Visualizing Categorical Data with R

Book Plan and Outline

Michael Friendly
York University

David Meyer
Wirtschaftsuniversität Wien

with contributions,
Achim Zeileis
Universität Insbruck

December 9, 2013

Overview and rationale for VCDR

Visualizing Categorical Data with R (VCDR) is the successor to my book *Visualizing Categorical Data* (2000) published by SAS Institute. In the interim, there has been much development in the analysis and visualization of categorical data, and the bulk of that has been implemented in R.

The 2000 edition of *Visualizing Categorical Data* (VCD) stemmed from the premise that, while graphical methods for quantitative data are well-developed in most statistical software and widely used in practice, corresponding graphical methods for categorical data—counts, frequencies and discrete variables—were still in relative infancy at that time.

VCD was designed to present an overview of the analysis of categorical data focused on the graphical methods designed for data exploration, model building, model diagnostics, etc., analogous to the graphical techniques that are now commonly used for quantitative data. That book, along with published journal articles (e.g., Friendly 1994, 1999; Emerson, Green, Schloerke, Crowley, Cook, Hofmann, and Wickham 2013) has been highly influential in statistical practice.

The special nature of discrete variables and frequency data vis-a-vis statistical graphics is now more widely accepted, and many of these methods (e.g., mosaic displays, fourfold plots, diagnostic plots for generalized linear models) have become, if not main stream, then at least more widely used in research and teaching. As well, VCD spurred the implementation of many of these methods in R (e.g., the **vcd** package), and there has been considerable growth in both statistical methods for the analysis of categorical data (e.g., generalized linear models, zero-inflation models, mixed models for hierarchical and longitudinal data with discrete outcomes), along with some new graphical methods for visualizing and interpreting the results (3D mosaic plots, effect plots, diagnostic plots, etc.)

Thus, the time is right for a thorough revision of the central organization and framework of VCD to a modern perspective, with a focus on R.

Features

- Provides an accessible introduction to the major methods of categorical data analysis for data exploration, statistical testing and statistical models.
- The emphasis throughout is on computing, visualizing, understanding and communicating the results of these analyses.
- As opposed to more theoretical books, the goal here is to help the reader to translate theory into practical application, by providing skills and software tools for carrying out these methods.
- Includes many examples using real data, often treated from several perspectives.
- Supported directly by R packages **vcd** and **vcdExtra**, along with numerous other R packages

- All materials will be available online; the design of the book will support simultaneous publication as an ebook, possibly with hyperlinks to references and related material in the text or elsewhere.
- As far as possible, each chapter will contain one or more lab exercises, which work through applications of some of the methods presented in that chapter. This will make the book more suitable for both self-study and classroom use.

Audience

This book assumes basic understanding of statistical concepts at least at an intermediate undergraduate level including regression and analysis of variance (for example, at the level of Neter, Wasserman, and Kutner (1990); Mendenhall and Sincich (2003)).

It is written to appeal to two audiences:

- Students and methodologists in the social and health sciences, epidemiology, economics, business and (bio)statistics
- Substantive researchers in various disciplines wanting to be able to apply these methods to their own data

It is also assumed that the reader has at least basic knowledge of the R language and environment, including interacting with the R console (RGui for Windows, R.app for Mac OS X) or other graphical user interface (e.g., RStudio), using R functions in packages, getting help for these from R, etc. One introductory chapter (Chapter 2) is devoted to covering those topics beyond such basic skills needed in the book.

The book will be written so that it can be used as a primary or secondary text in courses dealing with categorical data analysis at the upper undergraduate and graduate levels. For example, in Winter, 2015, I will teach a graduate course in the Quantitative Methods area in Psychology at York, where this book will serve as the main text. The most important markets would include most of the applied areas of statistics, but principally:

- Statistics: Biostatistics and Epidemiology
- Statistics: Statistics for the Social and Behavioral Sciences

Relation to other books

There are now quite a few modern texts covering categorical data analysis from varying perspectives. Among these, Agresti (2013), *Categorical Data Analysis* is probably among the most complete, but advanced treatments, and his earlier *An Introduction to Categorical Data Analysis* (Agresti 1996) remains an accessible introductory text at a lower level.

Powers & Xie 2008, *Statistical Methods for Categorical Data Analysis*, covers a wide variety of these topics and others (multilevel models for binary data, event history data) at an advanced, graduate level, with emphasis on social science data.

Simonoff (2003), *Analysing Categorical Data*, is somewhat similar, with a different range of topics and more geared to advanced students in statistics.

Christensen (1997), *Log-Linear Models and Logistic Regression*, is also a somewhat advanced-level book, with coverage largely restricted to the methods in the title.

Stokes, Davis, and Koch (2000), *Categorical Data Analysis with the SAS System*, covers most of the non-parametric and model building methods of analysis of categorical data, but the emphasis is almost entirely on presenting the underlying theory, using SAS software to perform the analysis, and on interpreting the results from the numerical output. There are only a handful of graphs in the entire book.

None of these books feature graphical methods for categorical data; in fact, most of these show very few data graphs. A few of these contain a brief appendix mentioning software, or have a related web site with some data sets and software examples. Moreover, none actually describe how to do these analyses and graphics with R.

Yet, for categorical data, just as for quantitative data, there are many aspects of the relationships among variables, the adequacy of a fitted model, and possibly unusual features of the data which can best (or in some cases, only) be seen and appreciated from an informative graphical display.

Thus, there is a clear need to present these modern methods for the graphical display and analysis of categorical data. The existing books present the opportunity to describe and illustrate the graphical approach without the necessity to discuss as much of the underlying theory as would be required otherwise, and VCDR will be written to complement, rather than compete with, them. So, where sufficient theoretical background already exists, I will present short summaries and point readers to the other sources. Material that is unique to VCDR (e.g., correspondence analysis, GLIMs, GEE, CART models, etc.) will be developed more fully. This strategy will also help keep the book more manageable in both size and writing time.

$\TeX$nical issues and production details

The book will be written using the **knitr** package and other tools for writing, reporting and reproducible research in R. This allows the writing to mix $\LaTeX$ for the text, equations, index entries, etc. with R code that generates tables and graphs for examples, guaranteeing that all of the examples, tables and graphs in the book are reproducible and up-to-date. It also makes it relatively easy to selectively export the R code and data for examples and exercises to a form that readers can download and work with on their own, and produce alternative (e.g., ebook) versions.

Color: Another important consideration is the use of color in the book. By its nature, the book will include many graphs, and I plan to use color liberally, particularly where it is essential for communication of the ideas, methods, and understanding of the techniques described and illustrated. Ideally, an all-color book would be best, but cost/price considerations might lead to some compromise.

Pages: As a rough guess, I expect that the book would come to approximately 400–450 printed pages.

Preferred format: In terms of format, structure and integration of R content, one book (on a different topic) that is similar to what I have in mind is James, Witten, Hastie, and Tibshirani (2013), *An Introduction to Statistical Learning with Applications in R*. This book, in the Springer Texts in Statistics series, is published in **full color** (even in the text), and sells for \$80 (hardcover), \$60 (ebook), with $\sim$ 20% discounts on Amazon.

Another related book using R that I admire in terms of layout and rich use of color graphics is Gower, Lubbe, and Roux (2011), *Understanding Biplots*. Unfortunately, too much of the content of this book is devoted to documentation of the methods in their **UBbipl** package, that should have been relegated to the package itself

Outline

The provisional outline below is based on the structure of topics in VCD, updated with a new introductory chapter (Chapter 2) and three new substantive topics in Chapters, 10–12. For each chapter, I give an overview of the content and a list of sections; some of the details given here will certainly change as writing progresses.

1 Chapter 1: Introduction

"Categorical data" means different things in different contexts. I introduce the topic with some examples illustrating (a) types of categorical variables: binary, nominal, and ordinal, and (b) the main types of categorical data: counted data and frequency data.

Methods for the analysis of categorical data also fall into two quite different categories, described and illustrated next: the simple non-parametric, and randomization-based methods typified by the classical Pearson χ^2, Fisher's exact test, and Mantel-Haenszel tests, and the model-based methods represented by logistic regression and generalized linear models. Chapters 3–6 are mostly related to the non-parametric methods, Chapters 7–12 to the model-based methods.

Finally, I describe some important differences between categorical data and quantitative data, discuss the implications of these differences for visualization techniques, and outline a strategy of data analysis focussed on visualization.

1.1. Data visualization and categorical data
1.2. What is categorical data?
1.3. Strategies for categorical data analysis
1.4. Graphical methods for categorical data
1.5. Visualization = Graphing + Fitting + Graphing

2 Chapter 2: Working with categorical data

Categorical data can be represented in various forms: case form, frequency form, and table form. This chapter describes and illustrates the skills and techniques in R needed to input, create and manipulate R data objects to represent categorical data, and convert these from one form to another for the purposes of statistical analysis and visualization which are the subject of the remainder of the book.

2.1. Forms of categorical data: case form, frequency form and table form
2.2. Ordered factors and reordered tables
2.3. Generating tables with `table()` and `xtabs()`
2.4. Printing tables with `structable()` and `ftable()`
2.5. Collapsing over table factors: `aggregate()`, `margin.table()` and `apply()`
2.6. Converting among frequency tables and data frames
2.7. A complex example
2.8. Lab exercises

3 Chapter 3: Fitting and graphing discrete distributions

Discrete frequency distributions often involve counts of occurrences, such as accident fatalities, words in passages of text, or blood cells with some characteristic. Often interest is focussed on how closely such data follow a particular probabiliy distribution, such as the Poisson, binomial, or geometric distribution. Understanding and visualizing such distributions in the simplest case of an unstructured sample provides a building block for generalized linear models where they serve as one component.

This chapter describes the well-known discrete frequency distributions: the binomial, Poisson, negative binomial, geometric, and logarithmic series distributions in the simplest case of an unstructured sample. The chapter begins with simple graphical displays (line graphs and histograms) to view the distributions of empirical data and theoretical frequencies from a specified discrete distribution.

It then describes methods for fitting data to a distribution of a given form and simple, effective graphical methods than can be used used to visualize goodness of fit, to diagnose an appropriate model (e.g., does a given data set follow the Poisson or negative binomial?) and determine the impact of individual observations on estimated parameters.

4 Chapter 4: Two-way contingency tables

This chapter begins with an overview of statistical tests for association in two-way frequency tables and extensions of these tests for the case of multi-way tables, where two primary variables are stratified by one or more others.

Several schemes for representing contingency tables graphically are based on the fact that when the row and column variables are independent, the estimated expected frequencies, e_{ij}, are products of the row and column totals (divided by the grand total). Then, each cell can be represented by a rectangle whose area shows the cell frequency, f_{ij}, or deviation from independence.

This chapter describes a number of relatively simple visualization techniques based on this relation (Sieve diagram, Association plot), and several more specialized techniques for particular data structures.

5 Chapter 5: Mosaic displays for n-way tables

When there are more than two classification variables, the visualization of categorical data becomes increasingly difficult. This chapter extends the use of the fourfold display to a collection of 2×2 tables, and introduces the mosaic display.

The mosaic display, proposed by Hartigan & Kleiner 1981 and extended by Friendly (1994, 1999), represents the counts in a contingency table directly by tiles whose size is proportional to the cell frequency. One important design goal is that this display should apply extend naturally to three-way and higher-way tables. Another design feature is to serve both exploratory goals (by showing the pattern of observed frequencies in the full table), and model building goals (by displaying the residuals from a given log-linear model).

The use of the mosaic display in connection with loglinear models is introduced here and extended in Chapter 7.

6 Chapter 6: Correspondence analysis

Correspondence analysis is an exploratory technique related to to principal components analysis which finds a multidimensional representation of the association between the row and column categories of a two-way contingency table.

This chapter illustrates the use of correspondence analysis in understanding the nature of association in two-way tables, and describes how informative plots can be produced from the results of the **ca** package. Extensions of correspondence analysis to multi-way tables and related biplot methods are then described and illustrated.

7 Chapter 7: Loglinear and logit models

Loglinear models provide a comprehensive scheme to describe and understand the associations among two or more categorical variables, particularly when no one variable is singled out as a response to be predicted from the remaining explanatory variables.

For larger tables (three or more variables), however, it becomes difficult to interpret the nature of these associations from tables of parameter estimates. I first illustrate how results from such models may be more easily understood from plots of predicted log odds and probabilities.

The chapter then shows how mosaic displays and correspondence analysis plots can be used to complement the description provided by loglinear models, and how to construct diagnostic plots to determine if a few cells are having undue influence on the overall model.

A collection of mosaic plots, stratified by one (or more) variable(s) is used to display the partial associations among the remaining variables. A mosaic scatterplot matrix is introduced, showing all pairwise associations among variables.

8 Chapter 8: Logistic regression

Logistic regression describes the relationship between a dichotomous response variable and a set of explanatory variables. The explanatory variables may be continuous or (with dummy variables) discrete.

This chapter describes the general logistic regression model and illustrates how the analysis of dichotomous (and polytomous) response data can be enhanced by graphical display.

For interpreting and understanding the results of a fitted model, I emphasize plotting predicted probabilities and predicted log odds. For model criticism and diagnosis, I introduce some discrete analogs of the influence and other plots useful in ordinary least squares regression.

8.1. Introduction
8.2. The logistic regression model
8.3. Models for quantitative predictors
8.4. Logit models for qualitative predictors
8.5. Multiple logistic regression models
8.6. Influence and diagnostic plots
8.7. Polytomous response models
8.8. The Bradley-Terry-Luce Model for Paired Comparisons
8.9. Power and sample size for logistic regression
8.10. Chapter summary
8.11. Further reading
8.12. Lab exercises

9 Chapter 9: Generalized linear models

Generalized linear models extend the familiar linear models of regression and ANOVA to include counted data, frequencies, and other data for which the assumptions of independent, normal errors are not reasonable. I rely on the analogies between ordinary and generalized linear models (GLIMs) to develop visualization methods to display the fitted relations and check model assumptions.

9.1. Varieties of GLIMs
9.2. GLIMs for binary data
9.3. GLIMs for count data
9.4. Diagnostic plots for model checking
9.5. Chapter summary
9.6. Further reading
9.7. Lab exercises

10 Chapter 10: Regression models for count data

An important special case of GLMs occurs when the response variable is a frequency or count of some event. Some examples are: the number of physician office visits by medical patients, number of deaths from an infectious disease, number of insects found on plants in an agricultural experiment.

