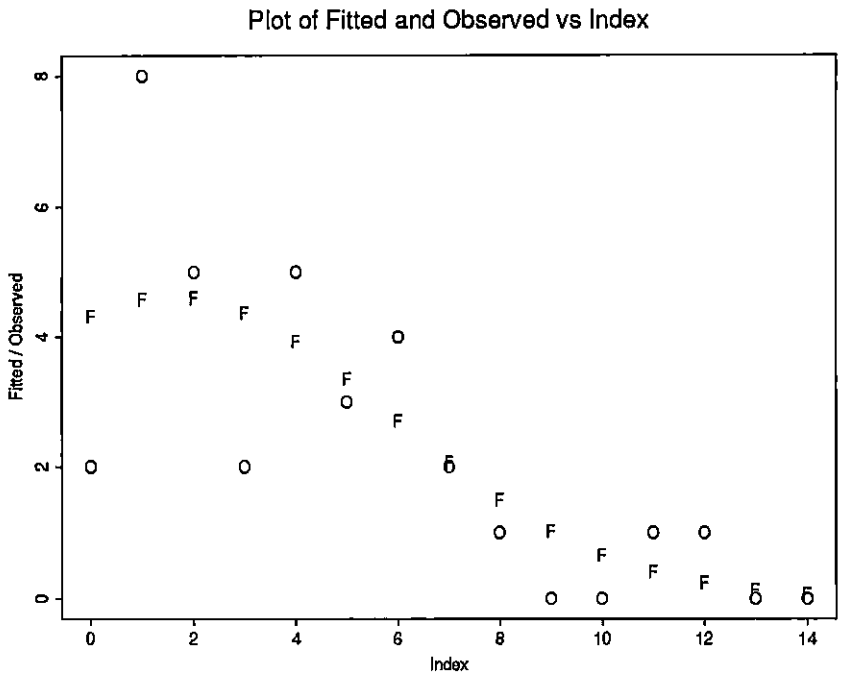



Figure 4.3: Plot of observed/fitted values for truncated normal model of traffic data.

```
> matplot(point.t, cbind(point.poisson, point.glm$fitted), pch="OF", main="Plot of Fitted and
Observed vs Index", xlab="Index", ylab="Fitted / Observed")
```

The plot (Figure 4.3) shows that periods with zero, one, and three events are especially poorly fitted. We conclude that the number of vehicles passing in a small period of time may depend on previous traffic density. This may quite possibly be due to traffic congestion, whereby a period with many vehicles is followed by another similar period since all vehicles are slowed down.

## 4.3 Kaplan-Meier Estimation

The study of inter-arrival times (times-to-event, inter-occurence times, or duration data), requires the introduction of a specific terminology [9]. Variables describing inter-arrival times have two characteristics which distinguish them from other traditional forms of statistical analysis.

1. All inter-arrival times must be non-negative.
2. For some individuals, *no event* may occur within the observation period.

Item two above is commonly known as *censoring*. It is of little or no importance for renewal processes, unless some of the time points are not exactly known (an exception is repairing machines

before breakdown), but is central to the study of survival distributions in the next chapter.

Suppose that the distribution of inter-arrival times has *probability density* $f(y)$. Then, the *cumulative probability function* is

$$F(y) = Pr(Y \leq y) = \int_0^y f(u)du \tag{4.3}$$

Similarly, the *survivor function* is

$$S(y) = Pr(Y > y) = 1 - F(y) \tag{4.4}$$

and the *hazard function* or *failure rate* is

$$h(y) = \frac{f(y)}{S(y)} \tag{4.5}$$

Any one of these three functions uniquely determines a probability distribution. The survivor function gives the probability of no event occurring before time $y$, while the hazard function gives the instantaneous rate of events occurring at time $y$ given that no event has occurred until just before time $y$.

In the analysis of duration data, the first task is to obtain an empirical estimate of the survivor function. This is provided by the *Kaplan-Meier estimates*, non-parametric estimates which assume no specific distributional form for the data. We use the surv.fit() function to obtain the Kaplan-Meier estimate of the survivor curve for the traffic data in Table 4.1.

The surv.fit(times, status=boolean) function requires two input parameters: a numerical vector of failure times, and a boolean vector of censoring status where true indicates an uncensored observation. In addition, surv.fit() accepts several optional parameters, of which strata is the most important. Using the strata parameter, we can analyze two sets of survival data simultaneously. We will use the strata feature in later sections of this chapter.

Because none of our inter-arrival times are censored, our status parameter is just a vector of ones, with the same length as our data. Because the survival curve is the default plot action for surv.fit, we can use plot() to plot the empirical survivor curve directly; however, we must construct the empirical hazard curve by hand.

The empirical hazard is an estimate of the probability of an event at time $y$ given that the event has not occurred just prior to time $y$. We calculate is as:

$$\hat{h}(y) = \frac{\#\text{ of events occurring at time } y}{\#\text{ of inter-arrival times greater than } y}$$

The denominator is commonly known as the number "at risk" at time $y$.

```
> point.kmsp_surv.fit(point.data, status = rep(1, length(point.data)))
> print(point.kmsp)
> plot(point.kmsp, main = "Empirical Survivor Curve")
> plot(point.kmsp$time, point.kmsp$n.event / point.kmsp$n.risk, type = "l", main = "Empirical
Hazard Curve")
```

```
95 percent confidence interval is of type "log"
 time n.risk n.event  survival     std.dev lower 95% CI upper 95% CI
  0.2    128       1 0.9921875 0.007781923    0.9770518    1.0000000
  0.5    127       1 0.9843750 0.010961887    0.9631229    1.0000000
  0.8    126       3 0.9609375 0.017124685    0.9279531    0.9950943
  1.0    123       1 0.9531250 0.018682741    0.9172020    0.9904550
  1.1    122       1 0.9453125 0.020096793    0.9067329    0.9855336
  1.2    121       5 0.9062500 0.025763488    0.8571355    0.9581788
    .                  .          .
    .                  .          .
    .                  .          .
 88.6      4       1 0.0234375 0.013372133  0.007660605   0.07170666
 91.7      3       1 0.0156250 0.010961887  0.003950485   0.06180016
119.8      2       1 0.0078125 0.007781923  0.001108971   0.05503767
125.3      1       1 0.0000000          NA           NA           NA
```

Since we only have one group of individuals under consideration, the strata parameter of surv.fit() is not necessary. As well, none of the observations are censored, so the status parameter of surv.fit() is just a vector of ones. The second column of the table gives the number still at risk after each successive length of time. The fourth column shows the estimates for the survivor curve, also plotted in the first graph, and the fifth column shows the standard error of these estimates. The original data table (Table 4.1) shows that there are a large number of small durations and a few very long intervals; this is reflected in the sharp descent of the survival curve (Figure 4.4).

The hazard function is vaguely U-shaped. It shows that as the time since the last car passed increases, so does the risk of another car going by. But there is also a higher risk of cars passing in very rapid succession.

## 4.4 Probability Plots

Once we have obtained an estimate for the empirical probability distribution, given by the Kaplan-Meier survivor curve, we usually wish to determine which theoretical probability distribution might fit the data. The best preliminary approach is to plot the data in a number of ways. Traditionally, this was done on various types of probability paper, corresponding to the different distributions. With a computer, it is now possible to do the plotting directly with little effort. In this way, the fit of more of the common distributions describing durations or survival can easily be inspected. One notable exception is the inverse Gaussian or Wald distributions which yield no simple form of plot. A more detailed description of the various distributions will be given in the next chapter.

The output of the surv.fit() function can be used to generate numerous probability plots automatically using the custom function plotsurv(). The custom function generates probability plots by transforming, $G(S(t))$, the empirical survivor curve so that it is linear in time, based on the theoretical survivor curve of the distribution being tested.

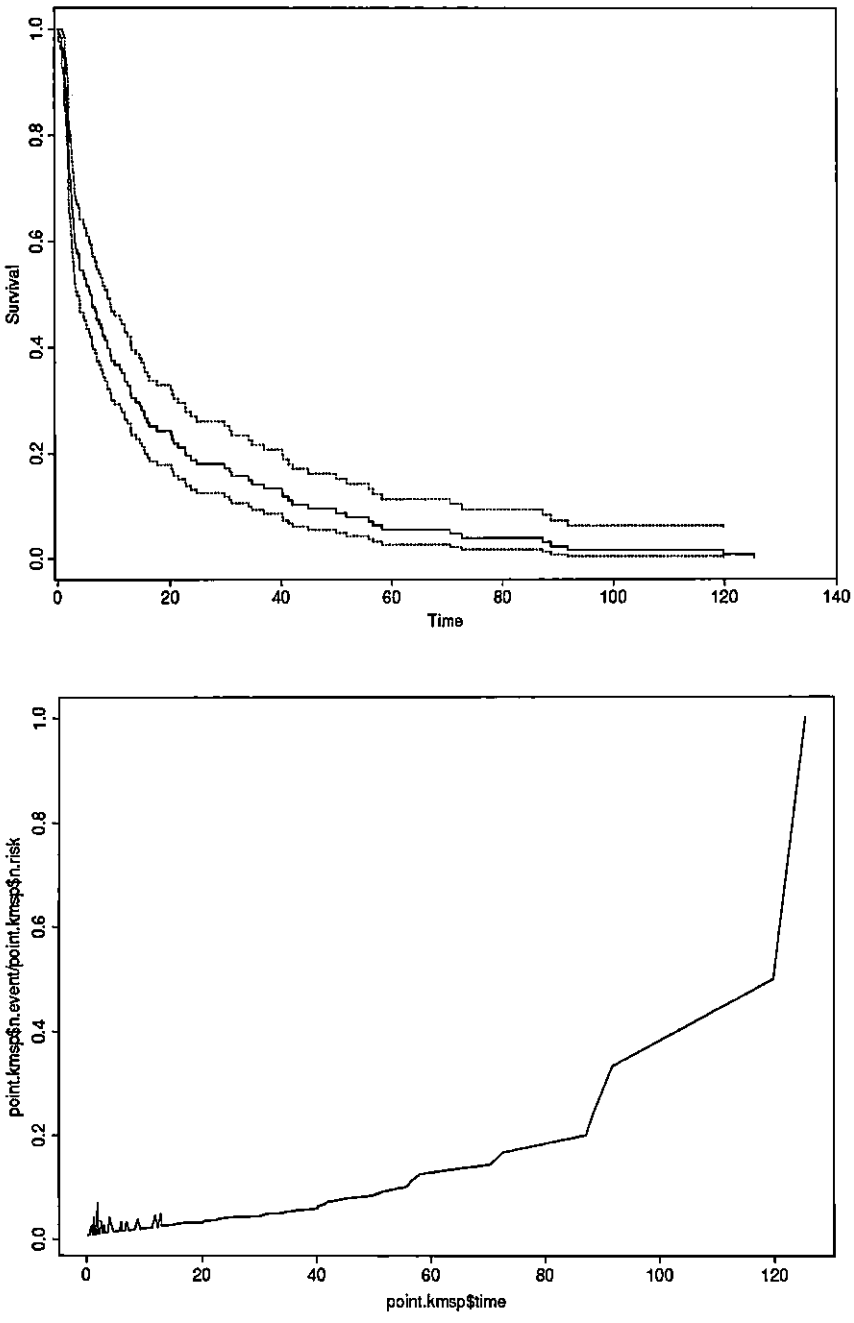



Figure 4.4: Empirical survivor and hazard plots for traffic data.

| Distribution | Survivor Function | Linear Relationship Between $G(S(t))$ and $t$ |
|---|---|---|
| Exponential | $S(t)=e^{-\lambda t}$ | $\log S(t)=-\lambda t$ |
| Pareto | $S(t)=c^a t^{-a}$ | $\log S(t)=a\log c - a\log t$ |
| Extreme Value | $S(t)=e^{-e^{(t-\mu)/\sigma}}$ | $\log(-\log S(t))=\frac{t-\mu}{\sigma}$ |
| Weibull | $S(t)=e^{-(\lambda t)^p}$ | $\log(-\log S(t))=p\log\lambda + p\log t$ |
| Normal | $S(t)=1-\Phi(\frac{t-\mu}{\sigma})$ | $\Phi^{-1}(1-S(t))=\frac{t-\mu}{\sigma}$ |
| Log-Normal | $S(t)=1-\Phi(\frac{\log t-\mu}{\sigma})$ | $\Phi^{-1}(1-S(t))=\frac{\log t}{\sigma}-\frac{\mu}{\sigma}$ |
| Logistic | $S(t)=\frac{1}{1+e^{\frac{t-\mu}{\sigma}}}$ | $\log(\frac{1-S(t)}{S(t)})=\frac{t-\mu}{\sigma}$ |
| Log-Logistic | $S(t)=\frac{e^{-\frac{\mu}{\sigma}}t^{\frac{1}{\sigma}}}{1+e^{-\frac{\mu}{\sigma}}t^{\frac{1}{\sigma}}}$ | $\log(\frac{1-S(t)}{S(t)})=-\frac{\mu}{\sigma}+\frac{1}{\sigma}\log t$ |

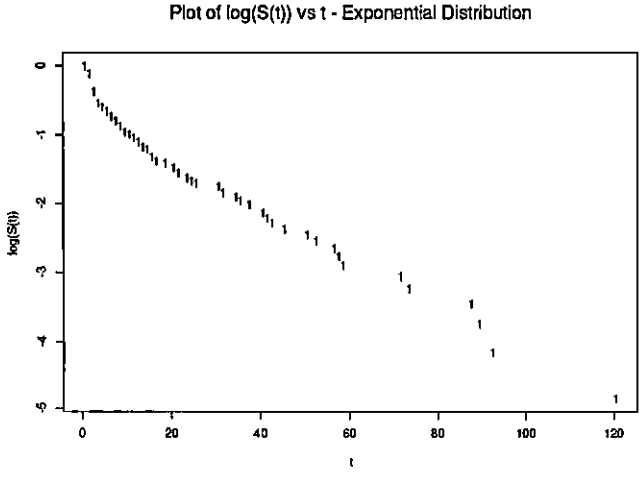
Table 4.2: Table of survivor functions and linear relationships between $S(t)$ and time for common probability distributions.

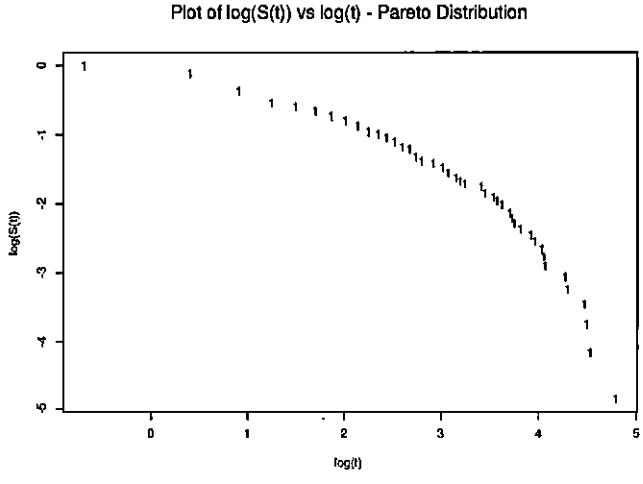
For example, the theoretical survivor curve of the exponential distribution is

$$S(t) = e^{-\lambda t} \quad \Rightarrow \quad \log S(t) = -\lambda t$$

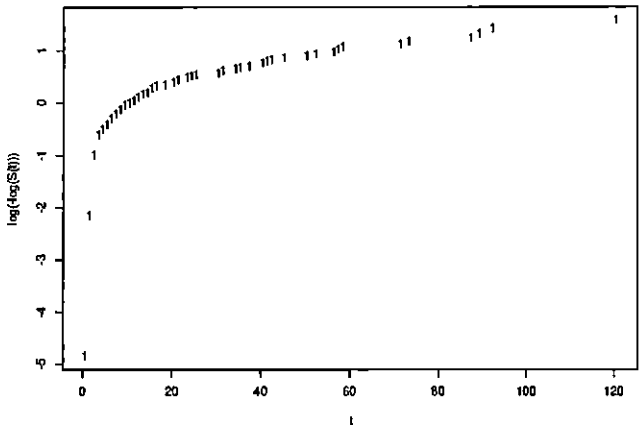
So, if our data are exponential, we would expect a plot of the log of the empirical survivor curve, $\log \hat{S}(t)$, versus time, $t$, would yield a straight line plot with slope $-\lambda$. Examples of other distributions, their survivor curves and transformed survivor functions are given in Table 4.2.

We continue with the analysis of Table 4.1, by plotting the results of the previous section with plotsurv(point.kmsp):

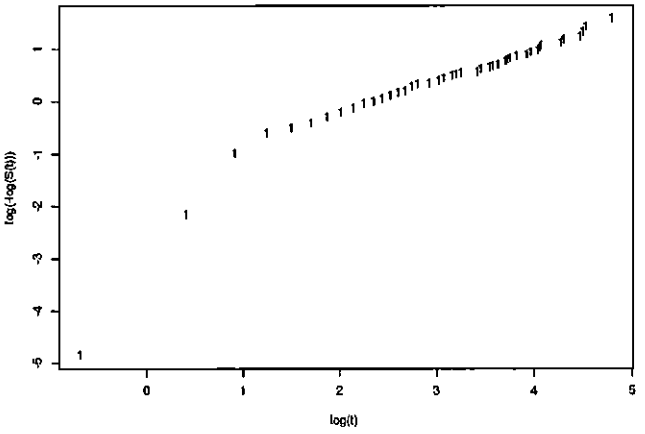


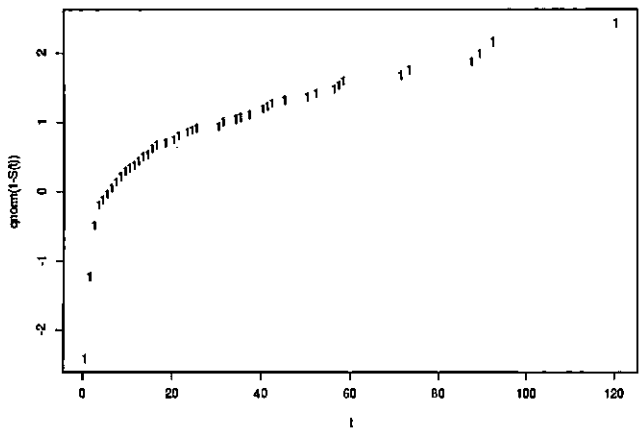
Plot of log(-log(S(t))) vs t - Extreme Value Distribution



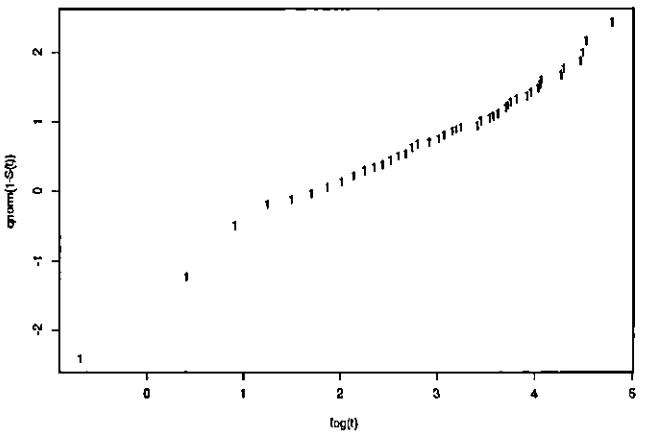

Plot of log(-log(S(t))) vs log(t) - Weibull Distribution



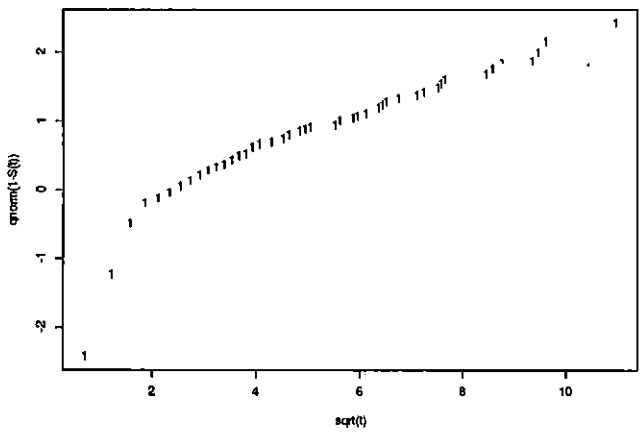

Plot of qnorm(1-S(t)) vs t - Normal Distribution



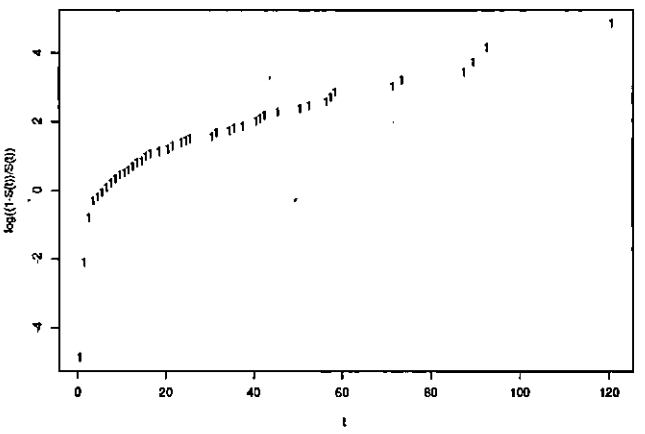

Plot of qnorm(1-S(t)) vs log(t) - Log Normal Distribution

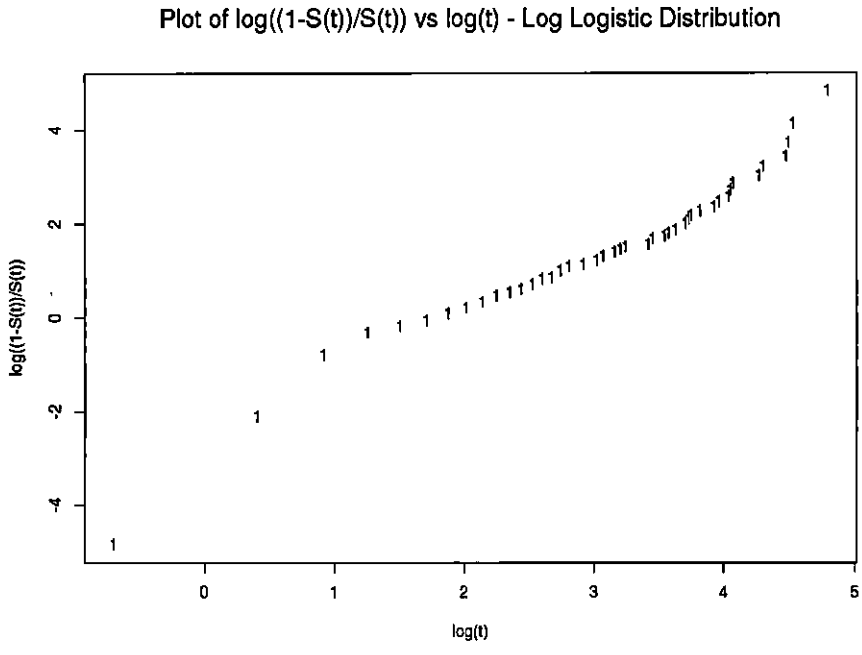



Plot of qnorm(1-S(t)) vs sqrt(t) - Gamma Distribution




Plot of log((1-S(t))/S(t)) vs t - Logistic Distribution






Of the various distributions, the log normal appears to have the straightest plot, followed by the log logistic. The exponential plot is curved for small time intervals, corresponding to the result previously obtained that a Poisson process is not applicable.

## 4.5 Fitting a Distribution

In Chapter 2, we briefly saw how to fit distributions in the exponential family to frequency data. We apply this method to our traffic data. Many distributions describing durations or survival are members of the exponential family. Of those for which probability plots were provided in the previous section, the Weibull, extreme value, and log-logistic distributions are not exponential, and must be fitted with iterative procedures.

We set up the model and fit five of the distributions (exponential, Pareto, normal, log-normal, gamma) listed in Section 2.5 using log-linear models.

```
> i_seq(150) - 0.5
> distrib.data_data.frame(counts = hist(point.data, breaks = seq(from = -0.5, to = 149.5, by = 1),
plot = F)$counts, i = i, i2 = i ^ 2, ri = 1/i, ri2 = 1/i ^ 2, lni = log(i), lni2 = log(i) ^ 2)
```

**Exponential**

```
> glm(counts ~ i, family=poisson(), data=distrib.data)
```

```
Coefficients:
 (Intercept)           i
    2.065044 -0.06159045

Degrees of Freedom: 150 Total; 148 Residual
Residual Deviance: 155.8708
```

### Pareto

```
> glm(counts ~ lni, family=poisson(), data=distrib.data)

Coefficients:
 (Intercept)         lni
    2.721323 -0.8949747

Degrees of Freedom: 150 Total; 148 Residual
Residual Deviance: 186.5983
```

### Normal

```
> glm(counts ~ i + i2, family=poisson(), data=distrib.data)

Coefficients:
 (Intercept)           i            i2
    2.406513 -0.1009787 0.0004685181

Degrees of Freedom: 150 Total; 147 Residual
Residual Deviance: 133.7991
```

### Log-Normal

```
> glm(counts ~ lni + lni2 + offset(-log(i)), family=poisson(), data=distrib.data)

Coefficients:
 (Intercept)       lni        lni2
     2.39437 1.276194 -0.3121826

Degrees of Freedom: 150 Total; 147 Residual
Residual Deviance: 115.3379
```

### Gamma

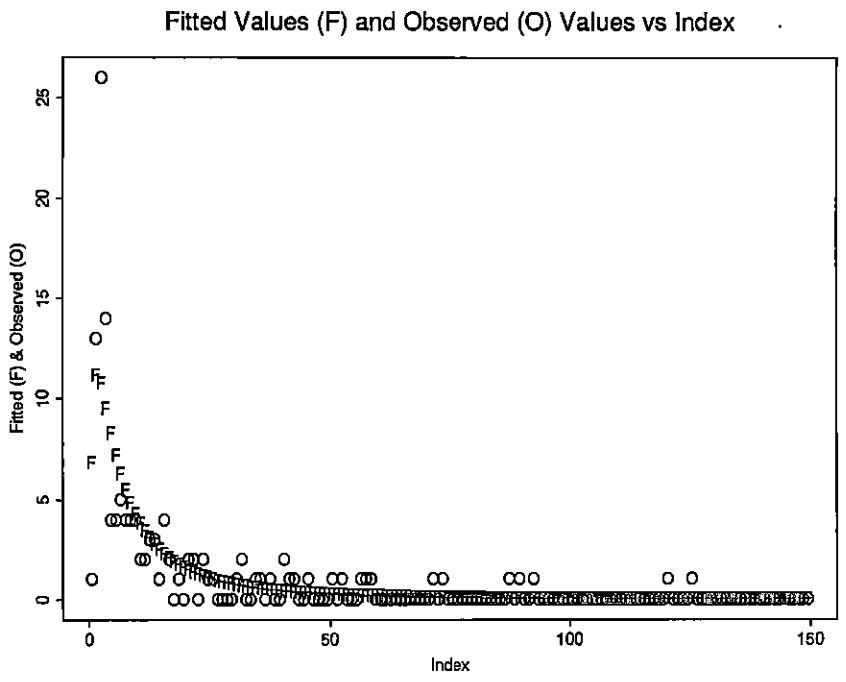



Figure 4.5: Plot of observed and fitted versus time for log-normal model of traffic data.

```
> glm(counts ~ i + lni, family=poisson(), data=distrib.data)

Coefficients:
 (Intercept)           i         lni
    2.515486 -0.03847874 -0.3898402

Degrees of Freedom: 150 Total; 147 Residual
Residual Deviance: 142.5613
```

The best fit is given by the log-normal distribution, as it has the smallest residual deviance. We refit the model, and then plot the resulting distribution (Figure 4.5).

```
> distrib.glm_glm(counts ~ lni2 + lni + offset(-log(i)), family = poisson(), data = distrib.data)
> matplot(distrib.data$i, cbind(distrib.glm$fitted, distrib.data$counts), pch="FO", main="Fitted
Values (F) and Observed (O) Values vs Index", ylab="Fitted (F) & Observed (O)", xlab="Index")
```

The log-normal distribution is the best fit among the common members of the exponential family. This confirms the results of our probability plots.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 152 | 123 | 2 | 124 | 12 | 4 | 10 | 216 | 80 | 12 | 33 | 66 |
| 232 | 826 | 40 | 12 | 29 | 190 | 97 | 65 | 186 | 23 | 92 | 197 |
| 431 | 16 | 154 | 95 | 25 | 19 | 78 | 202 | 36 | 110 | 276 | 16 |
| 88 | 225 | 53 | 17 | 538 | 187 | 34 | 101 | 41 | 139 | 42 | 1 |
| 250 | 80 | 3 | 324 | 56 | 31 | 96 | 70 | 41 | 93 | 24 | 91 |
| 143 | 16 | 27 | 144 | 45 | 6 | 208 | 29 | 112 | 43 | 193 | 134 |
| 420 | 95 | 125 | 34 | 127 | 218 | 2 | 0 | 378 | 36 | 15 | 31 |
| 215 | 11 | 137 | 4 | 15 | 72 | 96 | 124 | 50 | 120 | 203 | 176 |
| 55 | 93 | 59 | 315 | 59 | 61 | 1 | 13 | 189 | 345 | 20 | 81 |
| 286 | 114 | 108 | 188 | 233 | 28 | 22 | 61 | 78 | 99 | 326 | 275 |
| 54 | 217 | 113 | 32 | 388 | 151 | 361 | 312 | 354 | 307 | 275 | 78 |
| 17 | 1205 | 644 | 467 | 871 | 48 | 123 | 456 | 498 | 49 | 131 | 182 |
| 255 | 194 | 224 | 566 | 462 | 228 | 806 | 517 | 1643 | 54 | 326 | 1312 |
| 348 | 745 | 217 | 120 | 275 | 20 | 66 | 292 | 4 | 368 | 307 | 336 |
| 19 | 329 | 330 | 312 | 536 | 145 | 75 | 364 | 37 | 19 | 156 | 47 |
| 129 | 1630 | 29 | 217 | 7 | 18 | 1358 | 2366 | 952 | 632 | | |

Table 4.3: Days between successive coal-mining disasters (at least 10 deaths) in Great Britain, 15 March, 1851 - 22 March, 1962. (Jarrett, 1979)

## 4.6 A Nonhomogeneous Point Process

A non-homogeneous point process is a renewal process in which the rate of events is not constant over time. In this section we deal with non-homogeneous processes in which the rate is constant over segments of the process; that is, the process runs with rate $\lambda_1$ up until a certain point in time, at which it changes to rate $\lambda_2$.

With the methods of this chapter, the easiest way to accommodate changes in the rate of a time series is to cut it into several segments within which the rate does not vary. Then, the segments can be analysed simultaneously with the methods above, using, as a supplementary variable, a factor variable indicating the segments. If the rate change must be analysed as a continuous (linear or quadratic) trend, then the methods of the next chapter should be used.

As an example, we take the time in days between successive coal-mining disasters in Great Britain between 1851 and 1962, given in Table 4.3. After first reading the data, we construct the bar chart of the number of disasters in successive 400 day periods, as for the previous example.

```
> coal.count.data_tsfr(coal.data,400)

 [1] 3 8 2 0 4 3 3 1 6 4 3 3 0 5 4 2 7 5 5
[20] 3 1 4 4 4 6 4 3 2 6 2 2 3 5 2 2 3 2 1
```

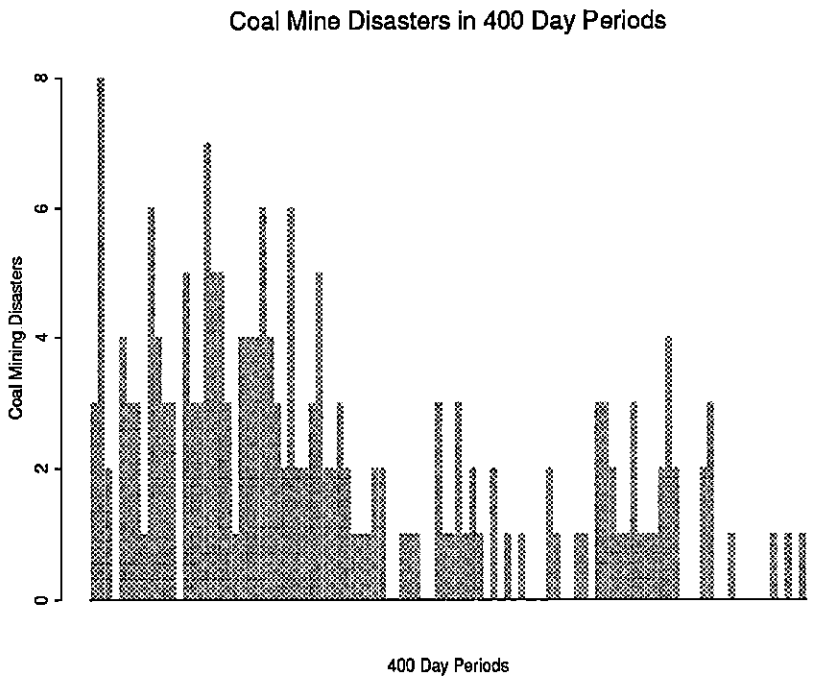



Figure 4.6: Bar chart of the number of British coal disasters in successive 400 day intervals.

```
[39] 1 1 2 2 0 0 1 1 1 0 0 3 1 1 3 1 2 1 0
[58] 2 0 1 0 1 0 0 0 2 1 0 0 1 1 0 3 3 2 1
[77] 1 3 1 1 1 2 4 2 0 0 0 2 3 0 0 1 0 0 0
[96] 0 0 1 0 1 0 1
```

We immediately see that the frequency of coal mine disasters decreases with time (Figure 4.6). There appear to be considerably more in the first 36 time periods than later, i.e. a change after about $36 \times 400 = 14400$ days.

We plot the cumulated number of events against time. The graph provides an indication of a change in frequency of the events over time.

```
> plot(cumsum(coal.count.data$counts),main="Cumulated Events Over Time", xlab="Time",
ylab="Events")
```

The graph (Figure 4.7) appears to be constructed of two reasonably straight lines. In other words, the frequency of events appears to change after about the 125th event or after about 15000 days. This is similar to the break point found in the barplot above. We fit linear and quadratic trends, and also a segmented model, distinguishing the first 36 periods.

```
> coal.time_seq(coal.count.data$counts)
> summary(glm(coal.count.data$counts ~ coal.time, family=poisson()))
```

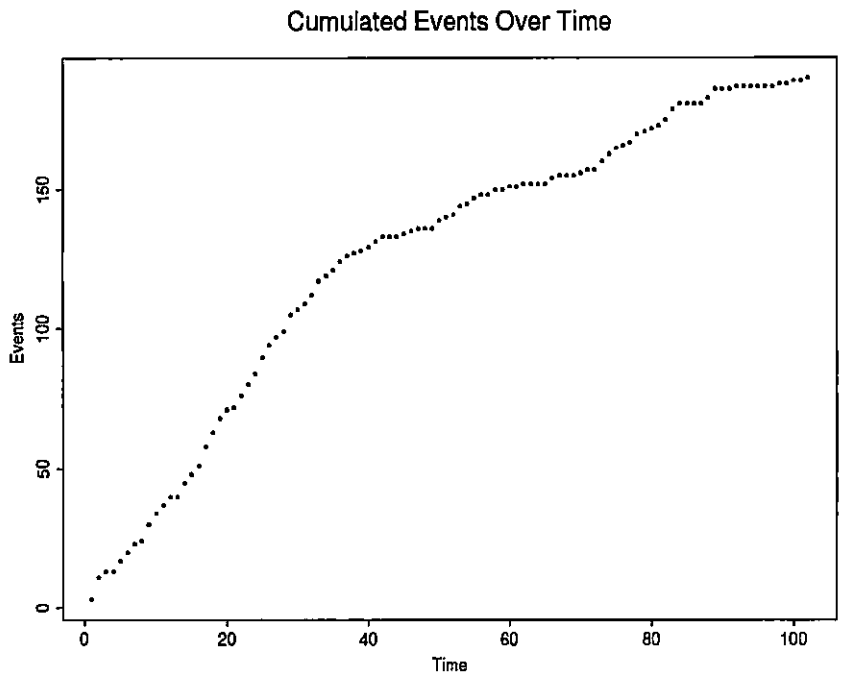



Figure 4.7: Cumulated number of coal mine disasters over time.

```
Coefficients:
                Value  Std. Error    t value
(Intercept)  1.48554094 0.120139246  12.365160
  coal.time -0.02003481 0.002716396  -7.375512

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 187.7471 on 101 degrees of freedom
Residual Deviance: 127.8476 on 100 degrees of freedom
```

```
> summary(glm(coal.count.data$counts ~ coal.time + coal.time ^ 2, family=poisson()))
```

```
Coefficients:
                   Value   Std. Error     t value
(Intercept)  1.483861e+00 0.1702760106  8.71444476
  coal.time -1.990845e-02 0.0094754131 -2.10106446
coal.time^2 -1.418487e-06 0.0001019559 -0.01391276

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 187.7471 on 101 degrees of freedom
Residual Deviance: 127.8474 on 99 degrees of freedom
```

```
> coal.split_factor(coal.time>36)
```

```
> summary(glm(coal.count.data$counts ~ coal.split, family = poisson()))

Coefficients:
               Value Std. Error    t value
(Intercept)  1.236763  0.0898005  13.772336
 coal.split -1.236760  0.1522551  -8.122949

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 187.0675 on 101 degrees of freedom
Residual Deviance: 116.7299 on 100 degrees of freedom
```

There is evidence for a linear trend but not a quadratic effect over time. However, the simple cut after 36 periods provides the only acceptably fitting model and has the same number of parameters as the linear trend model.

We attempt to fit a Poisson process to each of the two segments, using the method of Section 4.2. We create the appropriate table and fit the models for the complete series and for the two separate segments.

```
> coal.split.counts_data.frame(counts = c(hist(coal.count.data$counts[coal.split == F], breaks =
seq(0, 15) - 0.5, plot = F)$counts, hist(coal.count.data$counts[coal.split == T], breaks = seq(0, 15)
- 0.5, plot = F)$counts), split = factor(c(rep(1, 15), rep(2, 15))), index = c(1:15, 1:15) - 0.5)
```

| | counts | split | index |
|---|---|---|---|
| 1 | 2 | 1 | 0.5 |
| 2 | 2 | 1 | 1.5 |
| 3 | 6 | 1 | 2.5 |
| 4 | 11 | 1 | 3.5 |
| 5 | 6 | 1 | 4.5 |
| 6 | 4 | 1 | 5.5 |
| 7 | 3 | 1 | 6.5 |
| 8 | 1 | 1 | 7.5 |
| 9 | 1 | 1 | 8.5 |
| 10 | 0 | 1 | 9.5 |
| 11 | 0 | 1 | 10.5 |
| 12 | 0 | 1 | 11.5 |
| 13 | 0 | 1 | 12.5 |
| 14 | 0 | 1 | 13.5 |
| 15 | 0 | 1 | 14.5 |
| 16 | 25 | 2 | 0.5 |
| 17 | 24 | 2 | 1.5 |
| 18 | 10 | 2 | 2.5 |
| 19 | 6 | 2 | 3.5 |

```
20      1     2   4.5
21      0     2   5.5
22      0     2   6.5
23      0     2   7.5
24      0     2   8.5
25      0     2   9.5
26      0     2  10.5
27      0     2  11.5
28      0     2  12.5
29      0     2  13.5
30      0     2  14.5
```

Since the offset is based on the time index ($-\log y_k!$), it must be constructed of two identical halves, then concatenated together.

```
> coal.offset_-cumsum(log(c(1, 1:14)))
> coal.offset_c(coal.offset, coal.offset)
> summary(glm(counts ~ split + offset(coal.offset), family = poisson(), data = coal.split.counts))

Coefficients:
               Value Std. Error  t value
(Intercept) 2.5835586  0.1659252 15.57062
      split 0.6060962  0.2065979  2.93370

(Dispersion Parameter for Poisson family taken to be 1 )
    Null Deviance: 145.5621 on 29 degrees of freedom
Residual Deviance: 136.6068 on 28 degrees of freedom

> summary(glm(counts ~ index * split + offset(coal.offset), family = poisson(), data =
coal.split.counts))

Coefficients:
                Value Std. Error    t value
(Intercept) -0.4793511 0.39149297  -1.224418
      index  1.2367747 0.08980839  13.771260
      split  3.6690057 0.45000430   8.153268
index:split -1.2367747 0.15234886  -8.118044

(Dispersion Parameter for Poisson family taken to be 1 )
    Null Deviance: 145.5621 on 29 degrees of freedom
Residual Deviance: 5.8894 on 26 degrees of freedom
```

We see that the hypothesis of a Poisson process is acceptable (residual deviance of 5.8894 on 26 degrees of freedom, p-value of 0.9999867), but that a different distribution is necessary for each

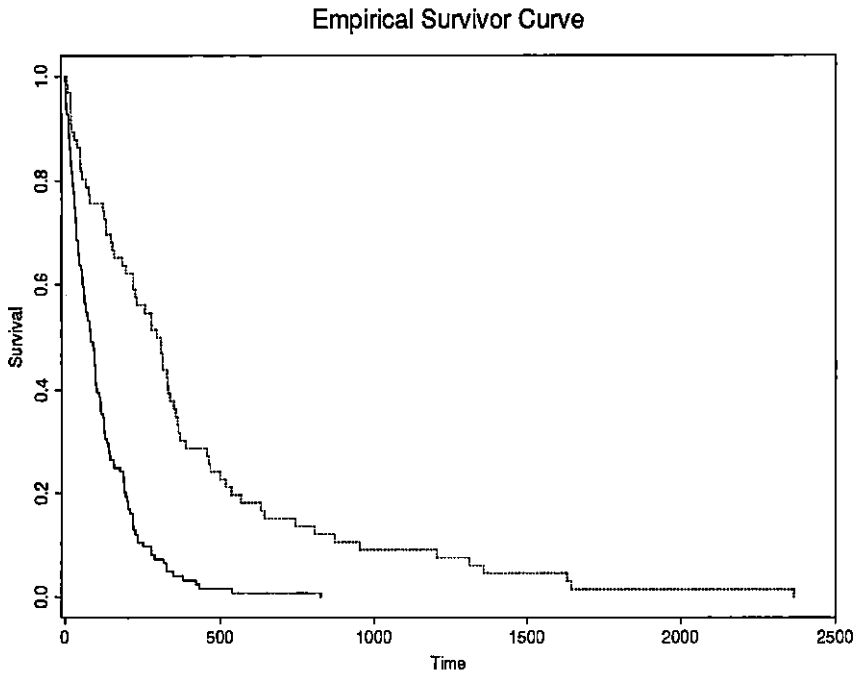



Figure 4.8: Empirical survivor curves of coal mine disaster data, split into two time periods.

of the two time segments. Thus, while we do not have a renewal process for the complete series, it is possible to fit one for each segment.