The simplest, classical case is that of Poisson regression, where the conditional distribution of the response given the explanatory variables is Poisson, but this model is often overly restrictive. Negative binomial regression can be used for over-dispersed count data, that is, when the conditional variance exceeds the conditional mean. Zero-inflated models attempt to account for situations in which there is an excess of zero counts, by positing an additional sub-model to deal with the excess zeros.

10.1. Poisson regression models

10.2. Negative binomial models
10.3. Zero-inflated models
10.4. Zero-truncated models
10.5. Chapter summary
10.6. Further reading
10.7. Lab exercises

11 Chapter 11: Repeated measures and Longitudinal data

This chapter develops several somewhat different forms of analysis and graphical display related to repeated measures and longitudinal data with categorical responses.

Model-based methods are most easily visualized by plotting predicted probabilities. Sequential analysis pertains to the sequences of behavioral events (e.g., statements, actions of children and parents) observed over time and classified into categories of a classification scheme. Generalized Estimating Equations (GEE) provide one approach to extending GLIMs to longitudinal observations.

11.1. Analysis of marginal probabilities
11.2. Multiple populations
11.3. Sequential analysis
11.4. GEE models
11.5. Chapter summary
11.6. Further reading
11.7. Lab exercises

12 Chapter 12: Classification and regression trees

Recursive partitioning methods provide an alternative to (generalized) linear models for categorical responses, particularly when there are numerous potential predictors and/or there are important interactions among predictors. These methods attempt to define a set of rules to classify observations into mutually exclusive subsets based on combinations of the explanatory variables, and tend to work well when there are important non-linearities or interactions in the data.

12.1. Introduction to recursive partitioning methods
12.2. Recursive partitioning trees
12.3. Conditional inference for classification trees
12.4. Chapter summary
12.5. Further reading
12.6. Lab exercises

Appendix A: Other material

My intention is that all of the datasets, and R functions from the book will be contained in publicly available R packages, where they will be fully documented, with some examples. Nevertheless, the reader of the book will find it useful to have some of this information and other material summarized or listed in print form, in ways that will enhance reading and use of the book, and allow easy reference from the chapters where they are used. Some of this material might better appear in specialized indexes, e.g., a dataset index, an R index, etc.

References

Agresti, A. (1996). *An Introduction to Categorical Data Analysis*. New York: Wiley, 2nd edn.

Agresti, A. (2013). *Categorical Data Analysis*. Wiley Series in Probability and Statistics. New York: Wiley-Interscience [John Wiley & Sons], 3rd edn.

Christensen, R. (1997). *Log-Linear Models and Logistic Regression*. New York, NY: Springer, 2nd edn.

Emerson, J. W., Green, W. A., Schloerke, B., Crowley, J., Cook, D., Hofmann, H., and Wickham, H. (2013). The generalized pairs plot. *Journal of Computational and Graphical Statistics*, 22(1), 79–91.

Friendly, M. (1994). Mosaic displays for multi-way contingency tables. *Journal of the American Statistical Association*, 89, 190–200.

Friendly, M. (1999). Extending mosaic displays: Marginal, conditional, and partial views of categorical data. *Journal of Computational and Graphical Statistics*, 8(3), 373–395.

Gower, J., Lubbe, S., and Roux, N. (2011). *Understanding Biplots*. Wiley.

Hartigan, J. A. and Kleiner, B. (1981). Mosaics for contingency tables. In W. F. Eddy, ed., *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface*, (pp. 268–273). New York, NY: Springer-Verlag.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. New York: Springer.

Mendenhall, W. and Sincich, T. (2003). *A Second Course in Statistics: Regression Analysis*. Prentice Hall / Pearson Education.

Neter, J., Wasserman, W., and Kutner, M. H. (1990). *Applied Linear Statistical Models : Regression, Analysis of Variance, and Experimental Designs*. Homewood, IL: R. D. Irwin, Inc., 3rd edn.

Powers, D. A. and Xie, Y. (2008). *Statistical Methods for Categorical Data Analysis*. Bingley, UK: Emerald, 2nd edn.

Simonoff, J. S. (2003). *Analyzing Categorical Data*. Springer Texts in Statistics. New York: Springer.

Stokes, M. E., Davis, C. S., and Koch, G. G. (2000). *Categorical Data Analysis Using the SAS System*. Cary, NC: SAS Institute, 2nd edn.

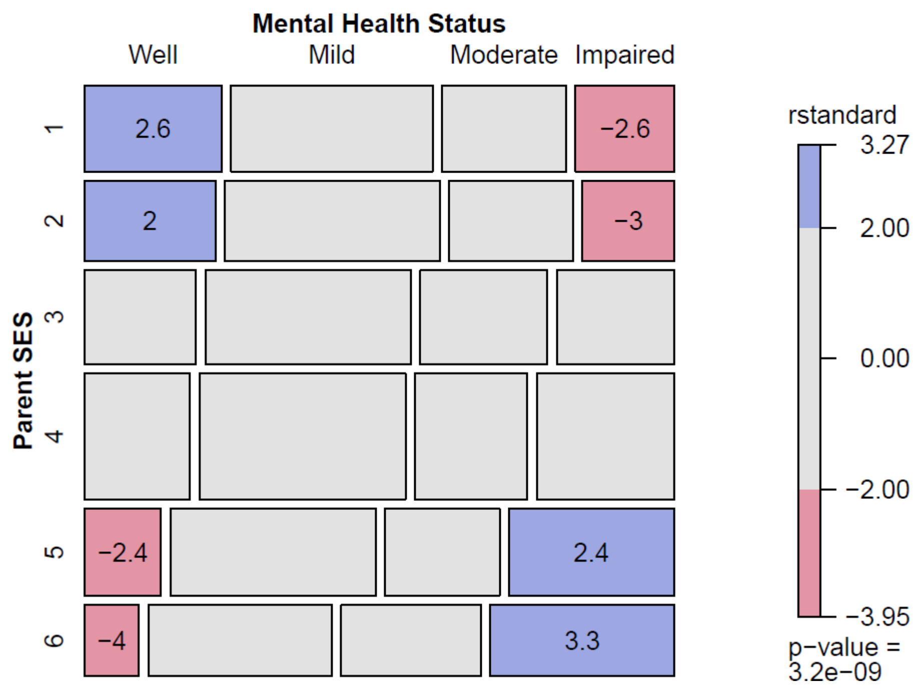
Visualizing Categorical Data with R

Michael Friendly
York University

David Meyer
Wirtschaftsuniversität Wien

with contributions,
Achim Zeileis
Universität Insbruck

December 18, 2013



Contents

Chapter 1
Introduction

Categorical data consists of variables whose values comprise a set of discrete categories. Such data require different statistical and graphical methods than commonly used for quantitative data. The focus of this book is on visualization techniques and graphical methods designed to reveal patterns of relationships among categorical variables. This chapter outlines the basic orientation of the book and some key distinctions regarding the analysis and visualization of categorical data.

1.1 Data visualization and categorical data: Overview

Beauty is truth; truth, beauty.
That is all ye know on Earth, all ye need to know.

John Keats, *Ode on a Grecian urn*

"Data visualization" can mean many things, from popular press infographics, to maps of voter turnout or party choice. Here we use this term in the narrower context of statistical analysis. As such, we refer to an approach to data analysis that focuses on *insightful* graphical display in the service of both *understanding* our data and *communicating* our results to others.

We may display the raw data, some summary statistics, or some indicators of the quality or adequacy of a fitted model. The word "insightful" suggests that the goal is (hopefully) to reveal some aspects of the data which might not be perceived, appreciated, or absorbed by other means. As in the quote from Keats, the overall aims include both beauty and truth, though each of these are only as perceived by the beholder.

Methods for visualizing quantitative data have a long history and are now widely used in both data analysis and in data presentation, and in both popular and scientific media. Graphical methods for categorical data, however, have only a more recent history, and are consequently not as widely used. The goal of this book is to show concretely how data visualization may be usefully applied to categorical data.

"Categorical" data means different things in different contexts. We introduce the topic in Section 1.2 with some examples illustrating (a) types of categorical variables: binary, nominal, and ordinal, (b) data in case form vs. frequency form, (c) frequency data vs. count data, (d) univariate, bivariate, and multivariate data, and (e) the distinction between explanatory and response variables.

Statistical methods for the analysis of categorical data also fall into two quite different categories, described and illustrated in Section 1.3: (a) the simple randomization-based methods typified by the classical Pearson χ^2, Fisher's exact test, and Cochran-Mantel-Haenszel tests, and (b) the model-based methods represented by logistic regression, loglinear, and generalized linear models. In this book, Chapters 3–**??** are mostly related to the randomization-based methods; Chapters **??**–**??** illustrate the model-based methods.

In Section 1.4 we describe some important similarities and differences between categorical data and quantitative data, and discuss the implications of these differences for visualization techniques. Section 1.5 outlines a strategy of data analysis focused on visualization.

In a few cases we show R code or results as illustrations here, but the fuller discussion of using R for categorical data analysis is postponed to Chapter 2.

1.2 What is categorical data?

A ***categorical variable*** is one for which the possible measured or assigned values consist of a discrete set of categories, which may be *ordered* or *unordered*. Some typical examples are:

- `Gender`, with categories "Male", "Female".
- `Marital status`, with categories "Never married", "Married", "Separated", "Divorced", "Widowed".
- `Fielding position` (in baseball), with categories "Pitcher", "Catcher", "1st base", "2nd base", . . ., "Left field".
- `Side effects` (in a pharmacological study), with categories "None", "Skin rash", "Sleep disorder", "Anxiety", ...
- `Political attitude`, with categories "Left", "Center", "Right".
- `Party preference` (in Canada), with categories "NDP", "Liberal", "Conservative", "Green".
- `Treatment outcome`, with categories "no improvement", "some improvement", or "marked improvement".
- `Age`, with categories "0-9", "10-19", "20-29", "30-39", ...
- `Number of children`, with categories $0, 1, 2, \ldots$.

As these examples suggest, categorical variables differ in the number of categories: we often distingish ***binary variables*** such as `Gender` from those with more than two categories (called ***polytomous variables***). For example, Table 1.1 gives data on 4526 applicants to graduate departments at the University of California at Berkeley in 1973, classified by two binary variables, gender and admission status.

Table 1.1: Admissions to Berkeley graduate programs

	Admitted	Rejected	Total
Males	1198	1493	2691
Females	557	1278	1835
Total	1755	2771	4526

Some categorical variables (`Political attitude`, `Treatment outcome`) may have ordered categories (and are called ***ordinal***), while other (***nominal***) variables like `Marital` `status` have unordered categories.[1] For example, Table 1.2 shows a $2 \times 2 \times 3$ table of ordered outcomes ("none", "some" or "marked" improvement) to an active treatment for rheumatoid arthritis compared to a placebo for men and women.

Table 1.2: Arthritis treatment data

		Improvement			
Treatment	Sex	None	Some	Marked	Total
Active	Female	6	5	16	27
	Male	7	2	5	14
Placebo	Female	19	7	6	32
	Male	10	0	1	11
Total		42	14	28	84

Finally, such variables differ in the fineness or level to which some underlying observation has been categorized for a particular purpose. From one point of view, *all* data may be considered categorical because the precision of measurement is necessarily finite, or an inherently continuous variable may be recorded only to limited precision.

But this view is not helpful for the applied researcher because it neglects the phrase "for a particular purpose". Age, for example, might be treated as a quantitative variable in a study of native language vocabulary, or as an ordered categorical variable with decade groups (0-10, 11-20, 20-30, ...) in terms of the efficacy or side-effects of treatment for depression, or even as a binary variable ("child" vs. "adult") in an analysis of survival following an epidemic or natural disaster. In the analysis of data using categorical methods, continuous variables are often recoded into ordered categories with a small set of categories for some purpose.[2]

1.2.1 Case form vs. frequency form

In many circumstances, data is recorded on each individual or experimental unit. Data in this form is called case data, or data in ***case form***. The data in Table 1.2, for example, were derived from the individual data listed in the data set `Arthritis` from the vcd package. The following lines show the first five of $N = 84$ cases in the `Arthritis` data,

```
data("Arthritis", package="vcd")
head(Arthritis, 5)

##   ID Treatment  Sex Age Improved
## 1 57   Treated Male  27     Some
## 2 46   Treated Male  29     None
## 3 77   Treated Male  30     None
## 4 17   Treated Male  32   Marked
## 5 36   Treated Male  46   Marked
```

[1]An ordinal variable may be defined as one whose categories are *unambiguously* ordered along a *single* underlying dimension. Both marital status and fielding position may be weakly ordered, but not on a single dimension, and not unambiguously.

[2]This may be wasteful of information available in the original variable, and should be done for substantitive reasons, not meer convenience. For example, some researchers unfamiliar with regression methods often perform a "median-split" on quantitative predictors so they can use ANOVA methods. Doing this precludes the possibility of determining if those variables have non-linear relations with the outcome.

Whether or not the data variables, and the questions we ask, call for categorical or quantitative data analysis, when the data are in case form, we can always trace any observation back to its individual identifier or data record (for example, if the case with `ID==57` turns out to be unusual or noteworthy).

Data in ***frequency form*** has already been tabulated, by counting over the categories of the table variables. The same data shown as a table in Table 1.2 appear in frequency form as shown below.

```
as.data.frame(xtabs(~Treatment+Sex+Improved, data=Arthritis))

##    Treatment    Sex Improved Freq
## 1    Placebo Female     None   19
## 2    Treated Female     None    6
## 3    Placebo   Male     None   10
## 4    Treated   Male     None    7
## 5    Placebo Female     Some    7
## 6    Treated Female     Some    5
## 7    Placebo   Male     Some    0
## 8    Treated   Male     Some    2
## 9    Placebo Female   Marked    6
## 10   Treated Female   Marked   16
## 11   Placebo   Male   Marked    1
## 12   Treated   Male   Marked    5
```

Data in frequency form may be analyzed by methods for quantitative data if there is a quantitative response variable (weighting each group by the cell frequency, with a `weight` variable). Otherwise, such data are generally best analyzed by methods for categorical data, where statistical models are often expressed as models for the frequency variable, in the form of an R formula like `Freq ~ .`.

In any case, an observation in a data set in frequency form refers to all cases in the cell collectively, and these cannot be identified individually. Data in case form can always be reduced to frequency form, but the reverse is rarely possible. In Chapter 2, we identify a third format, ***table form***, which is the R representation of a table like Table 1.2.