We consider modelling the distribution of the original data, days between successive coal mining disasters, for each of the two segments separately. Before we fit the various renewal distributions to the split data, we use the surv.fit() function to calculate the empirical survivor functions before and after the rate change after event number 124.

```
> coal.surv_surv.fit(coal.data, status = rep(1,length(coal.data)), strata = factor(c(rep(1, 124), rep(2,
length(coal.data) - 124))))
> plot(coal.surv, lty=c(1, 2), main = "Empirical Survivor Curve")
```

We see that the two survivor curves in Figure 4.8 are very different. The lower curve (solid line), with shorter intervals between disasters, is that for the first segment of about forty years, when accidents were more frequent.

The estimated hazard functions plotted in Figure 4.9 are also very distinct and indicate increasing risk of disaster as the time without a disaster increases. This is not characteristic of a Poisson process.

```
> plot(coal.surv$time, coal.surv$n.event / coal.surv$n.risk, type="n", main = "Empirical Hazard
Function")
> text(coal.surv$time, coal.surv$n.event / coal.surv$n.risk, coal.surv$strata)
```

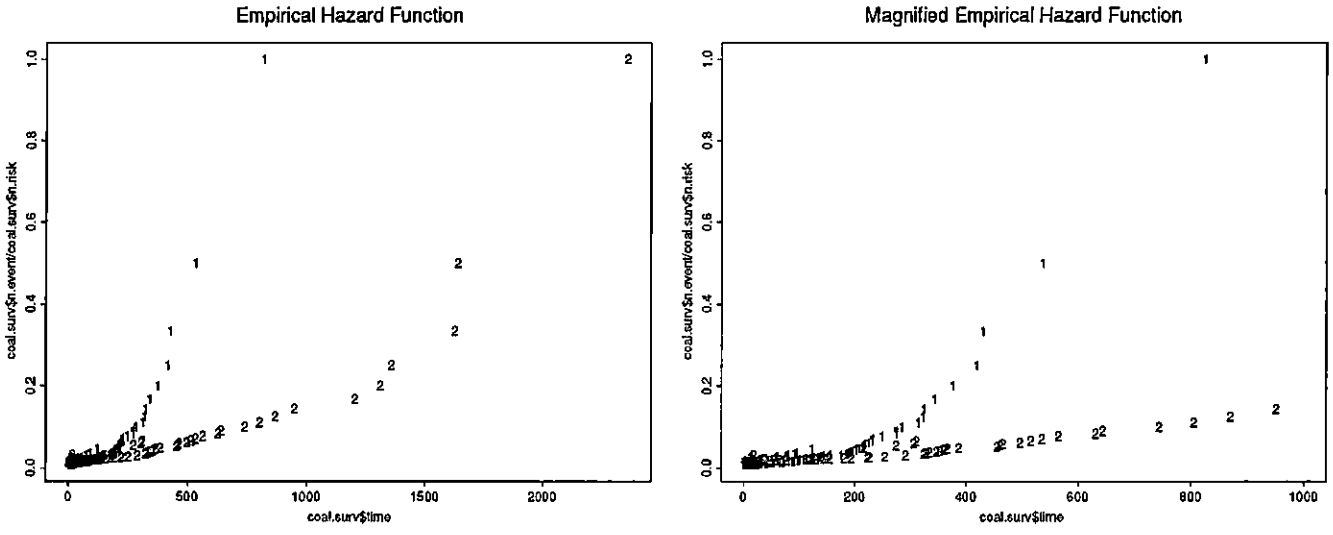



Figure 4.9: Empirical hazard functions for coal mine disaster data. Regular and magnified plots.

Most of the probability plots, with the exception of the Pareto, extreme value, and logistic distributions, provide reasonably straight lines for the two segments. Only that for the exponential distribution is presented in Figure 4.10, as a representative plot.

```
> plotsurv(coal.surv)
```

We fit various members of the exponential family as log-linear models. (Only the results for the exponential, normal, and gamma distributions are presented.) We will fit the distributions as log-linear models, as in Section 2.5, so we must first transform our time interval data into count data by taking a histogram over fixed interval widths, in this case 20 days.

```
> coal.counts_c(hist(coal.data[1:124], breaks = seq(from = 0, to = 2500, by = 20), plot = F)$counts,
hist(coal.data[125:length(coal.data)], breaks = seq(from = 0, to = 2500, by = 20), plot = F)$counts)
> coal.counts.split_factor(c(rep(1, length(coal.counts)/2), rep(2, length(coal.counts)/2)))
> coal.time_rep(seq(from = 10, to = 2490, by = 20),2)
> coal.dataframe_data.frame(counts = coal.counts, time = coal.time, split = coal.counts.split, time2
= coal.time ^ 2, timer = 1/coal.time, timer2 = 1/coal.time ^ 2, timeln = log(coal.time), timeln2 =
log(coal.time) ^ 2)
```

**Exponential**

```
> summary(glm(counts ~ time * split, poisson(), data = coal.dataframe))

Coefficients:
                    Value    Std. Error     t value
```

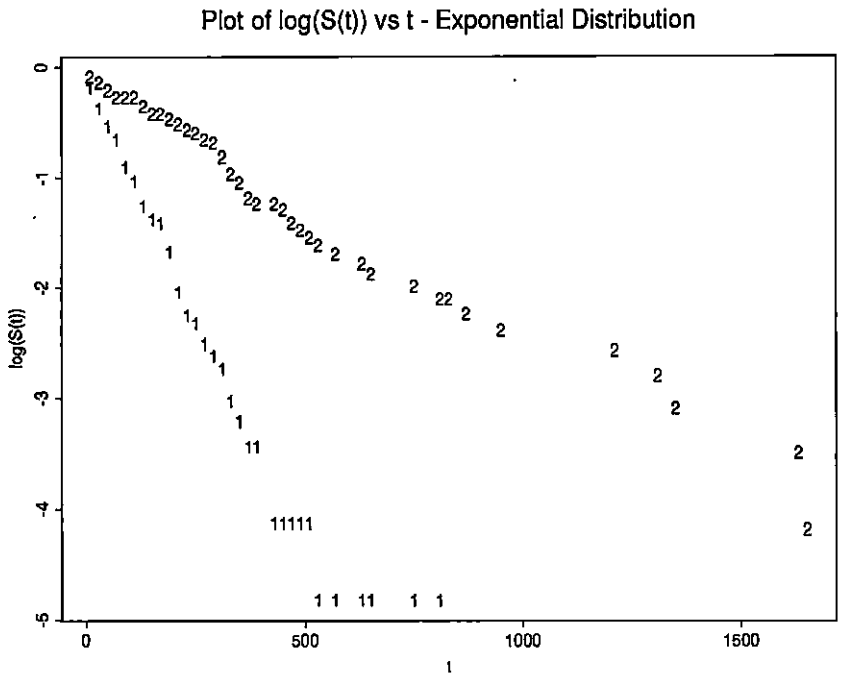



Figure 4.10: Probability plot of empirical survival estimates against exponential distribution.

```
(Intercept)  3.081968102 0.1272418397  24.221342
       time -0.008779550 0.0007893175 -11.122964
      split -1.893024407 0.2176605873  -8.697139
 time:split  0.006297241 0.0008510622   7.399272

(Dispersion Parameter for Poisson family taken to be 1 )
    Null Deviance: 754.6903 on 249 degrees of freedom
Residual Deviance: 108.1434 on 246 degrees of freedom
```

### Normal

```
> summary(glm(counts ~ (time + time2) * split, poisson(), data = coal.dataframe))
```

```
Coefficients:
                   Value   Std. Error     t value
(Intercept)  3.134645e+00 1.426076e-01  21.9809154
       time -9.666927e-03 1.353244e-03  -7.1435205
      time2  1.819918e-06 2.062649e-06   0.8823207
      split -1.739333e+00 2.530775e-01  -6.8727284
 time:split  6.091784e-03 1.540161e-03   3.9552907
time2:split -1.131548e-06 2.100121e-06  -0.5388016
```

```
(Dispersion Parameter for Poisson family taken to be 1 )
    Null Deviance: 754.6903 on 249 degrees of freedom
Residual Deviance: 105.0881 on 244 degrees of freedom
```

**Gamma**

```
> summary(glm(counts ~ (time + timeln) * split, poisson(), data = coal.dataframe))

Coefficients:
                    Value  Std. Error    t value
 (Intercept)  3.454975151 0.484174701  7.1358027
        time -0.007852951 0.001374401 -5.7137276
      timeln -0.113971426 0.143451797 -0.7944928
       split -1.170209312 0.822862172 -1.4221207
  time:split  0.006018961 0.001456681  4.1319706
timeln:split -0.138367592 0.208257775 -0.6644054

(Dispersion Parameter for Poisson family taken to be 1 )

    Null Deviance: 754.6903 on 249 degrees of freedom

Residual Deviance: 104.9936 on 244 degrees of freedom
```

Neither the (truncated) normal nor the gamma distribution fits significantly better than the exponential distribution. This confirms the result that a Poisson process is applicable to these data. Thus, disasters occur more or less at random, but with a different frequency in the two periods. Within each period, the risk of a mining disaster appears to remain constant, independently of the time since a previous such disaster. Only the empirical hazard functions appear to contradict this conclusion.

## 4.7 An Example with Periodicity

In addition to trend, time series very often show cyclical or periodic effects such as seasonality. If the cycle in the data can be conveniently divided into segments, the methods of the previous section can be readily applied. Otherwise, those of the next chapter may be used. However, the first problem is to identify whether periodicity is present.

As an example, we consider the times of admission of patients to a hospital, given in Section 10.2. (Another way of approaching these data was presented in Section 3.2). These data are distinct from the others we have studied in that we have the actual times of arrival, not the intervals between them. Before we begin, we have to construct the time intervals with SPlus.

One possible source of periodicity in these data may be a variation in number of admissions throughout the day—some hours may be busier than others. We construct a table for the frequency of arrival during each hour. Note that, although the times are apparently given in decimal form, the value after the period is the minutes and not the fraction of the hour.

```
> entr.data_data.frame(scan(file="Chpt3-7",what=list(day=0, month=0, year=0, time=0)))
> entr.data$time_floor(entr.data$time+0.99)
> table(entr.data$time)
```

```
 hour: 1  2  3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
count: 5 12 10 6 2 1 1 7 3  7 10 19 11 15 10 24 16 17 13 16 16 16  8  9
```

We set up and fit a log-linear model for constant frequency of admission throughout the 24 hours of the day.

```
> entr.tab_glmtab(index = entr.data$time)
> entr.glm1_glm(Counts ~ 1, family = poisson(), data = entr.tab)
> summary(entr.glm1)
```

```
Coefficients:
             Value Std. Error  t value
(Intercept) 2.35928  0.1165479 20.24301

(Dispersion Parameter for Gaussian family taken to be 3.450188)
    Null Deviance: 89.88451 on 23 degrees of freedom
Residual Deviance: 89.88451 on 23 degrees of freedom
```

Based on a residual deviance of 89.9 on 23 degrees of freedom (p-value of 0), the model is rejected.

```
> residuals(entr.glm1)
```

```
 [1] -1.9152654  0.4262579 -0.1809967 -1.5350619 -3.2406996 -3.8010664
 [7] -3.8010664 -1.1745138 -2.7572917 -1.1745138 -0.1809967  2.3243690
[13]  0.1272520  1.2766204 -0.1809967  3.5309757  1.5467811  1.8111835
[19]  0.7169788  1.5467811  1.5467811  1.5467811 -0.8302074 -0.4996611
```

We see that the residuals are predominantly positive for the hours between eleven in the morning and ten in the evening so we create a variable distinguishing these two segments of the day.

```
> entr.t2_factor((as.numeric(entr.tab$index) <= 11) + (as.numeric(entr.tab$index) > 22) + 1)
> entr.glm2_glm(Counts ~ entr.t2, poisson(), data = entr.tab)
> summary(entr.glm2)
```

```
Coefficients:
                Value Std. Error    t value
(Intercept)  2.7553963 0.07602859  36.241581
    entr.t2 -0.9258965 0.13462727  -6.877481

(Dispersion Parameter for Poisson family taken to be 1 )
    Null Deviance: 89.88451 on 23 degrees of freedom
Residual Deviance: 38.65288 on 22 degrees of freedom
```

Although this model still does not fit sufficiently well (p-value of 0.015), it is a very significant improvement on that with a constant admission rate.

residuals(entr.glm2)

```
         1        2        3       4         5         6         7
-0.5108029 2.047267 1.386811 -0.0930 -1.978921 -2.608168 -2.608168
         8        9       10      11        12        13        14
 0.3021314 -1.44090 0.302131 1.38681 0.7988532 -1.260763 -0.184829
        15       16       17      18        19        20        21
 -1.548651 1.934452 0.068573 0.31674 -0.709171 0.0685731 0.0685731
        22       23       24
0.06857311 0.678679 1.039512
```

We see that there are still more arrivals than expected at two and three o'clock in the morning and four in the afternoon and less at six and seven in the morning. There seems to be a cyclical effect with two waves during the day.

A second periodic effect in the data could be that of the seasons. We shall construct a table for frequency of admission by month.

> table(entr.data$month)

```
 month:  1  2  3  4  5  6  7  8  9 10 11 12
counts: 23 30 23 12 18 23 16 15 17 17 28 32
```

> entr.tab_glmtab(index = entr.data$month)

```
  Counts index
1     23     1
2     30     2
3     23     3
4     12     4
```

```
 5     18     5
 6     23     6
 7     16     7
 8     15     8
 9     17     9
10     17    10
11     28    11
12     32    12
```

Here we must take care, since February and part of March are represented for both 1963 and 1964. Thus, there are more total days observed for these months than usual. In any case, the differing number of days in each month must be taken into account. A model of constant arrival rate does not give the same number of admissions in each month because of this. We handle the problem by creating the appropriate offset. Thus the expected number of arrivals in month $j$, $e_j$, is modelled as $e_j = n_j e^{\mu}$ where $n_j$ is the number of days in month $j$ represented in the data set.

```
> entr.offset_log(c(31,56,47,30,31,30,31,31,30,31,30,31))
> entr.glm3_glm(Counts ~ 1 + offset(entr.offset), family = poisson(), data = entr.tab)
> summary(entr.glm3)

Coefficients:
                Value Std. Error   t value
(Intercept) -0.4763806 0.06272241 -7.595062

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 19.65171 on 11 degrees of freedom
Residual Deviance: 19.65171 on 11 degrees of freedom

> tsplot(residuals(entr.glm3), main = "Residuals of Constant Rate Model")
> abline(0, 0, lty=2)
```

Although the model appears just acceptable, we see from the plot in Figure 4.11 that admissions for November and December are seriously underestimated with the constant arrival rate model. As with the daily model above, we create a segment model and refit.

```
> entr.glm4_glm(Counts ~ factor(as.numeric(index) < 11) + offset(entr.w), family = poisson(), data
= entr.tab)
> summary(entr.glm4)

Coefficients:
                                  Value Std. Error    t value
                    (Intercept) -0.0165293  0.1290994 -0.1280355
factor(as.numeric(index) < 11) -0.5678150  0.1477181 -3.8439096
```

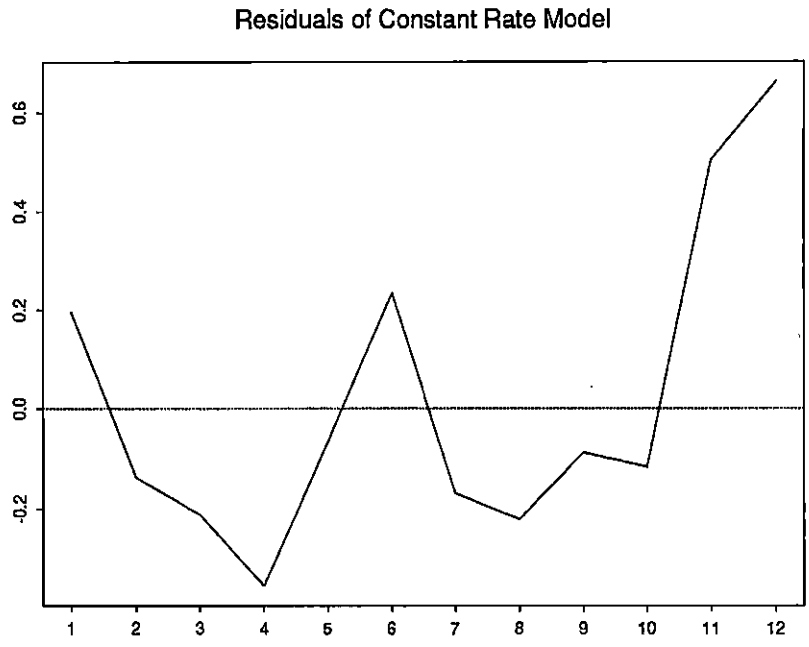



Figure 4.11: Residual plot for constant rate model of monthly hospital arrival counts.

```
(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 19.65171 on 11 degrees of freedom
Residual Deviance: 6.359328 on 10 degrees of freedom
```

Our model now fits extremely well. However, with such a short time series, we do not know if this indicates a systematic difference for these two months or a trend whereby admissions are increasing. To see this a little more clearly, we separate the data for the two February-March periods and plot the number of arrivals in successive one week intervals. However, we must first calculate the inter-arrival times from the entry times, as mentioned above. We use days as the unit of measurement, including fractions of days for the hour of admission. We must also separate minutes from hours and transform them to a fraction of an hour. All these steps will be carried out with a special function, interv(), which has been put together specifically to transform these data quickly. Once we have our inter-arrival times, we apply the tsfr() function to the data to calculate the frequencies of events in successive 7 day intervals.

```
function(yrr, mnth, dy, tm)
{
        tm <- floor(tm) + (tm - floor(tm))/0.6
        dy <- julian(mnth, dy, yrr)
        dy <- dy - min(dy)
```

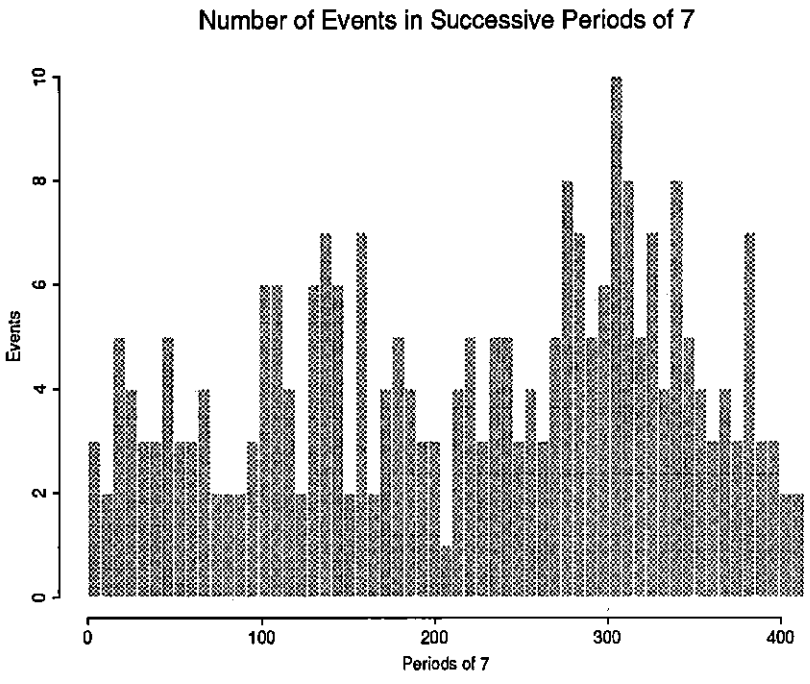



Figure 4.12: Histogram of time intervals between hospital arrivals.

```
        dy <- dy + tm/24
        dy <- dy[-1] - dy[ - length(dy)]
        dy
}

> entr.interv_interv(entr.data$year, entr.data$month, entr.data$day, entr.data$time)
> entr.tsfr_tsfr(entr.interv, 7)

$breaks:
 [1]   0   7  14  21  28  35  42  49  56  63  70  77  84  91  98 105 112 119 126
[20] 133 140 147 154 161 168 175 182 189 196 203 210 217 224 231 238 245 252 259
[39] 266 273 280 287 294 301 308 315 322 329 336 343 350 357 364 371 378 385 392
[58] 399 406 413

$counts:
 [1] 3 2 5 4 3 3 5 3 3 4 2 2 2 3 6 6 4 2 6 7 6 2 7 2 4
[26] 5 4 3 3 1 4 5 3 5 5 3 4 3 5 8 7 5 6 10 8 5 7 4 8 5
[51] 4 3 4 3 7 3 3 2 2
```

In Figure 4.12 we clearly see the increased frequency of admissions in November and December (days 273 to 350), with no evidence of a continuing trend into the following year.

We test for linear and quadratic trends and for the segmented model.

```
> entr.time_seq(entr.tsfr)
> entr.dataframe_data.frame(counts = entr.tsfr, time = entr.time, time2 = entr.time ^ 2, split =
factor((entr.time > 39) & (entr.time < 50)))
```

### Linear

```
> summary(glm(counts ~ time, family = poisson(), data = entr.dataframe))

Coefficients:
                  Value  Std. Error  t value
(Intercept) 1.272944784 0.133214040 9.555635
       time 0.005927274 0.003703088 1.600630

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 50.32488 on 58 degrees of freedom
Residual Deviance: 47.75506 on 57 degrees of freedom
```

### Quadratic

```
> summary(glm(counts ~ time + time2, family=poisson(), data = entr.dataframe))

Coefficients:
                   Value   Std. Error   t value
(Intercept)  0.9793236858 0.2220156225  4.411058
       time  0.0328805039 0.0160356669  2.050461
      time2 -0.0004385058 0.0002526588 -1.735565

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 50.32488 on 58 degrees of freedom
Residual Deviance: 44.66807 on 56 degrees of freedom
```

### Segmented

```
> summary(glm(counts ~ split, poisson(), data = entr.dataframe))

Coefficients:
               Value Std. Error   t value
(Intercept) 1.3285408 0.07340161 18.099613
      split 0.5883819 0.14172497  4.151575

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 50.32488 on 58 degrees of freedom
Residual Deviance: 34.72639 on 57 degrees of freedom
```

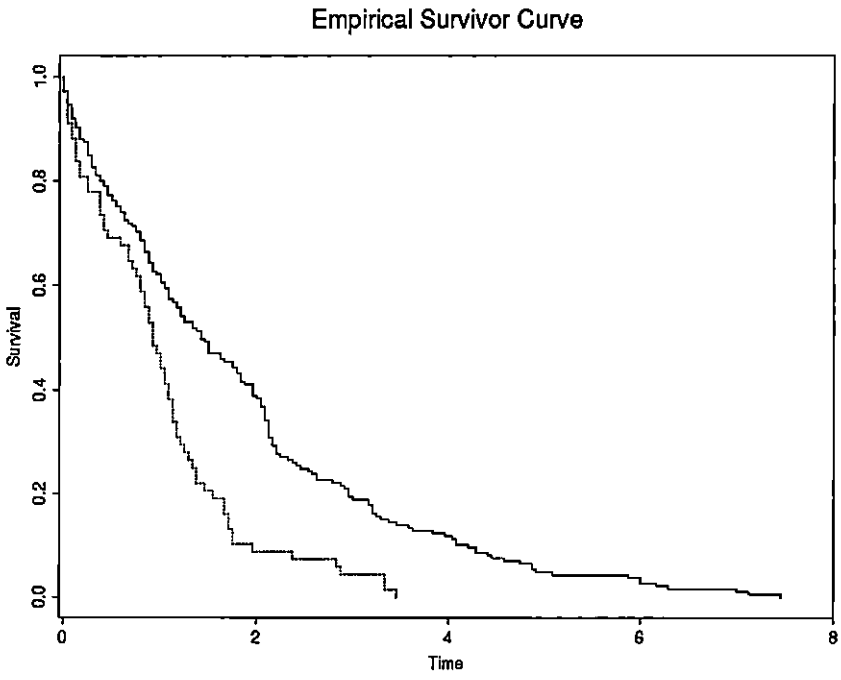



Figure 4.13: Empirical survivor curve for inter-arrival times at a hospital.

Although all models fit well, the segmented model is significantly better than the others, as would be expected.

If we test for a Poisson process as in Section 4.2, we obtain results very similar to those for vehicles passing on the road. The quadratic term yields a significantly better (residual deviance 10.4 with 24 df), although the linear term alone has a reasonable fit (residual deviance 17.1 with 26 df). The results for the distributions of duration are also very similar, with here the (truncated) normal distribution giving the best fit.

We next calculate the Kaplan-Meier estimates of the inter-arrival times. Only the empirical survivor curve is displayed in Figure 4.13.

```
> plot(surv.fit(entr.interv, status = rep(1, length(entr.interv)), strata = factor((seq(entr.interv) >
149) & (seq(entr.interv) < 218))), lty=c(1, 2), main = "Empirical Survivor Curve")
```

The lower curve is for the two winter months. We can also fit various distributions to the interarrival times, as in Section 4.6. The exponential distribution does fit the data reasonably well (residual deviance of 36.6 with 36 df) as compared to the normal distribution with residual deviance of 29.3 and 34 degrees of freedom.

# Chapter 5

# Survival Curves

## 5.1 Censored Data

In the two previous chapters, we studied event data where events were observed over time on one or more subjects (eg. mines, a hospital, voters). In Chapter 3, each event resulted in a possible switch from one of a relatively limited number of states to another. In Chapter 4, we looked at the series of inter-occurrence times between the events. In such series, no event is absorbing: it is always possible to pass from any state to another. A different situation arises when an absorbing state exists. The process ends when the individual enters such a state. In the simplest case, only two states exist and the event of interest is the passage from the one to the other. Thus, we are interested in the time during which an individual survives in the first state before passing into the absorbing state, the *survival time.*

With such data, we only have one observation per individual, since the process ends when the event occurs. This contrasts with the situation in the previous chapters where a sequence of events on one or more units was observed. For this reason, for survival data we require observations on a number of individuals in order to be able to proceed.

Survival data also differ from renewal processes in another way. Some individuals may survive longer than the period available for observation or drop out for some reason before the decisive event occurs. Hence, we shall be able to obtain complete survival information for some individuals, but, for others, all we know is that they survived at least for a given time. Such censoring may occur in various planned ways [9]. For example, all recording may be stopped after a fixed interval of time (Type I or time censoring). Another possibility is that the study may be continued until complete information is available on a fixed number of individuals; at that time recording is stopped and the remaining individuals are censored (Type II or failure censoring). Finally, individuals may drop out for reasons not related to the study or beyond the control of the research worker.

Survival data, then, have two specific characteristics: they are non-negative and some values may be censored. The first characteristic implies that only certain distributions will be suitable for

the study of survival curves. A number of these were introduced, at least by name, in the previous chapter. The second characteristic, censoring, means that special methods may often be required in order to apply a model and analyse the data.

In this chapter, we see how to fit a series of survival models to censored data. SPlus can fit parametric survival models for a number of common distributions; for more esoteric distributions, customised functions are required.

If $s_i$ is a failure indicator with $s_i = 1$ indicating that observation $i$ is uncensored and $s_i = 0$ that it is censored, then the probability of a sample is given by

$$\prod_i f(y_i)^{s_i} S(y_i)^{1-s_i} \tag{5.1}$$

where $f(y)$ is the density and $S(y) = P(Y \geq y)$ is the survivor function defined by equations 4.3 and 4.4 in Section 4.3.

As a preliminary step in any analysis of survival data, the procedures of the previous chapter, based on the Kaplan-Meier estimates, should be applied in order to determine which distribution might be suitable. Care must be taken, since the distribution which appears suitable when no explanatory variables are present will not necessarily remain so with the introduction of these variables into the model.

## 5.2 The Hazard Function

The hazard function,

$$h(t) = \lim_{y \to t+} \frac{f(y)}{S(t)} = \frac{f(t)}{S(t)}$$

is the limiting probability of an event occurring after time $t$ given survival just up to time $t$.

As an example, we use the times, in months, to recidivism of all people released from prison in North Carolina in 1978 and 1980, given in Section 10.3. Here, our only explanatory variable is the year. In these data, we have two forms of censoring. The study stopped after 81 months for the 1978 data and after 57 months for 1980. In addition, near the end of each of these periods, a certain number of individuals disappeared from the study.

First, we read the data and set up the model:

```
> rd.data_data.frame(scan(file = "data/Chpt4-2", what = list(year = "", censor = 0, freq = 0,
month = 0), skip = 1)
```

We also eliminate individuals with a zero recidivism period, since they returned immediately to prison after release and because many survival distributions do not accept zero values.

```
> rd.data_rd.data[rd.data$month != 0,]
```

Our first task is to plot the empirical hazard function using the empirical survivor functions provided in SPlus.

The data we have been provided with for analysis is laid out in a frequency table. This is economical for presentation and storage; unfortunately, SPlus processes survival data only in a raw form—one data point for each observation. Before proceding we must first expand the frequency data, making a separate record for each count in the frequency table. We do this using the apply() function to recursively replicate each cell in the frequency data:

```
> rd.exp.data_data.frame(apply(rd.data[,c(1, 2, 4)], 2, rep, rd.data[,3]))
> rd.exp.data$month_as.numeric(rd.exp.data$month)
> rd.exp.data$censor_as.numeric(rd.exp.data$censor)
```

The expanded data consists of a data frame with three variables: time, censor, and year. Do not be surprised if the conversion process takes a few seconds, even on a fast computer—the new data set consists of over ten thousand observations. To calculate the empirical survivor function using the Kaplan-Meier method, we employ the surv.fit() function provided with SPlus.

```
> attach(rd.exp.data)
> rd.empirical_surv.fit(month, censor, strata = year)
> plot(rd.empirical$time, rd.empirical$n.event / rd.empirical$n.risk, type="n", main="Empirical
Hazard Curves")
> text(rd.empirical$time, rd.empirical$n.event / rd.empirical$n.risk, labels = ifelse(rd.empirical$strata
== 1978, "*", "o"))
```

The text() function used above allows us to mark in the data points for the plot (Figure 5.1) with different labels for the 1978 and 1980 experiments. The curve for the 1978 experiment is marked using * and the curve for 1980 is marked using **o**. Note that the 1980 curve peaks higher, indicating a higher rate of early recidivism, but that both curves tail off to zero at about the same level. Since the hazard curves both go to zero, we can infer that once an inmate has been free long enough there is very little chance that he or she will return to crime.

## 5.3 Exponential Distribution

In a renewal process, an exponential distribution of renewal times corresponds to the Poisson distribution for the frequency of events in successive fixed time periods. In the context of survival distributions, an exponential distribution of survival times corresponds to a constant hazard function

$$h(y;\theta) = \frac{f(y)}{S(y)} = \frac{\theta e^{-\theta y}}{e^{-\theta y}} = \theta \tag{5.2}$$

In other words, the rate of mortality does not change with time; there is no aging effect.

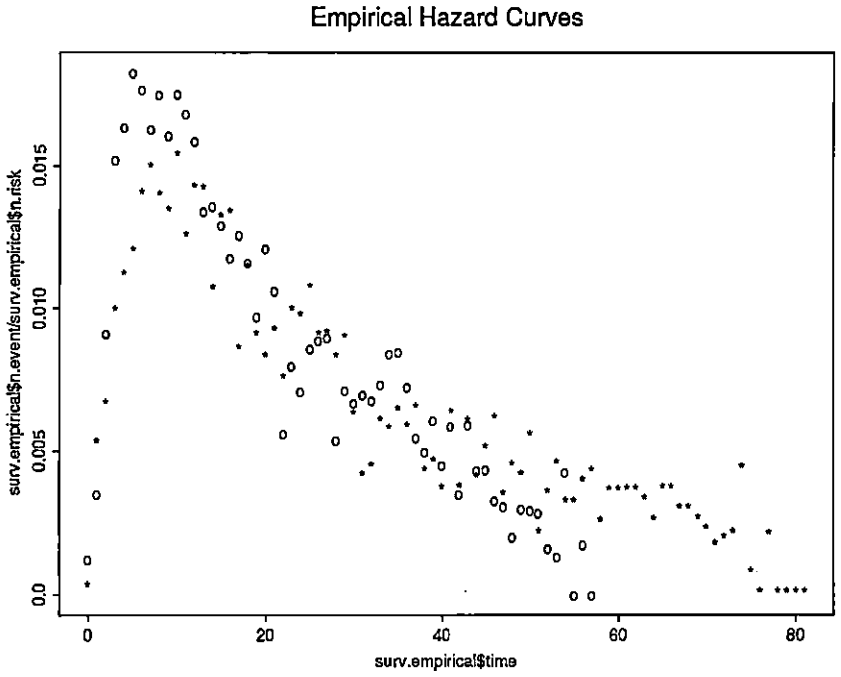



Figure 5.1: Empirical hazard curves for recidivism rates of inmates over two studies.

SPlus does not fit the exponential distribution directly, but uses the fact that the exponential can be constructed from a transformed extreme value distribution with a scale parameter of one. If $X$ is distributed as extreme value, then the probability density function of $X$ is

$$f_X(x; m, \sigma) = \frac{1}{\sigma} \exp\left(\frac{x-m}{\sigma} - e^{\frac{x-m}{\sigma}}\right)$$

where $m$ is the location/shape parameter and $\sigma$ is the scale parameter. Taking $\sigma = 1$ and making use of the transformation $Y = e^X$, we obtain

$$f_Y(y; m) = \frac{1}{y} \exp\left(\log y - m - e^{\log y - m}\right) = e^{-m} e^{y e^{-m}}$$

$$f_Y(y; \theta) = \theta e^{-\theta y} \qquad \text{where} \qquad \theta = e^{-m}$$

To fit a parametric survival model we will use the survreg() function. survreg() can fit models for the following distributions without any special techniques: extreme value, logistic, and gaussian. The exponential and Rayleigh distributions can also be fitted, but are special cases of the extreme value distribution. We will see in the following sections of this chapter that other distributions can be fitted by using transformations of the basic distributions available. The invocation of survreg() looks like

```
> survreg(formula, data = sys.parent(), subset, na.action, link = "log", dist = "extreme", fixed, eps
= 0.0001, init, iter.max = 10, control, model = F, x = F, y = T, ...)
```

Note that the default link is the log link. In fitting the exponential distribution, we use dist = "extreme", link="log", and fixed=list(scale=1). Our model looks like:

$$\log y_{ij} = \mu + \alpha_i + \sigma w_{ij} = m_i + \sigma w_{ij} \qquad \sigma > 0$$

where $y_{ij}$ is the recidivism time for the $j$'th individual in the $i$'th year ($i = 1$ for 1978 and $i = 2$ for 1980), and $w_{ij}$ has a standard extreme value distribution with $\sigma = 1$. That is, $S_W(w) = e^{-e^w}$. We use the treatment contrast to parameterize the model so that $\alpha_1 = 0$. $y_{ij}$ then has an exponential distribution with $S_Y(y) = e^{-\lambda y}$ where $\lambda_i = e^{-(\mu+\alpha_i)}$ is the hazard function.

We use the Surv() function to convert our time and censor vectors into a survival data object that SPlus can process.

```
> rd.exponential_survreg(Surv(month,censor) ~ year, dist = "extreme", link = "log", fixed = list(scale
= 1))
> summary(rd.exponential)


Coefficients:
             Value Std. Error z value        p
(Intercept)  4.955     0.0241   205.6 0.00e+00
      year -0.305     0.0324    -9.4 5.63e-21

Extreme value distribution: Dispersion (scale) fixed at 1

    Null Deviance: 16184 on 10138 degrees of freedom
Residual Deviance: 16095 on 10137 degrees of freedom
Number of Newton-Raphson Iterations: 4
```

The estimate of the hazard function is computed and plotted for each of the two years.

$$h_1(y) = \theta_1 = e^{-m_1} = e^{-(4.955)} = 0.007048$$

$$h_2(y) = \theta_2 = e^{-(4.955-0.305)} = 0.00955$$

```
> plot(month, ifelse(year == 1980, 0.00955, 0.00748), type = "n", main = "Exponential Distribution
- Hazard Functions", ylim = c(0, 0.02))
> text(month, ifelse(year == 1980, 0.0096, 0.007), labels = ifelse(year == 1980, "2", "1"))
```

Note in Figure 5.2 that inmates in 1978 had a lower rate of recidivism than inmates in 1980. The exponential distribution has a constant hazard rate, hence both graphs are horizontal straight lines.

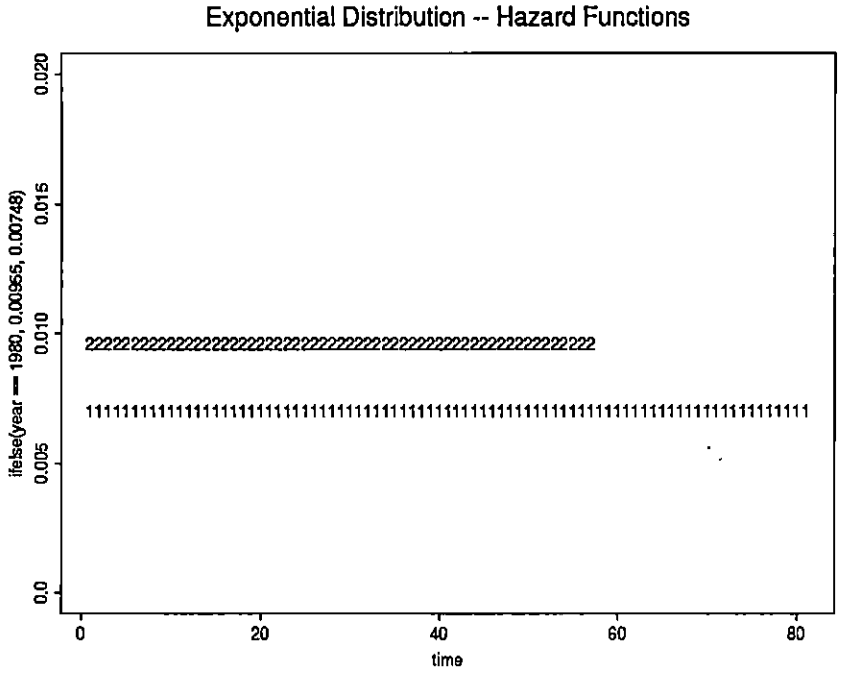



Figure 5.2: Estimated hazard function for recidivism data assuming an exponential survival distribution.

## 5.4 Pareto Distribution

SPlus does not include a built-in method for fitting the Pareto distribution to survival data. However, by taking logarithms, Pareto distributed data can be transformed into exponentially distributed data which can be fitted with the methods of the previous section.