1.2.2 Frequency data vs. count data

In many cases the observations represent the classifications of events or variables are recorded from *operationally independent* experimental units or individuals, typically a sample from some population. The tabulated data may be called ***frequency data***. The data in Table 1.1 and Table 1.2 are both examples of frequency data because each observation tabulated comes from a different person.

However, if several events or variables are observed for the same units or individuals, those events are not operationally independent, and it is useful to use the term ***count data*** in this situation. These terms (following Lindsey (1995)) are by no means standard, but the distinction is often important, particularly in statistical models for categorical data.

For example, in a tabulation of the number of male children within families (Table 1.3, described in Section 1.2.3 below), the number of male children in a given family would be a *count* variable, taking values $0, 1, 2, \ldots$. The number of independent families with a given number of male children is a *frequency* variable. Count data also arise when we tabulate a sequence of events over time or under different circumstances in a number of individuals.

Table 1.3: Number of Males in 6115 Saxony Families of Size 12

Males	0	1	2	3	4	5	6	7	8	9	10	11	12
Families	3	24	104	286	670	1033	1343	1112	829	478	181	45	7

1.2.3 Univariate, bivariate, and multivariate data

Another distinction concerns the number of variables: one, two or (potentially) many shown in a data set or table, or used in some analysis. Table 1.1 is an example of a bivariate (two-way) contingency table and Table 1.2 classifies the observations by three variables. Yet, we will see later that the Berkeley admisssions data also recorded the department to which potential students applied (giving a three-way table), and in the arthritis data, the age of subjects was also recorded.

Any contingency table (in frequency or table form) therefore records the *marginal totals*, summed over all variables not represented in the table. For data in case form, this means simply ignoring (or not recording) one or more variables; the "observations" remain the same. Data in frequency form, however, result in smaller tables when any variable is ignored; the "observations" are the cells of the contingency table. For example, in the `Arthritis` data, ignoring *`Sex`* gives the smaller 2×3 table for *`Treatment`* and *`Improved`*.

```
as.data.frame(xtabs(~Treatment + Improved, data=Arthritis))

##   Treatment Improved Freq
## 1   Placebo     None   29
## 2   Treated     None   13
## 3   Placebo     Some    7
## 4   Treated     Some    7
## 5   Placebo   Marked    7
## 6   Treated   Marked   21
```

In the limiting case, only one table variable may be recorded or available, giving the categorical equivalent of univariate data. For example, Table 1.3 gives data on the distribution of the number of male children in families with 12 children (discussed further in Example 3.2). These data were part of a large tabulation of the sex distribution of families in Saxony in the 19th century, but the data in Table 1.3 have only one discrete classification variable, number of males. Without further information, the only statistical questions concern the form of the distribution. We discuss methods for fitting and graphing such discrete distributions in Chapter 3. The remaining chapters relate to bivariate and multivariate data.

1.2.4 Explanatory vs. Response variables

Most statistical models make a distinction between ***response variables*** (or *dependent*, or *criterion* variables) and ***explanatory variables*** (or *independent*, or *predictor* variables).

In the standard (classical) linear models for regression and analysis of variance (ANOVA), for instance, we treat one (or more) variables as responses, to be explained by the other, explanatory variables. The explanatory variables may be quantitative or categorical (e.g., factors in R). This affects only the details of how the model is specified or how coefficients are interpreted for `lm()` or `glm()`. In these classical models, the response variable ("treatment outcome", for example), must be considered quantitative, and the model attempts to describe how the *mean* of the distribution of responses changes with the values or levels of the explanatory variables, such as age or gender.

However, when the response variable is categorical, however, the standard linear models do not apply, because they assume a normal (Gaussian) distribution for the model residuals. For example, in Table 1.2 the response variable is `Improvement`, and even if numerical scores were assigned to the categories "none", "some", "marked", it may be unlikely that the assumptions of the classical linear models could be met.

Hence, a categorical *response* variable generally requires analysis using methods for categorical data, but categorical *explanatory* variables may be readily handled by either method.

The distinction between response and explanatory variables also becomes important in the use of loglinear models for frequency tables (described in Chapter **??**), where models can be specified in a simpler way (as equivalent logit models) by focusing on the response variable.

1.3 Strategies for categorical data analysis

Methods of analysis for categorical data can be classified into two broad categories: those concerned with hypothesis testing *per se*, and those concerned with model building.

1.3.1 Hypothesis testing approaches

In many studies, the questions of substantive interest translate readily into questions concerning hypotheses about ***association*** between variables, a more general idea than that of correlation (*linear* association) for quantitative variables. If a non-zero association exists, we may wish to characterize the strength of the association numerically and understand the pattern or nature of the association.

For example, in Table 1.1, a main question is: "Is there evidence of gender-bias in admission to graduate school?" Another way to frame this: "Are males more likely to be admitted?" These questions can be expressed in terms of an association between gender and admission status in a 2×2 contingency table of applicants classified by these two variables. If there is evidence for an association, we can assess its strength by a variety of measures, including the difference in proportions admitted for men and women or the ratio of the odds of admission for men compared to women, as described in Section **??**.

Similarly, in Table 1.2, questions about the efficacy of the treatment for rheumatoid arthritis can be answered in terms of hypotheses about the associations among the table variables: `Treatment`, `Sex`, and the `Improvement` categories. Although the main concern might be focused on the overall association between Treatment and Improvement, one would also wish to know if this association is the same for men and women. A ***stratified analysis*** (Section 4.3) controls for the effects of background variables like Sex, and tests for ***homogeneity of association*** help determine if these associations are equal.

Questions involving tests of such hypotheses are answered most easily using a large variety of specific statistical tests, often based on randomization arguments. These include the familiar Pearson chi-square test for two-way tables, the Cochran-Mantel-Haenszel test statistics, Fisher's exact test, and a wide range of measures of strength of association. These tests make minimal assumptions, principally requiring that subjects or experimental units have been randomly assigned to the categories of experimental factors. The hypothesis testing approach is illustrated in

Chapter 4–**??**, though the emphasis is on graphical methods which help to understand the nature of association between variables.

EXAMPLE 1.1: Hair color and eye color

The data `HairEye` below records data on the the relationship between hair color and eye color in a sample of nearly 600 students.

```
library(vcd)
(HairEye <- margin.table(HairEyeColor, c(1, 2)))

##         Eye
## Hair    Brown Blue Hazel Green
##   Black    68   20    15     5
##   Brown   119   84    54    29
##   Red      26   17    14    14
##   Blond     7   94    10    16
```

The standard analysis (with `chisq.test()` or `assocstats()`) gives a Pearson χ^2 of 138.3 with nine degrees of freedom, indicating substantial departure from independence. Among the measures of strength of association, the ***phi coefficient***, $\phi = \sqrt{\chi^2/N} = 0.483$, indicates a substantial relationship between hair and eye color.

```
assocstats(HairEye)

##                     X^2 df P(> X^2)
## Likelihood Ratio 146.44  9        0
## Pearson          138.29  9        0
##
## Phi-Coefficient   : 0.483
## Contingency Coeff.: 0.435
## Cramer's V        : 0.279
```

The further (and perhaps more interesting question) is how do we understand the *nature* of this association between hair and eye color? Two graphical methods related to the hypothesis testing approach are shown in Figure 1.1.

The left panel of Figure 1.1 is a ***mosaic display*** (Chapter **??**), constructed so that the size of each rectangle is proportional to the observed cell frequency. The shading reflects the cell contribution to the χ^2 statistic—shades of blue when the observed frequency is substantially greater than the expected frequency under independence, shades of red when the observed freqency is substantially less, as shown in the legend.

The right panel of this figure shows the results of a correspondence analysis (Chapter **??**), where the deviations of the hair color and eye color points from the origin accounts for as much of the χ^2 as possible in two dimensions.

We observe that both the hair colors and the eye colors are ordered from dark to light in the mosaic display and along Dimension 1 in the correspondence analysis plot. The deviations between observed and expected frequencies have an opposite-corner pattern in the mosaic display, except for the combination of red hair and green eyes, which also stand out as the largest values on Dimension 2 in the Correspondence analysis plot. Displays such as these provide a means to understand *how* the variables are related. △

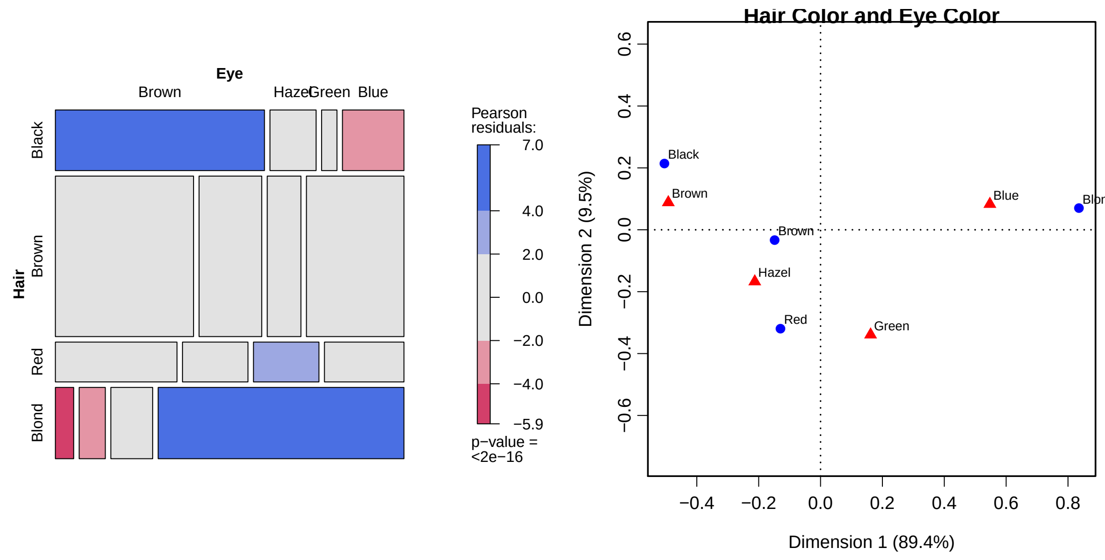


Figure 1.1: Graphical displays for the hair color and eye color data. Left: mosaic display; right: correspondence analysis plot

1.3.2 Model building approaches

Model-based methods provide tests of equivalent hypotheses about associations, but offer additional advantanges (at the cost of additional assumptions) not provided by the simpler hypotheses-testing approaches. Among these advantages, model-based methods provide estimates, standard errors and confidence intervals for parameters, and the ability to obtain predicted (fitted) values with associated measures of precision.

We illustrate this approach here for a dichotomous response variable, where it is often convenient to construct a model relating a function of the probability, π, of one event to a linear combination of the explanatory variables. Logistic regression uses the ***logit function***,

$$\text{logit}(\pi) \equiv \log_e \left(\frac{\pi}{1-\pi} \right)$$

which may be interpreted as the ***log odds*** of the given event. A linear logistic model can then be expressed as

$$\text{logit}(\pi) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$$

Statistical inferences from model-based methods provide tests of hypotheses for the effects of the predictors, x_1,x$_2, \ldots$, but they also provide estimates of parameters in the model, $\beta_1, \beta_2, \ldots$ and associated confidence intervals. Standard modeling tools allow us to graphically display the fitted response surface (with confidence or prediction intervals) and even to extrapolate these predictions beyond the given data. A particular advantage of the logit represention in the logistic regression model is that estimates of odds ratios (Section **??**) may be obtained directly from the parameter estimates.

Example 1.2: Space shuttle disaster

To illustrate the model-based approach, the graph in Figure 1.2 is based on a logistic regression model predicting the probability of a failure in one of the O-ring seals used in the 24 NASA space shuttles prior to the disasterous launch of the *Challenger* in January, 1986. The explanatory variable is the ambient temperature at the time of the flight. The sad story behind these data, and the lessons to be learned for graphical data display are related in Example **??**.

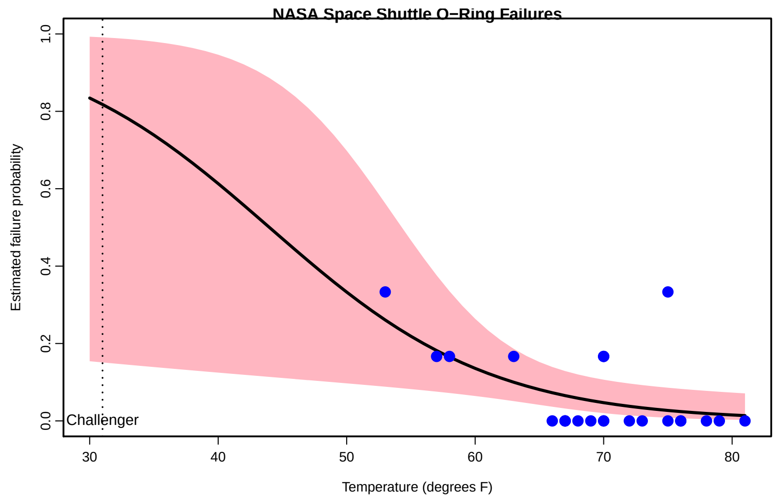


Figure 1.2: Space shuttle O-ring failure, observed and predicted probabilities. The dotted vertical line at 31° shows the prediction for the launch of the *Challenger*.

Here, we simply note that the fitted model, shown by the solid line in Figure 1.2, corresponds to the prediction equation (with standard errors shown in parentheses),

$$\text{logit}(\text{Failure}) = \underset{(3.06)}{5.09} - \underset{(0.047)}{0.116}\ \text{Temperature}$$

A hypothesis test that failure probability is unassociated with temperature is equivalent to the test that the coefficient for temperature in this model equals 0; this test has a p-value of 0.014, convincing evidence for rejection.

The parameter estimate for temperature, -0.116, however, gives more information. Each 1° increase in temperature decreases the log odds of failure by 0.116, with 95% confidence interval $(-0.208, -0.0235)$. The equivalent odds ratio is $\exp(-0.116) = 0.891$ (0.812–0.977). Equivalently, a 10° *decrease* in temperature corresponds to an odds ratio of a failure of $\exp(10 \times 0.116) = 3.18$, more than tripling the odds of a failure.

When the *Challenger* was launched, the temperature was only 31°. The shaded region in Figure 1.2 show 95% prediction intervals for failure probability. All previous shuttles (shown by the points in the figure) had been launched at much warmer temperatures, so the prediction interval (the dashed vertical line) at 31° represents a considerable extrapolation beyond the available data. Nonetheless, the model building approach does provide such predictions along with measures of their uncertainty. Figure 1.2 is a graph that might have saved lives.

△

ToDo: Perhaps replace this example with a similar one for the `Donner` data

1.4 Graphical methods for categorical data

You can see a lot, just by looking

Yogi Berra

The graphical methods for categorical data described in this book are in some cases straightforward adaptations of more familiar visualization techniques developed for quantitative data. Graphical principles and strategies, and the relations between the visualization approach and traditional statistical methods are described in a number of sources, including Chambers *et al.* (1983), Cleveland (1993) and several influential books by Tufte (Tufte, 1983, 1990, 1997, **?**).

The fundamental ideas of statistical graphics as a comprehensive system of visual signs and symbols with a grammar and semantics was first proposed in Jacques Bertin's *Semiology of Graphics* (1983), These ideas were later extended to a computational theory in Wilkinson's *Grammar of Graphics* (2005), and implemented in R in Hadley Wickham's ggplot2 package (Wickham, 2009, Wickham and Chang, 2013).

Another perspective on visual data display is presented in Section **??**. However, the discrete nature of categorical data implies that some familiar graphic methods need to be adapted, while in other cases we require a new graphic metaphor for data display. These issues are illustrated in Section **??**.

1.5 Visualization = Graphing + Fitting + Graphing

1.6 Further reading

Chapter 2
Working with categorical data

Creating and manipulating categorical data sets requires some skills and techniques in R beyond those ordinarily used for quantitative data. This chapter illustrates these for the main formats for categorical data: case form, frequency form and table form.

Categorical data can be represented as data sets in various formats: case form, frequency form, and table form. This chapter describes and illustrates the skills and techniques in R needed to input, create and manipulate R data objects to represent categorical data, and convert these from one form to another for the purposes of statistical analysis and visualization which are the subject of the remainder of the book.

As mentioned earlier, this book assumes that you have at least a basic knowledge of the R language and environment, including interacting with the R console (Rgui for Windows, R.app for Mac OS X) or some other graphical user interface (e.g., RStudio), loading and using R functions in packages (e.g., `library(vcd)`) getting help for these from R (e.g., `help(matrix)`), etc. This chapter is therefore devoted to covering those topics beyond such basic skills needed in the book.[1]

2.1 Working with R data: vectors, matrices, arrays and data frames

R has a wide veriety of data structures for storing, manipulating and calculating with data. Among these, vectors, matrices, arrays and data frames are most important for the material in this book.

In R, a ***vector*** is a collection of values, like numbers, character strings, logicals (`TRUE, FALSE`) or dates, and often correspond to a variable in some analysis. Matrices are rectangular arrays like a traditional table, composed of vectors in their columns or rows. Arrays add additional dimensions, so that, for example, a 3-way table can be represented as composed of rows, columns and layers. An important consideration is that the values in vectors, matrices and arrays must all be of the same *mode*, e.g., numbers or character strings. A ***data frame*** is a rectangular table, like a traditional data set in other statistical environments, and composed of rows and columns like a matrix, but allowing variables (columns) of different types.

[1]Some excellent introductory treatments of R are: Fox and Weisberg (2011, Chapter 2), ... Tom Short's *R Reference Card*, `http://cran.us.r-project.org/doc/contrib/Short-refcard.pdf` is a handy 4-page summary of the main functions. The web sites Quick-R `http://www.statmethods.net/` and Cookbook for R `http://www.cookbook-r.com/` provide very helful examples, organized by topics and tasks.

2.1.1 Vectors

The simplest data structure in R is a ***vector***, a one-dimensional collection of elements of the same type. An easy way to create a vector is with the `c()`, which combines its arguments. The following examples create and print vectors of length 4, containing numbers, character strings and logical values respectively:

```
c(17, 20, 15, 40)

## [1] 17 20 15 40

c("female", "male", "female", "male")

## [1] "female" "male"   "female" "male"

c(TRUE, TRUE, FALSE, FALSE)

## [1]  TRUE  TRUE FALSE FALSE
```

To store these values in variables, R uses the assignment operator (<−) or equals sign (=). This creates a variable named on the left-hand side. An assignment doesn't print the result, but a bare expression does, so you can assign and print by surrpounding the assignment with `()`.

```
count <- c(17, 20, 15, 40)                            # assign
count                                                 # print

## [1] 17 20 15 40

(sex <- c("female", "male", "female", "male"))    # both

## [1] "female" "male"   "female" "male"

(passed <- c(TRUE, TRUE, FALSE, FALSE))

## [1]  TRUE  TRUE FALSE FALSE
```

Other useful functions for creating vectors are:

- The `:` operator for generating consecutive integer sequences, e.g., `1:10` gives the integers 1 to 10. The `seq()` function is more general, taking the forms `seq(from, to)`, `seq(from, to, by= )`, and `seq(from, to, length= )` where the optional argument `by` specifies the interval between adjacent values and `length` gives the desired length of the result.
- The `rep()` function generates repeated sequences, replicating its first argument (which may be a vector) a given number of `times`, to a given `length` or `each` a given multiple.

```
seq(10, 100, by=10)        # give interval

##  [1]  10  20  30  40  50  60  70  80  90 100

seq(0, 1, length=11)       # give length
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

(sex <- rep(c("female", "male"), times=2))

## [1] "female" "male"   "female" "male"

(sex <- rep(c("female", "male"), length.out=4))  # same

## [1] "female" "male"   "female" "male"

(passed <- rep(c(TRUE, FALSE), each=2))

## [1]  TRUE  TRUE FALSE FALSE
```

2.1.2 Matrices

A ***matrix*** is a two-dimensional array of elements of the same type composed in a rectangular array of rows and columns. Matrices can be created by the function `matrix(values, nrow, ncol)`, which takes the reshapes the elements in the first argument (`values`) to a matrix with `nrow` rows and `ncol` columns. By default, the elements are filled in columnwise, unless the optional argument `byrow=TRUE` is given.

```
(matA <- matrix(1:8, nrow=2, ncol=4))

##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8

(matB <- matrix(1:8, nrow=2, ncol=4, byrow=TRUE))

##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8

(matC <- matrix(1:4, nrow=2, ncol=4))

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

The last example illustrates that the values in the first argument are recycled as necessary to fill the given number of rows and columns.

All matrices have a dimensions attribute, a vector of length two giving the number of rows and columns, retrieved with the function `dim()`. Labels for the rows and columns can be assigned using `dimnames()`,[2] which takes a list of two vectors for the row names and column names respectively. To see the structure of a matrix (or any other R object) and its attributes, I frequently use the `str()` function, as shown in the example below.

[2] The `dimnames` can also be specified as an optional argument to `matrix()`.

```
dim(matA)

## [1] 2 4

str(matA)

##  int [1:2, 1:4] 1 2 3 4 5 6 7 8

dimnames(matA) <- list(c("M","F"), LETTERS[1:4])
matA

##   A B C D
## M 1 3 5 7
## F 2 4 6 8

str(matA)

##  int [1:2, 1:4] 1 2 3 4 5 6 7 8
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2] "M" "F"
##   ..$ : chr [1:4] "A" "B" "C" "D"
```

Additionally, names for the row and column *variables* themselves can also be assigned in the `dimnames` call by giving each dimension vector a name.

```
dimnames(matA) <- list(sex=c("M","F"), group=LETTERS[1:4])
matA

##    group
## sex A B C D
##   M 1 3 5 7
##   F 2 4 6 8

str(matA)

##  int [1:2, 1:4] 1 2 3 4 5 6 7 8
##  - attr(*, "dimnames")=List of 2
##   ..$ sex  : chr [1:2] "M" "F"
##   ..$ group: chr [1:4] "A" "B" "C" "D"
```

Matrices can also be created or enlarged by "binding" vectors or matrices together by rows or columns:

- `rbind(a, b, c)` creates a matrix with the vectors `a`, `b` and `c` as its rows, recycling the elements as necessary to the length of the longest one.
- `cbind(a, b, c)` creates a matrix with the vectors `a`, `b` and `c` as its columns.
- `rbind(mat, a, b, ...)` and `cbind(mat, a, b, ...)` add additional rows (columns) to a matrix `mat`, recycling or subsetting the elements in the vectors to conform with the size of the matrix.

```
rbind(matA, c(10,20))
```

```
##     A  B  C  D
## M   1  3  5  7
## F   2  4  6  8
##    10 20 10 20

cbind(matA, c(10,20))

##   A B C D
## M 1 3 5 7 10
## F 2 4 6 8 20
```

2.1.3 Arrays

Higher-dimensional arrays are less frequently encountered in traditional data analysis, but they are of great use for categorical data, where frequency tables of three or more variables can be naturally represented as arrays, with one dimension for each table variable.

The function `array(values, dim)` takes the elements in `values` and reshapes thse into an array whose dimensions are given in the vector `dim`. The number of dimensions is the length of `dim`. As with matrices, the elements are filled in with the first dimension (rows) varying most rapidly, then by the second dimension (columns) and so on for all further dimensions, which can be considered as layers. A matrix is just the special case of an array with two dimensions.

```
(arrayA <- array(1:16, dim=c(2, 4, 2)))      # 2 rows, 4 columns, 2 layers

## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]    9   11   13   15
## [2,]   10   12   14   16

str(arrayA)

##  int [1:2, 1:4, 1:2] 1 2 3 4 5 6 7 8 9 10 ...

(arrayB <- array(1:16, dim=c(2, 8)))         # 2 rows, 8 columns

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    3    5    7    9   11   13   15
## [2,]    2    4    6    8   10   12   14   16

str(arrayB)

##  int [1:2, 1:8] 1 2 3 4 5 6 7 8 9 10 ...
```

In the same way that we can assign labels to the rows, columns and variables in matrices, we can assign these attributes to `dimnames(arrayA)`, or include this information in a `dimnames=` argument to `array()`.

```
dimnames(arrayA) <- list(sex=c("M", "F"),
                         group=letters[1:4],
                         time=c("Pre", "Post"))
arrayA

## , , time = Pre
##
##    group
## sex a b c d
##   M 1 3 5 7
##   F 2 4 6 8
##
## , , time = Post
##
##    group
## sex  a  b  c  d
##   M  9 11 13 15
##   F 10 12 14 16

str(arrayA)

##  int [1:2, 1:4, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ sex  : chr [1:2] "M" "F"
##   ..$ group: chr [1:4] "a" "b" "c" "d"
##   ..$ time : chr [1:2] "Pre" "Post"
```

Arrays in R can contain any single type of elements— numbers, character strings, logicals. R also has a variety of functions (e.g., `table()`, `xtabs()`) for creating and manipulating "table" objects, which are spacialized forms of matrices and arrays containing integer frequencies in a contingency table. These are discussed in more detail below (Section 2.4).

2.1.4 data frames

Data frames are the most commonly used form of data in R and more general than matrices in that they can contain columns of different types. For statistical modeling, data frames play a special role, in that many modeling functions are designed to take a data frame as a `data=` argument, and then find the variables mentioned within that data frame. Another distinguishing feature is that discrete variables (columns) like character strings `("M", "F")` or integers `(1, 2, 3)` in data frames can be represented as ***factor***s, which simplifies many statistical and graphical methods.

A data frame can be created using keyboard input with the `data.frame()` function, applied to a list of objects, `data.frame(a, b, c, ...)`, each of which can be a vector, matrix or another data frame, but typically all containing the same humber of rows. This works roughly like `cbind()`, collecting the arguments as columns in the result.

The following example generates `n=100` random observations on three discrete factor variables, `A, B, sex`, and a numeric variable, `age`. As constructed, all of these are statistically independent, since none depends on any of the others. The function `sample()` is used here to generate `n` random samples from the first argument allowing repetitions (`rep=TRUE`). Finally, all four variables are combined into the data frame `mydata`.

```r
set.seed(12345)    # reproducibility
n=100
A <- factor(sample(c("a1","a2"), n, rep=TRUE))
B <- factor(sample(c("b1","b2"), n, rep=TRUE))
sex <- factor(sample(c("M", "F"), n, rep=TRUE))
age <- round(rnorm(n, mean=30, sd=5))
mydata <- data.frame(A, B, sex, age)
head(mydata,5)

##    A  B sex age
## 1 a2 b1   F  22
## 2 a2 b2   F  33
## 3 a2 b2   M  31
## 4 a2 b2   F  26
## 5 a1 b2   F  29

str(mydata)

## 'data.frame': 100 obs. of  4 variables:
##  $ A  : Factor w/ 2 levels "a1","a2": 2 2 2 2 1 1 1 2 2 2 ...
##  $ B  : Factor w/ 2 levels "b1","b2": 1 2 2 2 2 2 2 2 1 1 ...
##  $ sex: Factor w/ 2 levels "F","M": 1 1 2 1 1 1 2 2 1 1 ...
##  $ age: num  22 33 31 26 29 29 38 28 30 27 ...
```

For real data sets, it is usually most convenient to read these into R from external files, and this is easiest using plain text (ASCII) files with one line per observation and fields separated by commas (or tabs), and with a first header line giving the variable names– called *comma-separated* or CSV format. If your data is in the form of Excel, SAS, SPSS or other file format, you can almost always export that data to CSV format first.[3]

The function `read.table()` has many options to control the details of how the data are read and converted to variables in the data frame. Among these some important options are:

`header` (default: `FALSE`) indicates whether the first line contains variable names;
`sep` (default: `""` meaning white space, i.e., one or more spaces, tabs or newlines) specifies the separator character between fields;
`stringsAsFactors` (default: TRUE) determines whether character string variables should be converted to factors;
`na.strings` (default: `"NA"` one or more strings which are interpreted as missing data values (`NA`);

For delimited files, `read.csv()` and `read.delim()` are convenient wrappers to `read.table()`, with default values `sep=","` and `sep="\t"` respectively, and `header=TRUE`.

Example 2.1: Arthritis treatment

The file `Arthritis.csv` contains data in CSV format from Koch and Edwards (1988), representing a double-blind clinical trial investigating a new treatment for rheumatoid arthritis with 84 patients. The first ("header") line gives the variable names. Some of the lines in the file are shown below, with `...` representing omitted lines:

[3]The foreign package contains specialized functions to *directly* read data stored by Minitab, SAS, SPSS, Stata, Systat and other software. There are also a number of packages for reading (and writing) Excel spreadsheets directly (gdata, XLConnect, xlsx). The R manual, *R Data Import/Export* covers many other variations, including data in relational data bases.