Suppose we have a random variable $X$ which has a Pareto distribution. Then,

$$X \sim f_X(x) = \frac{\theta c^\theta}{x^{\theta+1}} \qquad x > c, \theta > 0$$

If we let $Y$ be a random variable such that $Y = \log X$ then

$$f_Y(y) = e^y f_X(e^y) = \frac{e^y \theta c^\theta}{(e^y)^{\theta+1}} = \theta e^{-\theta(y - \log c)} \qquad y > \log c$$

which is guaranteed exponential distribution, with guarantee $\log c$ which we take to be zero.

The survivor function of the Pareto is

$$S(y; \theta, c) = c^\theta / y^\theta \tag{5.3}$$

while the hazard function is inversely proportional to the time

$$h(y; \theta) = \frac{f(y)}{S(y)} = \frac{\theta c^\theta / y^{\theta+1}}{c^\theta / y^\theta} = \theta / y \tag{5.4}$$

Obviously, this distribution is not at all appropriate for the present data set, but we fit it for completeness.

As part of the fit, we need to transform our data by taking the logarithm. Unfortunately, there are 43 values of 1 in our data, which become zeros after the log transformation; we will have to remove these values before continuing with the fitting process.

```
> detach("rd.exp.data")
> rd.exp.data2_rd.exp.data[rd.exp.data$month != 1,]
> attach(rd.exp.data2)
```

Now, all of the potentially difficult data points have been removed and we can transform the data and fit it with survreg(). Note that the survreg invocation is exactly the same as that for the exponential, except that the time variable has been logarithmically transformed.

```
> rd.pareto_survreg(Surv(log(month), censor) ~ year, dist = "extreme", link = "log", fixed =
list(scale = 1))
> summary(rd.pareto)
```

```
Coefficients:
             Value Std. Error z value     p
(Intercept)  2.276     0.0243   93.79 0.000
       year -0.044     0.0326   -1.35 0.177

Extreme value distribution: Dispersion (scale) fixed at 1

    Null Deviance: 9966 on 10095 degrees of freedom
Residual Deviance: 9965 on 10094 degrees of freedom
Number of Newton-Raphson Iterations: 3
```

Again we plot the hazard function, which in contrast to that for the exponential distribution, decreases with time.

$$h_1(y) = \theta_1/y = e^{-(2.276)}/y = 0.10269/y$$
$$h_2(y) = \theta_2/y = e^{-(2.276-0.044)}/y = 0.10731/y$$

```
> plot(month, ifelse(year == 1980, 0.10731, 0.10269)/month, type = "n", main = "Pareto
Distribution – Hazard Functions", ylim = c(0,0.02))
> text(month, ifelse(year == 1980, 0.10731, 0.10269)/month, labels = ifelse(year == 1980, "2",
"1"))
```

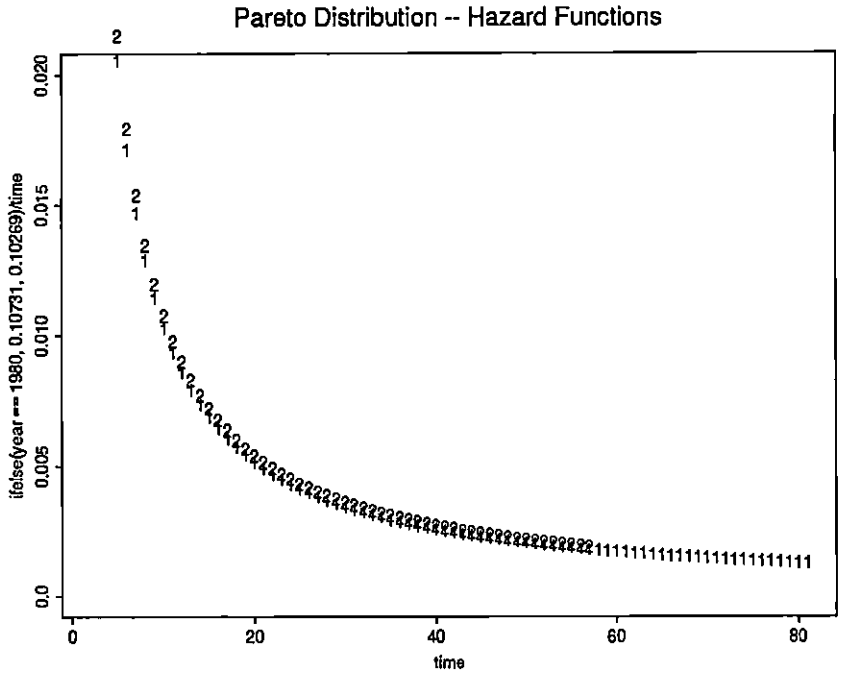



Figure 5.3: Estimated hazard function for recidivism data assuming a Pareto survival distribution.

## 5.5 Weibull Distribution

The Weibull distribution is suitable for a "weakest link" situation, whereby a variety of (unrecorded) causes may bring about the event which ends survival. The first one (the weakest) to occur determines the survival time.

If $X$ is a random variable with a Weibull distribution, then

$$f_X(x;\lambda,\alpha) = \alpha\lambda(\lambda x)^{\alpha-1}e^{-(\lambda x)^\alpha} \qquad \lambda,\alpha > 0 \tag{5.5}$$

$\lambda$ is both the *location parameter* and the *scale parameter* of the distribution, determining its position on the axis and its spread. The second parameter, $\alpha$, is called a *shape parameter*. When $\alpha = 1$, we have the exponential distribution and for $\alpha = 2$ the Rayleigh distribution.

As with the Pareto distribution, SPlus cannot directly fit the Weibull distribution to survival data. However, taking the logarithm of Weibull distributed data yields data with an extreme value distribution. The survreg() function can directly fit survival data to the extreme value distribution, using the dist = "extreme" parameter.

We derive the extreme value distribution from the Weibull distribution using the log transformation. If $Y$ is a random variable such that $Y = \log X$ then

$$f_Y(y) = e^y f_X(e^y) = e^y \alpha\lambda(\lambda e^y)^{\alpha-1}e^{-(\lambda e^y)^\alpha}$$

$$f_Y(y) = \frac{1}{\sigma} e^{\frac{y-m}{\sigma}} \exp\left(-e^{\frac{y-m}{\sigma}}\right)$$

which is the extreme value distribution where $\sigma = 1/\alpha$ and $m = -\log\lambda$.

The survivor function for the Weibull distribution is

$$S(y;\lambda,\alpha) = e^{-(\lambda y)^\alpha} \tag{5.6}$$

and the hazard function is

$$h(y) = \frac{f(y)}{S(y)} = \frac{\alpha\lambda(\lambda y)^{\alpha-1}e^{-(\lambda y)^\alpha}}{e^{-(\lambda y)^\alpha}} = \alpha\lambda(\lambda y)^{\alpha-1} \tag{5.7}$$

The hazard function is a power function of time, increasing for $\alpha > 1$ and decreasing for $\alpha < 1$.

```
> attach(rd.exp.data)
> rd.weibull_survreg(Surv(month, censor) ~ year, dist = "extreme", link = "log")
> summary(rd.weibull)
```

```
Coefficients:
             Value Std. Error z value        p
(Intercept)  5.144     0.0334  153.83 0.00e+00
      year -0.289     0.0401   -7.22 5.21e-13
 Log(scale)  0.212     0.0150   14.09 4.67e-45

Extreme value distribution: Dispersion (scale) = 1.235941

    Null Deviance: 14302 on 10138 degrees of freedom
Residual Deviance: 14250 on 10136 degrees of freedom  (LL= -11788 )
Number of Newton-Raphson Iterations: 5
```

From above, we know that

$$\alpha = 1/\sigma = 1/1.235941 = 0.8091001$$

$$\lambda_1 = e^{-\mu_1} = e^{-(5.144)} = 0.005834306$$

$$\lambda_2 = e^{-\mu_2} = e^{-(5.144-0.289)} = 0.007797127$$

Now, using the formula above, we can plot the hazard rate for the Weibull distribution.

```
> alpha_0.8091001
> lambda_ifelse(year == 1978, 0.005834306, 0.007797127)
> rd.weibull.y_lambda * alpha * (lambda * month) ** (alpha - 1)
> plot(month, rd.weibull.y, ylim = c(0, 0.02), type = "n", main = "Weibull Distribution – Hazard
Function")
> text(month, rd.weibull.y, labels = ifelse(year == 1978, "1", "2"))
```

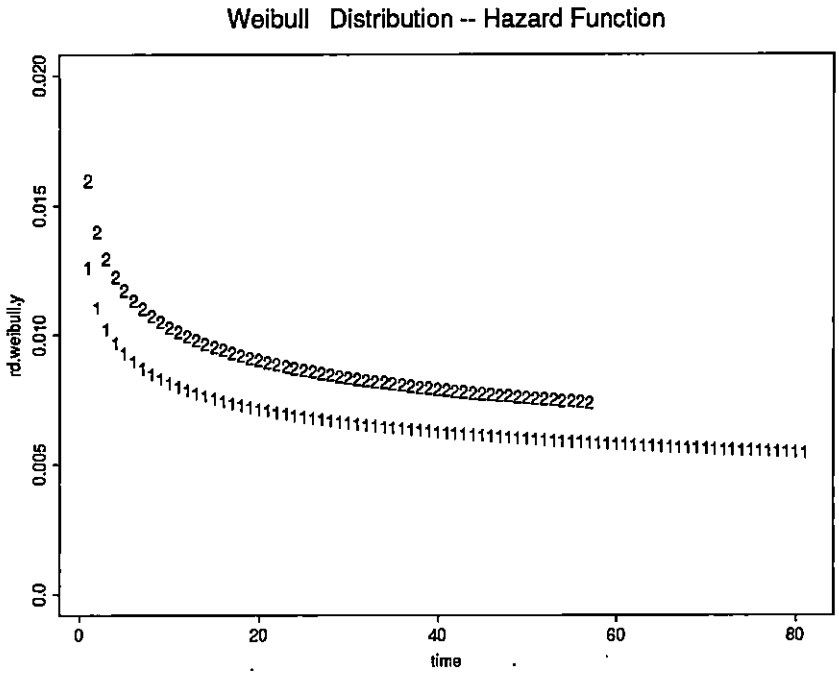



Figure 5.4: Estimated hazard function for recidivism data assuming a Weibull survival distribution.

## 5.6 Extreme Value

The extreme value density function is

$$f_Y(y;m,\sigma) = \frac{1}{\sigma}\exp\left(\frac{y-m}{\sigma} - e^{\frac{y-m}{\sigma}}\right) \qquad \sigma > 0 \tag{5.8}$$

The survivor function is

$$S(y;m,\sigma) = \exp\left(-e^{\frac{y-m}{\sigma}}\right) \tag{5.9}$$

and the hazard function is

$$h(y;m,\sigma) = \frac{1}{\sigma}e^{\frac{y-m}{\sigma}} \tag{5.10}$$

This function shows an increasing rate of failure with age.

We fit the extreme value distribution directly.

```
> rd.extreme_survreg(Surv(month,censor) ~ year, dist = "extreme", link = "identity")
> summary(rd.extreme)

Coefficients:
             Value Std. Error z value p
 (Intercept) 97.26     0.9576   101.6 0
```

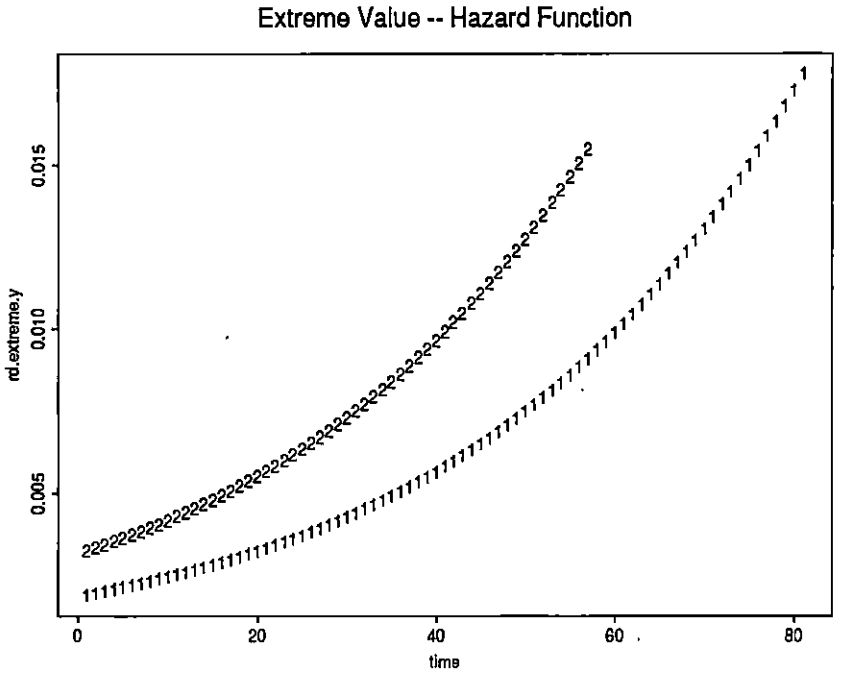



Figure 5.5: Estimated hazard function for recidivism data assuming an extreme value survival distribution.

```
        year -18.99     1.1635   -16.3 0
  Log(scale)   3.58     0.0149   240.9 0

Extreme value distribution: Dispersion (scale) = 35.82942

    Null Deviance: 14468 on 10138 degrees of freedom
Residual Deviance: 14199 on 10136 degrees of freedom
Number of Newton-Raphson Iterations: 7
```

Again, we plot the hazard function.

```
> sigma_35.82946
> m_ifelse(year == 1978, 97.26, 97.26 - 18.99)
> rd.extreme.y_(1 / sigma) * exp((month - m) / sigma)
> plot(month, rd.extreme.y, main = "Extreme Value – Hazard Function", type = "n")
> text(month, rd.extreme.y, labels = ifelse(year == 1978, "1", "2"))
```

## 5.7 Log Normal Distribution

Since survival times must be non-negative, most survival distributions are not symmetric, but have a long tail to the right. Thus, the normal distributions are not usually appropriate unless all survival times are very large. In contrast to this, the log normal distribution does have an appropriate form. This distribution arises when a larger number of unknown factors combine in a multiplicative fashion to produce the survival times. The log normal density has the form

$$f(x;\mu,\sigma^2) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(\log x-\mu)^2}{2\sigma^2}} \qquad \sigma > 0 \tag{5.11}$$

The survivor function is the same as that for a normal distribution, except for the logarithmic transformation:

$$S(x;\mu,\sigma^2) = 1 - \Phi\left(\frac{\log x - \mu}{\sigma}\right) \tag{5.12}$$

where $\Phi()$ is the standard normal cumulative distribution function. The hazard function for the log normal distribution differs from all those that have preceded in that it is not monotonic. It is zero at time zero, increases to a maximum, and then decreases back to zero with increasing time:

$$h(x;\mu,\sigma^2) = \frac{\phi(\frac{\log x-\mu}{\sigma})}{x\sigma\sqrt{(2\pi)(1-\Phi(\frac{\log x-\mu}{\sigma}))}} \tag{5.13}$$

where $\phi()$ is the standard normal probability density function.

The log normal distribution has a hazard function which is of the same shape as that observed empirically for our recidivism data, and may provide a better model than the distributions we have tried thus far. Without censoring, we need simply take the logarithms of the recidivism times and apply normal theory models, as in the first chapter. However, when censoring is present, the survival tools must be used.

```
> rd.lognorm_survreg(Surv(month, censor) ~ year, dist = "gaussian", link = "log")
> summary(rd.lognorm)

Coefficients:
             Value Std. Error z value        p
(Intercept)  4.757     0.0363  131.20 0.00e+00
      year -0.245     0.0436   -5.61 1.97e-08
 Log(scale)  0.625     0.0132   47.42 0.00e+00

Gaussian distribution: Dispersion (scale) = 1.867805

    Null Deviance: 11218 on 10138 degrees of freedom
Residual Deviance: 11187 on 10136 degrees of freedom
Number of Newton-Raphson Iterations: 4
```

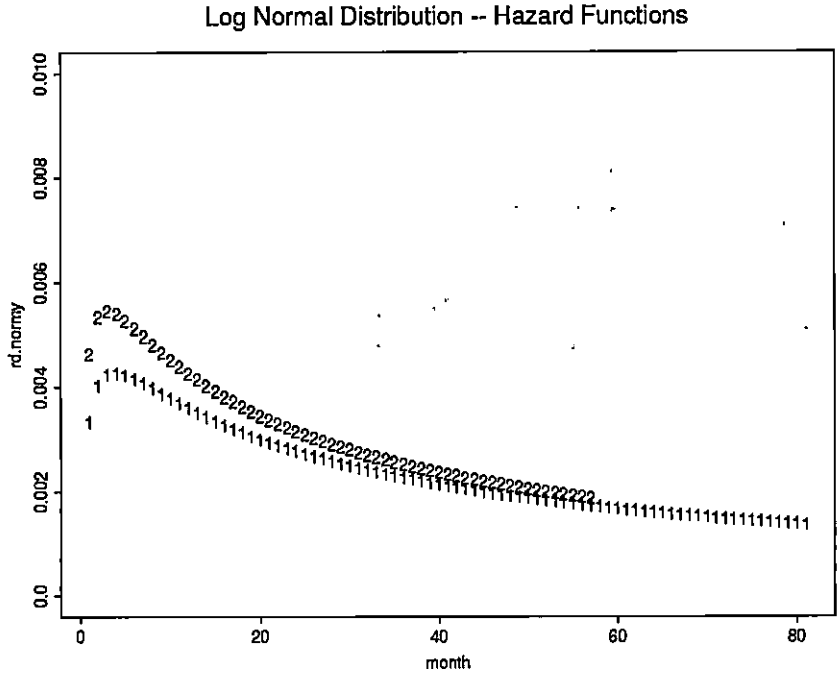



Figure 5.6: Estimated hazard function for recidivism data assuming a log normal survival distribution.

The parameter estimates yield conclusions similar to those for all of the other models as to the difference between years. We plot the hazard function.

```
> rd.normy_(log(month) - ifelse(year == 1978, 4.757, 4.757 - 0.245)) / 1.867805
> rd.normy_dnorm(rd.normy) / (month * 1.867805 * sqrt(2 * pi * (1 - pnorm(rd.normy))))
> plot(month, rd.normy, ylim = c(0,0.01), main = "Log Normal Distribution – Hazard Functions",
type = "n")
> text(month, rd.normy, labels = ifelse(year == 1978, "1", "2"))
```

The hazard function (Figure 5.6) has the form expected, but is much lower than the empirically derived estimates.

## 5.8 Log Logistic Distribution

The logistic distribution poses the same problems as the normal distribution, in that negative values of the variable are possible, though it can usually be applied successfully when all survival times are relatively large. The log logistic distribution is obtained from the logistic in the same way as the log normal from the normal and the Pareto from the exponential—by taking logarithms. The

log logistic distribution function looks like

$$f(x;\mu,\sigma) = \frac{e^{\frac{\log x-\mu}{\sigma}}}{\sigma x(1+e^{\frac{\log x-\mu}{\sigma}})^2} \qquad \sigma > 0 \tag{5.14}$$

where $\mu$ is a location parameter and $\sigma$ a scale parameter. The survivor function is

$$S(x;\mu,\sigma) = \frac{e^{\frac{\log x-\mu}{\sigma}}}{1+e^{\frac{\log x-\mu}{\sigma}}} \qquad \sigma > 0 \tag{5.15}$$

and the hazard function is

$$h(x;\mu,\sigma) = \frac{1}{\sigma x(1+e^{\frac{\log x-\mu}{\sigma}})} \qquad \sigma > 0 \tag{5.16}$$

This function is similar to that for the log normal distribution, increasing to a maximum, then decreasing.

We fit the log logistic model:

```
> rd.logistic_survreg(Surv(month, censor) ~ year, dist = "logistic", link = "log")
> summary(rd.logistic)
```

```
Coefficients:
              Value Std. Error z value        p
(Intercept)  4.7116     0.0344  136.81 0.00e+00
       year -0.2652     0.0431   -6.15 7.88e-10
 Log(scale)  0.0771     0.0144    5.37 7.73e-08

Logistic distribution: Dispersion (scale) est = 1.080194

    Null Deviance: 12138 on 10138 degrees of freedom
Residual Deviance: 12100 on 10136 degrees of freedom
Number of Newton-Raphson Iterations: 3
```

There is a significant difference in recidivism between the two years. However, when we plot the hazard function (Figure 5.7) there is no discernible difference between the two curves.

```
> rd.logistic.y_1 / (1.080194 * month * (1 + exp((log(month) - ifelse(year == 1978, 4.7116, 4.7116 -
0.2652))/1.080194)))
> plot(month, rd.logistic.y, main = "Logistic Distribution – Hazard Function")
```

Here, the hazard function appears to be monotonically decreasing because the maximum lies at a time less than one month. Hence, it resembles most of the other hazard functions which we have seen for these data, instead of that for the log normal distribution.

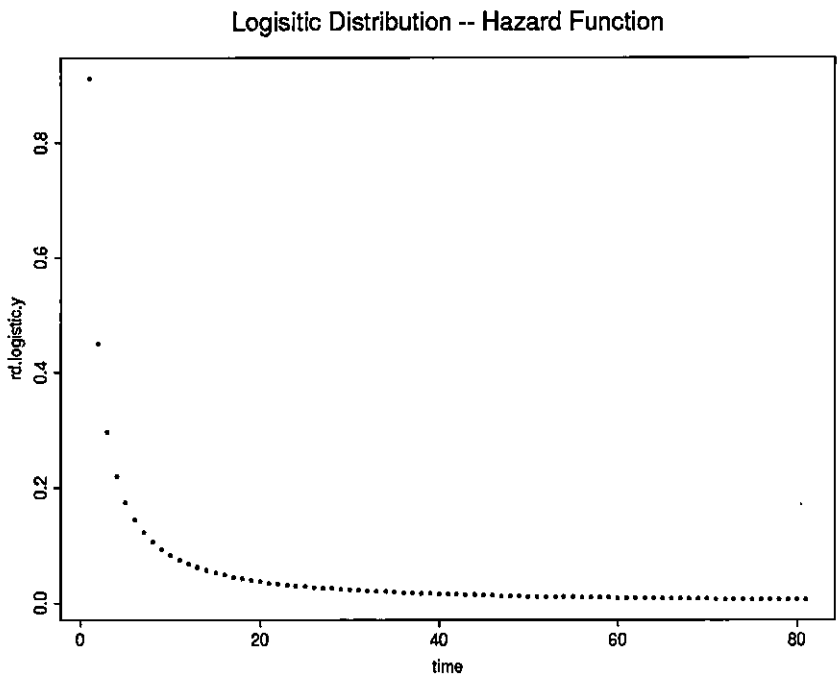



Figure 5.7: Estimated hazard function for recidivism data assuming a log logistic survival distribution.

## 5.9 Gamma Distribution

For two other commonly used survival distributions, the gamma and inverse Gaussian, there are no built-on SPlus functions to fit censored data. As we have seen in the first chapter, the gamma distribution is directly available to SPlus with uncensored data through the glm() function.

The gamma distribution is

$$f(x;\beta,\alpha) = \frac{x^{\alpha-1}e^{-x/\beta}}{\beta^{\alpha}\Gamma(\alpha)} \tag{5.17}$$

where $\Gamma(\alpha)$ is the gamma function. Here, $\beta$ is a scale parameter and $\alpha$ a shape parameter. This is a generalisation of the exponential distribution in a way somewhat similar to the Weibull distribution. When the parameter $\alpha$ is an integer this distribution is that of the sum of $\alpha$ independent exponential variables. Thus, the survival time can be thought of as depending on the sequential completion of $\alpha$ exponential times. Thus, for the Weibull distribution, we have a number of times in parallel, with the first completion (the weakest) determining the survival time. For the gamma, the times are in series, and their sum determines the total survival time.

The survivor function is

$$S(x;\beta,\alpha) = 1 - \int_0^x \frac{x^{\alpha-1}e^{-x/\beta}}{\beta^{\alpha}\Gamma(\alpha)}dx = 1 - \Gamma(x/\beta,\alpha)/\Gamma(\alpha) \tag{5.18}$$

where $\Gamma(x/\beta, \alpha)$ is the incomplete gamma function. The hazard function

$$h(x; \beta, \alpha) = \frac{x^{\alpha-1}e^{-x/\beta}}{\beta^{\alpha}(\Gamma(\alpha) - \Gamma(x/\beta, \alpha))} \tag{5.19}$$

decreases to a constant value $1/\beta$ for $\alpha < 1$ and increases from zero to this same constant for $\alpha > 1$. Note that this constant value is the mortality rate for the exponential distribution, i.e. when $\alpha = 1$.

Since censored observations are present in our data set, we cannot continue with the same example. In Chapter 2, we applied a gamma model to divorce data. Here, we apply it to the data on coal mine disasters of Section 4.6 (Table 4.3), although these are renewal and not survival data. Recall that we found the exponential distribution gave an acceptable fit to these data, separately within each of two periods. The gamma distribution also gave a good fit, although the extra parameters required to fit the two periods separately were statistically insignificant.

In addition to a model with a different distribution for the two time periods, we also try models where the distribution changes with time and with the number of events which have previously occurred. As in Chapter 2, we can fit these models directly, without a special custom function. The one zero observation is given a small positive value.

```
> coal.dataframe_data.frame(list(int = ifelse(coal.data == 0, 0.01, coal.data), br =
factor(seq(coal.data) > 124), event = seq(coal.data), time = cumsum(ifelse(coal.data == 0, 0.01,
coal.data))))
```

This model is the constant distribution model, in which the process maintains the same distribution throughout.

```
> summary(glm(int ~ 1, family = Gamma(), data = coal.dataframe))
```

```
Coefficients:
                 Value   Std. Error  t value
(Intercept) 0.004685688 0.0004994299 9.382073

(Dispersion Parameter for Gaussian family taken to be 2.15852)
    Null Deviance: 324.9827 on 189 degrees of freedom
Residual Deviance: 324.9827 on 189 degrees of freedom
```

This model is a segmented model, with a break where a rate change was identified in the analysis of Section 4.6.

```
> summary(glm(int ~ br, family = Gamma(), data = coal.dataframe))
```

```
Coefficients:
                  Value    Std. Error     t value
```

```
(Intercept)  0.008707531 0.0008398688  10.367728
         br -0.006199110 0.0009025682  -6.868301

(Dispersion Parameter for Gamma family taken to be 1.167795)
    Null Deviance: 324.9827 on 189 degrees of freedom
Residual Deviance: 253.7634 on 188 degrees of freedom
```

In this model, the survival distribution changes as a function of the number of events which have occurred previously.

```
> summary(glm(int ~ event, family = Gamma(), data = coal.dataframe))

Coefficients:
                    Value   Std. Error   t value
(Intercept)  1.105376e-02 1.208214e-03  9.148842
      event -5.058161e-05 7.645025e-06 -6.616279

(Dispersion Parameter for Gamma family taken to be 1.272343)
    Null Deviance: 324.9827 on 189 degrees of freedom
Residual Deviance: 264.6657 on 188 degrees of freedom
```

In this model, the survival distribution changes as a function of the amount of time passed.

```
> summary(glm(int ~ time, family = Gamma(), data = coal.dataframe))

Coefficients:
                    Value   Std. Error    t value
(Intercept)  8.769468e-03 8.167550e-04  10.736963
       time -1.981600e-07 2.589405e-08  -7.652724

(Dispersion Parameter for Gamma family taken to be 1.228825)
    Null Deviance: 324.9827 on 189 degrees of freedom
Residual Deviance: 265.685 on 188 degrees of freedom
```

Of the three models, the segmented model is superior, though none of the models is a particularly good fit in general. Recall from Section 2.4 that models with the gamma distribution use a reciprocal link. The parameters refer to the mortality rate instead of to the average survival time. Thus, the negative parameters indicate an increasing interval between disasters over time, as expected.

## 5.10 Inverse Gaussian Distribution

One remaining distribution, the inverse gaussian, is used for survival times. Suppose that we have an underlying stochastic process whereby survival is until first passage time to a fixed barrier. If

that process resembles Brownian motion with positive drift $\nu$ and variance per unit time $\theta$, the first passage to the barrier at $\lambda$ has an inverse gaussian distribution.

$$f(y;\lambda,\nu,\theta) = \frac{\lambda}{\sqrt{2\pi\theta y^3}} e^{-\frac{(\lambda-\nu y)^2}{2y\theta}} \tag{5.20}$$

Only two of the three parameters are identifiable, so we reparameterize as $\mu = 1/\nu$ and $\sigma^2 = \theta\lambda/\nu^3$, the mean and variance.

The survivor function is

$$S(y;\mu,\sigma^2) = 1 - \Phi\left((\frac{y}{\mu} - 1)\sqrt{\frac{\mu^3}{y\sigma^2}}\right) - e^{2\mu^2/\sigma^2}\Phi\left(-(\frac{y}{\mu} + 1)\sqrt{\frac{\mu^3}{y\sigma^2}}\right) \tag{5.21}$$

and as usual the hazard function is the ratio of $f(.)$ and $S(.)$. The hazard function is very complicated.

The inverse gaussian distribution can be fitted using the inverse.gaussian() family in the glm() function. In Chapter 4 we saw that this model is not suitable for the coal mining disaster data, therefore we fit it only as an illustrative example.

```
> summary(glm(int ~ 1, family = inverse.gaussian(), data = coal.dataframe))

Coefficients:
                   Value   Std. Error  t value
(Intercept) 2.195567e-05 2.340173e-06 9.382073

(Dispersion Parameter for Gaussian family taken to be 0.0101141)
    Null Deviance: 106.4916 on 189 degrees of freedom
Residual Deviance: 106.4916 on 189 degrees of freedom
```

## 5.11 Cox Proportional Hazards Model

The Cox proportional hazards model is a semi-parametric model which has been found to be very useful in the study of survival data. Here we are interested in modelling the hazard at time $y$ as a function of vector-valued explanatory variables $x$ and parameter $\beta$. The model is semi-parametric in the sense that the hazard when $x = 0$, called the *base-line hazard*, is an unspecified function of $y$. Therefore the proportional hazard has the form

$$h(y;x,\beta) = \psi(x,\beta)h_0(y) \tag{5.22}$$

where $h_0(y)$ is the base-line hazard and $\psi(x,\beta)$ is independent of $y$. Typically, $\psi(x,\beta) = e^{\boldsymbol{x}^T\boldsymbol{\beta}}$ so that $\psi(0,\beta) = 1$. The explanatory variables act multiplicatively on the base-line hazard, so that

two subjects with covariates $x_1$ and $x_2$ will have hazard functions which are proportional over time, $y$.

$$\frac{h(y;x_1,\beta)}{h(y;x_2,\beta)} = \frac{\psi(x_1,\beta)}{\psi(x_2,\beta)} \longleftarrow \text{Independent of Y} \tag{5.23}$$

Thus, this model is appropriate when the hazard functions over time are not expected to cross for different values of $x$.

We return to our data on recidivism and fit the Cox proportional hazards model. As previously, we eliminate the zero times and apply the appropriate SPlus function, coxph().

```
> attach(rd.exp.data)
> rd.cox_coxph(Surv(month, censor) ~ year)
> summary(rd.cox)

Call:
coxph(formula = Surv(month, censor) ~ year)

  n= 10139

      coef exp(coef) se(coef)    z      p
year 0.101      1.11   0.0333 3.03 0.0024

     exp(coef) exp(-coef) lower .95 upper .95
year      1.11      0.904      1.04      1.18

Rsquare= 0.001   (max possible= 0.999 )
Likelihood ratio test= 9.25  on 1 df,   p=0.00235
Wald test            = 9.21  on 1 df,   p=0.00241
Efficient score test = 9.22  on 1 df,   p=0.0024
```

We see that there is a significant difference between the two years (p-value of 0.0024). The parameter for the difference between the years is 0.101 (s.e. 0.0333). The hazard function for 1980 is 1.11 times the hazard for 1978, corresponding to shorter recidivism periods in 1980.

$$h_{1978} = h_0(y)$$

$$h_{1980} = h_0(y)e^{0.101} = 1.11h_0(y)$$

We plot the hazard function (Figure 5.8), which is calculated by applying the survfit() function to the object obtained from the coxph() process.

```
> rd.cox.fit_survfit(rd.cox)
> plot(rd.cox.fit$time, rd.cox.fit$n.event / rd.cox.fit$n.risk, main = "Cox Proportional Hazards –
Hazard Function",ylim=c(0,0.02))
```

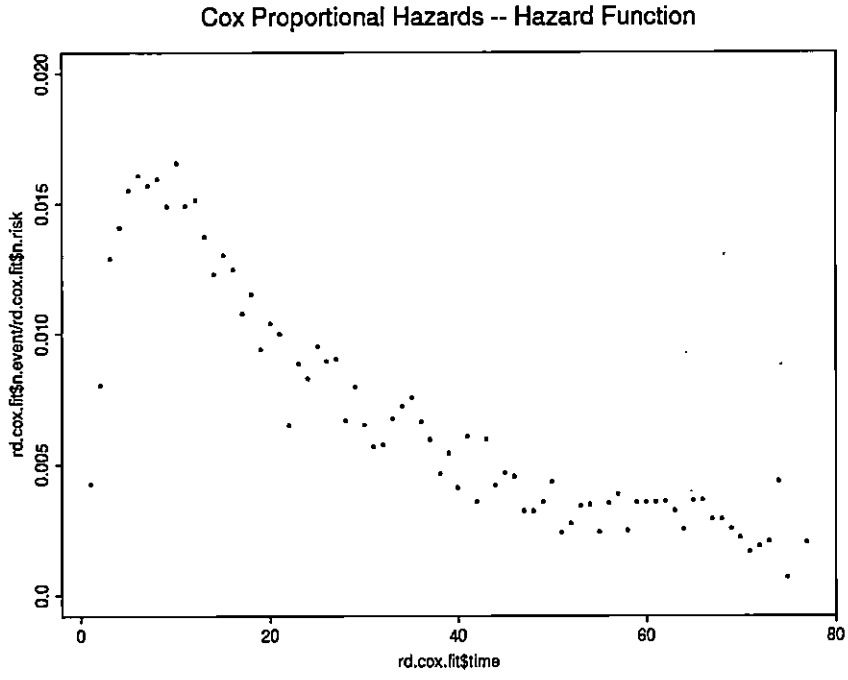



Figure 5.8: Hazard function for recidivism data calculated using Cox proportional hazards.

This hazard function is very similar to the empirical one. This is not surprising because of its close relationship to the Kaplan-Meier estimates.

## 5.12 Piecewise Exponential Distribution

A second semi-parametric distribution is also very useful. Suppose that we cut up the total time interval into a series of discrete sections. If the hazard function is constant within each section, we have an exponential distribution there. Thus, we have a series of different exponential distributions, with different (constant) hazard functions, for the different sections of time. This is known as the *piecewise exponential distribution.* The Cox proportional hazards model of the previous section may be considered as a special case of the piecewise exponential distribution where the cutting points are the distinct death times in the sample.

The proportional hazards assumption for the $i$'th individual can be written as

$$h_{ij} = \exp(\phi_j + \boldsymbol{\beta}^T \boldsymbol{x}_i)$$

Note that the hazard rate is constant over each interval, since the survival distribution over each interval is the exponential.

The log-linear model to fit the piecewise exponential is

$$\log \theta_{ij} = \log e_{ij} + \phi_j + \boldsymbol{\beta}^T \boldsymbol{x}_i \tag{5.24}$$

where $e_{ij}$ are the times at risk for the $i$'th individual in the $j$'th interval, $\phi_j$ are the hazard rate parameters, and $\boldsymbol{\beta}^T \boldsymbol{x}_i$ is the effect of the covariates. ([3] p.288-289)

Note that in the log-linear model above, there is a data point for the behaviour of each individual within each time interval. So, to fit a model with $j$ constant hazards and $n$ observations, the data will have to be expanded to $\sum_{i=1}^{n} c_i$ points before fitting, where $c_i$ is the number of intervals spanned by the $i$'th observation before death or censoring. To do this expansion and fitting, we use the custom piecewise.x() function, which correctly generates the expanded data vectors and fits the log-linear model.

```
function(time, censor, br, x = NULL, freq = NULL)
{
        delta <- br[-1] - br[ - length(br)]
        cm <- cut(time, br)
        cm[is.na(cm)]_0
        w <- rep(0, sum(cm))
        w[cumsum(cm)] <- censor
        phi <- seq(w) - rep(c(0, cumsum(cm[ - length(cm)])), cm)
        rownum <- rep(seq(time), cm)
        e <- delta[phi]
        e[cumsum(cm)] <- time - tapply(e, rownum, sum) + e[cumsum(cm)]
        phi <- factor(phi)
        if(is.null(freq))
                freq <- rep(1, sum(cm))
        else freq <- rep(freq, cm)
        model.formula <- ifelse(is.null(x), "w ~ phi + offset(log(e))",
                "w ~ x + phi + offset(log(e))")
        if(is.matrix(x))
                x <- apply(x, 2, rep, cm)
        else x <- rep(x, cm)
        fit <- glm(model.formula, family = poisson(), weights = freq, maxit
                 = 50)
        fit
}
```

```
> attach(rd.data)
> br_c(0, 2, 5, 10, 15, 20, 30, 40, 50, 60, 100)
> rd.pcexp_piecewise.x(month, censor, br, year, freq)
> summary(rd.pcexp)

Coefficients:
```

```
                Value Std. Error     t value
(Intercept) -5.15061865 0.08943583 -57.5901028
          x  0.09941967 0.03317469   2.9968532
       phi2  0.83418483 0.09999120   8.3425827
       phi3  0.94681251 0.09474523   9.9932473
       phi4  0.81286207 0.09641626   8.4307573
       phi5  0.57696276 0.09940277   5.8042928
       phi6  0.29311067 0.09607739   3.0507768
       phi7 -0.02703505 0.10002658  -0.2702786
       phi8 -0.33751497 0.10595702  -3.1853952
       phi9 -0.65258035 0.12778824  -5.1067326
      phi10 -0.80916409 0.13148896  -6.1538558

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 28558.36 on 1073 degrees of freedom
Residual Deviance: 27448.21 on 1063 degrees of freedom
```

The phi terms are treatment contrasts to the intercept. In order to plot the hazard function, we first generate the real $\hat{\phi}_j$ estimates.

```
> phi_-5.150619 + c(0, coefficients(rd.pcexp)[c(-1, -2)])
```

We now generate the overall hazard function $h_j$ for each interval, ignoring the effect of covariates in this instance.

```
> hazards_exp(phi)
```

Finally, we plot the hazard estimates against the center points of the break vector.

```
> midpoints_br[-length(br)] + (br[-1] - br[-length(br)]) / 2
> plot(midpoints, hazards, ylim = c(0, 0.05), main = "Piecewise Exponential Hazard Estimates")
```

Very often, the piecewise exponential model can provide a reasonably good fit to the data (Figure 5.9) with fewer parameters than the Cox proportional hazards model. However, the problem is to be able to choose the cutting points objectively. If the cutting points correspond theoretically to intervals in which the hazard can be considered constant, then the piecewise exponential is even more appropriate.

We can fit a hazard curve with extra points by increasing the number of intervals defined by the break vector. This will substantially reduce the degrees of freedom, but may more accurately model the shape of the hazard curve. We use a break vector which segments the data into all time periods in which an uncensored death occurs, which is analogous to fitting a Cox proportional hazards model (Section 5.11).

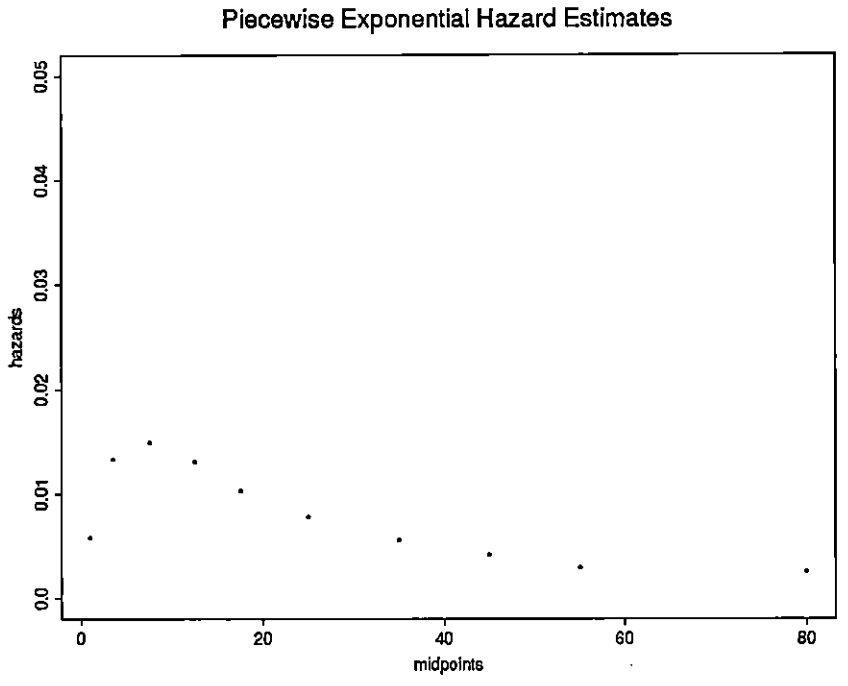



Figure 5.9: Hazard function for recidivism data calculated using piecewise exponential model.

```
> br_c(0,sort(unique(month[censor == 1])), 100)
> rd.pcexp2_piecewise.x(month, censor, br, year, freq)
> hazards_exp(coefficients(rd.pcexp2)[1] + c(0, coefficients(rd.pcexp2)[c(-1, -2)]))
> midpoints_br[-length(br)] + (br[-1] - br[-length(br)]) / 2
> plot(midpoints, hazards, ylim = c(0, 0.05), main = "Piecewise Exponential Hazard Estimates")
> print(rd.pcexp2)
```

The plot of the estimated hazard function (Figure 5.10) is similar to that for the Cox proportional hazards model. Also note that the coefficient for the effect of x (year of release) is 0.1032 for this model; for the Cox proportional hazard model, fitted directly, it was 0.101.

```
Coefficients:
(Intercept)         x      phi2       phi3       phi4       phi5       phi6
 -4.694036 0.1031979 -1.099108  -1.097124 -0.9928102 -0.7950492 -1.384784
      phi7       phi8       phi9      phi10      phi11      phi12      phi13
 -1.250506 -1.021346 -1.003448  -1.350292 -0.9772398 -0.8741373  -1.08143
     phi14      phi15      phi16      phi17      phi18      phi19      phi20
 -1.708296 -1.582138 -1.494228 -0.7452851   -2.59098  -5.095726 -1.506473
     phi21      phi22      phi23      phi24
 -5.095726 -5.095726 -5.095726  -5.095726
```

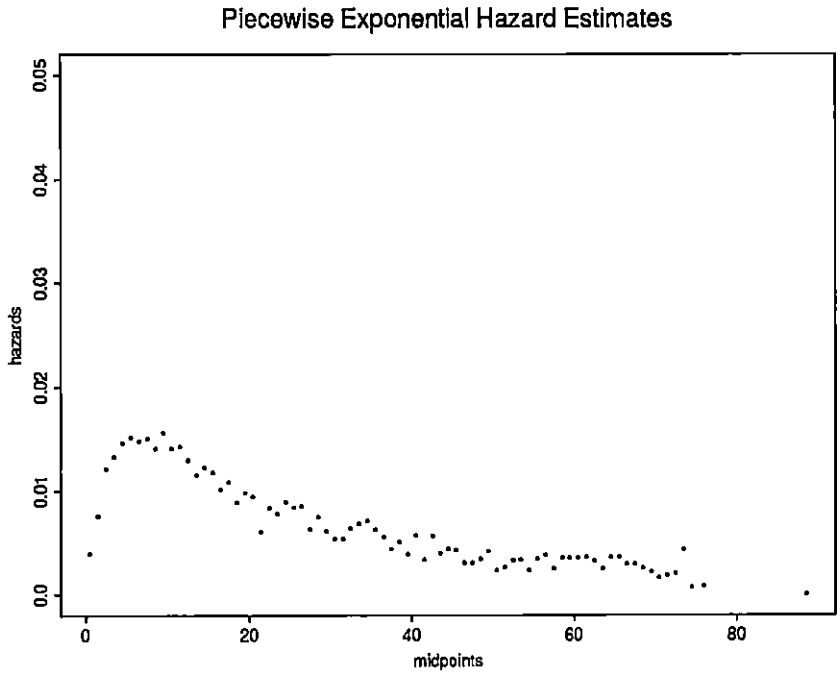



Figure 5.10: Hazard function for recidivism data calculated using piecewise exponential model with breaks for all intervals with uncensored deaths.

```
Degrees of Freedom: 793 Total; 768 Residual
Residual Deviance: 17667.78
```

# Chapter 6

# Growth Curves

## 6.1 Exponential Growth: Continuous Data

One special type of series measured in time concerns changes related to growth. Measurements are made repeatedly on one or more individuals. One goal of such studies is often to predict future growth. Naturally, for a given individual, the observations usually will not be independent, but will be correlated. If several individuals are concerned, the interrelation may even vary among the individuals. However, we shall ignore this problem, as has often been done with growth curves, until after we have looked at classical time series analysis in the next chapter. Here, we treat several examples of the simple case of one series of repeated measurements (on one individual) related to growth, leaving the case of measurements on several individuals for a later chapter, when we shall be able to handle the correlation.

One commonly used growth curve has the exponential form:

$$y = \alpha e^{\beta t} \tag{6.1}$$

As it stands, this is a deterministic function; the curve contains no stochastic element (probability function).

Two common models of this growth curve are frequently employed, both related to the normal distribution. One possibility is that $\log y$ has a normal distribution; that is, $y$ has a log normal distribution. Then, we model $\log y_i$ as

$$\log y_i = \log \alpha + \beta t_i + \epsilon_{1i} \tag{6.2}$$

where $\epsilon_{1i}$ are independent normal errors. The parameters are easily estimated by classical normal theory regression, using logarithms of the growth variable as the response variable, regressed on time.

The second possibility is to fit equation 6.1 directly as a non-linear regression:

$$y_i = \alpha e^{\beta t} + \epsilon_{2i} \tag{6.3}$$

| | | | | |
|---|---|---|---|---|
| 1422 | 1577 | 1700 | 1889 | 2106 |
| 2359 | 2552 | 2829 | 3103 | 3381 |
| 3492 | 3736 | 4120 | 4619 | 4731 |
| 4906 | 5860 | 6331 | 6686 | 7145 |

Table 6.1: Gross domestic fixed capital formation in the United Kingdom, 1948-1967, in millions of points. (Oliver, 1970)

where $\epsilon_{2i}$ are independent normal errors. Fortunately, this can also be easily accomplished in SPlus using nls(), the non-linear regression routines.

In classical terms, the first model has multiplicative errors since $y_i = \alpha e^{\beta t_i} e^{\epsilon_{1i}} = \alpha e^{\beta t_i} \delta_i$, and the second model has additive errors. The resulting models can differ significantly. However, if only one growth series is available the relative goodness of fit of the two models cannot be tested quantitatively. Using graphs of observed and fitted values, however, we can make a relative assessment of fit.

As an example, we choose the gross domestic fixed capital formation in the United Kingdom from 1948 to 1967 (Table 6.1), and predict that for the two following years.

In SPlus, prediction simply involves extending the vectors the required number of years and providing a weight vector with one for the observations and zero for the values to be predicted. We take 1960 as the base year with value 0. We first fit the log normal model (Equation 6.2) and plot the results.

```
> gdf.data_scan("data/Chpt5-1")
> gdf.data_data.frame(logdata = log(c(gdf.data, NA, NA)), year = seq(22) - 13)
> gdf.lognorm_lm(logdata ~ year, data = gdf.data, subset = year < 8)
> summary(gdf.lognorm)


Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)  8.3202   0.0093   896.2891   0.0000
       year  0.0841   0.0015    56.9405   0.0000

Residual standard error: 0.03809 on 18 degrees of freedom
Multiple R-Squared: 0.9945
F-statistic: 3242 on 1 and 18 degrees of freedom, the p-value is 0

> gdf.output_cbind(exp(gdf.data$logdata), exp(predict(gdf.lognorm, gdf.data)), exp(gdf.data$logdata)
- exp(predict(gdf.lognorm, gdf.data)))
> dimnames(gdf.output)_list(null(),c("Observed Values","Fitted Values", "Difference"))
> print(gdf.output)
```

```
      Observed Values Fitted Values Difference
 [1,]            1422       1496.631  -74.63101
 [2,]            1577       1627.946  -50.94583
 [3,]            1700       1770.782  -70.78224
 [4,]            1889       1926.151  -37.15115
 [5,]            2106       2095.152   10.84782
 [6,]            2359       2278.981   80.01860
 [7,]            2552       2478.940   73.06015
 [8,]            2829       2696.443  132.55729
 [9,]            3103       2933.029  169.97067
[10,]            3381       3190.374  190.62588
[11,]            3492       3470.298   21.70159
[12,]            3736       3774.783  -38.78334
[13,]            4120       4105.984   14.01614
[14,]            4619       4466.244  152.75600
[15,]            4731       4858.113 -127.11347
[16,]            4906       5284.366 -378.36567
[17,]            5860       5748.017  111.98265
[18,]            6331       6252.350   78.65004
[19,]            6686       6800.933 -114.93285
[20,]            7145       7397.649 -252.64855
[21,]              NA       8046.720         NA
[22,]              NA       8752.742         NA
```

```
> matplot(gdf.data$year, cbind(exp(predict(gdf.lognorm, gdf.data)), exp(gdf.data$lognorm)),
main="Exponential Growth with Log Normal Distribution", pch = "FO")
```

In the plot shown in Figure 6.1, the fitted values are the 'F's and the observed are the 'O's.

We now fit the additive errors model (Equation 6.3) using nls(), the non-linear least squares function. Because nls() does not accept weights, we cannot include an extended data vector with zero weights to obtain predicted values as we did for the multiplitative model. Data must be passed to nls() in a data frame, with starting values specified ahead of time in the data frame. Note that we can obtain starting values for the parameter values from the results of fitting the multiplicative errors model (Equation 6.2).

```
> gdf.nls.data_data.frame(gdf.data = gdf.data, gdf.year = gdf.year[1:20])
```

From the multiplicative model, we know that $\log \alpha = 8.32$, so $\alpha = e^{8.32} = 4105$. We will use a starting value close to 4105.

```
> param(gdf.nls.data, "alpha")_4000
```

From the multiplicative model, we know that $\beta = 0.084$. We will use a starting value close of 0.

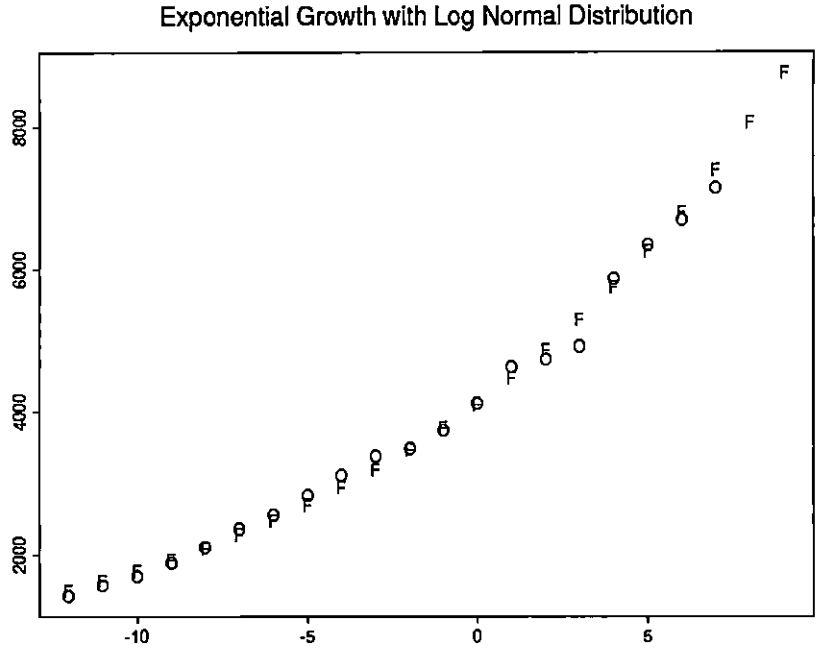



Figure 6.1: Plot of observed and fitted values in multiplicative errors fit of GDFC growth curve.

```
> param(gdf.nls.data, "beta")_0
> gdf.nls_nls(gdf.data ~ alpha * exp(beta * gdf.year), data = gdf.nls.data)
> summary(gdf.nls)

Formula: gdf.data ~ alpha * exp(beta * gdf.year)

Parameters:
           Value  Std. Error  t value
alpha 4106.170000 33.33720000 123.1710
 beta    0.081441  0.00160549  50.7265

Residual standard error: 134.997 on 18 degrees of freedom

> gdf.output_cbind(gdf.data, gdf.nls$fitted.values, gdf.data - gdf.nls$fitted.values)
> dimnames(gdf.output)_list(null(),c("Observed Values","Fitted Values","Difference"))
> print(gdf.output)

     Observed Values Fitted Values  Difference
[1,]            1422       1545.269 -123.269349
[2,]            1577       1676.384  -99.384196
[3,]            1700       1818.624 -118.624031
```

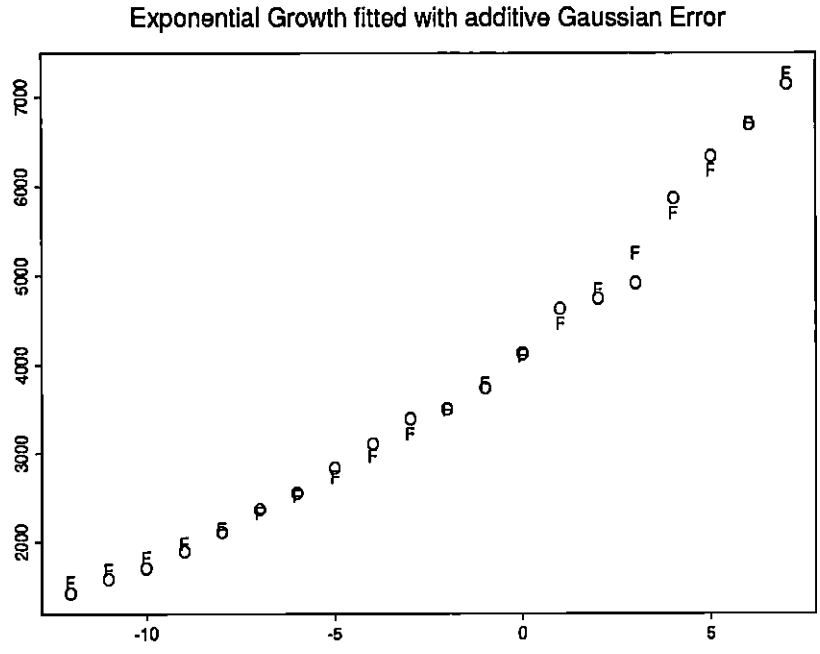



Figure 6.2: Plot of observed and fitted values in additive errors fit of GDFC growth curve.

```
 [4,]          1889      1972.933  -83.932801
 [5,]          2106      2140.335  -34.334544
 [6,]          2359      2321.940   37.059812
 [7,]          2552      2518.955   33.045076
 [8,]          2829      2732.686   96.313800
 [9,]          3103      2964.552  138.447599
[10,]          3381      3216.092  164.907740
[11,]          3492      3488.975    3.024928
[12,]          3736      3785.012  -49.011768
[13,]          4120      4106.167   13.833063
[14,]          4619      4454.572  164.428139
[15,]          4731      4832.539 -101.538659
[16,]          4906      5242.576 -336.575633
[17,]          5860      5687.404  172.596090
[18,]          6331      6169.976  161.024495
[19,]          6686      6693.493   -7.492906
[20,]          7145      7261.430 -116.430334
```

> matplot(gdf.year[1:20], cbind(gdf.nls$fitted.values, gdf.data), main = "Exponential Growth fitted with additive Gaussian Error", pch = "FO")

The graphs of the two models (Figure 6.1 and Figure 6.2) appear similar and both seem close to

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 |
| 1 | 0 | 2 | 0 | 1 | 1 | 5 | 4 | 3 | 0 | 5 | 7 |
| 6 | 5 | 8 | 4 | 5 | 6 | 10 | 14 | 7 | 14 | 8 | 16 |
| 16 | 15 | 18 | 16 | 14 | 12 | 17 | 24 | 23 | 25 | 22 | 21 |
| 25 | 30 | 26 | 35 | 38 | 27 | 24 | 29 | 39 | 34 | 38 | 47 |
| 33 | 44 | 35 | 35 | | | | | | | | |

Table 6.2: AIDS cases reported in the UK, January 1982 - April 1987 (Healy and Tillett, 1988)

the observed values. We can compare the sum of the prediction errors squared for the two models to obtain a measure of the quality of the fit of each model.

```
sum((exp(gdf.lognorm$fitted.values[1:20]) - gdf.data) ^ 2)
```

```
[1] 389804.9
```

```
sum((gdf.nls$fitted.values - gdf.data) ^ 2)
```

```
[1] 328034.8
```

The non-linear, additive errors model has a smaller sum of squares statistic than the multiplicative errors model.

## 6.2 Exponential Growth: Count Data

When the data are a series of counts, an alternative model may be more suitable than the normal or log normal models. The log linear model is most often applied to count data and a log linear regression model can directly yield an exponential growth model, for example as $\log E(\text{Count}) = \text{Time}$.

We take the example of the growth in the number of AIDS cases diagnosed in the United Kingdom between January 1982 and April 1987 (Table 6.2). These data contain a certain deficiency, in that not all recent cases diagnosed may have been reported at the time the data were collected, so that the most recent values may be underestimates.

We wish to predict the number of cases diagnosed for each month until the end of the year. We read the data, set up and fit two models.

```
> aids.data_scan("data/Chpt5-2")
> aids.dataframe_data.frame(data = c(aids.data, rep(0, 8)), time = 1:72, time2 = 1:72 * 1:72, w =
c(rep(1,64),rep(0,8)))
```

### Linear Model

```
> aids.fit.lin_glm(data ~ time, weights = w, data = aids.dataframe, family = poisson())
> summary(aids.fit.lin)

Coefficients:
                Value Std. Error   t value
(Intercept) 0.23209781 0.12215824  1.899977
       time 0.05825362 0.00240364 24.235581

(Dispersion Parameter for Poisson family taken to be 1 )
    Null Deviance: 878.5157 on 71 degrees of freedom
Residual Deviance: 96.70882 on 62 degrees of freedom
```

### Quadratic Model

```
> aids.fit.quad_glm(data ~ time + time2, weights = w, family = poisson(), data = aids.dataframe)
> summary(aids.fit.quad)

Coefficients:
                   Value   Std. Error   t value
(Intercept) -1.359019818 0.3258625747 -4.170531
       time  0.145166873 0.0154770970  9.379464
      time2 -0.001039484 0.0001773718 -5.860480

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 878.5157 on 71 degrees of freedom
Residual Deviance: 56.64329 on 61 degrees of freedom

> matplot(aids.dataframe$time, cbind(c(aids.data, rep(NA, 8)), aids.fit.quad$fitted.values),
main="Observed and Fitted Values vs Time", xlab="Time", ylim=c(0,60), pch="OF")
```

The simple exponential growth model, where $\log E(\text{Counts})$ is linear in time, did not fit very well—the residual deviance is 96.7 on 62 degrees of freedom. We added a quadratic term to the model, which significantly improved the fit; note the change in residual deviance from 96.7 to 56.6. We see from the plot of observed and fitted values (Figure 6.3) that the quadratic model predicts a significant levelling off of the curve in the latter half of 1987 (the limit of the ordinate has been set to 60 for easier comparison with some of the following graphs). However, this could be due in part to the under-reporting of the most recent months, as mentioned above. We eliminate the four last observations and refit the model.

```
> aids.dataframe$w_c(rep(1,60),rep(0,12))
```

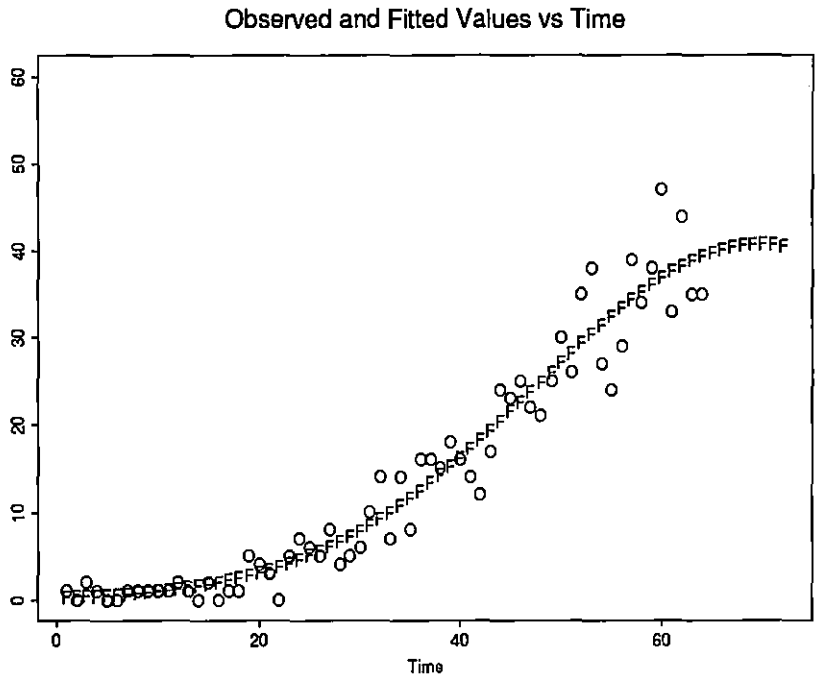



Figure 6.3: Plot of observed and fitted values for quadratic model of growth of AIDS incidence.

### Linear Model

```
> aids.fit.lin2_glm(data ~ time, weights = w, family = poisson(), data = aids.dataframe)
> summary(aids.fit.lin2)

Coefficients:
                Value  Std. Error    t value
(Intercept) 0.008156646 0.136965169  0.0595527
       time 0.064509488 0.002854147 22.6020217

(Dispersion Parameter for Poisson family taken to be 1 )
    Null Deviance: 766.683 on 71 degrees of freedom
Residual Deviance: 74.3246 on 58 degrees of freedom
```

### Quadratic Model

```
> aids.fit.quad2_glm(data ~ time + time2, weights = w, family = poisson(), data = aids.dataframe)
> summary(aids.fit.quad2)

Coefficients:
                  Value   Std. Error   t value
```

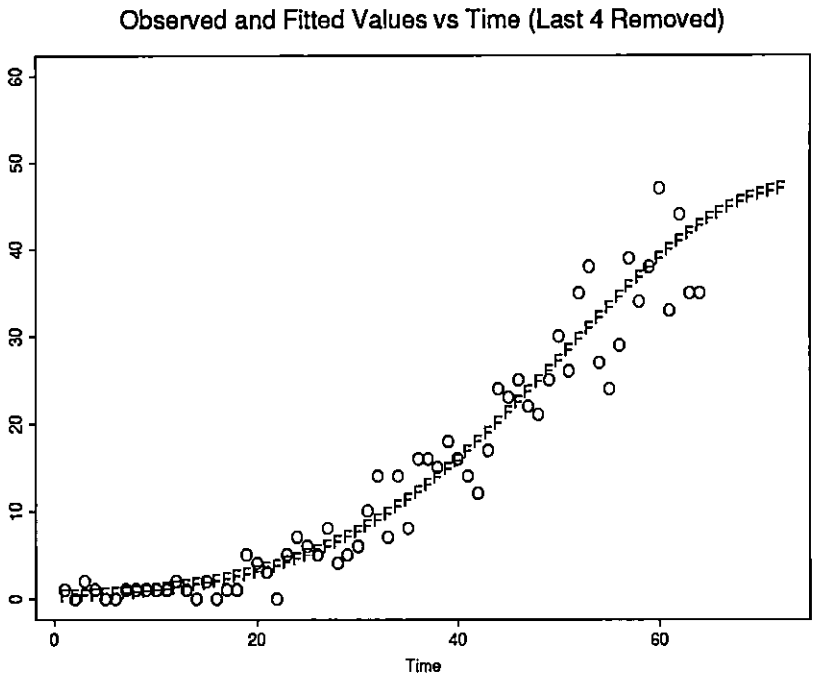



Figure 6.4: Plot of observed and fitted values for quadratic model (4 obs removed) of growth of AIDS incidence.

```
(Intercept) -1.2289288250 0.3448532890 -3.563628
       time  0.1365308898 0.0175180310  7.793735
      time2 -0.0009164733 0.0002145237 -4.272130

(Dispersion Parameter for Poisson family taken to be 1)
    Null Deviance: 766.683 on 71 degrees of freedom
Residual Deviance: 53.77199 on 57 degrees of freedom
```

```
> matplot(aids.dataframe$time, cbind(c(aids.data, rep(NA, 8)), aids.fit.quad2$fitted.values),
main="Observed and Fitted Values vs Time (Last 4 Removed)", xlab="Time", ylim=c(0,60),
pch="OF")
```

We see in Figure 6.4 that the predictions are now higher for 1987.

| **Observed** | **Predicted** |
|---|---|
| 33 | 40.0 |
| 44 | 41.0 |
| 35 | 41.9 |
| 35 | 42.7 |

Those for the last four observed months confirm the probable under-reporting. However, the quadratic term is still necessary in order to have a sufficiently well-fitting model.

## 6.3 The Logistic Growth Curve

The quadratic term in the exponential growth curve of the previous section was a rather ad hoc solution for obtaining a better fit. However, the quadratic model did indicate a leveling off of the curve. In such cases, a logistic is often more suitable. The logistic model takes the form

$$E(Y) = N\left(\frac{e^{\beta_0+\beta_1 t}}{1+e^{\beta_0+\beta_1 t}}\right) \tag{6.4}$$

which has an asymptote at $y = N$.

For count data, we may think of this model as $E(Y) = N\pi(t)$ at each time period $t$, where $\pi(t) \in [0,1]$ and we do not know the total, $N$. For the AIDS example, this would imply that the number diagnosed each month is a proportion of the unknown total number which will ever be diagnosed in one month—the asymptote. It is possible to estimate this total by an iterative procedure which minimizes the deviance. We fit models which assume that $y_t$ has a binomial distribution with parameters $N$ and $\pi(t)$, for a sequence of values for $N$.

We apply this approach to the AIDS data, using a special function for the iterations. Since the previous analysis indicated that we should ignore the last few observations, we leave them out here too.

We begin by creating vectors for our response and covariates, and then creating a vector of possible asymptote values to test, ranging from 50 to 100.

```
> aids.data_aids.data[1:60]
> aids.time_seq(60)
> aids.asymptotes_50:100
```

Next we make a temporary function, which will calculate the deviance of the model at a given asymptote value. Then we use the sapply() function to recursively generate the deviances of the model at each asymptote value in aids.asymptotes. Finally, we recover the asymptote value associated with the model with minimum deviance.

```
> tmp.function_function(asym, response, covariate) deviance(glm(cbind(response, rep(asym,
length(response)) - response) ~ covariate, family = binomial(link = logit)))
> aids.deviances_sapply(aids.asymptotes, tmp.function, aids.data, aids.time)
> aids.asymptote_aids.asymptotes[aids.deviances == min(aids.deviances)]
> print(aids.asymptote)
> plot(aids.asymptotes, aids.deviances, main = "Deviance Function", ylab = "Residual Deviance",
xlab = "N")

[1] 89
```

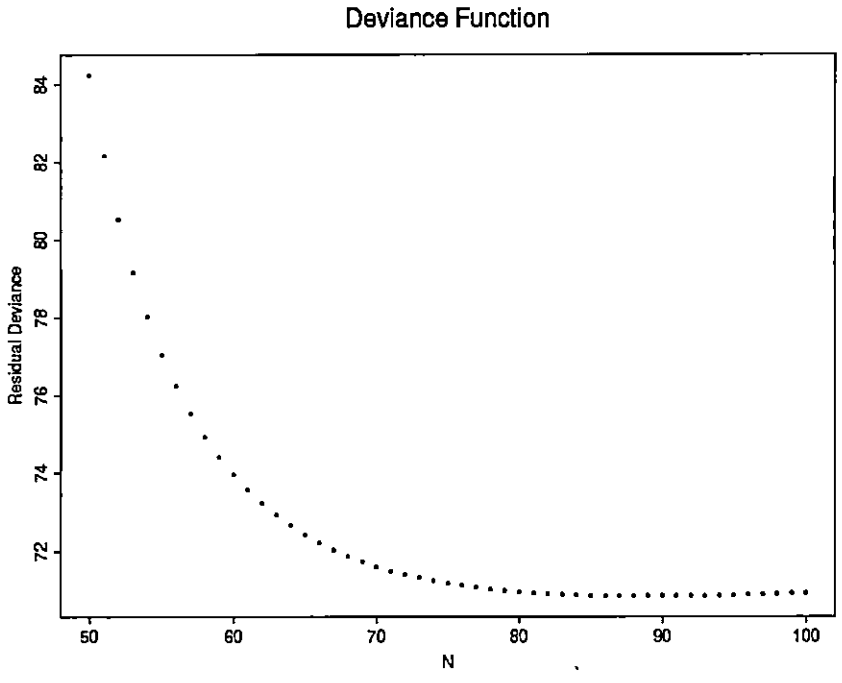



Figure 6.5: Deviance curve for binomial models of logistic growth with varying values of $N$.

The minimum deviance is for asymptote $N = 89$. However, as we see in Figure 6.5, the deviance curve is very flat, so there is no strongly defined minimum value. We fit the model with $N = 89$, display the parameter estimates, and plot fitted values (Figure 6.6).

```
> aids.glm_glm(cbind(aids.data, rep(89, length(aids.data)) - aids.data) ~ aids.time, family =
binomial(link = logit))
> matplot(aids.time, cbind(aids.data, fitted(aids.glm) * aids.asymptote), main = "Observed (O) and
Fitted (F) vs Time", xlab = "Time", pch = "OF", ylim = c(0, 60))
> summary(aids.glm)

Coefficients:
                Value  Std. Error   t value
(Intercept) -4.81148950 0.152574785 -31.53529
  aids.time  0.07918954 0.003328393  23.79212

(Dispersion Parameter for Binomial family taken to be 1)
    Null Deviance: 896.3648 on 59 degrees of freedom
Residual Deviance: 70.86086 on 58 degrees of freedom
```

The predicted values for the last four observed months are higher than those for the quadratic exponential model. The growth curve does not flatten off as much as for the previous model, although we know that it will be completely flat at 89 cases.

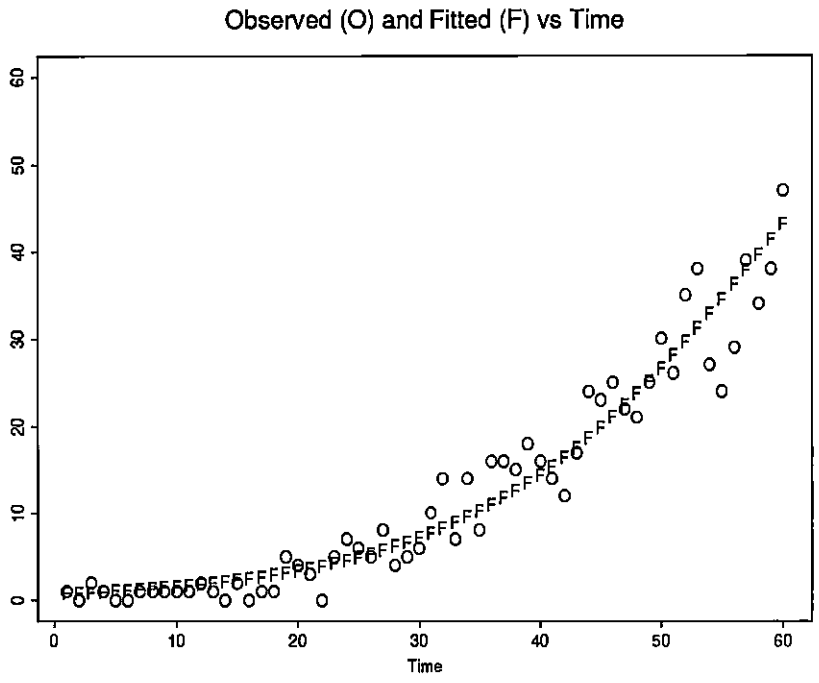



Figure 6.6: Plot of observed and fitted values for logistic model of growth of AIDS incidence.

## 6.4 The Gomperz Growth Curve

A second growth curve for count data, which has an asymptote, is the Gomperz curve:

$$E(Y) = N(1 - e^{-e^{\beta_0 + \beta_1 t}}) \tag{6.5}$$

The procedure is the same as for the logistic model. The only change required is to use the complementary log-log link, $\log(-\log(1-\pi(t)))$, instead of the (default) logistic link. However, for these data, the asymptote is very high and the deviance function very flat. We modify the iteration to take steps of 10 cases instead of 1.

```
> aids.asymptotes_seq(100,700,by=10)
```

Next we modify the temporary function, to use a complementary log-log link. Then we use the sapply() function to recursively generate the deviances of the model at each asymptote value in aids.asymptotes. Finally, we recover the asymptote value associated with the model with minimum deviance.

```
> tmp.function_function(asym, response, covariate) deviance(glm(cbind(response, rep(asym,
length(response)) - response) ~ covariate, family = binomial(link = cloglog)))
> aids.deviances_sapply(aids.asymptotes, tmp.function, aids.data, aids.time)
> aids.asymptote_aids.asymptotes[aids.deviances == min(aids.deviances)]
> print(aids.asymptote)
```

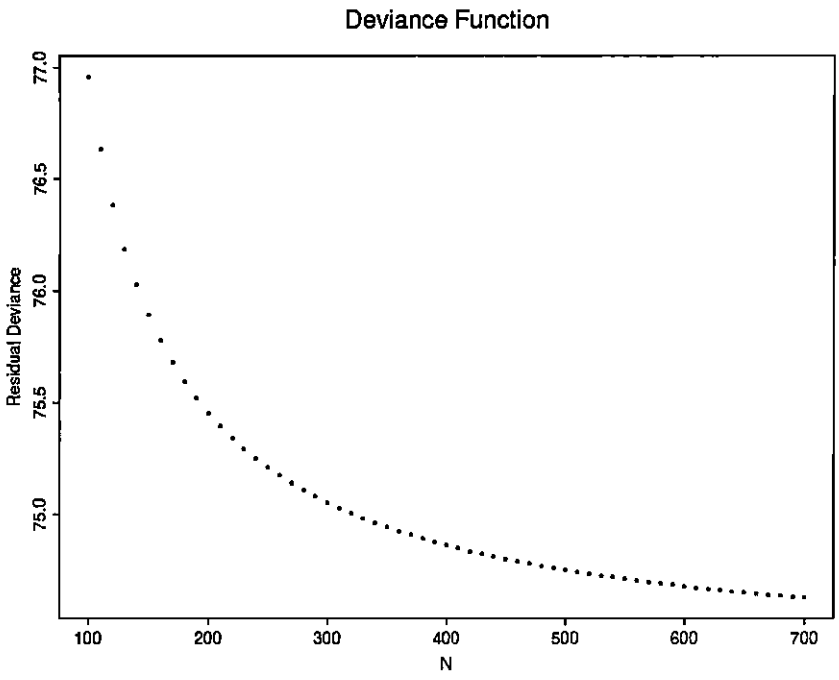



Figure 6.7: Deviance curve for binomial models of Gomperz growth with varying values of $N$.

```
[1] 700

> plot(aids.asymptotes, aids.deviances, main = "Deviance Function", ylab = "Residual Deviance",
xlab = "N")
```

The minimum deviance is for asymptote $N = 700$. Note that a minimum was not reached. The deviance curve (Figure 6.7) is very flat, and continues to decrease so there is no strongly defined minimum value. We fit the model with $N = 700$, display the parameter estimates, and plot fitted values (Figure 6.8).

```
> aids.glm_glm(cbind(aids.data, rep(700, length(aids.data)) - aids.data) ~ aids.time, family =
binomial(link=cloglog))
> matplot(aids.time, cbind(aids.data, fitted(aids.glm) * aids.asymptote), main = "Observed (O) and
Fitted (F) vs Time", xlab = "Time", pch = "OF", ylim = c(0, 60))
> summary(aids.glm)


Coefficients:
                Value  Std. Error   t value
(Intercept) -6.55505205 0.136832320 -47.90573
  aids.time  0.06515954 0.002853979  22.83112
```

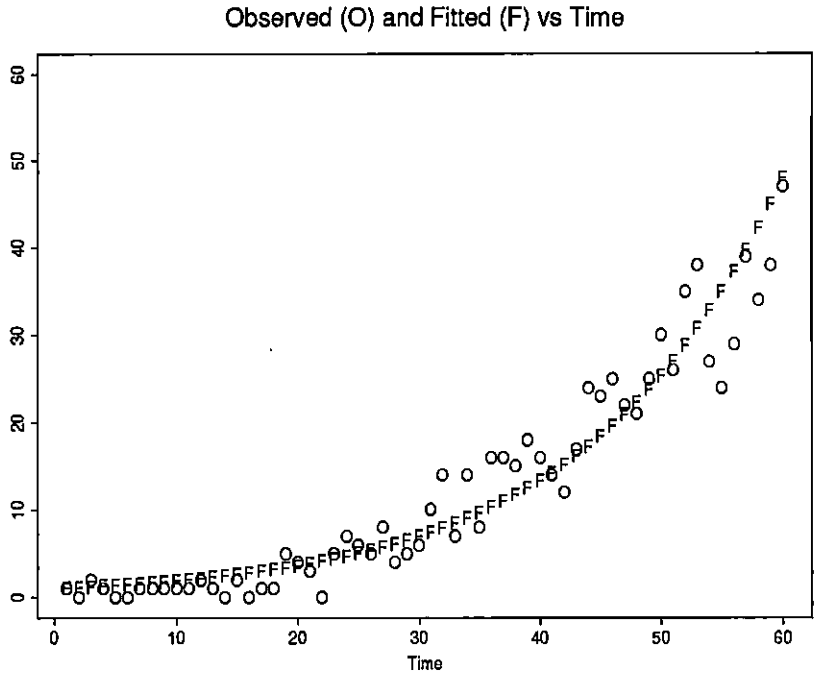



Figure 6.8: Plot of observed and fitted values for Gomperz model of growth of AIDS incidence.

```
(Dispersion Parameter for Binomial family taken to be 1)
    Null Deviance: 780.3207 on 59 degrees of freedom
Residual Deviance: 74.62824 on 58 degrees of freedom
```

This curve (Figure 6.8) climbs much more steeply than any of the others plotted. As with the exponential growth curve, we might want to add a quadratic term. Obviously, the predicted values for the last four observed months are also much higher than in the previous two cases.

## 6.5 Supplement: Fitting with nls()

The nls() function in SPlus has the ability to fit the logistic and Gomperz growth curves directly, assuming constant Gaussian error. The only added difficulty in fitting with nls() is specifying the starting values to the fitting algorithm. The starting values are specified within the data frame before the model is fit; here, we choose values which are similar in magnitude and sign to those we obtained using the iterative fitting procedures above. Once good starting values are chosen, we find that the non-linear fits are at least as good as those achieved by the iterative method of the previous two sections.

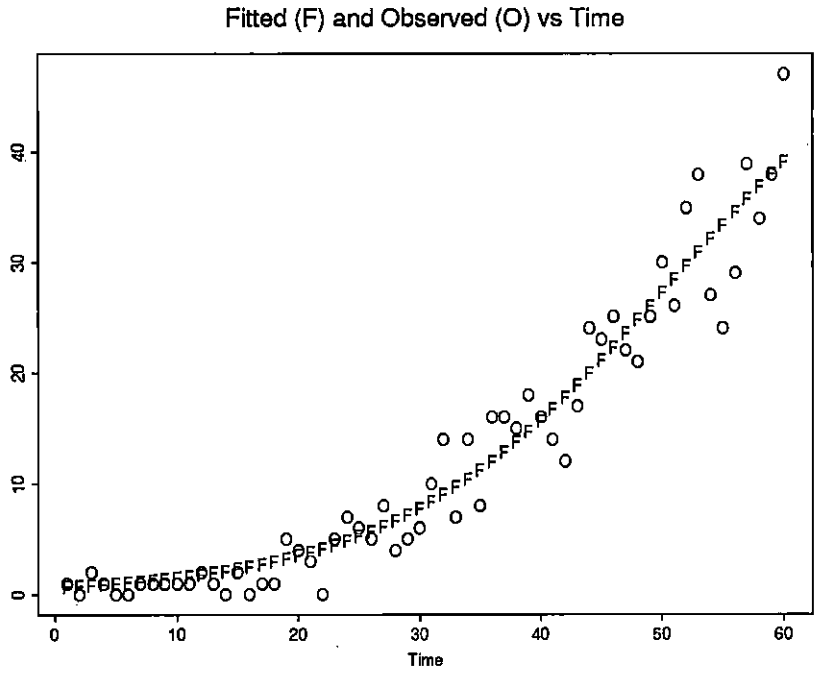



Figure 6.9: Plot of observed and fitted values for logistic model of growth of AIDS incidence.