```
ID,Treatment,Sex,Age,Improved
57,Treated,Male,27,Some
46,Treated,Male,29,None
77,Treated,Male,30,None
17,Treated,Male,32,Marked
 ...
42,Placebo,Female,66,None
15,Placebo,Female,66,Some
71,Placebo,Female,68,Some
1,Placebo,Female,74,Marked
```

We read this into R using `read.csv()` as shown below, using all the default options:

```
Arthritis <- read.csv("ch02/Arthritis.csv")
str(Arthritis)

## 'data.frame': 84 obs. of  5 variables:
##  $ ID       : int  57 46 77 17 36 23 75 39 33 55 ...
##  $ Treatment: Factor w/ 2 levels "Placebo","Treated": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Sex      : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Age      : int  27 29 30 32 46 58 59 59 63 63 ...
##  $ Improved : Factor w/ 3 levels "Marked","None",..: 3 2 2 1 1 1 2 1 2 2 ...
```

Note that the character variables *`Treatment`*, *`Sex`* and *`Improved`* were converted to factors, and the levels of those variables were ordered *alphabetically*. This often doesn't matter much for binary variables, but here, the response variable, *`Improved`* has levels that should be considered *ordered*, as `"None", "Some", "Marked"`. We can correct this here by re-assigning `Arthritis$Improved` using `ordered()`. The topic of re-ordering variables and levels in categorical data is considered in more detail in Section 2.3.

```
levels(Arthritis$Improved)

## [1] "Marked" "None"   "Some"

Arthritis$Improved <- ordered(Arthritis$Improved,
                              levels=c("None", "Some", "Marked"))
```

△

2.2 Forms of categorical data: case form, frequency form and table form

As we saw in Chapter 1, categorical data can be represented as ordinary data sets in case form, but the discrete nature of factors or stratifying variables allows the same information to be represented more compactly in summarized form with a frequency variable for each cell of factor combinations, or in tables. Consequently, we sometimes find data created or presented in one form (e.g., a spreadsheet data set, a two-way table of frequencies) and want to input that into R. Once we have the data in R, it is often necessary to manipulate the data into some other form for the purposes of statistical analysis, visualizing results and our own presentation. It is useful to understand the three main forms of categorical data in R and how to work with them for our purposes.

2.2.1 Case form

Categorical data in case form are simply data frames, with one or more discrete classifying variables or response variables, most conveniently represented as factors or ordered factors. In case form, the data set can also contain numeric variables (covariates or other response variables), that cannot be accommodated in other forms.

As with any data frame, X, you can access or compute with its attributes using nrow(X) for the number of observations, ncol(X) for the number of variables, names(X) or colnames(X) for the variable names and so forth.

EXAMPLE 2.2: Arthritis treatment

The Arthritis data is available in case form in the vcd package. There are two explanatory factors: Treatment and Sex. Age is a numeric covariate, and Improved is the response— an ordered factor, with levels "None" < "Some" < "Marked". Excluding Age, we would have a $2 \times 2 \times 3$ contingency table for Treatment, Sex and Improved.

```
data(Arthritis, package="vcd")  # load the data
names(Arthritis)      # show the variables

## [1] "ID"        "Treatment" "Sex"       "Age"
## [5] "Improved"

str(Arthritis)        # show the structure

## 'data.frame':  84 obs. of  5 variables:
##  $ ID       : int  57 46 77 17 36 23 75 39 33 55 ...
##  $ Treatment: Factor w/ 2 levels "Placebo","Treated": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Sex      : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Age      : int  27 29 30 32 46 58 59 59 63 63 ...
##  $ Improved : Ord.factor w/ 3 levels "None"<"Some"<..: 2 1 1 3 3 3 1 3 1 1 ...

head(Arthritis,5)     # first 5 observations, same as Arthritis[1:5,]

##   ID Treatment  Sex Age Improved
## 1 57   Treated Male  27     Some
## 2 46   Treated Male  29     None
## 3 77   Treated Male  30     None
## 4 17   Treated Male  32   Marked
## 5 36   Treated Male  46   Marked
```

△

2.2.2 Frequency form

Data in frequency form is also a data frame, containing one or more discrete factor variables and a frequency variable (often called Freq or count) representing the number of basic observations in that cell.

This is an alternative representation of a table form data set considered below. In frequency form, the number of cells in the equivalent table is nrowX, and the total number of observations is the sum of the frequency variable, sum(X$Freq), sum(X[,"Freq"]) or similar expression.

EXAMPLE 2.3: General social survey

For small frequency tables, it is often convenient to enter them in frequency form using `expand.grid()` for the factors and `c()` to list the counts in a vector. The example below, from Agresti (2002) gives results for the 1991 General Social Survey, with respondents classified by sex and party identification. As a table, the data look like this:

	party		
sex	dem	indep	rep
female	279	73	225
male	165	47	191

We use `expand.grid()` to create a 6×2 matrix containing the combinations of `sex` and `party` with the levels for `sex` given first, so that this varies most rapidly. Then, input the frequencies in the table by columns from left to right, and combine these two results with `data.frame()`.

```
# Agresti (2002), table 3.11, p. 106
GSS <- data.frame(
  expand.grid(sex=c("female", "male"),
              party=c("dem", "indep", "rep")),
  count=c(279,165,73,47,225,191))
GSS

##      sex party count
## 1 female   dem   279
## 2   male   dem   165
## 3 female indep    73
## 4   male indep    47
## 5 female   rep   225
## 6   male   rep   191

names(GSS)

## [1] "sex"   "party" "count"

str(GSS)

## 'data.frame': 6 obs. of  3 variables:
##  $ sex  : Factor w/ 2 levels "female","male": 1 2 1 2 1 2
##  $ party: Factor w/ 3 levels "dem","indep",..: 1 1 2 2 3 3
##  $ count: num  279 165 73 47 225 191

sum(GSS$count)

## [1] 980
```

The last line above shows that there are 980 cases represented in the frequency table. △

2.2.3 Table form

Table form data is represented as a matrix, array or table object whose elements are the frequencies in an n-way table. The number of dimensions of the table is the length, `length(dim(X))`, of its `dim` (or `dimnames`) attribute, and the sizes of the dimensions in the table are the elements of `dim(X)`. The total number of observations represented is the sum of all the frequencies, `sum(X)`.

EXAMPLE 2.4: Hair-eye color data

A classic data set on frequencies of hair color, eye color and sex is given in table form in `HairEyeColor` in the vcd package, reporting the frequencies of these categories for 592 students in a statistics course.

```
data(HairEyeColor, package="datasets")     # load the data
str(HairEyeColor)                          # show the structure

##  table [1:4, 1:4, 1:2] 32 53 10 3 11 50 10 30 10 25 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Hair: chr [1:4] "Black" "Brown" "Red" "Blond"
##   ..$ Eye : chr [1:4] "Brown" "Blue" "Hazel" "Green"
##   ..$ Sex : chr [1:2] "Male" "Female"

dim(HairEyeColor)                          # table dimension sizes

## [1] 4 4 2

dimnames(HairEyeColor)                     # variable and level names

## $Hair
## [1] "Black" "Brown" "Red"   "Blond"
##
## $Eye
## [1] "Brown" "Blue"  "Hazel" "Green"
##
## $Sex
## [1] "Male"   "Female"

sum(HairEyeColor)                          # number of cases

## [1] 592
```

Three-way (and higher-way) tables can be printed in a more convenient form using `structable()` and `ftable()` as described below in Section 2.5. △

Tables are often created from raw data in case form or frequency form using the functions `table()` and `xtabs()` described in Section 2.4. For smallish frequency tables that are already in tabular form, you can enter the frequencies in a matrix, and then assign `dimnames` and other attributes.

To illustrate, we create the GSS data as a table below, entering the values in the table by rows (`byrow=TRUE`), as they appear in printed form.

```
GSS.tab <- matrix(c(279, 73, 225,
                    165, 47, 191), nrow=2, ncol=3, byrow=TRUE)
dimnames(GSS.tab) <- list(sex=c("female", "male"),
                          party=c("dem", "indep", "rep"))
GSS.tab

##         party
## sex      dem indep rep
##   female 279    73 225
##   male   165    47 191
```

GSS.tab is a matrix, not an object of class("table"), and some functions are happier with tables than matrices.[4] You can coerce it to a table with as.table(),

```
GSS.tab <- as.table(GSS.tab)
str(GSS.tab)

##  table [1:2, 1:3] 279 165 73 47 225 191
##  - attr(*, "dimnames")=List of 2
##   ..$ sex  : chr [1:2] "female" "male"
##   ..$ party: chr [1:3] "dem" "indep" "rep"
```

Here is another similar example, entering data on job satisfactory classified by income and level of satisfaction from a 4×4 table given by Agresti (2002, Table 2.8, p. 57).

```
## A 4 x 4 table  Agresti (2002, Table 2.8, p. 57) Job Satisfaction
JobSat <- matrix(c(1,2,1,0, 3,3,6,1, 10,10,14,9, 6,7,12,11), 4, 4)
dimnames(JobSat) = list(income=c("< 15k", "15-25k", "25-40k", "> 40k"),
                satisfaction=c("VeryD", "LittleD", "ModerateS", "VeryS"))
JobSat <- as.table(JobSat)
JobSat

##         satisfaction
## income   VeryD LittleD ModerateS VeryS
##   < 15k      1       3        10     6
##   15-25k     2       3        10     7
##   25-40k     1       6        14    12
##   > 40k      0       1         9    11
```

2.3 Ordered factors and reordered tables

As we saw above (Example 2.1), factor variables in data frames (case form or frequency form) can be re-ordered and declared as ordered factors using ordered(). As well, the order of the factors themselves can be rearranged by sorting the data frame using sort().

However, in table form, the values of the table factors are ordered by their position in the table. Thus in the JobSat data, both income and satisfaction represent ordered factors, and the *positions* of the values in the rows and columns reflects their ordered nature, but only implicitly.

Yet, for analysis or graphing, there are occasions when you need *numeric* values for the levels of ordered factors in a table, e.g., to treat a factor as a quantitative variable. In such cases, you can simply re-assign the dimnames attribute of the table variables. For example, here, we assign numeric values to income as the middle of their ranges, and treat satisfaction as equally spaced with integer scores.

```
dimnames(JobSat)$income <- c(7.5,20,32.5,60)
dimnames(JobSat)$satisfaction <- 1:4
```

A related case is when you want to preserve the character labels of table dimensions, but also allow them to be sorted in some particular order. A simple way to do this is to prefix each label with an integer index using paste().

[4]There are quite a few functions in R with specialized methods for "table" objects. For example, plot(GSS.tab) gives a mosaic plot and barchart(GSS.tab) gives a divided bar chart.

```
dimnames(JobSat)$income <- paste(1:4, dimnames(JobSat)$income, sep=":")
dimnames(JobSat)$satisfaction <-
                            paste(1:4, dimnames(JobSat)$satisfaction, sep=":")
```

A different situation arises with tables where you want to *permute* the levels of one or more variables to arrange them in a more convenient order without changing their labels. For example, in the `HairEyeColor` table, hair color and eye color are ordered arbitrarily. For visualizing the data using mosaic plots and other methods described later, it turns out to be more useful to assure that both hair color and eye color are ordered from dark to light. Hair colors are actually ordered this way already: `"Black", "Brown", "Red", "Blond"`. But eye colors are ordered as `"Brown", "Blue", "Hazel", "Green"`. It is easiest to re-order the eye colors by indexing the columns (dimension 2) in this array to a new order, `"Brown", "Hazel", "Green", "Blue"`, giving the indices of the old levels in the new order (here: 1,3,4,2). Again `str()` is your friend, showing the structure of the result to check that the result is what you want.

```
data(HairEyeColor, package="datasets")
HEC <- HairEyeColor[, c(1,3,4,2), ]
str(HEC)

##  num [1:4, 1:4, 1:2] 32 53 10 3 10 25 7 5 3 15 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Hair: chr [1:4] "Black" "Brown" "Red" "Blond"
##   ..$ Eye : chr [1:4] "Brown" "Hazel" "Green" "Blue"
##   ..$ Sex : chr [1:2] "Male" "Female"
```

Finally, there are situations where, particularly for display purposes, you want to re-order the *dimensions* of an n-way table, and/or change the labels for the variables or levels. This is easy when the data are in table form: `aperm()` permutes the dimensions, and assigning to `names` and `dimnames` changes variable names and level labels respectively.

```
str(UCBAdmissions)

##  table [1:2, 1:2, 1:6] 512 313 89 19 353 207 17 8 120 205 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Admit : chr [1:2] "Admitted" "Rejected"
##   ..$ Gender: chr [1:2] "Male" "Female"
##   ..$ Dept  : chr [1:6] "A" "B" "C" "D" ...

UCB <- aperm(UCBAdmissions, c(2, 1, 3))
dimnames(UCB)[[2]] <- c("Yes", "No")
names(dimnames(UCB)) <- c("Sex", "Admitted", "Department")
str(UCB)

##  table [1:2, 1:2, 1:6] 512 89 313 19 353 17 207 8 120 202 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Sex       : chr [1:2] "Male" "Female"
##   ..$ Admitted  : chr [1:2] "Yes" "No"
##   ..$ Department: chr [1:6] "A" "B" "C" "D" ...
```

2.4 Generating tables with `table()` and `xtabs()`

With data in case form or frequency form, you can generate frequency tables from factor variables in data frames using the `table()` function; for tables of proportions, use the `prop.table()` function, and for marginal frequencies (summing over some variables) use `margin.table()`. The examples below use the same case-form data frame `mydata` used earlier (Section 2.1.4).

```
set.seed(12345)   # reproducibility
n=100
A <- factor(sample(c("a1","a2"), n, rep=TRUE))
B <- factor(sample(c("b1","b2"), n, rep=TRUE))
sex <- factor(sample(c("M", "F"), n, rep=TRUE))
age <- round(rnorm(n, mean=30, sd=5))
mydata <- data.frame(A, B, sex, age)
```

`table(...)` takes a list of variables interpreted as factors, or a data frame whose columns are so interpreted. It does not take a `data=` argument, so either supply the names of columns in the data frame, or select the variables using column indexes:

```
# 2-Way Frequency Table
table(mydata$A, mydata$B)             # A will be rows, B will be columns

##
##       b1 b2
##    a1 18 30
##    a2 22 30

(mytab <- table(mydata[,1:2]))        # same

##      B
## A     b1 b2
##    a1 18 30
##    a2 22 30
```

We can use `margin.table(X, margin)` to sum a table `X` for the indices in `margin`, i.e., over the dimensions not included in `margin`. A related function is `addmargins(X, margin, FUN=sum)`, which extends the dimensions of a table or array with the marginal values calculated by `FUN`.