### 6.5.1 Logistic

Here we fit the model

$$y_i = N\left(\frac{e^{\beta_0+\beta_1 t}}{1+e^{\beta_0+\beta_1 t}}\right) + \epsilon_{1i}$$

where the $\epsilon_{1i}$ are assumed to be normal independent errors.

```
> aids.dataframe_data.frame(aids.data, aids.time)
> param(aids.dataframe,"alpha")_-7
> param(aids.dataframe,"beta")_0.1
> param(aids.dataframe,"N")_60
> aids.nls_nls(aids.data ~ N * exp(alpha + beta * aids.time) / (1 + exp(alpha + beta * aids.time)),
data = aids.dataframe)
> matplot(aids.time, cbind(aids.data, fitted(aids.nls)), pch = "OF", main = "Fitted (F) and
Observed (O) vs Time", xlab = "Time")
> summary(aids.nls)


Parameters:
           Value Std. Error   t value
alpha  -4.4861500  0.2569820 -17.45710
```

```
 beta   0.0880546  0.0112902   7.79919
    N  56.7677000 10.5848000   5.36312

Residual standard error: 3.16759 on 57 degrees of freedom
```

### 6.5.2 Gomperz

We fit the model

$$y_i = N(1 - e^{-e^{\beta_0 + \beta_1 t}}) + \epsilon_{2i}$$

where the $\epsilon_{2i}$ are assumed to be normal independent errors. We change the starting parameter for $N$ before fitting the Gomperz growth curve, to better reflect the value obtained by the iterative procedure.

```
> param(aids.dataframe,"N")_600
> aids.nls_nls(aids.data ~ N - N * exp(-1 * exp(alpha + beta * aids.time)), data=aids.dataframe)
> matplot(aids.time, cbind(aids.data, fitted(aids.nls)), pch = "OF", main = "Fitted (F) and
Observed (O) vs Time", xlab = "Time")
```

```
Parameters:
           Value Std. Error   t value
alpha -4.0272000 0.22434800 -17.95070
 beta  0.0788255 0.00819401   9.61990
    N 45.1593000 6.51947000   6.92684

Residual standard error: 3.20917 on 57 degrees of freedom
```

In this case, the nls() fit is substantially different from the iterative fit obtained earlier, primarily because of the low value of $N$ calculated by the algorithm. The effect of the small $N$ is visible on the graph (Figure 6.10) of fitted values, where the growth of the fitted curve is beginning to slow as it nears the asymptote. Note that the estimated asymptote is less than the largest observation.

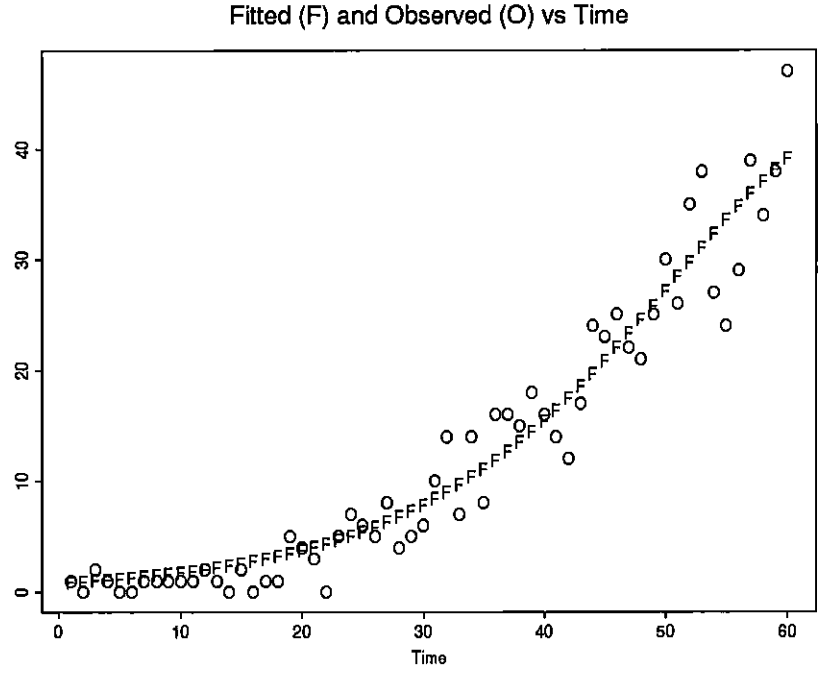



Figure 6.10: Plot of observed and fitted values for Gomperz model of growth of AIDS incidence.

# Chapter 7

# Time Series: The Time Domain

## 7.1 Trends and Correlograms

Many of the data sets which might best be handled by one of the approaches of the previous chapters would often be treated by a classical time series analysis, assuming a normal distribution. Most often, this simply amounts to fitting a multiple regression, with (a dynamic model) or without (a static model) lagged variables. Checks are then made on the various assumptions underlying the model; serial correlation of the residuals, non-normality, heteroscedasicity, and mis-specification of the linear model. Except for the first, these are the common verifications applied to any model.

The first thing to do with any time series is to plot the response variable against time (this is also true for all of the previous data sets presented). Let us consider a typical, fairly long, time series, the student enrolment at Yale University from 1796 to 1975 (Table 7.1). Note that an obvious approach to these data would be to model them using a growth curve, as in Chapter 6.

We read the data and then plot them against time using the tsplot() function.

```
> enrol.data_scan("data/Chpt6-1")
> tsp(enrol.data)_c(1796,1975,1)
> tsplot(enrol.data, xlab = "Time", ylab = "Enrolment")
```

We see that the series is fairly flat at first (Figure 7.1), up until about 1885, then rises sharply in this century, with drops for the two world wars. As in a number of previous data sets, this change in slope may indicate a modification of the process producing the observations. The enrolment mechanism may have changed at the end of the last century. In addition, we have important perturbations, here due to war.

In many of the models of the preceding chapters, we have assumed little or no effect of previous observations on the present ones, conditional on the explanatory variables. One exception was the Markov chains of Chapter 3. Here, we are more concerned with verifying that the hypothesis of

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 115 | 123 | 168 | 195 | 217 | 217 | 242 | 233 | 200 | 222 | 204 | 196 | 183 |
| 228 | 255 | 305 | 313 | 328 | 350 | 352 | 298 | 333 | 349 | 376 | 412 | 407 |
| 481 | 473 | 459 | 470 | 454 | 501 | 474 | 496 | 502 | 469 | 485 | 536 | 514 |
| 572 | 570 | 564 | 561 | 608 | 574 | 550 | 537 | 559 | 542 | 588 | 584 | 522 |
| 517 | 531 | 555 | 558 | 604 | 594 | 605 | 619 | 598 | 565 | 578 | 641 | 649 |
| 599 | 617 | 632 | 644 | 682 | 709 | 699 | 724 | 736 | 755 | 809 | 904 | 955 |
| 1031 | 1051 | 1021 | 1039 | 1022 | 1003 | 1037 | 1042 | 1096 | 1092 | 1086 | 1075 | 1134 |
| 1245 | 1365 | 1477 | 1645 | 1784 | 1969 | 2202 | 2350 | 2415 | 2615 | 2645 | 2674 | 2684 |
| 2542 | 2712 | 2816 | 3142 | 3138 | 3806 | 3605 | 3433 | 3450 | 3312 | 3282 | 3229 | 3288 |
| 3272 | 3310 | 3267 | 3262 | 2006 | 2554 | 3306 | 3820 | 3930 | 4534 | 4461 | 5155 | 5316 |
| 5626 | 5457 | 5788 | 6184 | 5914 | 5815 | 5631 | 5475 | 5362 | 5493 | 5483 | 5637 | 5747 |
| 5744 | 5694 | 5454 | 5036 | 5080 | 4056 | 3363 | 8733 | 8991 | 9017 | 8519 | 7745 | 7688 |
| 7567 | 7555 | 7369 | 7353 | 7664 | 7488 | 7665 | 7793 | 8129 | 8221 | 8404 | 8333 | 8614 |
| 8539 | 8654 | 8666 | 8665 | 9385 | 9214 | 9231 | 9219 | 9427 | 9661 | 9721 | | |

Table 7.1: Student enrolment at Yale University, 1796-1975. (Anscombe, 1981, p.130)

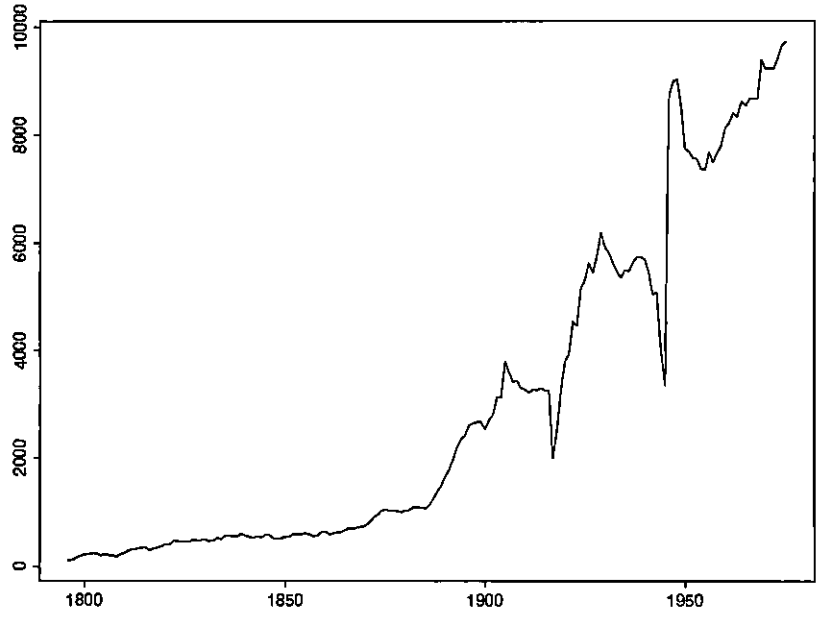



Figure 7.1: Time series plot of Yale enrolment data.

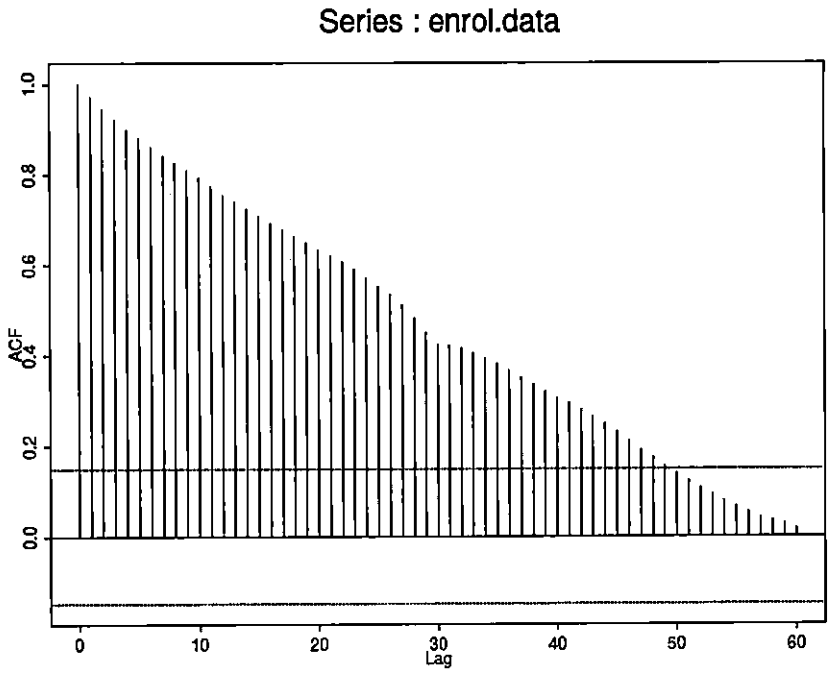



Figure 7.2: Autocorrelation plot of Yale enrolment data.

independence is correct, and if not, modifying it to take into account the lagged time effects. Such verification is often simpler in the present case, where we assume a normal distribution.

One of the simplest, but most important, means of examining the independence hypothesis is by plotting a *correlogram*, also known as the *autocorrelation function*. This provides the correlations of present observations with those of successive periods in the past. We use the acf() function to plot a correlogram to the active graphics device.

```
> acf(enrol.data, lag.max = 60)
```

Figure 7.2 shows that each observation has a strong dependency relationship with previous observations a long time in the past. The correlation with an observation 30 years before is 0.507 and 60 years before is 0.027. Such a correlogram is characteristic of time series with a trend. Points outside the two parallel horizontal lines indicate departures from white noise at the 5% level. ([6] p.39)

Let us now attempt to remove the trend by fitting a linear regression on time to the data. Obviously, we need at least a quadratic relationship.

```
> enrol.year_seq(enrol.data)
> enrol.year2_enrol.year ^ 2
> enrol.lm1_lm(enrol.data ~ enrol.year + enrol.year2)
```

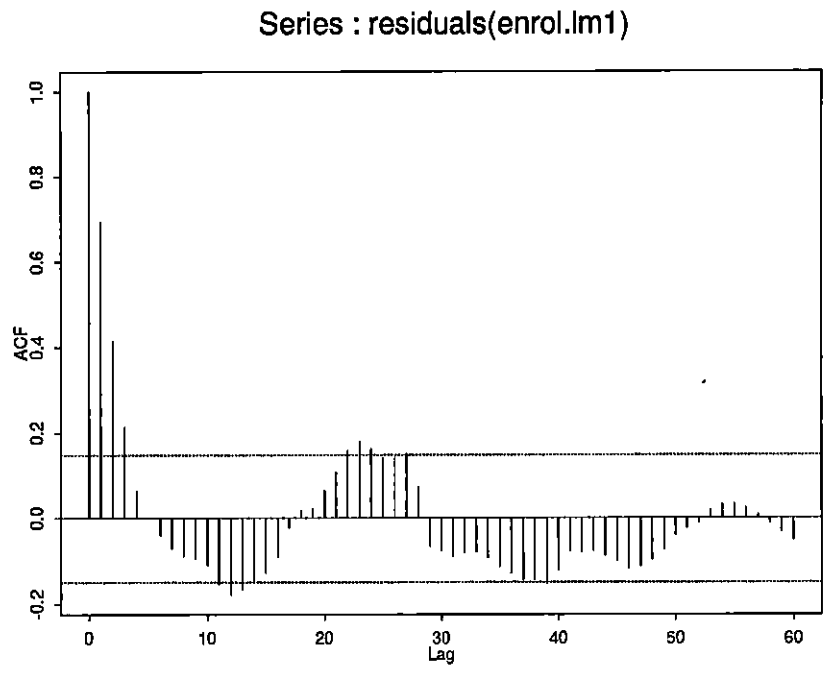



Figure 7.3: Residual autocorrelation of quadratic model of Yale enrolment data.

```
> summary(enrol.lm1)
> acf(residuals(enrol.lm1), lag.max = 60)
```

```
Coefficients:
               Value Std. Error   t value  Pr(>|t|)
(Intercept)  616.6735  132.8434     4.6421    0.0000
 enrol.year  -27.5132    3.3888    -8.1190    0.0000
enrol.year2    0.4422    0.0181    24.3830    0.0000

Residual standard error: 587.5 on 177 degrees of freedom
Multiple R-Squared: 0.962
F-statistic: 2239 on 2 and 177 degrees of freedom, the p-value is 0
```

The correlogram of residuals in Figure 7.3 shows improvement; however, strong correlation remains for the previous three years, which lie outside the 95% confidence limits, and a (non-significant) cyclic process appears. This is primarily due to the two world wars. Removal of the overall trend has caused this underlying effect to become more visible.

Another possibility is to make a break in the middle of the series, at about 90 years, and fit two separate linear regressions.

```
> enrol.split_factor(enrol.year > 90)
```

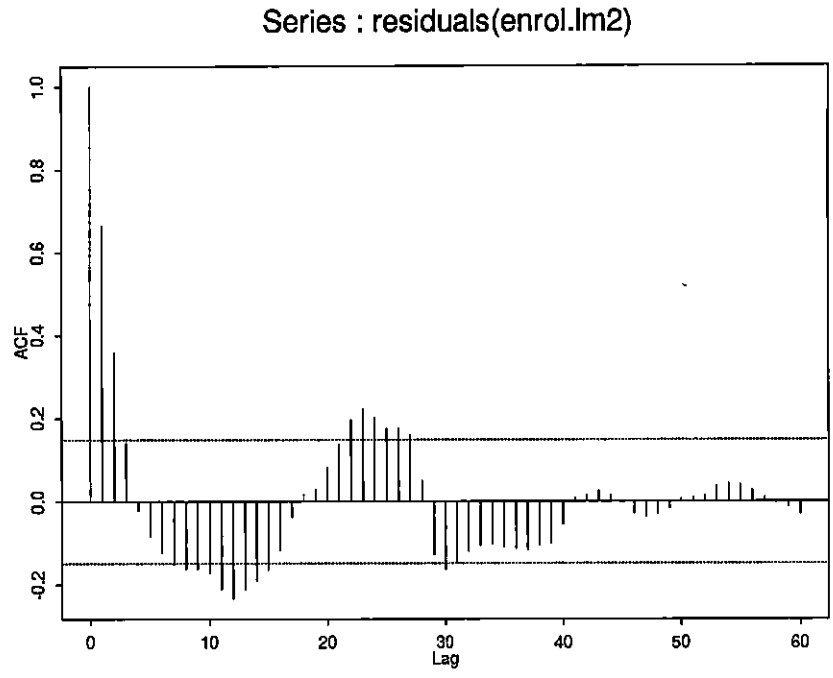



Figure 7.4: Residual autocorrelation of split model of Yale enrolment data.

```
> enrol.lm2_lm(enrol.data ~ enrol.year*enrol.split)
> summary(enrol.lm2)
> acf(residuals(enrol.lm2), lag.max = 60)
```

```
Coefficients:
                          Value Std. Error    t value   Pr(>|t|)
           (Intercept)  127.4969   119.1795     1.0698     0.2862
            enrol.year    9.4421     2.2747     4.1510     0.0001
           enrol.split -7364.7835   335.6986   -21.9387     0.0000
enrol.year:enrol.split    83.2826     3.2169    25.8894     0.0000

Residual standard error: 560.6 on 176 degrees of freedom
Multiple R-Squared: 0.9656
F-statistic: 1645 on 3 and 176 degrees of freedom, the p-value is 0
```

This model fits slightly better than the quadratic (the residual standard error is somewhat smaller) but with one less degree of freedom. The correlogram in Figure 7.4 shows a little less recent correlation, but more variability, with increased (and now significant) correlation about 12 and 24 years previous. The cyclic effect is here more evident.

We could continue in this way, searching for a better fitting model, for example, by fitting a

different quadratic to each side of the break or one of the exponential growth curves of Chapter 5. Instead, we look at some other methods of checking the assumptions of the model.

## 7.2 Autoregression and Random Walks

Another approach to fitting time trends is to use lagged values of the response variable as explanatory variables. This is known as autoregression of given order, $p$, or an $AR(p)$ process. For example, the model

$$y_t = \phi_0 + \phi_1 y_{t-1} + \epsilon_t \qquad \epsilon_t \sim N(0, \sigma) \tag{7.1}$$

is a first order autoregression or an $AR(1)$. Since the explanatory variables are stochastic (have a distribution) instead of being fixed values, this adds complications to the model. Time series of this type can be fitted only if they are stationary; that is, only if the mean, $E(y_t)$, variance, $\mathrm{Var}(y_t)$, and covariance at different lags, $\mathrm{Cov}(y_t, y_{t-k})$, are time invariant.

It is clear from Figure 7.1 that the mean of the Yale enrolment data is increasing with time, so a time series cannot be directly fitted to the raw enrolment data. However, an autoregressive model of the *differences* between successive enrolments, $\Delta y_t = y_{t+1} - y_t$, might be stationary.

We fit an autoregressive model using the arima.mle() function. The arima.mle() function can be used to fit higher order autoregressive models and moving average, $MA(q)$, models as well. The term *ARIMA* refers to autoregressive, integrated, moving average models. The model we will be fitting is an ARIMA(1,1,0) model; that is, a model with one autoregressive term, first order differencing, and no moving average terms.

$$\Delta y_t = \phi_0 + \phi_1 \Delta y_{t-1} + \epsilon_t \qquad \epsilon_t \sim N(0, \sigma) \tag{7.2}$$

```
> enrol.arima1_arima.mle(enrol.data, model = list(order = c(1, 1, 0)))
> arima.diag(enrol.arima1)
```

From the diagnostic plot (Figure 7.5) we see that the autoregressive model has removed all serial correlation from the residuals, although there are a few substantial outliers visible in the standardized residual plot which correspond to the fluctuations of the world wars.

Unfortunately, arima.mle() does not have a summary() method, so we view the results of the fitting process using the unlist() function.

```
> unlist(enrol.ar1)
```

```
model.order1 model.order2 model.order3              model.ar model.ndiff
"1"          "1"          "0"          "-0.0209821437413409" "1"

             var.coef                method       series                aic
"0.00561550421148323" "Maximum Likelihood" "enrol.data" "2690.87134877452"
```

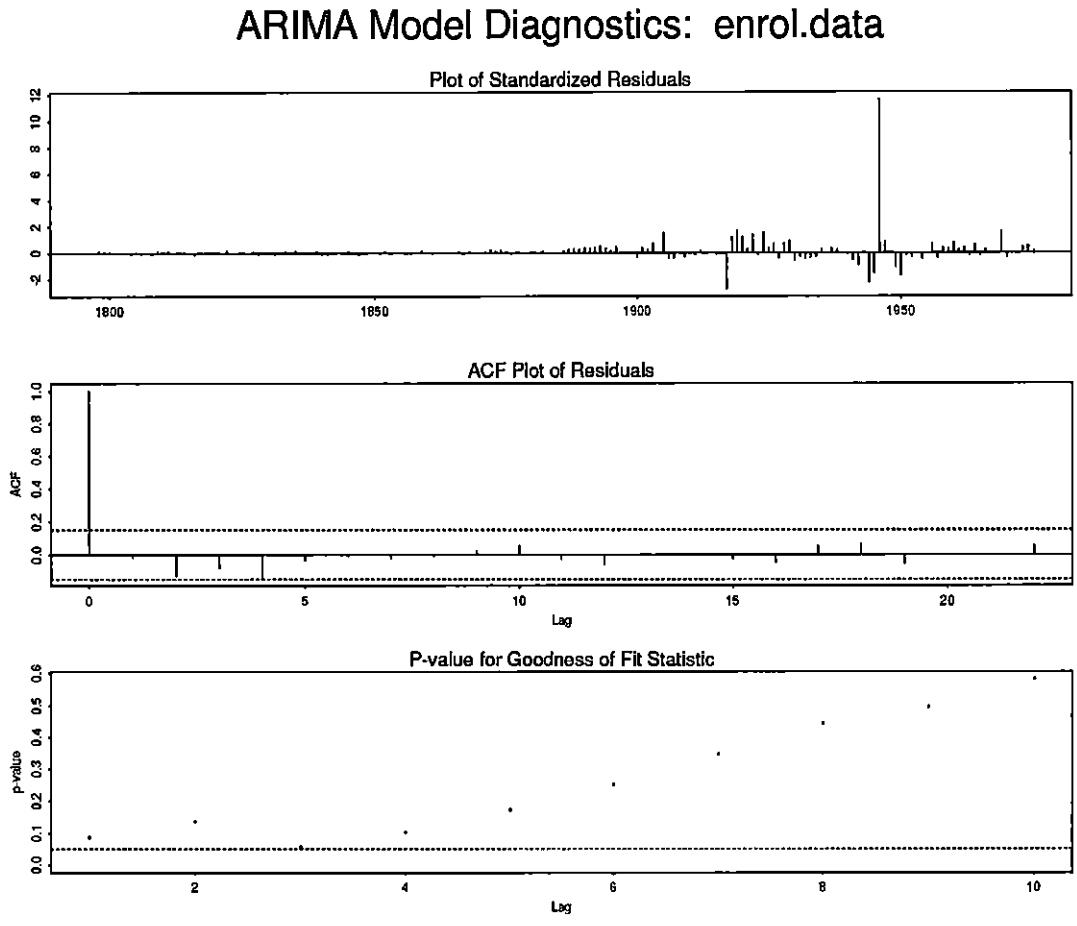



Figure 7.5: Diagnostic plot of ARIMA(1,1,0) model of Yale enrolment data.

```
          loglik             sigma2 n.used n.cond converged
"2688.87134877452" "212807.173711274" "178"  "2"    "TRUE"

                      conv.type
"relative function convergence"
```

Although the output from unlist() is not well formatted, we can see that our estimate for $\phi_1$ in Equation 7.2, model.ar, is $-0.021$, with variance 0.0056. The variance corresponds to a standard error of 0.07, so $\phi_1$ is non-significant. We can also read our estimate of $\sigma^2$ from above, 212807.17, so our estimate of $\sigma$ is 461. We can estimate $\phi_0$ by calculating the mean of the differences of the original series.

```
> mean(diff(enrol.data))

[1] 53.6648
```

Since our model has no significant autoregressive term, it is actually a random walk, with a drift term equal to the mean of the differenced series

$$\Delta y_t = 53.6 + \epsilon_t \qquad \epsilon_t \sim N(0, 461) \tag{7.3}$$

A second ARIMA model we could apply to the enrolment data is one in which the trend is a deterministic result of covariates, and the error terms are a first order autoregressive process.

$$y_t = \boldsymbol{\beta X} + z_t \tag{7.4}$$

$$z_t = \phi z_{t-1} + \epsilon_t \qquad \epsilon_t \sim N(0, \sigma)$$

Covariates can be added to an arima.mle model using the xreg parameter. We will fit a quadratic model, as in Section 7.1 on page 126, using autoregressive error term in place of the usual independent normal errors.

```
> enrol.arima2_arima.mle(enrol.data, model = list(order = c(1, 0, 0)), xreg = cbind(rep(1,
length(enrol.data)), enrol.year, enrol.year ^ 2))
> unlist(enrol.arima2)

model.order1 model.order2 model.order3           model.ar model.ndiff
"1"          "0"          "0"          "0.694549843682269" "0"

             var.coef                method       series                aic
"0.00289162298682087" "Maximum Likelihood" "enrol.data" "2677.51900745945"
```

```
           loglik             sigma2 n.used n.cond converged
"2669.51900745945" "175542.584950491" "179"  "1"    "TRUE"

                      conv.type          reg.coef1          reg.coef2
"relative function convergence" "690.195461815114" "-28.8548142470894"

         reg.coef3
"0.447172556190269"
                                                        reg.series
"cbind(rep(1, length(enrol.data)), enrol.year, enrol.year^2)"
```

Note that the coefficients for $\beta_0$, $\beta_1$ and $\beta_2$ (690.2, -28.8, 0.447) are similar to those for the normal linear regression on page 126, but that the AR term is also significant, at 0.69. The ARIMA model has fit the deterministic portion of the process and also taken into account the serial correlation of the residuals; the diagnostic plot of the model (Figure 7.6) shows that the remaining residuals are uncorrelated.

```
> arima.diag(enrol.arima2)
```

The two models can be compared using the AIC, or *Akaike information criterion*. The AIC of the ARIMA(1,1,0) model can be obtained using enrol.arima1$aic, to get 2690.871. The AIC of the ARIMA(1,0,0) model with a deterministic trend is 2677.519. Normally the model with a lower AIC is considered superior. However, AIC is only a general guide to model selection, and the AIC's of the two models are not very different.

## 7.3 Examination of the Distributional Assumptions

To examine the assumptions of normality and constancy of variance, we use methods which are of much wider application than simply to time series. These consist primarily of a study of the residuals. Consider the work of Nichols [17] on white collar wages (Table 7.2).

Nichols fits a model whereby the wage depends on the minimum wage, the unemployment rate, and the consumer price index, the latter also lagged for each of the previous two years. We set up the data, produce a pairs plot of the variables (Figure 7.7), and fit Nichols' model.

```
> wage.data_data.frame(scan("data/Chpt6-3", what = list(year = 0, wage = 0, CPI = 0, unem = 0,
minum = 0)))
> wage.data_data.frame(wage.data[3:20,1:5], CPI1 = wage.data$CPI[2:19], CPI2 =
wage.data$CPI[1:18])
> pairs(wage.data)
```

We now fit Nichol's model.

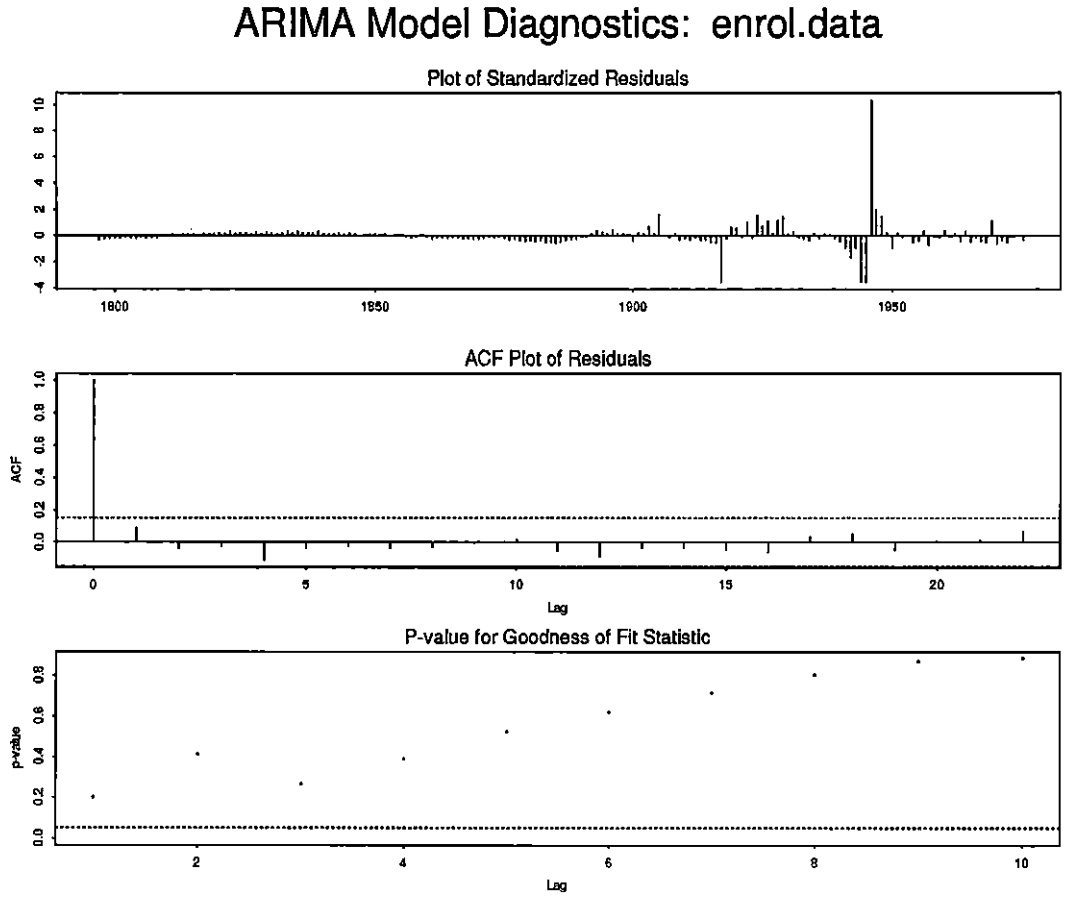



Figure 7.6: Diagnostic plot of ARIMA(1,0,0) model of Yale enrolment data with a quadratic deterministic component.

| year | wage | CPI | unem | minum |
|---|---|---|---|---|
| 1960 | 0.0 | 0.347 | 0.000 | 0.000 |
| 1961 | 0.0 | 1.499 | 0.000 | 0.000 |
| 1962 | 2.8 | 1.477 | 6.457 | 15.000 |
| 1963 | 2.7 | 1.120 | 6.192 | 0.000 |
| 1964 | 2.7 | 1.107 | 5.763 | 8.696 |
| 1965 | 2.2 | 1.424 | 5.018 | 0.000 |
| 1966 | 2.9 | 1.188 | 4.170 | 12.000 |
| 1967 | 4.5 | 2.775 | 3.725 | 12.000 |
| 1968 | 5.1 | 3.943 | 3.686 | 0.000 |
| 1969 | 5.5 | 3.943 | 3.686 | 0.000 |
| 1970 | 6.2 | 5.058 | 3.556 | 0.000 |
| 1971 | 6.2 | 6.019 | 5.395 | 0.000 |
| 1972 | 6.3 | 4.629 | 6.272 | 0.000 |
| 1973 | 5.5 | 3.506 | 5.433 | 0.000 |
| 1974 | 6.2 | 4.677 | 4.522 | 0.000 |
| 1975 | 9.1 | 10.247 | 6.163 | 31.250 |
| 1976 | 7.6 | 10.273 | 8.625 | 9.524 |
| 1977 | 6.9 | 6.147 | 7.877 | 0.000 |
| 1978 | 7.5 | 6.388 | 6.679 | 15.217 |
| 1979 | 7.2 | 6.510 | 5.494 | 9.434 |

Table 7.2: Annual percentage increase in average salaries of white collar workers in Britain, 1960-1979 (CPI: Consumer price index) (Kramer and Sonnberger, 1986, p.165)

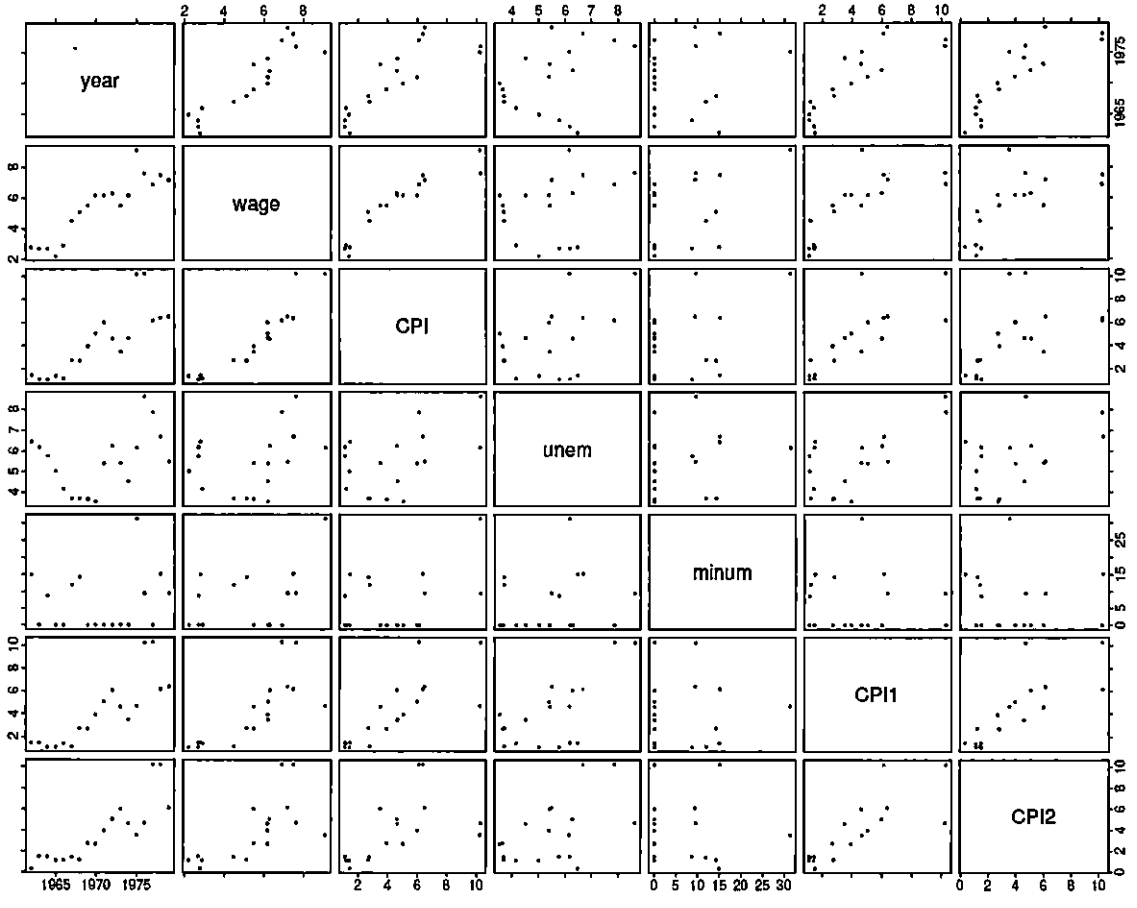



Figure 7.7: Pairs plot of white collar wage data.

```
> wage.lm_lm(wage ~ CPI + CPI1 + CPI2 + unem + minum, data = wage.data)
> summary(wage.lm)

Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  4.2754  0.4872     8.7757  0.0000
        CPI  0.5188  0.0860     6.0360  0.0001
       CPI1  0.1213  0.1166     1.0404  0.3187
       CPI2  0.2140  0.0589     3.6368  0.0034
       unem -0.4879  0.1103    -4.4240  0.0008
      minum  0.0316  0.0171     1.8516  0.0888

Residual standard error: 0.4191 on 12 degrees of freedom
Multiple R-Squared: 0.9699
F-statistic: 77.22 on 5 and 12 degrees of freedom, the p-value is 1.057e-08

> print(cbind(wage.data$year,resid(wage.lm)))

   1962  0.17773956    1971 -0.02380943
   1963  0.36433313    1972  0.86994920
   1964 -0.06531123    1973  0.20622711
   1965 -0.74012189    1974  0.28802073
   1966 -0.36706505    1975  0.20837467
   1967 -0.22640285    1976 -0.34310329
   1968  0.19717025    1977 -0.16141951
   1969  0.05561348    1978 -0.25729394
   1970 -0.02105678    1979 -0.16184415
```

A glance at the standard errors of the estimates indicates the effects of the variables—consumer price index with lag one and minimum wage are not significantly different from zero.

We also calculate some residual diagnostics. The Q-Q plot (Figure 7.8) indicates deviations in the residuals from the assumption of a normal error distribution.

```
> qqnorm(residuals(wage.lm))
> qqline(residuals(wage.lm), lty = 2)
```

The score residual test is used to detect outliers from the distribution. Pregibon [18] showed that the score test statistic of the hypothesis that the $i$'th observation is an outlier; i.e. $H_0 : \mu_j = \boldsymbol{x}^T\boldsymbol{\beta}$ for $j \neq i$, $\mu_i = \delta$ has the form $t_i^2$ where $t_i$ is the standardized residual

$$t_i = \frac{y_i - \hat{\mu}_i}{\hat{\sigma}\sqrt{1 - h_i}}$$

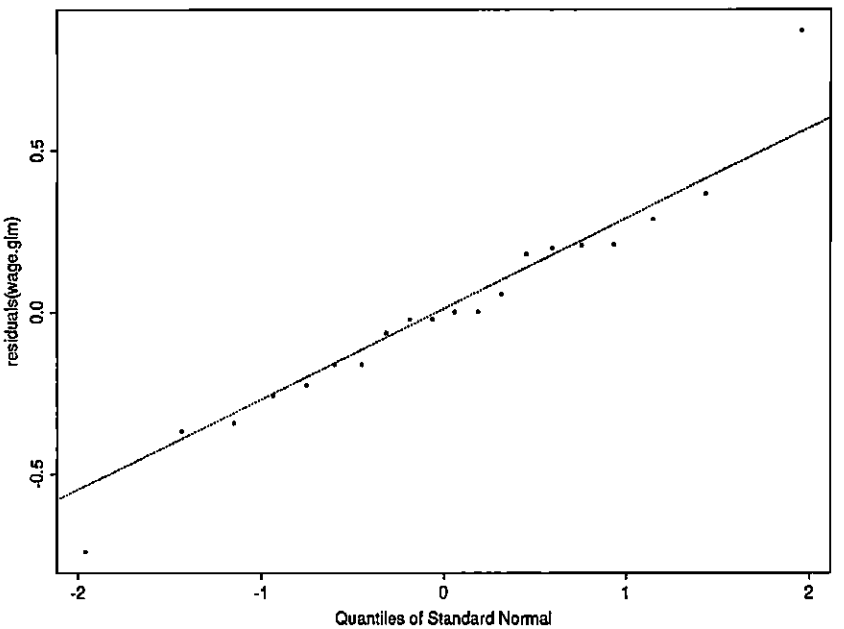



Figure 7.8: Normal quantile plot of residuals of a multiple linear regression on white collar wage data.

Here $y - \hat{\mu}_i$ is the $i$'th residual, $\hat{\sigma}$ is the estimate of $\sigma$ from the linear model, and $h_i$ is the $i$'th leverage, the $i$'th diagonal entry of the hat matrix $\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$. Therefore, the score test statistic has an approximate $\chi^2_{(1)}$ distribution. We graph the score statistics versus year together with the 95th quantile of the $\chi^2_{(1)}$ distribution in Figure 7.9. The score tests indicate that the observations for years 1965 and 1972 are outliers. Note also that the largest residuals occur in years 1965 and 1972.

```
> wage.score_resid(wage.lm) ^ 2 / ((deviance(wage.lm) / wage.lm$df) * (1 -
lm.influence(wage.lm)$hat))
> plot(wage.score, main = "Score Residual Plot")
> abline(qchisq(.95, 1), 0, lty = 2)
> identify(wage.score, labels = wage.data$year)
```

If we plot a graph of fitted values versus observed (Figure 7.10), and then identify the two points farthest from the line of equality, years 1965 and 1972 are again found to be the greatest outliers.

```
> plot(wage.lm)
> identify(fitted(wage.lm), wage.data$wage, labels = wage.data$year)
```

Finally, we test for serial correlation in the residuals, using the acf() function introduced earlier. None of the autocorrelations (Figure 7.11) are significantly different from zero.

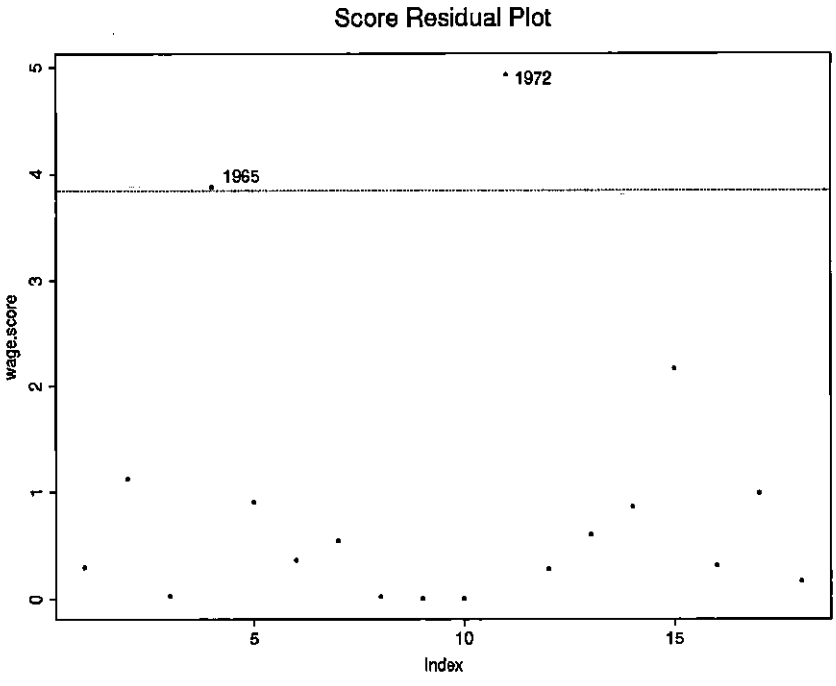



Figure 7.9: Score test plot of residuals of a multiple linear regression on white collar wage data.

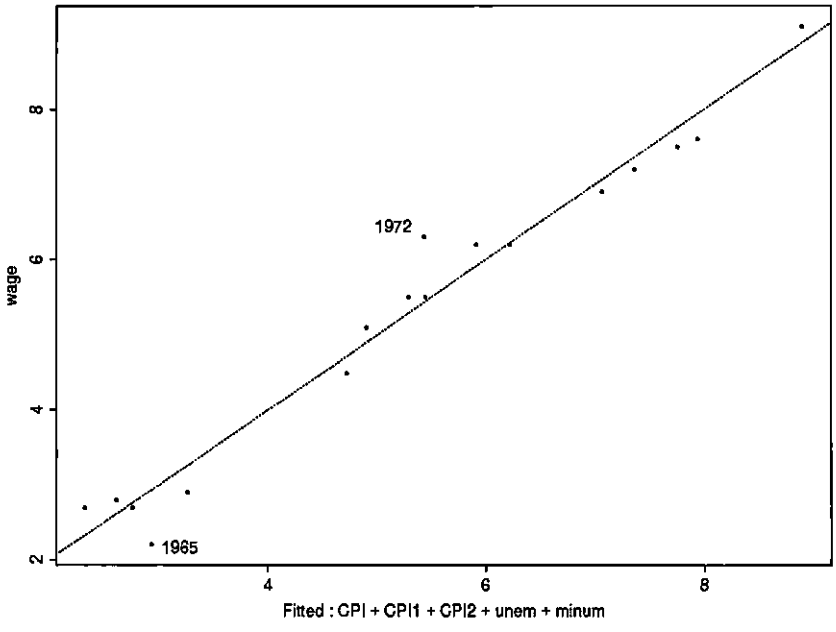



Figure 7.10: Plot of fitted values versus observed for multiple linear regression on white collar wage data.

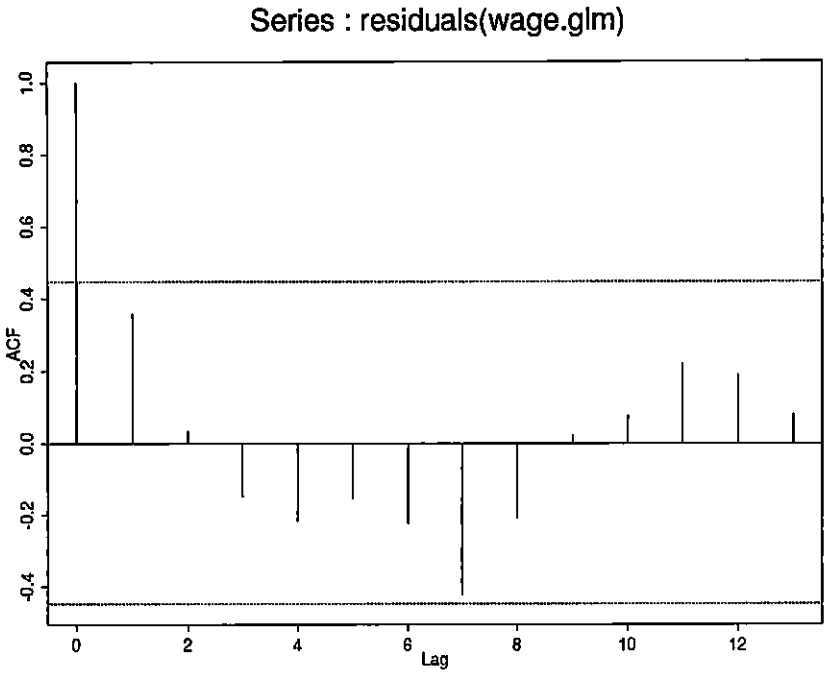



Figure 7.11: Autocorrelation plot of residuals for multiple linear regression on white collar wage data.

```
> acf(residuals(wage.glm))
```

## 7.4 Mis-specification of the Linear Model

Mis-specification of the linear model usually involves variables which are missing, including perhaps nonlinear terms. Most often, all available variables are routinely tested, or at least their linear effects. Inclusion of higher order terms is sometimes known as a *reset test* [19]. A second order reset test adds quadratic terms and a third order test adds the cubic terms as well. Logically, these should include cross-product terms as well, but very often econometricians do not include them. We apply these tests to the wage data using the update() function.

The update() function allows you to add or remove terms from a model without re-specifying the entire model—you only have to specify those terms you wish to add or remove.

```
> wage.lm2_update(wage.lm, . ~ . + CPI ^ 2 + CPI1 ^ 2 + CPI2 ^ 2 + unem ^ 2 + minum ^ 2)
```

To test the significance of the differences between these two models, we will employ a custom function f.test() (Section 9.5), which calculates the F statistic and assocated p-value for a pair of linear models.

```
> f.test(wage.lm, wage.lm2)

F test: 8.32654 with df 5 and 7 and probability 0.00735
```

We see that the F-test indicates that the second order terms add significantly to the model. Note however that this model is over-parameterized—we have only 18 observations and 11 parameters in the second order model. A good rule of thumb is that a model should have at least 5 times as many observations as parameters.

The second important approach to mis-specification involves checking if the form of the model changes over time. In the simplest case, we suspect a change at some specific point in time - a break such as those we have seen in several preceding chapters - and we test for a difference in regression parameters before and after. This is what we did in the first section for the Yale enrolment data. Such a verification is sometimes known as a *Chow test* [5]. Although such model testing can be performed directly, as we have seen several times, a function, chow() is presented here.

```
function(model.object, cut.point)
{
        dev1 <- deviance(model.object)
        df1 <- model.object$df
        cut.factor <- factor(seq(model.object$y) > cut.point)
        attach(data.frame(cut.factor = cut.factor))
        model.object.2 <- update(model.object, . ~ . * cut.factor, class =
                "glm")
        dev2 <- deviance(model.object.2)
        df2 <- model.object.2$df
        dispersion <- as.numeric(summary(model.object.2)$dispersion)
        score <- ((dev1 - dev2)/(df1 - df2))/dispersion
        cat("Chow test is", round(score, 5), "with", df1 - df2, "and", df2,
                "degrees of freedom for a break after", cut.point, "\n", sep =
                " \n")
        cat("P-value:", round(1 - pf(score, df1 - df2, df2), 5), "\n", sep =
                " ")
}
```

For the wage data, we test for a break after 12 years (in 1971-1972).

```
> chow(wage.glm, 12)

Chow test is 1.53718 with 6 and 6 degrees of freedom for a break after 12
P-value: 0.30736
```

Here the test provides no indication of a change in the model at the chosen time point.

If we do not know, *a priori*, where a break in the model might have occurred, we may proceed in another way. We can fit the model using an initial subset of the data, and use it to forecast future responses. Forecast errors are computed by subtracting the predicted responses from the observed responses which were not used to fit the model. If the model is invariant over time, then the forecast errors should behave like ordinary residuals. We can plot them over time to see if there are any noticeable patterns which would indicate a change in the underlying process. One technique which is used in *statistical process control* is to plot the cumulative sum of the errors [16, 21]. A change in the process is more quickly detected using cumulative sums rather than the original errors.

Here we use the first 12 observations to fit the model to the wage data, compute and scale the prediction errors, and produce a CUSUM chart. The variance of the errors is estimated using

$$Var(y_k - \hat{y}_k) = Var(y_k) + Var(\hat{y}_k) = \hat{\sigma}^2 + \text{se.fit}^2$$

where $\hat{\sigma}^2$ is the estimated variance from the fitted model, and se.fit is the estimated error of the predicted values. Note that $y_k$ and $\hat{y}_k$ are independent if $k$ is greater than the truncation point of the fitted model.

```
> attach(wage.data)
> wage.lm2_lm(wage ~ CPI + CPI1 + CPI2 + unem + minum, data = wage.data, subset = year <
1972)
> wage.lm2.pred_predict(wage.lm2, wage.data[year > 1971,], se.fit = T)
> sigma2_deviance(wage.lm2) / wage.lm2$df
> resahead_(wage[year > 1971] - wage.lm2.pred$fit) / sqrt(sigma2 + wage.lm2.pred$se.fit ^ 2)
> cusum(qcc(resahead, type = "xbar", sizes = 1, std.dev = 1), decision.int = 3, target = 0)
```

Figure 7.12 gives the CUSUM plot of the predicted residuals. Although the residuals begin to wander at point 17, they do not pass the critical values, so there is no evidence of model misspecifiction.

## 7.5 Serial Correlation in Regression Analysis

If the correlogram of residuals for the model indicates an important serial correlation when the full model is fitted, this must be taken into account in the model instead of fitting a classical normal theory model which supposes independence among the observations (conditional on the explanatory variables). In other words, the variance-covariance matrix is no longer diagonal. By pre-weighting our response and covariates, we can fit models which take serial correlation into account.

Recall that, in Section 7.3, we noted that the variables, consumer price index with lag one (CPI1) and minimum wage (minum), were probably not significantly different from zero in our full model. Let us explore this problem a little more before proceeding.

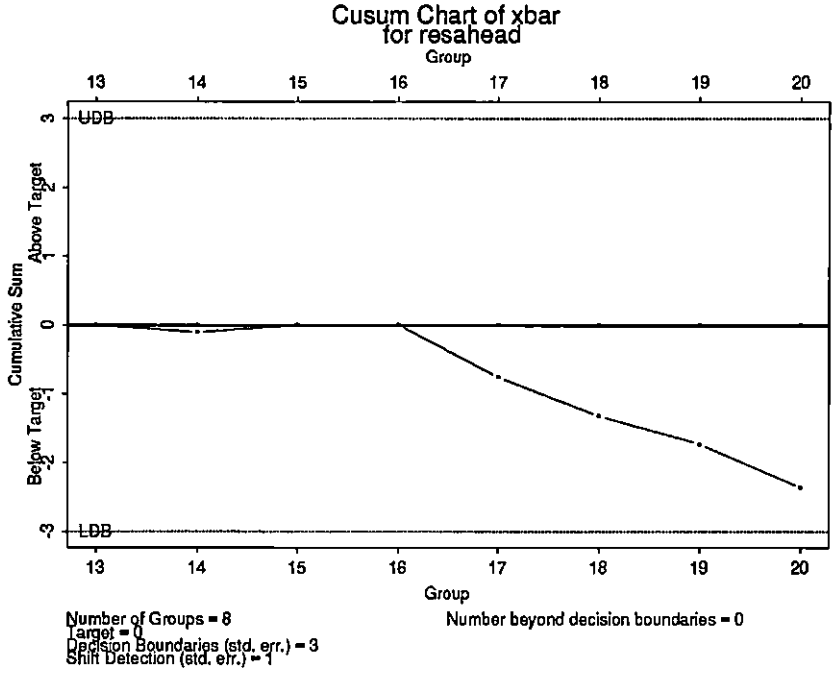



Figure 7.12: CUSUM plot of predicted residuals based on model fitted to first 12 points of white collar wage data.

We fit a full model, using all the explanatory variables, and then sequentially eliminate the least significant variables in the model. This is known as *backward elimination.* Note that statistically insignificant variables may be retained in a model where they are integral to the mathematical theory being examined.

```
> wage.sc1.lm_lm(wage ~ CPI + CPI1 + CPI2 + unem + minum, data = wage.data)
> summary(wage.sc1.lm)

Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  4.2754  0.4872     8.7757  0.0000
       CPI   0.5188  0.0860     6.0360  0.0001
      CPI1   0.1213  0.1166     1.0404  0.3187
      CPI2   0.2140  0.0589     3.6368  0.0034
      unem -0.4879  0.1103    -4.4240  0.0008
     minum   0.0316  0.0171     1.8516  0.0888

Residual standard error: 0.4191 on 12 degrees of freedom
Multiple R-Squared: 0.9699
F-statistic: 77.22 on 5 and 12 degrees of freedom, the p-value is 1.057e-08
```

The least significant explanatory variable here is CPI1, which only has a p-value of 0.3187. We remove CPI1 from the model and refit it.

```
> wage.sc2.lm_update(wage.sc1.lm, . ~ . - CPI1)
> summary(wage.sc2.lm)
```

```
Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)  3.9950    0.4071   9.8126   0.0000
        CPI  0.5908    0.0512  11.5371   0.0000
       CPI2  0.2499    0.0478   5.2233   0.0002
       unem -0.4157    0.0860  -4.8341   0.0003
      minum  0.0210    0.0138   1.5289   0.1503

Residual standard error: 0.4204 on 13 degrees of freedom
Multiple R-Squared: 0.9671
F-statistic: 95.65 on 4 and 13 degrees of freedom, the p-value is 1.663e-09
```

The minum variable is now non-significant, so we remove it and and refit the model.

```
> wage.sc3.lm_update(wage.sc2.lm, . ~ . - minum)
> summary(wage.sc3.lm)
```

```
Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)  3.9759    0.4259   9.3346   0.0000
        CPI  0.6288    0.0469  13.4208   0.0000
       CPI2  0.2227    0.0465   4.7908   0.0003
       unem -0.3993    0.0893  -4.4715   0.0005

Residual standard error: 0.44 on 14 degrees of freedom
Multiple R-Squared: 0.9612
F-statistic: 115.7 on 3 and 14 degrees of freedom, the p-value is 4.066e-10
```

All the explanatory variables are now significant at the 5% level.

To investigate the serial nature of the residuals, we plot the autocorrelation function of the residuals.

```
> acf(resid(wage.sc3.lm))
```

Although there is no significant autocorrelation present (Figure 7.13), for illustration let us introduce a serial correlation parameter into the model with only three explanatory variables. We

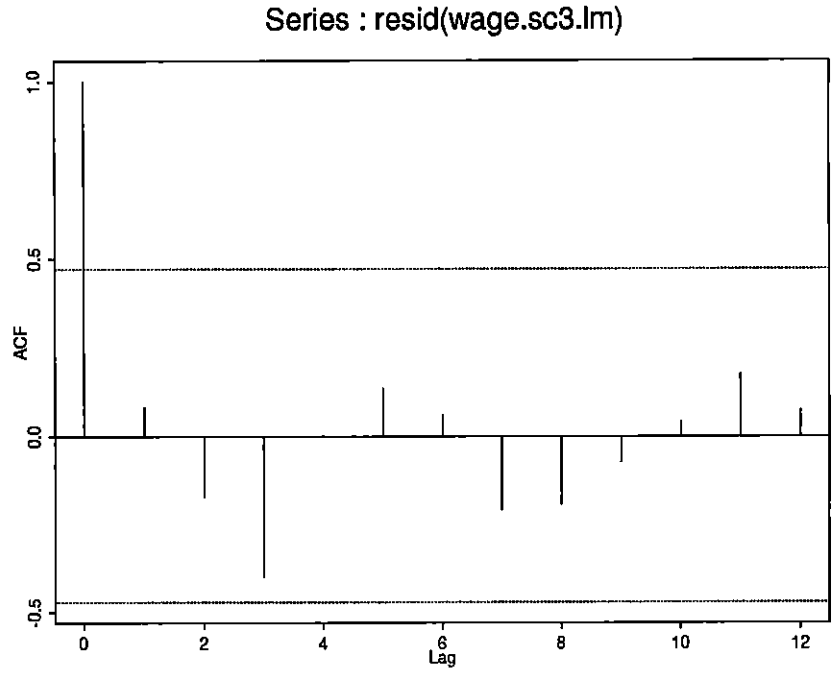



Figure 7.13: Autocorrelation plot of the residuals of a three parameter model of white collar wage data.

fit a model in which the residuals are correlated at a lag of one. To do this, we need two things: first, an estimate of the serial correlation of the residuals; second, a method of fitting a model in which a complete variance-covariance matrix is specified.

To obtain an estimate of the serial correlation of the residuals at lag one, we simply apply the acf() function with the plot option turned off and extract the lag one estimate.

```
> acf(resid(wage.sc1.glm,plot=F)$acf[2]

[1] 0.08177487
```

So, the lag one autocorrelation is estimated at 0.08177. In order to fit a model with correlation between residuals, we must transform our data with a weighting matrix.

An ordinary weighted linear regression calculates parameter estimates as follows:

$$\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{W}^{-1}\boldsymbol{X})\boldsymbol{X}^T\boldsymbol{W}^{-1}\boldsymbol{Y} \tag{7.5}$$

where $Var(\boldsymbol{Y}) = \boldsymbol{W}$ is the variance-covariance matrix of $\boldsymbol{Y}$. If the square root of the inverse weight matrix, $\boldsymbol{C}^T\boldsymbol{C} = \boldsymbol{W}^{-1}$, is calculated, then the linear estimates in equation 7.5 can be rewritten

$$\hat{\beta} = (\boldsymbol{Q}^T\boldsymbol{Q})\boldsymbol{Q}^T\boldsymbol{Z} \tag{7.6}$$

where $\boldsymbol{Z} = \boldsymbol{CY}$ and $\boldsymbol{Q} = \boldsymbol{CX}$. However, equation 7.6 is just an ordinary un-weighted linear regression, and we can fit that directly using lm().

We begin the fitting process by calculating the weight matrix, inverting it, and finding the square root. The weight matrix for a model with serial residual correlation at lag one has a diagonal of ones and minor diagonals of the lag one autocorrelation. To quickly construct the matrix, we use the custom function wdiag(), and also print out a smaller example matrix.

```
function(diag, rho, size)
{
        element <- c(diag, rho, rep(0, size - 2), rho)
        data <- rep(element, size)
        matr <- matrix(data[1:size^2], size)
        matr
}
```

```
> wage.wgts_wdiag(1, 0.08177, 18)
> print(wdiag(1, 0.5, 5))
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]  1.0  0.5  0.0  0.0  0.0
[2,]  0.5  1.0  0.5  0.0  0.0
[3,]  0.0  0.5  1.0  0.5  0.0
[4,]  0.0  0.0  0.5  1.0  0.5
[5,]  0.0  0.0  0.0  0.5  1.0
```

We calculate the matrix $\boldsymbol{C}$ using a singular-value decomposition of $\boldsymbol{W^{-1}}$.

```
> wage.wgts_solve(wage.wgts)
> wage.wgts_svd(wage.wgts)
> wage.c_t(wage.wgts$v %*% (t(wage.wgts$u) * sqrt(wage.wgts$d)))
```

Once the square root of the inverse weight matrix has been calculated, we calculate the response vector and covariate matrix using matrix multiplication, and then fit the final weighted model using lm().

```
> wage.z_wage.c %*% wage.data[,"wage"]
> wage.q_wage.c %*% as.matrix(wage.data[,c("CPI","unem","CPI2")])
> summary(lm(wage.z ~ wage.q))
```

```
Coefficients:
               Value Std. Error  t value Pr(>|t|)
```

```
(Intercept)   3.6559   0.4237    8.6290   0.0000
  wage.qCPI   0.6258   0.0496   12.6112   0.0000
 wage.qunem  -0.3854   0.0946   -4.0733   0.0011
 wage.qCPI2   0.2189   0.0488    4.4847   0.0005

Residual standard error: 0.4503 on 14 degrees of freedom
Multiple R-Squared: 0.954
F-statistic: 96.79 on 3 and 14 degrees of freedom, the p-value is 1.341e-09
```

As expected, adding the lag one effect does not change the model very much, since the effect was non-significant to start with.

# Chapter 8

# Conclusion

The conversion project was personally educational in both the statistical and computational fields. The breadth of statistical application to which the project exposed me will undoubtedly serve me well in the future. Similarly, the experience of programming in an object oriented environment may transfer well to other object oriented programming paradigms.

Transferring the techniques used by Lindsey in his text to SPlus equivalents involved more than a rote transposition of language syntax. Because SPlus is a substantially more developed package than GLIM, a large proportion of Lindey's GLIM macros were superseded by built-in SPlus functions.

The procedures Lindsey developed for GLIM were characterised by an ingenious use of the limited GLIM syntax to achieve exceptional results. In SPlus, such algorithmic gymnastics were unnecessary and even superfluous; the greater portion of the GLIM macro collection has been jetisoned in favour of native SPlus variants. The remainder of the collection has been largely re-tooled to accommodate results provided by SPlus object classes.

In re-tooling the procedures of the Lindsey book to take advantage of the SPlus environment, I have had opportunity to learn many new techniques, both statistical and computational:

- The project covered a wide range of mathematical models, many of which I had learned about in previous courses. But although Markov chains, Poisson processes, renewal processes, and non-homogeneous processes were all familiar to me mathematically, I had never dealt with the possibility of fitting them to data.
- The project also included models to which I had not been exposed in my course-work, such as survival models, Cox proportional hazards, and non-linear models. Understanding the mathematics behind these models, at least cursorily, was a necessary prerequisite to converting computational procedures from GLIM to SPlus. As well, the examples which had to be worked through gave me a practical understanding of the application of survival and non-linear models.

- Integrating custom functions into the SPlus environment gave me an opportunity to become familiar with the object oriented nature of SPlus. Many of the new functions are simply additional tools for manipulating or displaying existing SPlus object classes. The linear model fitter used to fit a linear model with serially correlated residuals, actually overrides an existing intermediate step in the model fitting process and cleanly deals with a weighting matrix—something none of the current linear fitters in SPlus can do.

- Several of the techniques Lindsey presents in his book are predicated on a method of fitting distributions to data using log-linear models. This method is easily applicable to all distributions in the exponential family, and can be used to simultaneously evaluate the fit of several distributions to the data.

  The more I learn about it, the more I think this would be an interesting process to automate in SPlus, and a useful one. With a little more knowledge of the details it could be done fairly successfully—details around the granularity of continuous data grouping need to be clarified [13].

- In fitting non-linear models, I had the opportunity to reproduce Lindsey's method of minimizing residual deviance in binomial models as well as the more direct SPlus method of fitting a non-linear least squares model. Both methods yield very similar fitted values.

When I first approached this project, it was with the intention of reproducing Lindsey's macros in SPlus form. However, the macros tended to be redundant to SPlus, so initial effort was wasted in reproducing them precisely in SPlus—moving immediately to converting the *techniques* in the book to SPlus would have been more fruitful.

In reproducing Lindsey's algorithms in SPlus, fatal errors were common due to a reliance on the forgiving nature of the GLIM package. The most common errors occurred when Lindsey's algorithms spawned divide-by-zero situations—GLIM handles divide-by-zero by returning zero, SPlus halts the process.

In retrospect, the structure of the book itself is somewhat deficient. Some topics, such as survival distributions, are split across chapters. Others, which could be given more detailed treatment, such as Cox proportional hazards, are only briefly touched on.

The custom functions designed in SPlus can still be integrated more completely into the SPlus environment—final distribution of the new functions will include a written summary of the functions as well as an SPlus library incorporating help files for each function.

If I were advising someone starting this project afresh, I would ensure that they began macro conversion by converting the macros to pseudo-code before programming them in SPlus. Because Lindsey coded the macros in GLIM, he was extremely restricted in choosing variable names, and a conversion to pseudo-code, with expressive variable names, would make understanding the algorithmic structure must easier. Similarly, I would advise that a great deal of care be taken in conversion to avoid GLIM control structures. GLIM has no real control structures, and iteration has to be cobbled together with sub-macros—for this reason, starting a conversion with a pseudo-code step would result in a faster attainment of clean SPlus code.

In working through this project I found the survival libraries written by Terry Therneau, of the Mayo Foundation, to be particularly helpful. Because so much of the book concentrates on survival distributions, the survival libraries were extremely useful.

Similarly, the existing functions of the SPlus package were extremely useful as templates and learning devices for structuring my own functions, particularly in the area of error handling and parameter checking.

Finally, the patient and knowledgeable assistance of my advisor, Prof. Mary Lesperance, has been invaluable to me.

# Chapter 9

# Custom Functions

## 9.1 glmtab()

```
function(..., exclude = c(NA, NaN))
{
        if(nargs() == 0)
                stop("No arguments")
        bin <- 0
        lens <- NULL
        dims <- numeric(0)
        pd <- 1
        dn <- list()
        args <- list(...)
        for(iarg in seq(along = args)) {
                i <- args[[iarg]]
                imode <- mode(i)
                if(is.null(lens))
                        lens <- length(i)
                if(length(i) != lens)
                        stop("All arguments must have the same length")
                if(!is.numeric(i))
                        i <- as.numeric(i)
                if(!is.category(i))
                        cat <- category(i, exclude = exclude)
                else cat <- i
                l <- levels(cat)
                dims <- c(dims, length(l))
                mode(l) <- imode
                dn <- c(dn, list(l))
```

```
                bin <- bin + pd * (as.numeric(cat) - 1)
                pd <- pd * length(l)
        }
        names(dn) <- names(args)
        lentab <- prod(dims)
        bin <- bin[!is.na(bin)]
        x <- 1
        colnames <- null()
        for(idim in seq(along = dims)) {
                tmp.vec <- null()
                for(ilab in seq(dims[idim])) {
                        tmp.vec <- c(tmp.vec, rep(dn[[idim]][ilab], x))
                }
                if(x == 1)
                        tab <- data.frame(factor(rep(tmp.vec, length = lentab))
                                )
                else tab <- data.frame(tab, factor(rep(tmp.vec, length = lentab
                                )))
                colnames <- c(colnames, paste("Col", idim, sep = ""))
                x <- x * dims[idim]
        }
        tab <- data.frame(as.numeric(tabulate(bin + 1, pd)), tab)
        if(is.null(names(args)))
                names(tab) <- c("Counts", colnames)
        else names(tab) <- c("Counts", names(args))
        tab
}
```

## 9.2 tsfr()

```
function(times, interval)
{
#
# This function is used to transform data from a form of "times between events"
# to a form of "number of events per interval".
# The "times" variable is a vector of times between events.
# The "interval" variable is the interval in which you want to count events.
# The interval should be given in the same time units as the original
# time data.
#
        if(length(interval) > 1) stop("Interval must be a scalar.")
        if(!is.vector(times))
```

```
                stop("The times between events must be in a vector.")
        input.cumsum <- cumsum(times)
        output.breaks <- seq(from = 0, to = sum(times) + interval, by =
                interval)
        output <- hist(input.cumsum, breaks = output.breaks, plot = F)
        output.xlab <- paste("Periods of", interval)
        output.main <- paste("Number of Events in Successive", output.xlab)
        if(!(dev.cur() == 1))
                hist(input.cumsum, breaks = output.breaks, main = output.main,
                        xlab = output.xlab, ylab = "Events")
        output
}
```

## 9.3 mpct()

```
function(counts, before, after, time, data)
{
#
# The mpct function calculates the first order transition probabilities for a
# Markov chain and tests for stationarity.
# To invoke mpct, pass it the following data:
# counts = vector of counts in the markov matrindx
# before = vector of states before transition, must be coded in
# integers starting at 1
# after = vector of states after transition, must be coded in integers
# starting at 1
# time = time index of transition matrices.
#
        if(nargs() == 0) stop("No arguments")
        if(!missing(data))
                attach(data)
        if(!is.numeric(counts))
                stop("counts must be a numeric vector")
        dims <- numeric(0)
        args <- list(before, after, time)
        input.length <- length(before)
        for(iarg in seq(along = args)) {
#
# validate data
#
                i <- args[[iarg]]
                if(length(i) != input.length)
```

```
                stop("All arguments must have the same length")
        if(!is.factor(i))
                args[[iarg]] <- as.factor(i)
    }
    before <- args[[1]]
    after <- args[[2]]
    input.time <- args[[3]]
    input.levels <- length(levels(before))
    output.length <- input.levels * input.levels
    input.index <- rep(seq(output.length), input.length)
    output.vector <- rep(0, output.length)
    output.divisor <- rep(0, output.length)
    for(indx in seq(along = counts)) {
        output.vector[input.index[indx]] <- output.vector[input.index[
            indx]] + counts[indx]
    }
    for(indx in seq(along = counts)) {
        output.divisor[before[indx]] <- output.divisor[before[indx]] +
            counts[indx]
    }
# calculate transition probabilities
    output.index <- rep(as.vector(matrix(1:input.levels, nrow = ifelse(
        before[1] == before[2], input.levels, 1), ncol = input.levels,
        byrow = T)), length = output.length)
    output.vector <- output.vector/output.divisor[output.index]
    # calculate transition probabilities
    output <- matrix(output.vector, byrow = T, nrow = input.levels, ncol =
        input.levels)
    cat("First Order Markov Chain",
        "Estimated Stationary Transition Probabilities", "\n", sep =
        "\n")
    output
}
```

## 9.4 plotsurv()

```
function(input)
{
#
# Check arguments for validity
#
    ct <- cat
```

```
        if(nargs() == 0)
                stop("No arguments")
        if(!inherits(input, "surv.fit"))
                stop("Must provide surv.fit object")
        input.length <- length(input$time)
        input.time <- input$time
        input.time.log <- log(input$time)
        input.survival <- input$surv
        input.survival.log <- log(input$surv)
        input.strata <- input$strata
        if(is.null(input.strata)) input.strata <- rep("*", input.length)
#
#
# Plot of Empirical Survivor Curve
#
        plot(input.time, input.survival, type = "n", main =
                "Plot of S(t) vs. t - Empirical Survivor Curve", ylab = "S(t)",
                xlab = "t")
        text(input.time, input.survival, input.strata)  #
#
# Plot of Hazard Function
#
        input.tmp <- (c(input.survival[input.length], input.survival[seq(1,
                input.length - 1)]) - input.survival)/input.survival
        plot(input.time, input.tmp, type = "n", main =
                "Empirical Hazard Function", xlab = "t", ylab = "h(t)", ylim =
                c(0, 1), ask = T)
        text(input.time, input.tmp, input.strata)       #
#
# Prob Plot - Exponential Distribution
#
        plot(input.time, input.survival.log, type = "n", main =
                "Plot of log(S(t)) vs t - Exponential Distribution", ylab =
                "log(S(t))", xlab = "t", ask = T)
        text(input.time, input.survival.log, input.strata)      #
#
# Prob Plot - Pareto Distribution
#
        plot(input.time.log, input.survival.log, type = "n", main =
                "Plot of log(S(t)) vs log(t) - Pareto Distribution", ylab =
                "log(S(t))", xlab = "log(t)", ask = T)
        text(input.time.log, input.survival.log, input.strata)  #
#
```

```
# Prob Plot - Extreme Value Distribution
#
        input.survival.log <- log( - input.survival.log)
        plot(input.time, input.survival.log, type = "n", main =
                "Plot of log(-log(S(t))) vs t - Extreme Value Distribution",
                ylab = "log(-log(S(t)))", xlab = "t", ask = T)
        text(input.time, input.survival.log, input.strata)      #
#
# Prob Plot - Weibull Distribution
#
        plot(input.time.log, input.survival.log, type = "n", main =
                "Plot of log(-log(S(t))) vs log(t) - Weibull Distribution",
                ylab = "log(-log(S(t)))", xlab = "log(t)", ask = T)
        text(input.time.log, input.survival.log, input.strata)  #
#
# Prob Plot - Normal Distribution
#
        input.survival.log <- qnorm(1 - input.survival)
        plot(input.time, input.survival.log, type = "n", main =
                "Plot of qnorm(1-S(t)) vs t - Normal Distribution", ylab =
                "qnorm(1-S(t))", xlab = "t", ask = T)
        text(input.time, input.survival.log, input.strata)      #
#
# Prob Plot - Log Normal Distribution
#
        input.survival.log <- qnorm(1 - input.survival)
        plot(input.time.log, input.survival.log, type = "n", main =
                "Plot of qnorm(1-S(t)) vs log(t) - Log Normal Distribution",
                ylab = "qnorm(1-S(t))", xlab = "log(t)", ask = T)
        text(input.time.log, input.survival.log, input.strata)  #
#
# Prob Plot - Gamma Distribution
#
        input.tmp <- sqrt(input.time)
        plot(input.tmp, input.survival.log, type = "n", main =
                "Plot of qnorm(1-S(t)) vs sqrt(t) - Gamma Distribution", ylab
                 = "qnorm(1-S(t))", xlab = "sqrt(t)", ask = T)
        text(input.tmp, input.survival.log, input.strata)       #
#
# Prob Plot - Logistic Distribution
#
        input.survival.log <- log((1 - input.survival)/input.survival)
        plot(input.time, input.survival.log, type = "n", main =
```

```
                "Plot of log((1-S(t))/S(t)) vs t - Logistic Distribution", ylab
                 = "log((1-S(t))/S(t))", xlab = "t", ask = T)
        text(input.time, input.survival.log, input.strata)      #
#
# Prob Plot - Log Logistic Distribution
#
        input.survival.log <- log((1 - input.survival)/input.survival)
        plot(input.time.log, input.survival.log, type = "n", main =
          "Plot of log((1-S(t))/S(t)) vs log(t) - Log Logistic Distribution",
          ylab = "log((1-S(t))/S(t))", xlab = "log(t)", ask = T)
        text(input.time.log, input.survival.log, input.strata)  #
}
```

## 9.5 f.test()

```
function(mdl1, mdl2)
{
        if(!(inherits(mdl1, "lm") & inherits(mdl2, "lm")))
                stop("This function only takes models of class lm.")
        dev1 <- deviance(mdl1)
        dev2 <- deviance(mdl2)
        df1 <- mdl1$df
        df2 <- mdl2$df
        scale <- summary.lm(mdl2)$sigma^2
        score <- (dev1 - dev2)/(df1 - df2)/scale
        prob <- pf(score, df1 - df2, df2)
        cat("F test:", round(score, 5), "with df", df1 - df2, "and", df2,
                "and probability", round(1 - prob, 5), "\n", sep = " ")
}
```

# Chapter 10

# Data Tables

## 10.1 Liège Divorce Data

Data on divorces in the city of Liège, France. **A** is the origin of the divorce application (1 = Husband, 2 = Wife, 3 = Mutual). **HA** is the husband's age. **WA** is the wife's age. **LM** is the length of the marriage, in years. **LP** is the length of the divorce proceding, in months. **C** is the number of children.