```
margin.table(mytab)       # sum over A & B

## [1] 100

margin.table(mytab, 1)    # A frequencies (summed over B)

## A
## a1 a2
## 48 52

margin.table(mytab, 2)    # B frequencies (summed over A)

## B
## b1 b2
## 40 60
```

```
addmargins(mytab)         # show all marginal totals

##        B
## A       b1  b2 Sum
##   a1    18  30  48
##   a2    22  30  52
##   Sum   40  60 100
```

The function `prop.table()` expresses the table entries as a fraction of a given marginal table.

```
prop.table(mytab)         # cell percentages

##      B
## A       b1   b2
##   a1 0.18 0.30
##   a2 0.22 0.30

prop.table(mytab, 1)      # row percentages

##      B
## A         b1     b2
##   a1 0.3750 0.6250
##   a2 0.4231 0.5769

prop.table(mytab, 2)      # column percentages

##      B
## A       b1   b2
##   a1 0.45 0.50
##   a2 0.55 0.50
```

`table()` can also generate multidimensional tables based on 3 or more categorical variables. In this case, use the `ftable()` or `structable()` function to print the results more attractively as a "flat" (2-way) table.

```
# 3-Way Frequency Table
mytab <- table(mydata[,c("A", "B", "sex")])
ftable(mytab)

##         sex  F  M
## A  B
## a1 b1        9  9
##    b2       15 15
## a2 b1       12 10
##    b2       19 11
```

`table()` ignores missing values by default, but has optional arguments `useNA` and `exclude` that can be used to control this. See `help(table)` for the details.

2.4.1 `xtabs()`

The `xtabs()` function allows you to create crosstabulations of data using formula style input. This typically works with case-form or frequency-form data supplied in a data frame or a matrix.

The result is a contingency table in array format, whose dimensions are determined by the terms on the right side of the formula. As shown below, the `summary` method for tables produces a simple χ^2 test of independence of all factors.

```
# 3-Way Frequency Table
mytable <- xtabs(~A+B+sex, data=mydata)
ftable(mytable)     # print table

##        sex  F  M
## A  B
## a1 b1       9  9
##    b2      15 15
## a2 b1      12 10
##    b2      19 11

summary(mytable)    # chi-square test of independence

## Call: xtabs(formula = ~A + B + sex, data = mydata)
## Number of cases in table: 100
## Number of factors: 3
## Test for independence of all factors:
##  Chisq = 1.5, df = 4, p-value = 0.8
```

When the data have already been tabulated in frequency form, include the frequency variable (usually `count` or `Freq`) on the left side of the formula, as shown in the example below for the GSS data.

```
(GSStab <- xtabs(count ~ sex + party, data=GSS))

##         party
## sex       dem indep rep
##   female 279    73 225
##   male   165    47 191

summary(GSStab)

## Call: xtabs(formula = count ~ sex + party, data = GSS)
## Number of cases in table: 980
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 7, df = 2, p-value = 0.03
```

For "table" objects, the `plot` method produces basic mosaic plots using the `mosaicplot()` function. With the option `shade=TRUE`, the cells are shaded according to the deviations (residuals) from an independence model. Mosaic plot are discussed in detail in Chapter **??**.

```
plot(mytable)
plot(GSStab, shade=TRUE)
```

2.5 Printing tables with `structable()` and `ftable()`

For 3-way and larger tables, the functions `ftable()` (in the stats package) and `structable()` (in vcd) provide a convenient and flexible tabular display in a “flat” (2-way) format.

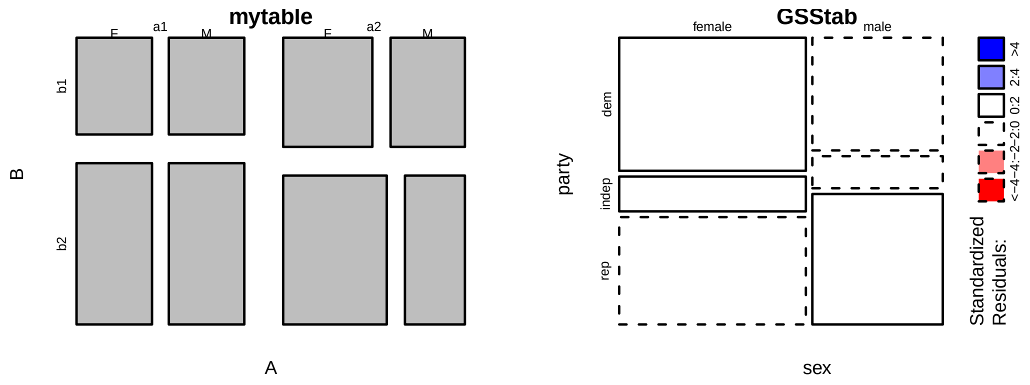


Figure 2.1: Mosaic plot of tables using the plot method for table objects

With `ftable(X, row.vars=, col.vars=)`, variables assigned to the rows and/or columns of the result can be specified as the integer numbers or character names of the variables in the array `X`. By default, the last variable is used for the columns. The formula method, in the form `ftable(colvars ~ rowvars, data)` allows a formula, where the left and right hand side of formula specify the column and row variables respectively.

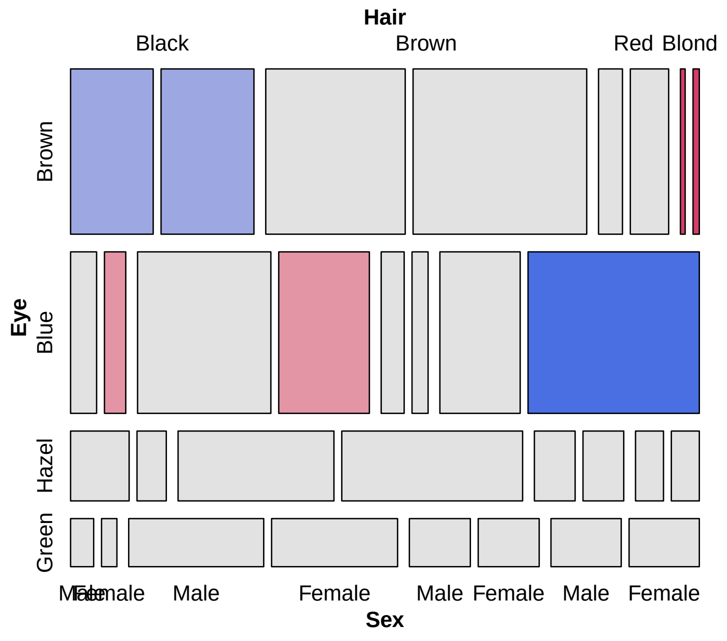
```
 ftable(UCB)                             # default

##                    Department   A   B   C   D   E   F
## Sex    Admitted
## Male   Yes                     512 353 120 138  53  22
##        No                      313 207 205 279 138 351
## Female Yes                      89  17 202 131  94  24
##        No                       19   8 391 244 299 317

#ftable(UCB, row.vars=1:2)        # same result
 ftable(Admitted + Sex ~ Department, data=UCB)    # formula method

##            Admitted  Yes         No
##            Sex      Male Female Male Female
## Department
## A                    512     89  313     19
## B                    353     17  207      8
## C                    120    202  205    391
## D                    138    131  279    244
## E                     53     94  138    299
## F                     22     24  351    317
```

The `structable()` function is similar, but more general, and uses recursive splits in the vertical or horizontal directions (similar to the construction of mosaic displays). It works with both data frames and table objects.

```
structable(HairEyeColor)                            # show the table: default

##               Eye Brown Blue Hazel Green
## Hair  Sex
## Black Male          32   11    10     3
##       Female        36    9     5     2
## Brown Male          53   50    25    15
##       Female        66   34    29    14
## Red   Male          10   10     7      7
##       Female        16    7     7      7
## Blond Male           3   30     5      8
##       Female         4   64     5      8

structable(Hair+Sex ~ Eye, HairEyeColor)   # specify col ~ row variables

##        Hair Black        Brown         Red         Blond
##        Sex    Male Female  Male Female Male Female  Male Female
## Eye
## Brown           32     36    53     66   10     16     3      4
## Blue            11      9    50     34   10      7    30     64
## Hazel           10      5    25     29    7      7     5      5
## Green            3      2    15     14    7      7     8      8
```

It also returns an object of class "structable" for which there are a variety of special methods. For example, the transpose function t() interchanges rows and columns, so that t(structable(HairEyeColor)) produces the second result shown just above; "structable" objects can be subset using the [and [[operators, using either level indices or names. There are also plot methods, so that plot() and mosaic() produce mosaic plots.

```
HSE <- structable(Hair+Sex ~ Eye, HairEyeColor)   # save structable object
HSE[1:2,]                                         # subset rows

##        Hair Black        Brown         Red         Blond
##        Sex    Male Female  Male Female Male Female  Male Female
## Eye
## Brown           32     36    53     66   10     16     3      4
## Blue            11      9    50     34   10      7    30     64

mosaic(HSE, shade=TRUE, legend=FALSE)             # plot it
```



2.5.1 Publishing tables to LaTeXor HTML

OK, you've read your data into R, done some analysis, and now want to include some tables in a LaTeXdocument or in a web page in HTML format. Formatting tables for these purposes is often tedious and error-prone.

There are a great many packages in R that provide for nicely formatted, publishable tables for a wide variety of purposes; indeed, most of the tables in this book are generated using these tools. See Leifeld (2013) for description of the texreg package and a comparison with some of the other packages.

Here, we simply illustrate the xtable package, which, along with capabilities for statistical model summaries, time-series data, and so forth, has a `xtable.table` method for one-way and two-way table objects.

The `HorseKicks` data is a small one-way frequency table described in Example 3.4 and containing the frequencies of 0, 1, 2, 3, 4 deaths per corps-year by horse-kick among soldiers in 20 corps in the Prussian army.

```
data(HorseKicks, package="vcd")
HorseKicks

## nDeaths
##   0   1   2   3   4
## 109  65  22   3   1
```

By default, `xtable()` formats this in LaTeXas a vertical table, and prints the LaTeXmarkup to the R console. This output is shown below (without the usual `##` comment used to indicate R output).

```
library(xtable)
xtable(HorseKicks)

% latex table generated in R 3.0.1 by xtable 1.7-1 package
% Wed Dec 18 17:08:43 2013
\begin{table}[ht]
\centering
\begin{tabular}{rr}
  \hline
 & nDeaths \\
  \hline
0 & 109 \\
  1 &  65 \\
  2 &  22 \\
  3 &   3 \\
  4 &   1 \\
   \hline
\end{tabular}
\end{table}
```

When this is rendered in a LaTeXdocument, the result of `xtable()` appears as shown in the table below.

```
xtable(HorseKicks)
```

The table above isn't quite right, because the column label "nDeaths" belongs to the first column, and the second column should be labeled "Freq". To correct that, we convert the

	nDeaths
0	109
1	65
2	22
3	3
4	1

HorseKicks table to a data frame (see Section 2.7 for details), add the appropriate colnames, and use the xtable.print method to supply some other options.

```
tab <- as.data.frame(HorseKicks)
colnames(tab) <- c("nDeaths", "Freq")
print(xtable(tab), include.rownames=FALSE, include.colnames=TRUE)
```

nDeaths	Freq
0	109
1	65
2	22
3	3
4	1

Finally, in Chapter 3, we display a number of similar one-way frequency tables in a transposed form to save display space. Table 3.3 is the finished version we show there. The code below uses the following techniques: (a) addmargins() is used to show the sum of all the frequency values; (b) t() transposes the data frame to have 2 rows; (c) rownames() assigns the labels we want for the rows; (d) using the caption argument provides a table caption, and a numbered table in LaTeX. (d) column alignment ("r" or "l") for the table columns is computed as a character string used for the align argument.

```
horsetab <- t( as.data.frame( addmargins( HorseKicks ) ) )
rownames( horsetab ) <- c( "Number of deaths", "Frequency" )
horsetab <- xtable( horsetab, digits = 0,
     caption = "von Bortkiewicz's data on deaths by horse kicks",
     align = paste0("l|", paste(rep("r", ncol(horsetab)), collapse=""))
     )
print(horsetab, include.colnames=FALSE)
```

Number of deaths	0	1	2	3	4	Sum
Frequency	109	65	22	3	1	200

Table 2.1: von Bortkiewicz's data on deaths by horse kicks

2.6 Collapsing over table factors: aggregate(), margin.table() and apply()

It sometimes happens that we have a data set with more variables or factors than we want to analyse, or else, having done some initial analyses, we decide that certain factors are not important, and so should be excluded from graphic displays by collapsing (summing) over them. For example, mosaic plots and fourfold displays are often simpler to construct from versions of the data collapsed over the factors which are not shown in the plots.

The appropriate tools to use again depend on the form in which the data are represented— a case-form data frame, a frequency-form data frame (aggregate()), or a table-form array or table object (margin.table() or apply()).

When the data are in frequency form, and we want to produce another frequency data frame, aggregate() is a handy tool, using the argument FUN=sum to sum the frequency variable over the factors *not* mentioned in the formula.

EXAMPLE 2.5: Dayton survey

The data frame DaytonSurvey in the vcdExtra package represents a 2^5 table giving the frequencies of reported use ("ever used?") of alcohol, cigarettes and marijuana in a sample of 2276 high school seniors, also classified by sex and race.

```
data(DaytonSurvey, package="vcdExtra")
str(DaytonSurvey)

## 'data.frame':	32 obs. of  6 variables:
##  $ cigarette: Factor w/ 2 levels "Yes","No": 1 2 1 2 1 2 1 2 1 2 ...
##  $ alcohol  : Factor w/ 2 levels "Yes","No": 1 1 2 2 1 1 2 2 1 1 ...
##  $ marijuana: Factor w/ 2 levels "Yes","No": 1 1 1 1 2 2 2 2 1 1 ...
##  $ sex      : Factor w/ 2 levels "female","male": 1 1 1 1 1 1 1 1 2 2 ...
##  $ race     : Factor w/ 2 levels "white","other": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Freq     : num  405 13 1 1 268 218 17 117 453 28 ...

head(DaytonSurvey)

##   cigarette alcohol marijuana    sex  race Freq
## 1       Yes     Yes       Yes female white  405
## 2        No     Yes       Yes female white   13
## 3       Yes      No       Yes female white    1
## 4        No      No       Yes female white    1
## 5       Yes     Yes        No female white  268
## 6        No     Yes        No female white  218
```

To focus on the associations among the substances, we want to collapse over sex and race. The right-hand side of the formula used in the call to aggregate() gives the factors to be retained in the new frequency data frame, Dayton.ACM.df. The left-hand side is the frequency variable (Freq), and we aggregate using the FUN=sum.

```
# data in frequency form: collapse over sex and race
Dayton.ACM.df <- aggregate(Freq ~ cigarette+alcohol+marijuana,
                           data=DaytonSurvey, FUN=sum)
Dayton.ACM.df

##   cigarette alcohol marijuana Freq
## 1       Yes      Yes        Yes  911
## 2        No      Yes        Yes   44
## 3       Yes       No        Yes    3
## 4        No       No        Yes    2
## 5       Yes      Yes         No  538
## 6        No      Yes         No  456
## 7       Yes       No         No   43
## 8        No       No         No  279
```

△

When the data are in table form, and we want to produce another table, `apply()` with `FUN=sum` can be used in a similar way to sum the table over dimensions not mentioned in the `MARGIN` argument. `margin.table()` is just a wrapper for `apply()` using the `sum()` function.

EXAMPLE 2.6: Dayton survey

To illustrate, we first convert the `DaytonSurvey` to a 5-way table using `xtabs()`, giving `Dayton.tab`.

```
# convert to table form
Dayton.tab <- xtabs(Freq~cigarette+alcohol+marijuana+sex+race,
                    data=DaytonSurvey)
structable(cigarette+alcohol+marijuana ~ sex+race, data=Dayton.tab)

##              cigarette Yes                 No
##              alcohol   Yes       No      Yes       No
##              marijuana Yes  No  Yes  No  Yes  No  Yes  No
## sex    race
## female white            405 268    1  17   13 218    1 117
##        other             23  23    0   1    2  19    0  12
## male   white            453 228    1  17   28 201    1 133
##        other             30  19    1   8    1  18    0  17
```

Then, use `apply()` on `Dayton.tab` to give the 3-way table `Dayton.ACM.tab` summed over sex and race. The elements in this new table are the column sums for `Dayton.tab` shown by `structable()` just above.

```
# collapse over sex and race
Dayton.ACM.tab <- apply(Dayton.tab, MARGIN=1:3, FUN=sum)
Dayton.ACM.tab <- margin.table(Dayton.tab, 1:3)    # same result
structable(cigarette+alcohol ~ marijuana, data=Dayton.ACM.tab)

##           cigarette Yes       No
##           alcohol   Yes  No Yes  No
## marijuana
## Yes                 911   3  44   2
## No                  538  43 456 279
```

△

Many of these operations can be performed using the `**ply()` functions in the plyr package. For example, with the data in a frequency form data frame, use `ddply()` to collapse over unmentioned factors, and `plyr::summarise()`[5] as the function to be applied to each piece.

[5]Ugh. This plyr function clashes with a function of the same name in vcdExtra. In this book I will use the explicit double-colon notation to keep them separate.

```
Dayton.ACM.df <- ddply(DaytonSurvey, .(cigarette, alcohol, marijuana),
                       plyr::summarise, Freq=sum(Freq))
```

2.7 Converting among frequency tables and data frames

As we've seen, a given contingency table can be represented equivalently in case form, frequency form and table form. However, some R functions were designed for one particular representation. Table 2.2 shows some handy tools for converting from one form to another.

Table 2.2: Tools for converting among different forms for categorical data

	To this		
From this	Case form	Frequency form	Table form
Case form		`Z <- xtabs( A+B)` `as.data.frame(Z)`	`table(A,B)`
Frequency form	`expand.dft(X)`		`xtabs(count~A+B)`
Table form	`expand.dft(X)`	`as.data.frame(X)`	

2.7.1 Table form to frequency form

A contingency table in table form (an object of class "table") can be converted to a data frame in frequency form with `as.data.frame()`.[6] The resulting data frame contains columns representing the classifying factors and the table entries (as a column named by the `responseName` argument, defaulting to `Freq`. The function `as.data.frame()` is the inverse of `xtabs()`, which converts a data frame to a table.

Example 2.7: General social survey

Convert the `GSStab` in table form to a data.frame in frequency form. By default, the frequency variable is named `Freq`, and the variables `sex` and `party` are made factors.

```
as.data.frame(GSStab)

##      sex party Freq
## 1 female   dem  279
## 2   male   dem  165
## 3 female indep   73
## 4   male indep   47
## 5 female   rep  225
## 6   male   rep  191
```

△

In addition, there are situations where numeric table variables are represented as factors, but you need to convert them to numerics for calculation purposes.

[6]Because R is object-oriented, this is actually a short-hand for the function `as.data.frame.table()`.

Example 2.8: Horse kicks data

For example, We might want to calculate the weighted mean of `nDeaths` in the `HorseKicks` data. Using `as.data.frame()` won't work here, because the variable `nDeaths` becomes a factor.

```
str(as.data.frame(HorseKicks))

## 'data.frame': 5 obs. of  2 variables:
##  $ nDeaths: Factor w/ 5 levels "0","1","2","3",..: 1 2 3 4 5
##  $ Freq   : int  109 65 22 3 1
```

One solution is to use `data.frame()` directly and `as.numeric()` to coerce the table names to numbers.

```
horse.df <- data.frame(nDeaths = as.numeric(names(HorseKicks)),
                       Freq = as.vector(HorseKicks))
str(horse.df)

## 'data.frame': 5 obs. of  2 variables:
##  $ nDeaths: num  0 1 2 3 4
##  $ Freq   : int  109 65 22 3 1

horse.df

##   nDeaths Freq
## 1       0  109
## 2       1   65
## 3       2   22
## 4       3    3
## 5       4    1
```

Then, `weighted.mean()` works as we would like:

```
weighted.mean(horse.df$nDeaths, weights=horse.df$Freq)

## [1] 2
```

△

2.7.2 Case form to table form

Going the other way, we use `table()` to convert from case form to table form.

Example 2.9: Arthritis treatment

Convert the `Arthritis` data in case form to a 3-way table of `Treatment` × `Sex` × `Improved`. We select the desired columns with their names, but could also use column numbers, e.g., `table(Arthritis[,c(2,3,5)])`.

```
Art.tab <- table(Arthritis[,c("Treatment", "Sex", "Improved")])
str(Art.tab)
```

```
##  'table' int [1:2, 1:2, 1:3] 19 6 10 7 7 5 0 2 6 16 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Treatment: chr [1:2] "Placebo" "Treated"
##   ..$ Sex      : chr [1:2] "Female" "Male"
##   ..$ Improved : chr [1:3] "None" "Some" "Marked"

ftable(Art.tab)

##                  Improved None Some Marked
## Treatment Sex
## Placebo   Female            19    7      6
##           Male              10    0      1
## Treated   Female             6    5     16
##           Male               7    2      5
```

△

2.7.3 Table form to case form

There may also be times that you will need an equivalent case form data frame with factors representing the table variables rather than the frequency table. For example, the `mca()` function in package MASS only operates on data in this format. The function `expand.dft()`[7] in vcdExtra does this, converting a table into a case form.

EXAMPLE 2.10: Arthritis treatment

Convert the `Arthritis` data in table form (`Art.tab`) back to a `data.frame` in case form, with factors `Treatment`, `Sex` and `Improved`.

```
Art.df <- expand.dft(Art.tab)
str(Art.df)

## 'data.frame': 84 obs. of  3 variables:
##  $ Treatment: Factor w/ 2 levels "Placebo","Treated": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Sex      : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Improved : Factor w/ 3 levels "Marked","None",..: 2 2 2 2 2 2 2 2 2 2 ...
```

△

2.8 A complex example

If you've followed so far, congratulations! You're ready for a more complicated example that puts together a variety of the skills developed in this chapter: (a) reading raw data, (b) creating tables, (c) assigning level names to factors and (d) collaping levels or variables for use in analysis.

For illustration of these steps, we use the dataset `tv.dat`, supplied with the initial implementation of mosaic displays in R by Jay Emerson. In turn, they were derived from an early, compelling example of mosaic displays (Hartigan and Kleiner, 1984), that illustrated the method with data on a large sample of TV viewers whose behavior had been recorded for the Neilson ratings. This data set contains sample television audience data from Neilsen Media Research for the week starting November 6, 1995.

[7]The original code for this function was provided by Marc Schwarz on the R-Help mailing list.

The data file, `tv.dat` is stored in frequency form as a file with 825 rows and 5 columns. There is no header line in the file, so when we use `read.table()` below, the variables will be named `V1` – `V5`. This data represents a 4-way table of size $5 \times 11 \times 5 \times 3 = 825$ where the table variables are `V1` – `V4`, and the cell frequency is read as `V5`.

The table variables are:
`V1`– values 1:5 correspond to the days Monday–Friday;
`V2`– values 1:11 correspond to the quarter hour times 8:00PM through 10:30PM;
`V3`– values 1:5 correspond to ABC, CBS, NBC, Fox, and non-network choices;
`V4`– values 1:3 correspond to transition states: turn the television Off, Switch channels, or Persist in viewing the current channel.

2.8.1 Creating data frames and arrays

The file `tv.dat` is stored in the `doc/extdata` directory of vcdExtra; it can be read as follows:

```
tv.data<-read.table(system.file("doc","extdata","tv.dat",package="vcdExtra"))
str(tv.data)

## 'data.frame': 825 obs. of  5 variables:
##  $ V1: int  1 2 3 4 5 1 2 3 4 5 ...
##  $ V2: int  1 1 1 1 1 2 2 2 2 2 ...
##  $ V3: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V4: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V5: int  6 18 6 2 11 6 29 25 17 29 ...

head(tv.data,5)

##   V1 V2 V3 V4 V5
## 1  1  1  1  1  6
## 2  2  1  1  1 18
## 3  3  1  1  1  6
## 4  4  1  1  1  2
## 5  5  1  1  1 11
```

To read such data from a local file, just use `read.table()` in this form:

```
tv.data<-read.table("C:/R/data/tv.dat")
```

We could use this data in frequency form for analysis by renaming the variables, and converting the integer-coded factors `V1` – `V4` to R factors. The lines below use the function `within()` to avoid having to use `TV.dat$Day <- factor(TV.dat$Day)` etc., and only supplies labels for the first variable.

```
TV.df <- tv.data
colnames(TV.df) <- c("Day", "Time", "Network", "State", "Freq")
TV.df <- within(TV.df, {Day <- factor(Day,
                                  labels=c("Mon", "Tue", "Wed", "Thu", "Fri"))
                        Time <- factor(Time)
                        Network <- factor(Network)
                        State <- factor(State)})
```

Alternatively, we could just reshape the frequency column (`V5` or `tv.data[,5]`) into a 4-way array. In the lines below, we rely on the facts that the (a) the table is complete— there are no missing cells, so nrow(tv.data)=825; (b) the observations are ordered so that V1 varies most rapidly and V4 most slowly. From this, we can just extract the frequency column and reshape it into an array using the dim argument. The level names are assigned to dimnames(TV) and the variable names to names(dimnames(TV)).

```
TV <- array(tv.data[,5], dim=c(5,11,5,3))
dimnames(TV) <- list(c("Mon","Tue","Wed","Thu","Fri"),
                     c("8:00","8:15","8:30","8:45","9:00","9:15","9:30",
                       "9:45","10:00","10:15","10:30"),
                     c("ABC","CBS","NBC","Fox","Other"),
                     c("Off","Switch","Persist"))
names(dimnames(TV))<-c("Day", "Time", "Network", "State")
```

More generally (even if there are missing cells), we can use xtabs() (or plyr::daply()) to do the cross-tabulation, using V5 as the frequency variable. Here's how to do this same operation with xtabs():

```
TV <- xtabs(V5 ~ ., data=tv.data)
dimnames(TV) <- list(Day=c("Mon","Tue","Wed","Thu","Fri"),
                     Time=c("8:00","8:15","8:30","8:45","9:00","9:15","9:30",
                            "9:45","10:00","10:15","10:30"),
                     Network=c("ABC","CBS","NBC","Fox","Other"),
                     State=c("Off","Switch","Persist"))
```

Note that in the lines above, the variable names are assigned directly as the names of the elements in the dimnames list.

2.8.2 Subsetting and collapsing

For many purposes, the 4-way table TV is too large and awkward to work with. Among the networks, Fox and Other occur infrequently, so we will remove them. We can also cut it down to a 3-way table by considering only viewers who persist with the current station.[8]

```
TV <- TV[,,1:3,]      # keep only ABC, CBS, NBC
TV <- TV[,,,3]        # keep only Persist -- now a 3 way table
structable(TV)

##              Time 8:00 8:15 8:30 8:45 9:00 9:15 9:30 9:45 10:00 10:15 10:30
## Day Network
## Mon ABC            146  151  156   83  325  350  386  340   352   280   278
##     CBS            337  293  304  233  311  251  241  164   252   265   272
##     NBC            263  219  236  140  226  235  239  246   279   263   283
## Tue ABC            244  181  231  205  385  283  345  192   329   351   364
##     CBS            173  180  184  109  218  235  256  250   274   263   261
##     NBC            315  254  280  241  370  214  195  111   188   190   210
## Wed ABC            233  161  194  156  339  264  279  140   237   228   203
##     CBS            158  126  207   59   98  103  122   86   109   105   110
##     NBC            134  146  166   66  194  230  264  143   274   289   306
## Thu ABC            174  183  197  181  187  198  211   86   110   122   117
##     CBS            196  185  195  104  106  116  116   47   102    84    84
##     NBC            515  463  472  477  590  473  446  349   649   705   747
## Fri ABC            294  281  305  239  278  246  245  138   246   232   233
##     CBS            130  144  154   81  129  153  136  126   138   136   152
##     NBC            195  220  248  160  172  164  169   85   183   198   204
```

[8]This relies on the fact that that indexing an array drops dimensions of length 1 by default, using the argument drop=TRUE; the result is coerced to the lowest possible dimension.

Finally, for some purposes, we might also want to collapse the 11 times into a smaller number. Here, we use `as.data.frame.table()` to convert the table back to a data frame, `levels()` to re-assign the values of `Time`, and finally, `xtabs()` to give a new, collapsed frequency table.