| A | HA | WA | LM | LP | C |
|---|---|---|---|---|---|
| 3 | 37 | 35 | 11 | 13 | 1 |
| 3 | 38 | 35 | 14 | 12 | 1 |
| 3 | 41 | 37 | 15 | 12 | 2 |
| 3 | 32 | 48 | 07 | 12 | 1 |
| 3 | 39 | 36 | 17 | 12 | 2 |
| 3 | 59 | 57 | 38 | 12 | 2 |
| 3 | 30 | 30 | 08 | 12 | 1 |
| 3 | 45 | 42 | 21 | 12 | 3 |
| 3 | 32 | 31 | 09 | 12 | 0 |
| 3 | 36 | 26 | 06 | 12 | 0 |
| 3 | 35 | 32 | 05 | 12 | 0 |
| 3 | 37 | 37 | 11 | 12 | 2 |
| 3 | 32 | 32 | 13 | 12 | 2 |
| 3 | 37 | 28 | 05 | 12 | 0 |
| 3 | 33 | 30 | 04 | 12 | 0 |
| 3 | 35 | 30 | 09 | 12 | 0 |
| 3 | 38 | 39 | 20 | 12 | 2 |
| 3 | 45 | 55 | 22 | 12 | 1 |
| 3 | 38 | 35 | 16 | 12 | 1 |
| 3 | 26 | 26 | 05 | 12 | 0 |

| A | HA | WA | LM | LP | C |
|---|---|---|---|---|---|
| 3 | 34 | 34 | 11 | 13 | 1 |
| 3 | 35 | 33 | 14 | 12 | 1 |
| 3 | 44 | 41 | 22 | 12 | 2 |
| 3 | 41 | 39 | 16 | 12 | 1 |
| 3 | 44 | 27 | 08 | 12 | 0 |
| 3 | 33 | 33 | 09 | 12 | 1 |
| 3 | 29 | 26 | 07 | 12 | 1 |
| 3 | 36 | 32 | 11 | 12 | 0 |
| 3 | 38 | 35 | 13 | 12 | 0 |
| 3 | 38 | 39 | 15 | 12 | 0 |
| 3 | 30 | 26 | 04 | 12 | 0 |
| 3 | 41 | 40 | 14 | 12 | 3 |
| 3 | 32 | 35 | 09 | 12 | 1 |
| 3 | 52 | 44 | 24 | 12 | 0 |
| 3 | 25 | 26 | 04 | 12 | 0 |
| 3 | 33 | 27 | 05 | 12 | 1 |
| 3 | 36 | 31 | 08 | 12 | 1 |
| 3 | 41 | 34 | 12 | 12 | 2 |
| 3 | 28 | 25 | 06 | 12 | 0 |
| 3 | 30 | 26 | 07 | 12 | 0 |

| A | HA | WA | LM | LP | C |
|---|---|---|---|---|---|
| 3 | 33 | 30 | 10 | 12 | 2 |
| 3 | 35 | 31 | 08 | 12 | 1 |
| 3 | 28 | 28 | 07 | 12 | 2 |
| 3 | 27 | 26 | 06 | 12 | 1 |
| 3 | 33 | 34 | 12 | 12 | 1 |
| 3 | 38 | 37 | 15 | 12 | 3 |
| 3 | 57 | 57 | 24 | 12 | 1 |
| 3 | 30 | 29 | 09 | 12 | 0 |
| 3 | 32 | 28 | 09 | 12 | 0 |
| 3 | 33 | 39 | 05 | 12 | 0 |
| 3 | 34 | 34 | 08 | 12 | 2 |
| 3 | 37 | 33 | 14 | 12 | 1 |
| 3 | 38 | 33 | 04 | 12 | 0 |
| 3 | 27 | 26 | 04 | 12 | 0 |
| 3 | 54 | 54 | 30 | 12 | 1 |
| 3 | 35 | 31 | 13 | 12 | 2 |
| 3 | 26 | 28 | 06 | 12 | 1 |
| 3 | 30 | 26 | 07 | 12 | 1 |
| 3 | 42 | 41 | 16 | 12 | 0 |
| 3 | 35 | 40 | 04 | 12 | 0 |

| | | |
|---|---|---|
| 3 36 31 09 12 0 | 3 53 51 32 12 1 | 3 38 36 14 12 2 |
| 3 34 29 10 12 3 | 3 50 44 24 12 2 | 3 28 27 07 12 2 |
| 3 36 36 11 12 1 | 3 35 26 06 12 0 | 3 55 49 19 12 0 |
| 3 32 27 07 12 0 | 3 37 29 10 12 0 | 3 39 38 13 12 0 |
| 3 28 25 04 12 0 | 3 31 25 08 12 0 | 3 48 49 26 12 1 |
| 3 46 42 23 12 1 | 3 35 28 11 12 1 | 3 34 37 12 12 1 |
| 3 40 38 15 12 1 | 3 49 45 20 12 2 | 3 34 31 12 12 2 |
| 3 54 51 27 12 0 | 3 60 58 39 12 0 | 3 28 30 04 12 0 |
| 3 47 51 27 12 0 | 3 43 41 19 12 2 | 3 48 44 24 12 2 |
| 3 36 37 12 12 1 | 3 33 33 12 12 1 | 3 38 38 14 12 1 |
| 3 57 52 06 12 0 | 3 33 31 05 12 0 | 3 32 25 06 12 0 |
| 3 35 33 16 06 2 | 3 27 26 15 08 0 | 3 37 36 13 08 0 |
| 3 43 43 23 08 3 | 3 41 37 17 08 0 | 3 51 45 13 17 0 |
| 3 44 41 19 17 2 | 3 29 25 06 14 1 | 3 37 35 13 14 1 |
| 3 31 34 08 13 1 | 3 29 33 07 12 0 | 3 35 31 11 12 1 |
| 3 34 37 10 12 3 | 3 38 36 14 12 1 | 3 33 30 12 12 2 |
| 3 29 26 09 12 1 | 3 28 26 07 12 2 | 3 36 37 16 12 1 |
| 3 46 38 20 12 1 | 3 29 27 07 12 0 | 3 31 28 06 12 0 |
| 3 35 25 04 12 0 | 3 32 33 10 12 0 | 3 35 35 13 12 0 |
| 3 27 24 04 12 0 | 3 31 29 10 14 1 | 3 32 28 06 12 1 |
| 3 31 25 06 12 2 | 3 29 25 06 12 1 | 3 40 36 06 12 2 |
| 3 37 36 12 12 0 | 3 34 30 08 12 0 | 3 29 25 05 12 0 |
| 3 30 26 05 12 0 | 3 36 36 07 12 0 | 3 27 27 03 12 0 |
| 3 30 30 07 12 1 | 3 35 31 11 12 2 | 3 39 36 12 12 2 |
| 3 37 36 16 12 2 | 3 35 35 12 12 2 | 3 32 31 13 12 2 |
| 3 41 40 19 12 1 | 3 28 30 04 12 0 | 3 30 28 08 12 0 |
| 3 31 34 07 12 0 | 3 36 30 06 12 0 | 3 52 46 06 12 0 |
| 3 30 24 06 12 0 | 3 40 35 05 12 0 | 3 26 25 04 12 0 |
| 3 40 39 21 12 2 | 3 32 29 11 12 2 | 3 27 28 08 12 2 |
| 3 33 31 10 12 2 | 3 35 35 10 12 1 | 3 38 32 10 12 1 |
| 3 33 28 10 12 0 | 3 26 26 08 12 0 | 3 40 38 14 12 0 |
| 3 31 27 06 12 0 | 3 34 37 08 12 0 | 3 28 25 04 12 0 |
| 3 40 39 15 12 0 | 3 32 32 12 12 0 | 3 44 41 22 12 1 |
| 3 41 38 17 12 1 | 3 29 27 04 12 1 | 3 36 33 22 12 1 |
| 3 35 34 13 12 2 | 3 35 34 08 12 1 | 3 34 31 10 12 1 |
| 3 50 45 27 12 1 | 3 32 40 09 12 2 | 3 54 45 22 12 1 |
| 3 39 32 13 12 0 | 3 34 35 13 12 0 | 3 31 31 11 12 1 |
| 3 47 38 20 12 3 | 3 34 36 12 12 2 | 3 39 37 17 12 1 |
| 3 37 35 15 12 2 | 3 28 24 07 12 1 | 3 28 25 06 12 2 |
| 3 28 28 09 12 1 | 3 34 32 12 12 1 | 3 38 37 13 12 2 |
| 3 46 37 18 12 2 | 3 31 28 09 12 2 | 3 36 37 13 12 3 |
| 3 31 26 05 12 1 | 3 38 36 13 12 2 | 3 45 42 23 12 1 |
| 3 32 31 11 12 1 | 3 28 24 06 12 0 | 3 25 25 04 12 0 |

| | | |
|---|---|---|
| 3 26 26 04 12 0 | 3 38 38 11 12 0 | 3 34 31 10 12 0 |
| 3 27 25 05 12 1 | 3 28 26 06 12 0 | 3 32 26 07 12 1 |
| 3 33 32 13 12 1 | 3 31 25 04 12 0 | 3 47 45 22 12 0 |
| 3 29 25 06 12 0 | 3 29 28 05 12 0 | 3 31 34 12 12 2 |
| 3 30 28 09 12 1 | 3 40 39 14 12 2 | 3 45 34 13 12 1 |
| 3 37 28 07 12 1 | 3 38 38 17 12 2 | 3 39 34 19 12 1 |
| 3 58 54 32 12 0 | 3 33 26 08 12 1 | 3 35 34 08 12 1 |
| 3 40 38 15 12 3 | 3 45 46 21 12 1 | 3 42 45 12 12 1 |
| 3 41 40 20 12 2 | 3 25 38 06 12 0 | 3 28 29 07 12 0 |
| 3 38 51 15 12 0 | 3 36 35 13 12 0 | 3 40 44 04 12 0 |
| 3 37 38 03 12 0 | 3 31 29 05 12 0 | 3 32 33 13 12 0 |
| 3 25 26 05 12 0 | 3 49 44 25 12 0 | 3 33 30 04 12 0 |
| 3 50 42 08 12 0 | 3 27 29 04 12 0 | 3 27 30 05 12 1 |
| 3 37 39 16 12 2 | 3 25 27 05 12 1 | 3 33 29 12 12 1 |
| 3 34 34 09 12 1 | 3 34 31 08 12 2 | 3 43 40 10 12 0 |
| 3 27 25 05 12 0 | 3 71 72 46 12 0 | 3 55 54 35 12 0 |
| 3 43 41 25 12 0 | 3 32 32 11 12 0 | 3 35 33 13 12 2 |
| 3 30 32 06 12 1 | 3 33 30 11 12 1 | 3 39 30 09 12 1 |
| 3 37 25 05 12 1 | 3 37 38 15 12 1 | 3 41 37 18 12 2 |
| 3 37 35 15 12 1 | 3 35 35 14 12 2 | 3 32 32 11 12 1 |
| 3 35 32 06 12 1 | 3 26 29 05 12 2 | 3 31 30 11 12 1 |
| 3 46 47 23 12 2 | 3 38 30 09 12 1 | 3 27 25 06 12 2 |
| 3 31 32 12 12 2 | 3 36 31 11 12 1 | 3 35 31 12 12 2 |
| 3 33 32 14 12 0 | 3 26 25 05 12 0 | 3 53 45 16 12 0 |
| 3 58 40 15 12 0 | 3 41 44 10 12 0 | 3 29 30 06 12 0 |
| 3 25 25 05 12 0 | 3 42 38 17 12 0 | 3 37 35 14 12 0 |
| 3 49 37 10 12 0 | 3 39 36 13 12 2 | 3 45 42 20 12 3 |
| 3 47 38 16 12 2 | 3 38 35 14 12 2 | 3 42 42 14 12 2 |
| 3 36 38 08 12 1 | 3 32 29 09 12 1 | 3 43 37 15 12 2 |
| 3 27 26 04 12 0 | 3 74 59 38 12 0 | 3 31 32 08 12 0 |
| 3 36 31 04 12 0 | 3 47 35 11 12 0 | 3 43 39 20 12 1 |
| 3 33 33 11 12 1 | 3 37 35 13 12 1 | 3 41 37 17 12 1 |
| 3 33 29 08 12 1 | 3 45 40 21 12 3 | 3 28 29 05 12 1 |
| 3 29 25 06 12 0 | 3 30 26 05 12 0 | 3 29 34 06 12 0 |
| 3 35 33 13 12 1 | 3 40 37 17 12 2 | 3 26 33 12 12 2 |
| 3 48 45 23 12 2 | 3 26 25 05 12 1 | 3 35 33 13 12 2 |
| 3 46 30 04 12 0 | 3 34 30 11 12 1 | 3 45 45 19 12 1 |
| 3 39 38 15 12 2 | 3 36 36 13 12 2 | 3 39 35 11 12 1 |
| 3 33 31 09 12 0 | 3 30 25 07 12 0 | 3 29 33 05 12 0 |
| 3 32 27 07 12 0 | 3 33 29 11 12 0 | 3 30 33 04 12 0 |
| 3 38 38 18 12 3 | 3 38 36 17 12 2 | 3 27 28 06 12 1 |
| 3 30 27 03 12 0 | 3 34 30 10 12 0 | 3 28 25 03 12 0 |
| 3 37 37 15 12 0 | 3 29 27 04 12 0 | 3 31 29 08 12 0 |

```
3 49 40 22 12 2     3 37 41 17 12 1     3 35 34 12 12 3
3 37 32 17 12 1     3 30 30 11 12 1     3 31 29 09 12 1
3 26 27 07 12 1     3 33 34 10 12 0     3 29 27 07 12 0
3 34 30 08 12 0     3 70 68 12 12 0     3 33 27 03 12 0
3 46 39 21 12 0     3 40 41 19 12 2     3 36 29 07 12 0
3 35 34 07 12 1     3 39 40 16 12 2     3 31 26 04 12 1
3 37 33 14 12 1     3 46 47 23 12 2     3 47 42 23 12 0
3 37 36 15 12 1     3 53 48 27 12 1     3 37 32 07 12 2
3 28 25 04 12 1     3 47 47 23 12 1     3 36 32 12 12 1
3 35 30 13 12 2     3 30 27 05 12 0     3 25 33 04 12 0
3 31 28 08 12 0     3 37 40 10 12 0     3 25 25 06 12 0
3 32 32 10 12 0     3 39 35 13 12 1     3 37 36 17 12 3
3 40 41 16 12 2     3 49 40 24 12 3     3 37 27 12 12 1
3 34 36 13 12 1     3 29 25 07 12 1     3 42 39 16 12 0
3 50 48 26 12 0     3 43 42 15 12 0     3 32 31 12 12 1
3 40 44 18 12 1     3 30 29 11 12 1     3 27 26 09 12 1
3 42 48 21 12 2     3 37 27 07 12 0     3 29 28 10 12 0
3 29 24 04 12 0     3 29 28 08 13 1     3 29 29 08 13 0
3 46 44 23 13 0     3 29 25 04 13 0     3 31 31 11 13 1
3 47 43 20 12 3     3 33 30 08 12 1     3 37 36 15 12 1
3 31 31 10 12 2     3 40 37 19 12 2     3 30 26 05 12 0
3 48 46 22 12 0     3 26 24 04 12 0     3 34 34 12 12 0
3 35 34 13 12 1     3 30 27 06 12 1     3 33 26 07 12 1
3 44 38 12 12 1     3 48 44 26 12 1     3 37 33 12 12 1
3 34 25 08 12 2     3 36 32 12 12 2     3 33 32 11 12 0
3 28 24 07 12 0     3 29 31 09 12 0     3 28 26 05 12 0
3 38 37 15 12 0     3 29 28 07 12 1     3 57 43 25 12 1
3 33 33 13 12 1     3 53 50 30 12 3     3 33 27 07 12 2
3 40 40 20 12 3     3 39 37 21 12 2     3 31 30 06 12 2
3 37 30 10 12 1     3 31 27 09 12 1     3 29 34 09 12 1
3 27 27 07 12 0     3 57 45 17 12 0     3 26 31 03 12 0
3 29 28 03 13 0     3 57 51 32 12 0     3 40 41 19 12 0
3 51 49 27 12 0     3 30 33 09 12 0     3 32 30 12 12 0
3 36 26 04 12 0     3 31 28 08 12 0     3 37 36 13 13 2
3 38 35 11 13 2     3 34 31 09 13 1     3 38 33 13 13 1
3 28 24 06 12 1     3 27 29 06 12 2     3 34 32 11 12 1
3 34 31 10 12 2     3 32 32 10 12 1     3 54 41 19 12 2
3 38 37 14 12 2     3 35 36 13 12 2     3 27 25 08 12 2
3 27 28 04 12 0     3 49 36 12 12 1     3 34 33 11 12 2
3 38 35 99 20 9     3 35 37 99 19 9     3 50 48 99 19 9
3 32 27 99 19 9     3 30 29 99 19 9     3 27 28 99 18 9
3 45 42 99 17 9     3 40 37 99 17 9     3 57 53 99 17 9
3 35 32 99 14 9     3 56 55 99 11 9     3 34 31 99 11 9
```

| | | |
|---|---|---|
| 3 48 44 99 09 9 | 3 35 31 99 09 9 | 3 43 50 99 08 9 |
| 3 51 47 99 08 9 | 3 24 25 99 07 9 | 3 44 42 99 06 9 |
| 3 38 37 99 06 9 | 3 32 29 05 12 1 | 3 45 37 16 12 1 |
| 3 28 28 08 12 0 | 3 27 26 03 12 0 | 3 46 47 08 12 0 |
| 3 32 34 07 12 1 | 3 32 30 09 12 1 | 3 34 29 12 12 2 |
| 3 34 36 05 12 0 | 3 42 40 20 12 3 | 3 34 30 09 11 1 |
| 3 30 26 07 12 2 | 3 44 38 18 12 1 | 3 30 32 08 12 2 |
| 3 33 32 12 12 1 | 3 43 42 22 12 3 | 3 37 35 16 12 2 |
| 3 49 50 30 12 0 | 3 29 29 07 12 0 | 3 42 41 19 12 3 |
| 3 38 37 16 12 1 | 3 34 29 08 12 1 | 3 34 31 11 12 1 |
| 3 30 28 08 12 1 | 3 32 32 11 12 1 | 3 41 40 20 12 1 |
| 3 31 28 09 12 1 | 3 30 29 06 12 0 | 3 79 64 10 12 0 |
| 3 27 28 06 12 0 | 3 31 39 12 12 0 | 3 30 57 04 12 0 |
| 3 27 26 15 12 1 | 3 47 44 22 12 2 | 3 34 28 06 12 2 |
| 3 26 25 03 12 0 | 3 48 44 20 12 0 | 3 41 37 17 12 0 |
| 3 31 28 05 12 0 | 3 29 31 05 12 0 | 3 44 42 20 12 2 |
| 3 40 29 11 12 1 | 3 37 39 17 12 2 | 3 32 30 12 12 3 |
| 3 36 35 15 12 1 | 3 32 29 07 12 1 | 3 32 29 08 12 0 |
| 3 29 29 07 12 0 | 3 36 33 11 12 0 | 3 28 30 04 12 0 |
| 3 33 36 05 12 0 | 3 29 28 04 12 0 | 3 25 24 05 12 0 |
| 3 38 33 13 12 2 | 3 27 24 06 12 1 | 3 26 28 06 12 1 |
| 3 45 46 24 12 1 | 3 47 46 21 12 2 | 3 50 49 25 12 4 |
| 3 38 35 13 12 2 | 3 44 38 16 12 2 | 3 32 29 04 12 1 |
| 3 46 43 20 12 2 | 3 30 25 06 12 1 | 3 42 40 21 12 1 |
| 3 30 27 05 12 1 | 3 37 34 15 12 1 | 3 45 42 21 12 1 |
| 3 35 30 05 12 1 | 3 36 36 12 12 2 | 3 37 34 14 12 1 |
| 3 31 32 11 12 1 | 3 31 33 13 12 1 | 3 32 31 12 12 2 |
| 3 30 28 11 12 3 | 3 28 28 07 12 1 | 3 37 24 05 12 1 |
| 3 30 33 09 12 1 | 3 26 33 03 12 0 | 3 44 42 11 12 0 |
| 3 41 37 09 12 0 | 3 31 29 07 12 0 | 3 42 32 03 12 0 |
| 3 31 29 08 12 0 | 3 45 30 03 12 0 | 3 29 24 08 12 0 |
| 3 25 26 03 12 0 | 3 47 42 19 12 1 | 3 31 31 10 12 1 |
| 3 40 37 19 12 1 | 3 38 36 12 12 1 | 3 35 37 11 12 2 |
| 3 28 26 05 12 1 | 3 29 33 07 12 1 | 3 37 35 13 12 1 |
| 3 24 29 03 12 0 | 3 34 30 09 12 0 | 3 28 25 03 12 0 |
| 3 28 27 07 12 0 | 3 28 28 07 12 0 | 3 27 24 06 12 0 |
| 3 27 31 03 12 1 | 3 28 24 05 12 0 | 3 51 46 26 12 2 |
| 3 41 34 12 12 1 | 3 39 31 12 12 1 | 3 32 32 10 12 2 |
| 3 33 31 08 12 1 | 3 34 32 11 12 1 | 3 28 26 06 12 0 |
| 3 25 24 03 12 0 | 3 37 37 12 12 0 | 3 45 45 23 12 1 |
| 3 41 40 20 12 2 | 3 29 27 07 12 1 | 3 37 33 12 12 2 |
| 3 54 41 11 12 1 | 3 35 33 11 12 1 | 3 38 27 03 12 0 |
| 3 29 27 04 12 0 | 3 55 55 33 12 0 | 3 32 24 07 12 1 |

| | | |
|---|---|---|
| 3 47 39 13 12 0 | 3 25 28 04 12 0 | 3 37 36 15 12 1 |
| 3 43 41 20 12 0 | 3 34 32 12 12 2 | 3 44 44 21 12 2 |
| 3 45 39 16 12 1 | 3 43 37 19 12 3 | 3 31 25 07 12 1 |
| 3 51 49 24 12 1 | 3 25 26 06 12 0 | 3 37 36 17 12 0 |
| 3 28 24 05 12 0 | 3 29 26 06 12 0 | 3 27 27 05 12 0 |
| 3 53 49 12 12 0 | 3 49 48 30 12 0 | 3 45 45 23 12 1 |
| 3 30 28 06 12 2 | 3 39 36 18 12 1 | 3 36 34 15 12 1 |
| 3 36 34 15 12 3 | 3 31 27 09 12 3 | 3 37 37 13 12 2 |
| 3 38 34 17 12 1 | 3 34 32 11 12 2 | 3 36 36 14 12 1 |
| 3 27 26 08 12 2 | 3 30 25 05 12 1 | 3 31 29 11 12 1 |
| 3 27 28 07 12 1 | 3 31 27 08 12 0 | 3 39 34 17 12 0 |
| 3 26 24 04 12 0 | 3 33 31 06 12 0 | 3 37 34 13 12 0 |
| 3 26 27 05 12 0 | 3 35 29 05 12 0 | 3 33 37 03 12 0 |
| 3 38 32 11 12 0 | 3 32 28 09 12 0 | 3 38 38 99 13 9 |
| 3 33 36 99 15 9 | 3 29 24 99 14 9 | 3 26 27 99 14 9 |
| 3 41 35 99 14 9 | 3 29 32 99 15 9 | 3 39 37 99 15 9 |
| 3 53 42 99 07 9 | 3 26 25 99 07 9 | 3 40 43 18 12 1 |
| 3 39 36 18 12 1 | 3 31 33 09 12 2 | 3 46 34 06 12 1 |
| 3 34 31 11 12 1 | 3 46 34 13 12 1 | 3 29 30 09 12 2 |
| 3 46 35 10 12 1 | 3 45 44 20 12 2 | 3 37 37 15 12 1 |
| 3 43 38 18 12 3 | 3 30 25 08 12 0 | 3 31 28 03 12 0 |
| 3 26 26 04 12 0 | 3 34 54 03 12 0 | 3 24 24 03 12 0 |
| 3 29 24 03 12 0 | 3 25 26 04 12 0 | 3 41 37 09 12 0 |
| 3 33 30 09 12 0 | 3 26 31 03 12 0 | 3 26 26 03 12 0 |
| 2 31 28 09 10 0 | 2 39 36 14 99 0 | 2 46 40 19 26 0 |
| 1 35 31 13 07 0 | 2 30 28 06 12 0 | 2 44 38 15 12 0 |
| 1 57 52 30 03 0 | 1 44 40 22 03 1 | 1 37 52 15 02 0 |
| 1 24 24 05 10 0 | 1 57 53 35 03 1 | 2 48 48 24 09 0 |
| 2 50 61 16 10 0 | 2 29 27 06 39 0 | 2 46 41 23 19 0 |
| 2 28 27 07 08 0 | 2 52 48 21 22 0 | 2 30 27 09 37 0 |
| 1 36 53 15 06 1 | 1 65 64 34 06 0 | 2 34 26 06 13 0 |
| 2 45 42 20 12 0 | 2 32 32 09 55 1 | 2 41 40 18 31 0 |
| 1 66 63 40 04 0 | 1 66 69 43 05 0 | 2 28 27 08 49 0 |
| 2 43 38 19 03 1 | 1 46 39 19 02 0 | 1 28 26 09 13 0 |
| 2 33 31 07 24 0 | 1 53 53 32 07 0 | 2 42 31 06 08 0 |
| 1 34 32 05 09 0 | 2 29 26 03 03 0 | 2 32 29 10 10 0 |
| 2 33 34 13 14 0 | 2 44 38 20 10 0 | 1 26 21 03 04 0 |
| 1 41 32 07 04 0 | 1 36 41 03 06 0 | 2 30 29 08 13 0 |
| 1 26 24 03 08 0 | 2 46 51 24 72 2 | 2 44 40 19 08 0 |
| 2 30 29 08 25 0 | 2 44 37 13 34 0 | 2 42 38 19 14 1 |
| 1 26 26 08 21 0 | 1 38 36 16 03 0 | 1 28 27 08 09 0 |
| 2 36 38 15 59 0 | 2 52 44 24 15 0 | 2 76 66 42 23 0 |
| 2 42 38 21 04 5 | 2 57 55 16 04 0 | 1 49 55 15 04 0 |

| | | |
|---|---|---|
| 1 37 36 13 04 0 | 1 43 52 18 04 1 | 2 38 36 16 05 0 |
| 2 41 40 14 12 0 | 1 33 29 09 04 0 | 2 55 53 29 04 2 |
| 2 28 26 07 14 0 | 2 33 30 10 03 1 | 1 54 54 31 03 0 |
| 2 40 33 09 07 0 | 2 34 26 09 05 0 | 2 34 29 12 03 1 |
| 1 42 40 20 09 1 | 1 55 53 33 07 0 | 2 49 49 29 11 0 |
| 2 35 29 09 31 0 | 2 39 36 13 04 0 | 1 31 29 09 07 0 |
| 2 40 37 17 08 0 | 2 31 27 07 06 0 | 2 29 22 05 31 0 |
| 2 45 41 22 07 0 | 1 30 28 08 12 0 | 1 35 31 08 04 0 |
| 1 46 25 09 10 0 | 1 64 64 40 04 0 | 2 38 34 14 13 1 |
| 1 47 45 24 03 2 | 1 53 54 32 04 0 | 1 49 48 23 03 1 |
| 1 64 59 37 03 0 | 2 40 39 19 11 0 | 1 59 56 37 08 0 |
| 2 42 37 18 18 0 | 2 64 47 22 99 0 | 2 55 37 12 41 0 |
| 1 38 43 14 13 0 | 1 50 55 11 82 0 | 2 33 34 09 17 0 |
| 2 27 26 06 30 0 | 1 33 34 04 32 0 | 2 36 36 17 33 0 |
| 1 44 37 08 30 0 | 1 55 52 13 11 0 | 1 30 30 09 24 0 |
| 2 45 43 20 10 1 | 1 55 49 35 02 0 | 2 35 32 15 08 2 |
| 2 26 27 06 15 0 | 2 38 33 14 02 2 | 2 54 47 13 02 0 |
| 1 39 43 18 03 2 | 2 53 53 30 03 0 | 1 40 37 04 13 0 |
| 1 54 63 31 02 0 | 1 28 28 06 08 0 | 1 35 31 12 07 0 |
| 2 36 26 09 07 0 | 1 35 31 12 10 0 | 1 25 22 03 07 0 |
| 2 36 32 17 03 0 | 1 51 44 26 04 0 | 1 41 43 23 04 0 |
| 1 28 27 10 11 0 | 1 26 22 02 15 0 | 2 24 20 05 04 0 |
| 2 31 28 08 03 0 | 2 38 32 11 43 0 | 2 61 50 17 22 2 |
| 2 29 27 06 20 0 | 1 36 27 10 13 0 | 2 39 39 11 03 0 |
| 2 35 31 14 11 0 | 1 29 26 09 22 0 | 2 27 24 03 19 0 |
| 2 39 40 13 06 0 | 1 47 44 23 23 0 | 2 32 26 09 06 0 |
| 2 33 25 05 34 0 | 2 31 37 08 22 0 | 1 48 47 25 09 0 |
| 2 44 44 19 04 0 | 1 61 57 39 04 0 | 1 38 40 11 04 0 |
| 1 64 58 39 20 0 | 1 46 38 22 17 0 | 1 39 35 14 09 0 |
| 2 37 32 17 03 1 | 1 50 48 23 74 0 | 2 31 30 10 08 1 |
| 2 53 49 28 10 0 | 1 51 41 21 08 4 | 1 48 45 24 13 0 |
| 1 41 40 21 20 0 | 2 41 39 13 31 0 | 1 27 26 02 06 0 |
| 1 61 40 15 34 0 | 2 32 28 05 27 0 | 1 31 31 03 26 0 |
| 2 36 35 13 04 0 | 2 32 28 09 03 0 | 1 38 36 17 31 0 |
| 1 53 58 33 07 1 | 2 45 38 17 28 0 | 1 40 57 15 05 0 |
| 1 48 47 25 50 0 | 2 32 30 09 05 0 | 2 40 37 17 09 1 |
| 1 62 58 37 03 0 | 1 59 56 38 03 0 | 2 40 41 18 05 0 |
| 2 49 48 12 24 0 | 2 41 35 16 10 2 | 2 41 39 22 17 0 |
| 1 26 23 05 24 0 | 2 36 33 15 30 0 | 2 37 35 12 32 0 |
| 2 43 39 19 42 0 | 2 33 30 09 03 1 | 1 34 25 08 06 0 |
| 1 27 27 08 65 0 | 1 30 27 07 11 0 | 2 40 41 17 18 0 |
| 1 31 26 08 04 0 | 1 56 47 27 18 0 | 1 31 31 06 12 0 |
| 1 22 21 02 07 0 | 1 33 29 09 04 0 | 1 46 47 21 06 0 |

| | | |
|---|---|---|
| 1 63 65 38 08 0 | 1 44 59 15 04 0 | 2 48 28 08 24 0 |
| 2 30 27 10 10 0 | 2 38 36 13 10 0 | 1 49 47 30 13 0 |
| 2 58 58 34 50 0 | 2 44 35 17 07 0 | 1 61 60 38 14 4 |
| 2 33 30 07 10 0 | 1 37 47 10 07 0 | 1 85 64 07 69 0 |
| 1 62 57 37 04 0 | 2 36 30 09 10 1 | 2 27 24 02 08 0 |
| 2 36 32 09 14 0 | 1 35 37 09 04 0 | 1 50 43 21 06 0 |
| 1 34 33 14 24 0 | 2 34 36 14 25 0 | 2 39 38 15 05 0 |
| 1 41 31 11 10 0 | 1 51 36 11 20 0 | 2 50 48 24 20 0 |
| 1 50 48 26 17 1 | 2 38 41 16 25 0 | 2 43 44 19 03 0 |
| 1 33 28 11 13 0 | 1 67 52 31 10 0 | 2 39 36 16 08 1 |
| 2 40 37 08 24 0 | 2 24 26 03 29 0 | 1 40 38 15 45 0 |
| 2 30 22 04 27 0 | 1 49 40 22 47 0 | 2 43 33 12 43 0 |
| 2 36 35 07 28 0 | 1 46 45 24 17 0 | 2 40 25 07 59 0 |
| 1 55 57 34 06 0 | 1 74 71 52 36 0 | 1 47 44 26 03 0 |
| 2 27 23 06 20 0 | 2 36 39 12 05 0 | 1 27 26 09 37 0 |
| 2 29 27 09 31 1 | 2 42 41 23 09 0 | 1 49 47 26 07 0 |
| 2 32 33 09 33 0 | 2 43 35 13 10 1 | 2 33 34 12 03 1 |
| 2 44 41 20 06 0 | 1 32 28 07 03 0 | 1 54 52 04 06 0 |
| 1 56 51 27 04 1 | 1 57 56 31 20 0 | 1 36 35 14 06 2 |
| 1 60 58 40 02 0 | 1 56 56 30 08 1 | 1 51 50 29 03 0 |
| 2 48 48 27 02 0 | 1 59 57 38 04 0 | 1 41 54 24 03 0 |
| 1 28 29 08 09 0 | 2 31 26 10 03 0 | 1 30 28 05 05 0 |
| 1 50 38 19 18 0 | 2 33 38 08 16 0 | 2 34 36 13 15 0 |
| 1 39 35 13 05 0 | 2 39 37 18 05 0 | 1 31 27 07 05 0 |
| 1 33 27 09 26 0 | 1 31 26 08 04 0 | 1 32 31 10 14 0 |
| 1 30 24 04 04 0 | 2 26 24 03 06 0 | 1 37 37 14 20 0 |
| 2 48 48 28 14 0 | 2 34 32 13 99 0 | 2 54 53 28 99 0 |
| 1 59 62 36 36 0 | 2 58 54 33 99 0 | 2 30 29 09 29 0 |
| 1 24 21 06 32 0 | 2 42 36 16 17 0 | 1 35 30 08 17 0 |
| 1 53 55 33 46 0 | 2 42 42 17 07 0 | 1 33 26 08 17 0 |
| 1 33 36 02 06 0 | 2 35 34 11 02 0 | 1 53 63 23 04 0 |
| 2 45 44 21 11 0 | 2 39 34 06 09 0 | 1 33 31 07 07 0 |
| 2 47 48 29 10 0 | 2 36 29 07 05 1 | 1 39 37 17 06 0 |
| 2 35 30 12 15 0 | 1 45 40 09 11 1 | 2 51 62 11 09 0 |
| 2 21 24 03 22 0 | 2 33 32 08 35 0 | 1 45 41 14 10 0 |
| 1 26 23 05 13 0 | 2 49 38 10 10 0 | 1 53 51 30 13 1 |
| 2 55 50 31 11 0 | 1 39 33 10 03 0 | 1 51 51 33 05 0 |
| 1 65 55 33 04 0 | 2 40 23 05 18 0 | 2 48 43 17 04 0 |
| 2 30 28 08 33 0 | 2 30 31 10 05 0 | 2 33 33 08 07 0 |
| 1 37 34 16 19 0 | 2 38 41 20 18 0 | 1 57 37 15 10 0 |
| 2 45 50 24 10 0 | 1 30 26 08 05 0 | 1 33 22 03 04 0 |
| 1 34 31 14 36 0 | 2 47 36 16 40 0 | 2 27 24 06 45 0 |
| 2 41 34 13 26 0 | 2 38 34 11 29 0 | 1 36 32 12 17 0 |

| | | |
|---|---|---|
| 2 46 41 16 09 0 | 2 31 28 10 38 0 | 2 38 35 11 09 0 |
| 2 42 38 17 84 0 | 1 27 25 05 33 0 | 1 37 35 14 03 0 |
| 1 40 28 10 06 0 | 1 36 35 09 06 0 | 2 42 41 21 21 0 |
| 2 29 24 07 23 0 | 2 27 21 03 13 0 | 2 33 30 12 02 1 |
| 1 38 34 11 11 0 | 1 49 41 21 11 0 | 2 53 57 18 05 0 |
| 2 55 55 33 02 1 | 2 40 33 11 42 0 | 2 45 44 23 24 0 |
| 2 31 28 09 30 0 | 1 28 25 05 19 0 | 2 29 33 04 33 0 |
| 1 48 46 24 23 0 | 2 24 25 06 30 0 | 1 39 38 19 20 0 |
| 2 32 32 08 38 0 | 2 29 26 06 55 0 | 2 37 35 15 28 0 |
| 1 59 56 35 12 0 | 2 48 32 11 04 0 | 2 47 46 24 15 0 |
| 1 52 46 28 04 0 | 1 26 23 04 07 0 | 1 27 36 04 03 0 |
| 2 44 43 22 05 0 | 2 49 49 25 09 0 | 2 41 38 20 17 0 |
| 2 29 27 06 06 0 | 2 31 28 12 13 0 | 1 43 40 20 05 0 |
| 2 31 30 08 24 0 | 2 42 39 20 18 0 | 1 55 39 20 18 0 |
| 1 37 31 11 04 0 | 1 37 30 14 23 0 | 1 36 37 18 39 0 |
| 2 41 36 15 61 0 | 1 47 37 16 13 0 | 2 33 30 08 06 0 |
| 2 32 28 11 22 0 | 2 30 28 09 07 0 | 1 29 28 08 06 0 |
| 1 59 49 33 03 1 | 2 30 29 10 05 0 | 2 41 38 19 03 2 |
| 2 56 48 12 06 0 | 2 23 22 05 09 0 | 1 49 51 28 06 0 |
| 2 47 41 20 05 0 | 1 46 41 20 07 2 | 1 48 48 26 02 0 |
| 1 48 46 23 02 0 | 1 55 40 10 06 0 | 1 65 63 43 03 0 |
| 2 28 27 05 33 0 | 2 46 44 24 05 0 | 2 29 26 06 50 0 |
| 1 42 30 14 05 0 | 1 27 25 05 17 0 | 2 44 30 12 06 1 |
| 1 29 27 07 30 0 | 1 32 32 08 39 0 | 2 39 32 13 43 0 |
| 1 38 34 14 42 0 | 2 48 42 23 23 0 | 1 62 61 36 06 0 |
| 1 60 53 34 04 0 | 2 37 37 17 21 0 | 2 36 30 10 73 0 |
| 2 53 48 30 14 0 | 1 67 58 34 17 0 | 1 39 32 15 40 0 |
| 1 46 50 16 25 0 | 1 45 40 23 10 0 | 2 26 23 05 31 0 |
| 2 40 42 17 08 0 | 1 45 37 15 38 0 | 2 55 52 15 31 2 |
| 2 50 41 09 03 0 | 1 39 39 18 08 0 | 1 44 41 22 09 0 |
| 2 42 35 19 03 2 | 2 34 30 12 12 0 | 2 38 33 13 08 0 |
| 2 46 35 17 12 0 | 1 38 32 17 07 0 | 2 31 26 08 16 0 |
| 2 26 27 06 42 0 | 1 38 34 16 13 0 | 1 49 50 27 04 0 |
| 1 32 25 11 20 0 | 1 39 33 11 09 0 | 2 38 31 14 58 0 |
| 1 24 21 04 05 0 | 2 64 67 41 18 0 | 2 39 40 18 07 0 |
| 1 53 52 29 04 1 | 1 58 55 36 07 0 | 1 45 41 08 08 0 |
| 1 31 21 03 11 0 | 2 52 51 21 06 0 | 2 29 31 11 42 0 |
| 2 33 35 14 19 0 | 1 33 29 03 18 0 | 2 30 28 06 05 0 |
| 1 30 28 05 03 0 | 1 37 36 13 06 0 | 1 27 24 04 04 0 |
| 2 39 38 17 22 0 | 1 27 23 04 04 0 | 1 59 38 11 05 0 |
| 1 24 25 03 25 0 | 2 49 48 26 06 0 | 2 36 35 16 07 0 |
| 2 42 35 19 48 0 | 2 27 28 06 11 0 | 1 29 26 06 55 0 |
| 1 30 30 07 04 0 | 2 35 38 16 15 0 | 2 35 28 08 69 0 |

| | | |
|---|---|---|
| 2 34 36 16 66 0 | 1 48 44 25 21 0 | 2 30 28 09 34 0 |
| 1 28 28 08 26 0 | 1 28 23 04 15 0 | 2 33 35 15 04 0 |
| 2 46 33 13 55 0 | 1 24 24 05 11 0 | 2 38 39 16 22 0 |
| 1 46 48 16 08 0 | 2 31 30 10 14 0 | 1 48 42 23 05 0 |
| 2 39 33 08 05 0 | 1 27 23 06 08 0 | 2 40 41 20 06 0 |
| 1 26 24 07 16 0 | 1 34 26 09 06 0 | 2 38 41 12 12 0 |
| 1 34 31 14 09 0 | 2 45 49 12 06 0 | 2 53 49 04 04 0 |
| 1 31 37 11 23 0 | 1 41 37 17 02 0 | 1 53 53 32 06 0 |
| 1 34 30 13 20 0 | 2 41 38 19 04 2 | 2 29 26 06 04 0 |
| 2 39 27 08 13 0 | 2 40 36 06 39 0 | 2 46 40 19 31 0 |
| 2 42 35 11 40 0 | 2 38 27 04 15 0 | 1 41 47 18 11 0 |
| 2 31 28 08 38 0 | 1 37 38 16 20 0 | 2 51 45 26 10 0 |
| 2 33 26 05 05 0 | 1 26 28 08 05 0 | 2 41 33 13 27 0 |
| 2 41 42 19 08 1 | 2 47 45 17 07 0 | 2 35 31 11 05 0 |
| 2 26 23 03 16 0 | 2 36 29 10 17 2 | 1 49 56 27 04 0 |
| 2 37 37 13 08 0 | 1 43 29 10 14 0 | 2 71 64 43 02 0 |
| 1 54 53 31 02 0 | 2 46 40 18 04 0 | 2 46 49 23 22 0 |
| 2 33 35 14 03 0 | 1 56 56 38 16 0 | 1 39 38 18 04 0 |
| 1 32 28 10 06 0 | 2 28 29 04 04 0 | 1 35 35 09 23 0 |
| 2 40 40 20 19 0 | 2 39 35 18 31 0 | 1 28 30 07 09 0 |
| 1 32 28 07 14 0 | 1 28 27 10 11 0 | 1 29 21 04 17 0 |
| 1 32 21 05 10 0 | 1 32 30 13 07 1 | 1 26 27 07 07 0 |
| 2 37 36 17 07 0 | 1 35 34 13 07 0 | 2 53 51 29 81 0 |
| 2 48 45 24 22 0 | 1 37 34 16 61 0 | 2 37 32 14 05 0 |
| 2 32 26 07 07 0 | 2 31 29 08 34 0 | 2 36 35 13 09 0 |
| 1 31 28 07 16 0 | 2 30 32 10 04 0 | 2 40 28 08 32 0 |
| 1 25 21 02 07 0 | 2 35 37 16 05 0 | 1 25 22 03 04 0 |
| 1 25 22 04 05 0 | 1 26 20 04 05 0 | 1 36 34 08 10 0 |
| 2 27 29 07 12 0 | 2 37 27 03 07 0 | 1 45 57 27 04 0 |
| 1 39 32 11 10 0 | 1 59 57 36 06 0 | 2 38 40 18 19 0 |
| 1 48 42 18 25 0 | 1 24 22 04 18 0 | 2 31 27 11 41 0 |
| 2 37 35 12 82 1 | 1 38 40 17 16 2 | 2 64 55 30 14 0 |
| 2 48 46 27 28 0 | 2 53 50 30 50 0 | 2 37 26 07 32 0 |
| 2 47 44 23 29 0 | 2 41 40 20 29 0 | 1 36 41 09 20 0 |
| 2 36 29 12 69 0 | 1 38 37 17 40 0 | 2 35 33 14 11 0 |
| 2 27 21 05 03 0 | 1 64 60 20 15 0 | 1 25 21 05 17 0 |
| 2 40 35 15 04 0 | 2 36 35 14 45 0 | 2 41 39 14 04 0 |
| 2 24 21 02 25 0 | 2 60 60 39 06 0 | 2 41 39 18 06 1 |
| 2 44 41 23 07 0 | 2 44 40 21 05 0 | 1 58 51 27 29 0 |
| 1 35 33 02 06 0 | 1 34 27 04 03 0 | 1 40 38 20 08 0 |
| 2 29 30 10 40 0 | 1 28 27 08 55 0 | 2 32 28 12 29 0 |
| 1 24 25 05 41 0 | 2 46 55 11 68 0 | 1 35 39 13 13 0 |
| 1 32 34 14 36 0 | 2 30 27 11 47 0 | 2 35 37 15 31 0 |

| | | |
|---|---|---|
| 1 55 55 03 25 0 | 1 31 32 07 24 0 | 2 26 24 03 09 0 |
| 2 24 22 04 28 0 | 1 29 26 08 61 0 | 2 33 29 11 58 0 |
| 2 29 26 07 20 0 | 2 27 26 02 08 0 | 1 38 29 07 07 0 |
| 1 49 44 23 09 0 | 2 40 36 15 05 0 | 2 38 36 17 24 0 |
| 2 45 42 20 11 0 | 1 49 39 18 05 0 | 1 24 25 06 08 0 |
| 1 29 29 09 03 0 | 1 29 25 07 19 0 | 1 26 26 06 52 0 |
| 2 32 31 07 02 0 | 2 40 40 21 14 0 | 2 30 26 08 43 0 |
| 2 33 38 08 19 0 | 2 43 45 13 08 0 | 2 41 43 17 09 0 |
| 1 46 43 24 03 0 | 1 64 64 42 02 0 | 2 43 39 18 65 1 |
| 2 38 42 13 05 2 | 2 38 36 10 49 0 | 2 37 33 13 03 0 |
| 1 45 49 24 03 0 | 2 46 33 14 13 0 | 1 61 64 41 07 0 |
| 2 31 30 10 20 0 | 1 28 27 06 15 0 | 2 28 24 04 23 0 |
| 2 39 38 19 17 0 | 1 28 25 04 08 0 | 2 28 26 08 17 0 |
| 2 28 29 09 10 0 | 2 31 34 08 04 0 | 1 35 33 15 06 0 |
| 1 51 51 33 08 0 | 2 32 35 09 05 0 | 1 62 61 37 01 0 |
| 2 40 38 17 08 0 | 1 36 35 09 08 0 | 2 27 29 08 04 0 |
| 2 37 33 08 09 0 | 2 26 23 03 08 0 | 2 36 31 15 09 0 |
| 2 34 28 10 35 0 | 1 42 43 20 16 0 | 1 34 30 11 21 0 |
| 2 39 31 13 25 0 | 2 43 36 15 44 0 | 1 46 45 24 13 0 |
| 2 26 22 01 03 0 | 2 40 32 11 03 0 | 2 32 41 10 29 0 |
| 2 41 33 06 19 0 | 1 30 29 04 04 0 | 1 53 55 15 05 1 |
| 1 41 40 19 04 1 | 2 44 41 19 29 0 | 1 63 56 31 04 0 |
| 2 37 34 15 05 3 | 1 51 49 30 07 0 | 2 53 52 30 03 0 |
| 2 41 36 20 03 1 | 2 29 26 05 09 0 | 2 38 34 16 10 0 |
| 1 48 47 29 04 0 | 1 32 29 11 08 2 | 2 28 24 08 35 0 |
| 1 53 53 32 13 0 | 1 27 28 08 03 0 | 2 30 27 10 11 2 |
| 1 43 50 08 09 0 | 2 31 29 09 20 0 | 1 40 43 18 16 0 |
| 1 39 34 16 36 0 | 2 36 36 16 20 0 | 2 29 24 07 25 0 |
| 2 41 31 05 21 0 | 2 27 27 08 12 0 | 1 49 42 24 64 0 |
| 2 50 54 28 31 0 | 2 62 48 26 39 0 | 2 39 39 19 79 0 |
| 2 32 34 10 28 0 | 2 30 29 10 44 0 | 2 45 45 23 25 0 |
| 2 32 24 05 37 0 | 2 36 33 11 50 0 | 1 50 53 28 06 0 |
| 2 30 24 05 35 0 | 2 30 28 08 48 0 | 2 40 37 18 06 0 |
| 2 28 26 08 13 0 | 1 64 56 36 12 0 | 2 36 31 12 07 0 |
| 1 27 29 06 08 0 | 1 34 34 12 16 0 | 2 58 55 30 05 0 |
| 2 38 35 10 07 0 | 1 34 31 08 06 0 | 2 33 29 11 05 0 |
| 2 40 41 16 06 0 | 1 50 54 23 04 0 | 2 37 35 18 06 0 |
| 2 42 39 20 23 0 | 1 28 22 05 09 0 | 1 51 37 13 46 0 |
| 2 28 24 07 17 0 | 2 25 22 03 21 0 | 1 35 34 15 08 0 |
| 1 29 21 04 21 0 | 2 36 31 15 12 0 | 2 27 27 06 07 0 |
| 1 37 34 15 08 0 | 1 37 37 17 09 0 | 1 38 38 02 08 0 |
| 2 36 24 03 07 0 | 2 45 42 24 11 0 | 1 34 31 12 27 0 |
| 2 32 26 05 06 0 | 1 60 65 39 23 0 | 1 45 45 24 07 0 |

| | | |
|---|---|---|
| 1 57 55 25 04 1 | 1 43 40 24 45 3 | 2 46 46 24 04 0 |
| 2 36 32 12 14 0 | 2 39 37 14 17 0 | 1 29 28 10 14 2 |
| 2 39 38 12 32 0 | 2 34 35 14 42 0 | 2 36 32 12 30 0 |
| 2 34 33 12 30 0 | 2 36 30 13 07 0 | 2 53 79 12 94 0 |
| 2 32 31 09 12 0 | 1 43 37 15 08 0 | 1 39 35 15 05 0 |
| 1 36 30 06 44 0 | 2 31 30 10 19 0 | 1 31 26 06 04 0 |
| 2 32 26 07 10 0 | 1 33 31 13 06 0 | 2 29 26 07 21 0 |
| 1 23 30 02 04 0 | 1 47 42 25 41 0 | 2 36 34 14 17 0 |
| 1 49 44 10 12 0 | 1 34 26 08 11 0 | 2 24 21 03 06 0 |
| 1 36 26 08 11 0 | 2 41 34 11 23 0 | 1 36 28 12 05 0 |
| 2 46 37 13 35 0 | 1 33 27 10 06 1 | 1 35 30 14 07 0 |
| 1 64 67 41 03 0 | 1 62 57 37 06 1 | 1 49 44 17 21 0 |
| 1 30 25 06 06 0 | 1 35 26 07 05 0 | 2 32 29 11 30 0 |
| 1 30 24 09 29 0 | 2 27 25 04 29 0 | 1 39 33 15 37 0 |
| 1 41 37 04 03 0 | 1 30 24 04 03 0 | 2 29 26 06 10 0 |
| 1 57 42 19 06 0 | 2 37 39 10 03 0 | 1 30 27 03 06 0 |
| 2 29 35 03 04 0 | 1 47 42 23 04 0 | 1 29 25 05 19 0 |
| 2 40 38 16 45 0 | 1 47 47 23 06 0 | 2 36 41 10 16 0 |
| 1 37 31 11 08 0 | 2 60 52 14 08 0 | 2 54 46 25 04 0 |
| 1 69 72 37 18 0 | 2 44 43 21 05 0 | 2 47 54 23 20 0 |
| 2 43 32 11 25 0 | 1 43 42 16 34 1 | 1 54 52 32 05 0 |
| 2 40 39 12 05 0 | 1 61 57 36 04 0 | 2 41 43 18 22 0 |
| 1 49 44 19 04 0 | 2 39 36 17 04 4 | 1 25 21 04 03 0 |
| 1 48 50 20 06 0 | 2 56 59 36 03 0 | 2 49 46 26 06 1 |
| 2 48 46 28 04 0 | 1 26 24 03 10 0 | 1 32 32 09 07 0 |
| 2 34 34 14 11 0 | 2 33 28 11 08 0 | 2 27 23 04 24 0 |
| 2 38 33 13 12 0 | 2 41 30 12 36 0 | 2 45 26 07 10 0 |
| 2 28 28 10 15 0 | 2 28 29 09 14 0 | 1 33 29 09 08 0 |
| 2 43 43 23 08 0 | 2 34 28 10 39 0 | 1 34 30 10 11 0 |
| 1 41 41 17 04 0 | 1 27 23 06 12 0 | 1 58 60 36 17 0 |
| 1 34 34 12 37 0 | 1 26 22 04 08 0 | 1 47 48 16 08 1 |
| 2 45 43 23 09 0 | 2 51 48 26 16 0 | 1 48 39 21 12 3 |
| 2 51 45 26 03 0 | 2 34 32 11 11 0 | 2 35 32 11 02 1 |
| 2 33 34 11 06 0 | 2 34 32 15 07 2 | 1 65 58 35 99 0 |
| 2 23 22 03 34 0 | 1 43 36 17 47 0 | 2 27 28 05 27 0 |
| 2 28 27 07 27 0 | 2 41 37 17 99 0 | 1 45 40 18 19 0 |
| 2 42 39 22 99 0 | 2 68 61 43 28 0 | 1 58 52 28 24 0 |
| 1 52 47 24 05 1 | 2 58 58 38 05 0 | 1 38 35 10 18 0 |
| 2 28 24 07 07 0 | 2 37 35 13 07 0 | 2 38 35 12 09 0 |
| 2 48 48 28 13 3 | 1 39 36 13 62 0 | 1 39 38 16 28 0 |
| 1 35 33 15 11 0 | 1 41 31 11 18 0 | 2 29 30 09 08 0 |
| 2 36 33 13 14 0 | 2 52 51 28 06 0 | 2 53 49 30 41 0 |
| 1 24 25 04 12 0 | 2 23 22 04 23 0 | 2 28 29 09 06 0 |

| | | |
|---|---|---|
| 2 32 29 07 04 0 | 1 33 28 09 04 0 | 1 32 35 04 29 0 |
| 2 36 36 17 16 0 | 1 30 32 05 07 0 | 2 50 48 27 12 0 |
| 1 30 33 03 11 0 | 2 39 37 10 12 0 | 2 37 30 12 03 0 |
| 1 54 57 29 07 0 | 1 41 40 21 16 0 | 2 33 39 13 09 0 |
| 2 45 43 22 59 0 | 1 44 45 22 20 0 | 2 44 41 18 45 0 |
| 2 27 24 05 30 0 | 2 39 35 03 20 0 | 2 51 46 25 08 0 |
| 1 34 33 12 14 0 | 2 46 46 25 11 0 | 1 41 33 17 04 0 |
| 1 39 39 16 04 0 | 2 38 32 13 04 0 | 2 45 44 19 07 0 |
| 2 33 34 12 22 0 | 1 36 34 12 09 0 | 1 30 29 06 04 0 |
| 2 38 44 26 24 0 | 1 49 47 26 15 0 | 2 33 28 11 35 0 |
| 1 37 34 17 52 0 | 2 45 33 08 22 0 | 2 48 36 17 18 0 |
| 2 33 34 13 07 0 | 2 33 31 13 11 0 | 2 32 27 09 15 0 |
| 2 26 25 06 07 0 | 2 24 23 03 11 0 | 1 34 31 12 09 0 |
| 1 52 54 31 02 0 | 1 47 48 29 02 0 | 2 37 36 17 08 0 |
| 2 29 29 09 14 0 | 1 30 31 07 11 0 | 1 38 37 19 06 0 |
| 1 56 56 37 07 0 | 2 36 35 08 08 0 | 2 35 28 11 19 1 |
| 2 36 34 14 03 1 | 2 59 49 16 69 0 | 2 45 43 23 51 0 |
| 2 40 34 14 12 0 | 1 30 30 05 04 0 | 2 38 40 10 07 0 |
| 2 33 24 06 26 0 | 1 33 32 04 09 0 | 1 25 24 04 13 0 |
| 1 34 30 12 09 0 | 2 40 40 20 99 2 | 2 63 63 42 02 0 |
| 2 40 41 11 23 0 | 2 54 38 17 04 6 | 2 56 47 26 04 0 |
| 1 39 46 21 70 3 | 1 46 47 23 50 0 | 1 28 31 09 05 0 |
| 1 49 46 26 11 2 | 2 36 36 10 15 0 | 2 49 42 21 59 0 |
| 1 42 33 12 12 0 | 2 62 53 32 09 4 | 2 63 55 36 02 0 |
| 1 49 50 28 02 1 | 1 55 54 33 04 1 | 2 31 35 10 63 0 |
| 1 40 32 03 07 0 | 2 36 36 16 26 0 | 2 28 29 10 12 0 |
| 1 53 49 26 11 0 | 1 40 43 14 18 0 | 1 28 25 07 06 0 |
| 1 31 27 08 05 0 | 2 49 36 12 17 0 | 2 23 20 02 16 0 |
| 2 43 29 11 06 0 | 2 35 32 12 36 0 | 2 62 56 35 57 0 |
| 2 40 39 18 64 0 | 1 45 48 22 08 7 | 1 42 35 04 31 0 |
| 2 44 30 04 20 0 | 1 38 39 14 45 0 | 2 24 48 04 23 0 |
| 1 29 32 05 48 0 | 2 32 31 10 28 0 | 2 32 30 08 25 0 |
| 2 34 35 11 57 0 | 1 46 35 14 26 0 | 2 35 29 13 40 0 |
| 1 41 41 11 33 0 | 1 37 33 14 09 0 | 2 32 30 08 16 0 |
| 1 64 61 35 32 0 | 2 34 33 15 11 0 | 1 48 47 18 15 0 |
| 2 33 33 10 12 0 | 2 38 31 14 64 0 | 2 30 29 06 08 0 |
| 2 31 31 13 22 0 | 2 29 28 10 07 0 | 1 50 52 28 05 0 |
| 1 63 60 33 04 0 | 2 32 27 09 16 0 | 1 41 40 19 10 0 |
| 1 44 45 24 05 1 | 1 60 60 38 02 0 | 1 36 37 14 12 0 |
| 2 54 62 24 05 0 | 2 48 46 15 02 0 | 2 53 48 26 05 3 |
| 2 35 33 16 12 0 | 1 55 51 29 09 1 | 2 25 20 01 14 0 |
| 1 66 65 43 09 0 | 2 32 26 04 17 0 | 1 50 51 28 21 0 |
| 1 35 35 08 14 0 | 2 34 31 05 22 0 | 2 26 24 02 07 0 |

2 26 26 04 25 0
2 28 27 06 14 0
1 64 62 34 03 0
1 29 22 04 06 0
1 37 31 10 14 0
2 31 29 07 07 0
2 26 24 07 13 0
2 38 39 16 05 0
1 54 54 32 02 0
2 27 27 08 12 0
2 38 39 15 08 2
1 39 39 18 10 0
1 45 44 17 03 2
2 48 38 18 36 0
1 35 44 10 04 0
1 28 23 04 33 0
1 75 62 24 21 0
1 51 56 24 03 0
2 82 80 12 05 0
2 32 26 08 15 0
2 38 35 15 24 0
2 36 31 15 79 0
2 28 27 07 20 0
1 38 38 16 64 0
2 51 40 06 10 0
2 27 26 09 15 0
2 35 35 13 59 0
2 38 37 14 08 0
2 38 34 09 08 0
1 69 62 28 99 0
1 51 47 23 04 0
1 44 32 06 03 0
2 56 66 24 02 0
2 53 52 35 03 0
2 38 36 13 11 0
2 37 32 13 39 0
2 35 35 11 06 0
2 34 29 11 19 0
2 61 58 38 99 0
2 37 36 17 52 0

1 29 27 10 12 0
1 46 40 18 06 0
2 23 21 04 26 0
1 36 36 11 21 0
1 70 67 22 04 0
1 25 23 05 08 0
2 54 51 29 94 0
2 30 27 07 09 0
2 29 25 08 05 0
2 28 26 04 13 0
1 37 37 17 11 0
2 54 58 35 08 0
2 30 29 12 13 0
1 72 74 48 05 1
2 52 50 29 11 0
2 36 38 13 25 0
1 52 53 28 03 0
1 54 65 17 03 0
2 34 32 13 09 0
2 53 62 16 09 0
2 29 29 06 05 0
2 45 43 09 27 0
1 75 70 26 17 0
2 29 25 07 44 0
2 30 27 02 11 0
2 34 35 13 09 0
2 37 33 09 23 0
1 35 28 08 03 0
1 76 73 48 04 0
1 43 38 22 12 0
2 54 52 31 04 0
1 46 44 23 03 1
2 23 33 03 17 0
2 39 36 19 06 4
1 30 28 10 76 0
1 29 28 01 08 0
2 33 32 13 10 0
1 62 57 32 06 0
2 38 39 16 78 0
2 33 32 11 11 0

2 34 33 15 43 0
2 63 60 42 08 0
2 63 55 37 10 0
2 48 50 25 01 0
2 34 30 11 08 0
1 33 31 11 08 0
2 30 28 10 23 0
1 64 64 43 06 0
1 37 32 14 06 0
2 32 31 11 10 0
2 29 32 05 16 0
1 50 48 31 03 0
2 35 32 14 99 0
1 53 52 33 03 0
2 51 45 28 14 0
1 30 28 09 15 0
1 41 42 19 02 2
1 51 51 30 07 0
2 30 30 08 10 0
2 39 35 20 09 0
1 45 40 23 17 0
1 27 22 03 10 0
2 48 35 14 33 0
2 27 26 02 14 0
2 55 56 34 07 0
2 34 32 12 16 0
2 34 31 13 23 0
1 31 20 02 12 0
2 43 37 16 15 1
2 53 63 17 08 0
2 33 34 12 07 0
2 37 33 12 03 1
1 34 27 11 07 0
1 23 19 02 07 0
2 46 37 18 13 0
2 39 40 19 06 0
2 35 33 15 07 0
1 48 30 02 18 0
2 38 29 11 22 0

## 10.2 Hospital Admissions Data

Data on hospital admission times. **DY** is the day of admission. **MT** is the month of admission. **YR** is the year of admission. **HR** is the hour of admission. **MI** is the minute of admission.

| DY | MT | YR | HR | MI | DY | MT | YR | HR | MI | DY | MT | YR | HR | MI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 63 | 11 | 00 | 4 | 2 | 63 | 17 | 00 | 8 | 2 | 63 | 23 | 15 |
| 11 | 2 | 63 | 10 | 00 | 16 | 2 | 63 | 12 | 00 | 18 | 2 | 63 | 8 | 45 |
| 18 | 2 | 63 | 16 | 00 | 20 | 2 | 63 | 10 | 00 | 20 | 2 | 63 | 15 | 30 |
| 21 | 2 | 63 | 20 | 20 | 25 | 2 | 63 | 4 | 00 | 25 | 2 | 63 | 12 | 00 |
| 28 | 2 | 63 | 2 | 20 | 1 | 3 | 63 | 12 | 00 | 3 | 3 | 63 | 5 | 30 |
| 7 | 3 | 63 | 7 | 30 | 7 | 3 | 63 | 12 | 00 | 9 | 3 | 63 | 16 | 00 |
| 15 | 3 | 63 | 16 | 00 | 16 | 3 | 63 | 1 | 30 | 17 | 3 | 63 | 11 | 05 |
| 20 | 3 | 63 | 16 | 00 | 22 | 3 | 63 | 19 | 00 | 24 | 3 | 63 | 17 | 45 |
| 24 | 3 | 63 | 20 | 20 | 24 | 3 | 63 | 21 | 00 | 28 | 3 | 63 | 12 | 00 |
| 28 | 3 | 63 | 12 | 00 | 30 | 3 | 63 | 18 | 00 | 2 | 4 | 63 | 22 | 00 |
| 2 | 4 | 63 | 22 | 00 | 6 | 4 | 63 | 22 | 05 | 9 | 4 | 63 | 12 | 45 |
| 9 | 4 | 63 | 19 | 30 | 10 | 4 | 63 | 18 | 45 | 11 | 4 | 63 | 16 | 15 |
| 15 | 4 | 63 | 16 | 00 | 16 | 4 | 63 | 20 | 30 | 23 | 4 | 63 | 23 | 40 |
| 28 | 4 | 63 | 20 | 20 | 29 | 4 | 63 | 18 | 45 | 4 | 5 | 63 | 16 | 30 |
| 6 | 5 | 63 | 22 | 00 | 7 | 5 | 63 | 8 | 45 | 11 | 5 | 63 | 19 | 15 |
| 13 | 5 | 63 | 15 | 30 | 14 | 5 | 63 | 12 | 00 | 14 | 5 | 63 | 18 | 15 |
| 16 | 5 | 63 | 14 | 00 | 18 | 5 | 63 | 13 | 00 | 19 | 5 | 63 | 23 | 00 |
| 20 | 5 | 63 | 19 | 15 | 22 | 5 | 63 | 22 | 00 | 23 | 5 | 63 | 10 | 15 |
| 23 | 5 | 63 | 12 | 30 | 24 | 5 | 63 | 18 | 15 | 25 | 5 | 63 | 21 | 05 |
| 28 | 5 | 63 | 21 | 00 | 30 | 5 | 63 | 0 | 30 | 1 | 6 | 63 | 1 | 45 |
| 1 | 6 | 63 | 12 | 20 | 3 | 6 | 63 | 14 | 45 | 5 | 6 | 63 | 22 | 30 |
| 10 | 6 | 63 | 12 | 30 | 10 | 6 | 63 | 13 | 15 | 12 | 6 | 63 | 17 | 30 |
| 13 | 6 | 63 | 11 | 20 | 13 | 6 | 63 | 17 | 30 | 16 | 6 | 63 | 23 | 00 |
| 18 | 6 | 63 | 10 | 55 | 18 | 6 | 63 | 13 | 30 | 21 | 6 | 63 | 11 | 00 |
| 21 | 6 | 63 | 18 | 30 | 22 | 6 | 63 | 11 | 05 | 24 | 6 | 63 | 4 | 00 |
| 24 | 6 | 63 | 7 | 30 | 25 | 6 | 63 | 20 | 00 | 25 | 6 | 63 | 21 | 30 |
| 26 | 6 | 63 | 6 | 30 | 27 | 6 | 63 | 17 | 30 | 29 | 6 | 63 | 20 | 45 |
| 30 | 6 | 63 | 22 | 00 | 2 | 7 | 63 | 20 | 15 | 2 | 7 | 63 | 21 | 00 |
| 8 | 7 | 63 | 17 | 30 | 9 | 7 | 63 | 19 | 50 | 10 | 7 | 63 | 2 | 00 |
| 12 | 7 | 63 | 1 | 45 | 13 | 7 | 63 | 3 | 40 | 13 | 7 | 63 | 4 | 15 |
| 13 | 7 | 63 | 23 | 55 | 20 | 7 | 63 | 3 | 15 | 21 | 7 | 63 | 19 | 00 |
| 23 | 7 | 63 | 21 | 45 | 24 | 7 | 63 | 21 | 30 | 27 | 7 | 63 | 0 | 45 |
| 27 | 7 | 63 | 2 | 30 | 29 | 7 | 63 | 15 | 30 | 1 | 8 | 63 | 21 | 00 |
| 2 | 8 | 63 | 8 | 45 | 3 | 8 | 63 | 14 | 30 | 3 | 8 | 63 | 17 | 00 |
| 7 | 8 | 63 | 3 | 30 | 7 | 8 | 63 | 15 | 45 | 7 | 8 | 63 | 17 | 30 |
| 11 | 8 | 63 | 14 | 00 | 13 | 8 | 63 | 2 | 00 | 13 | 8 | 63 | 11 | 30 |
| 13 | 8 | 63 | 17 | 30 | 19 | 8 | 63 | 17 | 10 | 21 | 8 | 63 | 21 | 20 |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | 8 | 63 | 3 | 00 | 31 | 8 | 63 | 13 | 30 | 2 | 9 | 63 | 23 | 00 |
| 5 | 9 | 63 | 20 | 20 | 7 | 9 | 63 | 23 | 15 | 8 | 9 | 63 | 20 | 00 |
| 10 | 9 | 63 | 16 | 00 | 10 | 9 | 63 | 18 | 30 | 11 | 9 | 63 | 21 | 00 |
| 13 | 9 | 63 | 21 | 10 | 15 | 9 | 63 | 17 | 00 | 16 | 9 | 63 | 13 | 25 |
| 18 | 9 | 63 | 15 | 05 | 21 | 9 | 63 | 14 | 10 | 23 | 9 | 63 | 19 | 15 |
| 24 | 9 | 63 | 14 | 05 | 24 | 9 | 63 | 22 | 40 | 27 | 9 | 63 | 9 | 30 |
| 28 | 9 | 63 | 17 | 30 | 1 | 10 | 63 | 12 | 30 | 2 | 10 | 63 | 17 | 30 |
| 3 | 10 | 63 | 14 | 30 | 3 | 10 | 63 | 16 | 00 | 6 | 10 | 63 | 14 | 10 |
| 8 | 10 | 63 | 14 | 00 | 12 | 10 | 63 | 15 | 30 | 13 | 10 | 63 | 4 | 30 |
| 19 | 10 | 63 | 11 | 50 | 20 | 10 | 63 | 11 | 55 | 20 | 10 | 63 | 15 | 20 |
| 20 | 10 | 63 | 15 | 40 | 22 | 10 | 63 | 11 | 15 | 23 | 10 | 63 | 2 | 15 |
| 26 | 10 | 63 | 11 | 15 | 30 | 10 | 63 | 21 | 30 | 31 | 10 | 63 | 3 | 00 |
| 1 | 11 | 63 | 0 | 40 | 1 | 11 | 63 | 10 | 00 | 4 | 11 | 63 | 9 | 45 |
| 4 | 11 | 63 | 23 | 45 | 5 | 11 | 63 | 10 | 00 | 6 | 11 | 63 | 7 | 50 |
| 7 | 11 | 63 | 13 | 30 | 8 | 11 | 63 | 12 | 30 | 9 | 11 | 63 | 13 | 45 |
| 9 | 11 | 63 | 19 | 30 | 11 | 11 | 63 | 0 | 15 | 12 | 11 | 63 | 7 | 45 |
| 15 | 11 | 63 | 15 | 20 | 15 | 11 | 63 | 18 | 40 | 15 | 11 | 63 | 19 | 50 |
| 16 | 11 | 63 | 23 | 55 | 17 | 11 | 63 | 1 | 45 | 18 | 11 | 63 | 10 | 50 |
| 19 | 11 | 63 | 7 | 50 | 22 | 11 | 63 | 15 | 30 | 23 | 11 | 63 | 18 | 00 |
| 23 | 11 | 63 | 23 | 05 | 24 | 11 | 63 | 19 | 30 | 26 | 11 | 63 | 19 | 00 |
| 27 | 11 | 63 | 16 | 10 | 29 | 11 | 63 | 10 | 00 | 30 | 11 | 63 | 2 | 30 |
| 30 | 11 | 63 | 22 | 00 | 1 | 12 | 63 | 21 | 50 | 2 | 12 | 63 | 19 | 10 |
| 3 | 12 | 63 | 11 | 45 | 3 | 12 | 63 | 15 | 45 | 3 | 12 | 63 | 16 | 30 |
| 3 | 12 | 63 | 18 | 30 | 5 | 12 | 63 | 10 | 05 | 5 | 12 | 63 | 20 | 00 |
| 7 | 12 | 63 | 13 | 35 | 7 | 12 | 63 | 16 | 45 | 8 | 12 | 63 | 2 | 15 |
| 9 | 12 | 63 | 20 | 30 | 11 | 12 | 63 | 14 | 00 | 12 | 12 | 63 | 21 | 15 |
| 13 | 12 | 63 | 18 | 45 | 14 | 12 | 63 | 14 | 05 | 14 | 12 | 63 | 14 | 15 |
| 15 | 12 | 63 | 1 | 15 | 16 | 12 | 63 | 1 | 45 | 17 | 12 | 63 | 18 | 00 |
| 20 | 12 | 63 | 14 | 15 | 20 | 12 | 63 | 15 | 15 | 21 | 12 | 63 | 16 | 15 |
| 22 | 12 | 63 | 10 | 20 | 23 | 12 | 63 | 13 | 35 | 23 | 12 | 63 | 17 | 15 |
| 24 | 12 | 63 | 19 | 50 | 24 | 12 | 63 | 22 | 45 | 25 | 12 | 63 | 7 | 25 |
| 25 | 12 | 63 | 17 | 00 | 28 | 12 | 63 | 12 | 30 | 31 | 12 | 63 | 23 | 15 |
| 2 | 1 | 64 | 10 | 30 | 3 | 1 | 64 | 13 | 45 | 5 | 1 | 64 | 2 | 30 |
| 6 | 1 | 64 | 12 | 00 | 7 | 1 | 64 | 15 | 45 | 7 | 1 | 64 | 17 | 00 |
| 7 | 1 | 64 | 17 | 00 | 10 | 1 | 64 | 1 | 30 | 10 | 1 | 64 | 20 | 15 |
| 11 | 1 | 64 | 12 | 30 | 12 | 1 | 64 | 15 | 40 | 14 | 1 | 64 | 3 | 30 |
| 14 | 1 | 64 | 18 | 35 | 15 | 1 | 64 | 13 | 30 | 17 | 1 | 64 | 16 | 40 |
| 19 | 1 | 64 | 18 | 00 | 20 | 1 | 64 | 20 | 00 | 21 | 1 | 64 | 11 | 15 |
| 24 | 1 | 64 | 16 | 40 | 25 | 1 | 64 | 13 | 55 | 29 | 1 | 64 | 21 | 00 |
| 30 | 1 | 64 | 7 | 45 | 31 | 1 | 64 | 22 | 30 | 5 | 2 | 64 | 16 | 40 |
| 5 | 2 | 64 | 23 | 10 | 6 | 2 | 64 | 19 | 15 | 7 | 2 | 64 | 11 | 00 |
| 11 | 2 | 64 | 0 | 15 | 11 | 2 | 64 | 14 | 40 | 12 | 2 | 64 | 15 | 45 |
| 17 | 2 | 64 | 12 | 45 | 18 | 2 | 64 | 17 | 00 | 18 | 2 | 64 | 18 | 00 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | 2 | 64 | 21 | 45 | 19 | 2 | 64 | 16 | 00 | 20 | 2 | 64 | 12 | 00 |
| 23 | 2 | 64 | 2 | 30 | 24 | 2 | 64 | 12 | 55 | 25 | 2 | 64 | 20 | 20 |
| 26 | 2 | 64 | 10 | 30 | 2 | 3 | 64 | 15 | 50 | 4 | 3 | 64 | 17 | 30 |
| 6 | 3 | 64 | 20 | 00 | 10 | 3 | 64 | 2 | 00 | 11 | 3 | 64 | 1 | 45 |
| 18 | 3 | 64 | 1 | 45 | 18 | 3 | 64 | 2 | 05 | | | | | |

## 10.3 Recidivism Data

Times, in months, to recidivism of all people released from prison in North Carolina in 1978 and 1980. **Year** is the year of release, **Cen** is the censoring indicator, **Fq** is the number of people who re-offended in this time period, and **Mn** is the number of months between release and re-offense.

| Year | Cen | Fq | Mn | Year | Cen | Fq | Mn | Year | Cen | Fq | Mn |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1978 | 1 | 1 | 0 | 1978 | 1 | 23 | 1 | 1978 | 1 | 29 | 2 |
| 1978 | 1 | 43 | 3 | 1978 | 1 | 48 | 4 | 1978 | 1 | 51 | 5 |
| 1978 | 1 | 59 | 6 | 1978 | 1 | 62 | 7 | 1978 | 1 | 57 | 8 |
| 1978 | 1 | 54 | 9 | 1978 | 1 | 61 | 10 | 1978 | 1 | 49 | 11 |
| 1978 | 1 | 55 | 12 | 1978 | 1 | 54 | 13 | 1978 | 1 | 40 | 14 |
| 1978 | 1 | 49 | 15 | 1978 | 1 | 49 | 16 | 1978 | 1 | 31 | 17 |
| 1978 | 1 | 41 | 18 | 1978 | 1 | 32 | 19 | 1978 | 1 | 29 | 20 |
| 1978 | 1 | 32 | 21 | 1978 | 1 | 26 | 22 | 1978 | 1 | 34 | 23 |
| 1978 | 1 | 33 | 24 | 1978 | 1 | 36 | 25 | 1978 | 1 | 30 | 26 |
| 1978 | 1 | 30 | 27 | 1978 | 1 | 27 | 28 | 1978 | 1 | 29 | 29 |
| 1978 | 1 | 20 | 30 | 1978 | 1 | 13 | 31 | 1978 | 1 | 14 | 32 |
| 1978 | 1 | 19 | 33 | 1978 | 1 | 18 | 34 | 1978 | 1 | 20 | 35 |
| 1978 | 1 | 18 | 36 | 1978 | 1 | 20 | 37 | 1978 | 1 | 13 | 38 |
| 1978 | 1 | 14 | 39 | 1978 | 1 | 11 | 40 | 1978 | 1 | 19 | 41 |
| 1978 | 1 | 11 | 42 | 1978 | 1 | 18 | 43 | 1978 | 1 | 12 | 44 |
| 1978 | 1 | 15 | 45 | 1978 | 1 | 18 | 46 | 1978 | 1 | 10 | 47 |
| 1978 | 1 | 13 | 48 | 1978 | 1 | 12 | 49 | 1978 | 1 | 16 | 50 |
| 1978 | 1 | 6 | 51 | 1978 | 1 | 10 | 52 | 1978 | 1 | 13 | 53 |
| 1978 | 1 | 9 | 54 | 1978 | 1 | 9 | 55 | 1978 | 1 | 11 | 56 |
| 1978 | 1 | 12 | 57 | 1978 | 1 | 7 | 58 | 1978 | 1 | 10 | 59 |
| 1978 | 1 | 10 | 60 | 1978 | 1 | 10 | 61 | 1978 | 1 | 10 | 62 |
| 1978 | 1 | 9 | 63 | 1978 | 1 | 7 | 64 | 1978 | 1 | 10 | 65 |
| 1978 | 1 | 10 | 66 | 1978 | 1 | 8 | 67 | 1978 | 1 | 8 | 68 |
| 1978 | 1 | 7 | 69 | 1978 | 1 | 6 | 70 | 1978 | 0 | 306 | 70 |
| 1978 | 1 | 4 | 71 | 1978 | 0 | 282 | 71 | 1978 | 1 | 4 | 72 |
| 1978 | 0 | 175 | 72 | 1978 | 1 | 4 | 73 | 1978 | 0 | 332 | 73 |
| 1978 | 1 | 7 | 74 | 1978 | 0 | 146 | 74 | 1978 | 1 | 1 | 75 |
| 1978 | 0 | 145 | 75 | 1978 | 0 | 327 | 76 | 1978 | 1 | 2 | 77 |
| 1978 | 0 | 198 | 77 | 1978 | 0 | 176 | 78 | 1978 | 0 | 199 | 79 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1978 | 0 | 238 | 80 | 1978 | 0 | 173 | 81 | 1980 | 1 | 7 | 0 |
| 1980 | 1 | 20 | 1 | 1980 | 1 | 52 | 2 | 1980 | 1 | 86 | 3 |
| 1980 | 1 | 91 | 4 | 1980 | 1 | 100 | 5 | 1980 | 1 | 95 | 6 |
| 1980 | 1 | 86 | 7 | 1980 | 1 | 91 | 8 | 1980 | 1 | 82 | 9 |
| 1980 | 1 | 88 | 10 | 1980 | 1 | 83 | 11 | 1980 | 1 | 77 | 12 |
| 1980 | 1 | 64 | 13 | 1980 | 1 | 64 | 14 | 1980 | 1 | 60 | 15 |
| 1980 | 1 | 54 | 16 | 1980 | 1 | 57 | 17 | 1980 | 1 | 52 | 18 |
| 1980 | 1 | 43 | 19 | 1980 | 1 | 53 | 20 | 1980 | 1 | 46 | 21 |
| 1980 | 1 | 24 | 22 | 1980 | 1 | 34 | 23 | 1980 | 1 | 30 | 24 |
| 1980 | 1 | 36 | 25 | 1980 | 1 | 37 | 26 | 1980 | 1 | 27 | 32 |
| 1980 | 1 | 37 | 27 | 1980 | 1 | 22 | 28 | 1980 | 1 | 29 | 29 |
| 1980 | 1 | 27 | 30 | 1980 | 1 | 28 | 31 | 1980 | 1 | 29 | 33 |
| 1980 | 1 | 33 | 34 | 1980 | 1 | 33 | 35 | 1980 | 1 | 28 | 36 |
| 1980 | 1 | 21 | 37 | 1980 | 1 | 19 | 38 | 1980 | 1 | 23 | 39 |
| 1980 | 1 | 17 | 40 | 1980 | 1 | 22 | 41 | 1980 | 1 | 13 | 42 |
| 1980 | 1 | 22 | 43 | 1980 | 1 | 16 | 44 | 1980 | 1 | 16 | 45 |
| 1980 | 1 | 12 | 46 | 1980 | 0 | 387 | 46 | 1980 | 1 | 10 | 47 |
| 1980 | 0 | 288 | 47 | 1980 | 1 | 6 | 48 | 1980 | 0 | 272 | 48 |
| 1980 | 1 | 8 | 49 | 1980 | 0 | 281 | 49 | 1980 | 1 | 7 | 50 |
| 1980 | 0 | 275 | 50 | 1980 | 1 | 6 | 51 | 1980 | 0 | 238 | 51 |
| 1980 | 1 | 3 | 52 | 1980 | 0 | 357 | 52 | 1980 | 1 | 2 | 53 |
| 1980 | 0 | 328 | 53 | 1980 | 1 | 5 | 54 | 1980 | 0 | 328 | 54 |
| 1980 | 0 | 272 | 55 | 1980 | 1 | 1 | 56 | 1980 | 0 | 297 | 56 |
| 1980 | 0 | 271 | 57 | | | | | | | | |

# Bibliography


[1] Abraham, E. and Ledolter, J. (1983) **Statistical Methods for Forecasting**. New York: Wiley.

[2] Agresti, A. (1990) **Categorical Data Analysis**. New York: Wiley.

[3] Aitkin, M., Anderson, D., Francis, B., and Hinde, J. (1990) **Statistical Modelling in GLIM**. New York: Oxford Science Publications

[4] Chatfield, C. (1989) **The Analysis of Time Series**. New York: Chapman & Hall.

[5] Davidson, R. and MacKinnon, J.G. (1993) **Estimation and Inference in Econometrics**. New York: Oxford University Press.

[6] Diggle, P.J. (1990) **Time Series: A Biostatistical Introduction**. Oxford: Oxford University Press.

[7] Dobson, A.J. (1990) **An Introduction to Generalized Linear Models**. New York: Chapman & Hall.

[8] Grimmet, G.R., and Stirzaker, D.R. (1983) **Probability and Random Processes**. New York: Oxford University Press.

[9] Kabfleisch, J.D., and Prentice, R.L. (1980) **The Statistical Analysis of Failure Time Data**. New York: Wiley.

[10] Lawless, J.F. (1982) **Statistical Modules and Methods for Lifetime Data**. New York: Wiley.

[11] Lindsey, J.K. (1974) "Comparison of probability distributions." **Journal of the Royal Statistical Society B** 36: 38-45

[12] Lindsey, J.K. (1989) **The Analysis of Categorical Data Using GLIM**. Berlin: Springer Verlag.

[13] Lindsey, J.K. & Mersch, G. (1992) "Fitting and comparing probability distributions with log linear models." **Computational Statistics and Data Analysis** 13: 373-384.

[14] Lindsey, J.K. (1992) **The Analysis of Stochastic Processes using GLIM**. Berlin: Springer-Verlag.

[15] McCullagh, P. and Nelder, J.A. (1983) **Generalized Linear Models**. London: Chapman & Hall.

[16] Montgomery, D. (1991) **Introduction to Statistical Quality Control: 2nd Edition**. New York: Wiley.

[17] Nichols, D.A. (1983) "Macroeconomic determinants of wage adjustments in white collar occupations." **Review of Economics and Statistics** 65:203-213.

[18] Pregibon, D. (1982) "Score Tests in GLIM." in Gilchrist, R. (1982, ed.) **GLIM82: Procedings of the International Conference on Generalized Linear Models**. New York: Springer-Verlag.

[19] Ramsey, J.B. (1969) "Tests for specification errors in classical linear least-squares regression analysis." **Journal of the Royal Statistical Society B** 31: 350-371

[20] Ross, S.M. (1993) **Introduction to Probability Models, Fifth Edition**. San Diego: Academic Press.

[21] Ryan, T. (1989) **Statistical Methods for Quality Improvement**: New York: Wiley.

[22] Weisberg, S. (1985) **Applied Linear Regression**. New York: Wiley.

# VITA

Surname: Ramsey Given Names: Paul Willard

Place of Birth: Duluth, Minnesota, United States of America

Educational Institutions Attended:

| | |
|---|---|
| University of Victoria | 1993 to 1996 |
| University of British Columbia | 1989 to 1993 |

Degrees Awarded:

B. Sc. University of British Columbia 1993

Honours and Awards:

Graduate Teaching Fellowship 1995-1996

Publications:

None.

## PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis (or dissertation) to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similary institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis/Dissertation:

**The Analysis of Stochastic Processes using Splus**

Author: [redacted]

Paul Willard Ramsey
April 29, 1996