```
TV.df <- as.data.frame.table(TV)
levels(TV.df$Time) <- c(rep("8:00-8:59",4),
                        rep("9:00-9:59",4), rep("10:00-10:44",3))
TV2 <- xtabs(Freq ~ Day + Time + Network, TV.df)
structable(Day ~ Time+Network,TV2)

##                       Day  Mon  Tue  Wed  Thu  Fri
## Time        Network
## 8:00-8:59   ABC             536  861  744  735 1119
##             CBS            1167  646  550  680  509
##             NBC             858 1090  512 1927  823
## 9:00-9:59   ABC            1401 1205 1022  682  907
##             CBS             967  959  409  385  544
##             NBC             946  890  831 1858  590
## 10:00-10:44 ABC             910 1044  668  349  711
##             CBS             789  798  324  270  426
##             NBC             825  588  869 2101  585
```

Congratulations! If you followed the operations described above, you are ready for the material described in the rest of the book. If not, try working through some of exercises below.

As a final step and a prelude to what follows, we construct a mosaic plot, below (Figure 2.2) that focuses on the associations between the combinations of `Day` and `Time` and the `Network` viewed. In terms of a loglinear model, this is represented by the model formula `~Day:Time + Network`, which asserts that `Network` is independent of the `Day:Time` combinations.

```
dimnames(TV2)$Time <- c("8", "9", "10")     # re-level for mosaic display
mosaic(~ Day + Network + Time, data=TV2, expected=~Day:Time + Network,
         legend=FALSE, gp=shading_Friendly)
```

The cells shaded in blue show positive associations (observed frequency $>$ expected) and red shows negative associations. From this it is easy to read how network choice varies with day and time. For example, CBS dominates in all time slots on Monday; ABC and NBC dominate on Tuesday, particularly in the later time slots; Thursday is an NBC day, while on Friday, ABC gets the greatest share.

2.9 Lab exercises

1. The packages vcd and vcdExtra contain many data sets with some examples of analysis and graphical display. The goal of this exercise is to familiarize yourself with these resources.

   You can get a brief summary of these using the function `datasets()`. Use the following to get a list of these with some characteristics and titles.

```
ds <- datasets(package=c("vcd", "vcdExtra"))
str(ds)
```

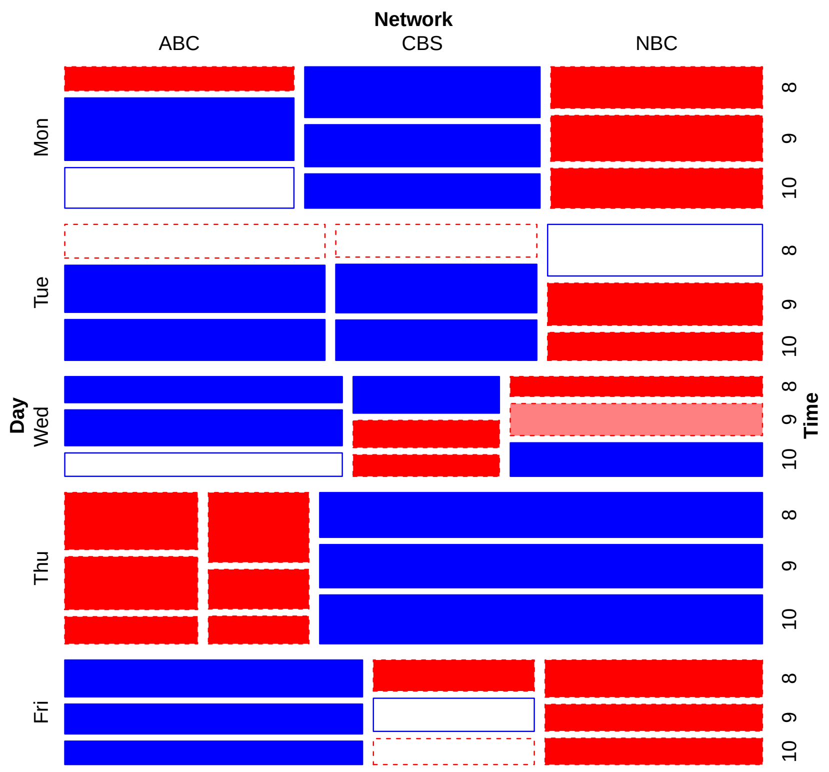


Figure 2.2: Mosaic plot for the TV data showing model of joint independence, `Day:Time + Network`

```
## 'data.frame': 63 obs. of  5 variables:
##  $ Package: chr  "vcd" "vcd" "vcd" "vcd" ...
##  $ Item   : chr  "Arthritis" "Baseball" "BrokenMarriage" "Bundesliga" ...
##  $ class  : chr  "data.frame" "data.frame" "data.frame" "data.frame" ...
##  $ dim    : chr  "84x5" "322x25" "20x4" "14018x7" ...
##  $ Title  : chr  "Arthritis Treatment Data" "Baseball Data" "Broken Marriage Data" '
```

(a) How many data sets are there altogether? How many are there in each package?
(b) Make a tabular display of the frequencies by `Package` and `class`.
(c) Choose one or two data sets from this list, and examine their help files (e.g., `help(Arthritis)` or `?Arthritis`). You can use, e.g., `example(Arthritis)` to run the R code for a given example.

2. The data set `UCBADdmissions` is a 3-way table of frequencies classified by *`Admit`*, *`Gender`* and *`Dept`*.

(a) Find the total number of cases contained in this table.
(b) For each department, find the total number of applicants.
(c) For each department, find the overall proportion of applicants who were admitted.
(d) Construct a tabular display of department (rows) and gender (columns), showing the proportion of applicants in each cell who were admitted.

3. The data set `DanishWelfare` in vcd gives a 4-way, $3 \times 4 \times 3 \times 5$ table as a data frame in frequency form, containing the variable *`Freq`* and four factors, *`Alcohol`*, *`Income`*, *`Status`* and *`Urban`*. The variable *`Alcohol`* can be considered as the response variable, and the others as possible predictors.

(a) Find the total number of cases represented in this table.
(b) In this form, the variables *`Alcohol`* and *`Income`* should arguably be considered *ordered* factors. Change them to make them ordered.
(c) Convert this data frame to table form, `DanishWelfare.tab`, a 4-way array containing the frequencies with appropriate variable names and level names.
(d) The variable *`Urban`* has 5 categories. Find the total frequencies in each of these. How would you collapse the table to have have only two categories, `City`, `Non-city`?
(e) Use `structable()` or `ftable()` to produce a pleasing flattened display of the frequencyes in the 4-way table. Choose the variables used as row and column variables to make it easier to compare levels of *`Alcohol`* across the other factors.

4. The data set `UKSoccer` in vcd gives the distributions of number of goals scored by the 20 teams in the 1995/96 season of the Premier League of the UK Football Association.

```
data(UKSoccer, package="vcd")
ftable(UKSoccer)

##      Away  0  1  2  3  4
## Home
## 0         27 29 10  8  2
## 1         59 53 14 12  4
## 2         28 32 14 12  4
## 3         19 14  7  4  1
## 4          7  8 10  2  0
```

This two-way table classifies all $20 \times 19 = 380$ games by the joint outcome (Home, Away), the number of goals scored by the Home and Away teams. The value 4 in this table actually represents 4 or more goals.

(a) Verify that the total number of games represented in this table is 380.
(b) Find the marginal total of the number of goals scored by each of the home and away teams.
(c) Express each of the marginal totals as proportions.
(d) Comment on the distribution of the numbers of home-team and away-team goals. Is there any evidence that home teams score more goals on average?

5. The one-way frequency table, Saxony in vcd records the frequencies of families with 0, 1, 2, ... 12 male children, among 6115 families with 12 children. This data set is used extensively in Chapter 3.

```
data(Saxony, package="vcd")
Saxony

## nMales
##    0    1    2    3    4    5    6    7    8    9   10   11
##    3   24  104  286  670 1033 1343 1112  829  478  181   45
##   12
##    7
```

Another data set, Geissler in the vcdExtra package, gives the complete tabulation of all combinations of boys and girls in families with a given total number of children size. The task here is to create an equivalent table, Saxony12 from the Geissler data.

```
data(Geissler, package="vcdExtra")
str(Geissler)

## 'data.frame': 90 obs. of  4 variables:
##  $ boys : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ girls: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ size : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Freq : int  108719 42860 17395 7004 2839 1096 436 161 66 30 ...
```

(a) Use subset() to create a data frame, sax12 containing the Geissler observations in families with size==12.
(b) Select the columns for boys and Freq.
(c) Use xtabs() with a formula, Freq ~ boys, to create the one-way table.
(d) Do the same steps again, to create a one-way table, Saxony11 containing similar frequencies for families of size==11.

ToDo: Cleanup some local variables

```
remove(list=objects(pattern="array|mat|my|\\.tab|\\.df"))
remove(list=c("A", "B", "age", "count", "ds", "n", "passed", "sex", "tab", "tv.data", "TV2"
ls()
```

Chapter 3
Fitting and graphing discrete distributions

Discrete data often follow various theoretical probability models. Graphic displays are used to visualize goodness of fit, to diagnose an appropriate model, and determine the impact of individual observations on estimated parameters.

Not everything that counts can be counted, and not everything that can be counted counts.

Albert Einstein

Discrete frequency distributions often involve counts of occurrences of events, such as accident fatalities, incidents of terorism or suicide, words in passages of text, or blood cells with some characteristic. Often interest is focussed on how closely such data follow a particular probabiliy distribution, such as the binomial, Poisson, or geometric distribution, which provide the basis for generating mechanisms that might give rise to the data. Understanding and visualizing such distributions in the simplest case of an unstructured sample provides a building block for generalized linear models (Chapter ?) where they serve as one component. The also provide the basis for a variety of recent extensions of regression models for count data (Chapter ?), allowing excess counts of zeros (zero-inflated models), left- or right- truncation often encountered in statistical practice.

This chapter describes the well-known discrete frequency distributions: the binomial, Poisson, negative binomial, geometric, and logarithmic series distributions in the simplest case of an unstructured sample. The chapter begins with simple graphical displays (line graphs and bar charts) to view the distributions of empirical data and theoretical frequencies from a specified discrete distribution.

It then describes methods for fitting data to a distribution of a given form and simple, effective graphical methods than can be used used to visualize goodness of fit, to diagnose an appropriate model (e.g., does a given data set follow the Poisson or negative binomial?) and determine the impact of individual observations on estimated parameters.

3.1 Introduction to discrete distributions

Discrete data analysis is concerned with the study of the tabulation of one or more types of events, often categorized into mutually exclusive and exhaustive categories. ***Binary events*** having two outcome categories include the toss of a coin (head/tails), sex of a child (male/female), survival of a patient following surgery (lived/died), and so forth. ***Polytomous events*** have more outcome categories, which may be *ordered* (rating of impairment: low/medium/high, by a physician) and possibly numerically-valued (number of dots (pips), 1–6 on the toss of a die) or *unordered* (political party supported: Liberal, Conservative, Greens, Socialist).

In this chapter, we focus largely on one-way frequency tables for a single numerically-valued variable. Probability models for such data provide the opportunity to describe or explain the *structure* in such data, in that they entail some data generating mechanism and provide the basis for testing scientific hypotheses, prediction of future results. If a given probability model does not fit the data, this can often be a further opportunity to extend understanding of the data or the underlying substantitive theory or both.

The remainder of this section gives a few substantive examples of situations where the well-known discrete frequency distributions (binomial, Poisson, negative binomial, geometric, and logarithmic series) might reasonably apply, at least approximately. The mathematical characteristics and properties of these theoretical distributions are postponed to Section 3.2.

In many cases, the data at hand pertain to two types of variables in a one-way frequency table. There is a basic outcome variable, k, taking integer values, $k = 0, 1, \ldots$, and called a ***count***. For each value of k, we also have a ***frequency***, n_k that the count k was observed in some sample. For example, in the study of children in families, the count variable k could be the total number of children or the number of male children; the frequency variable, n_k, would then give the number of families with that basic count k.

3.1.1 Binomial data

Binomial type data arise as the discrete distribution of the number of "success" events in n independent binary trials, each of which yields a success (yes/no, head/tail, lives/dies, male/female) with a constant probability p.

Sometimes, as in Example 3.1 below, the avaiable data record only the number of successes in n trials, with separate such observations recorded over time or space. More commonly, as in Example 3.2 and Example 3.3, we have available data on the frequency n_k of $k = 0, 1, 2, \ldots n$ successes in the n trials.

EXAMPLE 3.1: Arbuthnot data

Sex ratios— births of male to female children have long been of interest in population studies and demography. Indeed, in 1710, John Arbuthnot (Arbuthnot, 1710) used data on the ratios of male to female christenings in London from 1629–1710 to carry out the first known significance test. The data for these 82 years showed that in *every* year there were more boys than girls. He calculated that the under the assumption that male and female births were equally likely, the probablity of 82 years of more males than females was vanishingly small, ($\Pr \approx 4.14 \times 10^{-25}$). He used this to argue that a nearly constant birth ratio > 1 (or $\Pr(\text{Male}) > 0.5$) could be interpreted to show the guiding hand of a devine being.

Arbuthnot's data, along with some other related variables are available in `Arbuthnot` in the HistData package. For now, we simply display a plot of the probabilty of a male birth over time. The plot in Figure 3.1 shows the proportion of males over years, with horizontal lines at $\Pr(\text{Male}) = 0.5$ and the mean, $\Pr(\text{Male}) = 0.517$. Also shown is a (loess) smoothed curve, which suggests that any deviation from a constant sex ratio is relatively small.

```
spar()
data(Arbuthnot, package="HistData")
with(Arbuthnot, {
  prob = Males/(Males+Females)
  plot(Year, prob, type='b', ylim=c(0.5, 0.54), ylab="Pr (Male)")
  abline(h=0.5, col="red", lwd=2)
  abline(h=mean(prob), col="blue")
  text(x=1640, y=0.5, expression(H[0]: "Pr(Male)=0.5"), pos=3, col="red")
  Arb.smooth <- with(Arbuthnot, loess.smooth(Year,prob))
  lines(Arb.smooth$x, Arb.smooth$y, col="blue", lwd=2)
  })
```

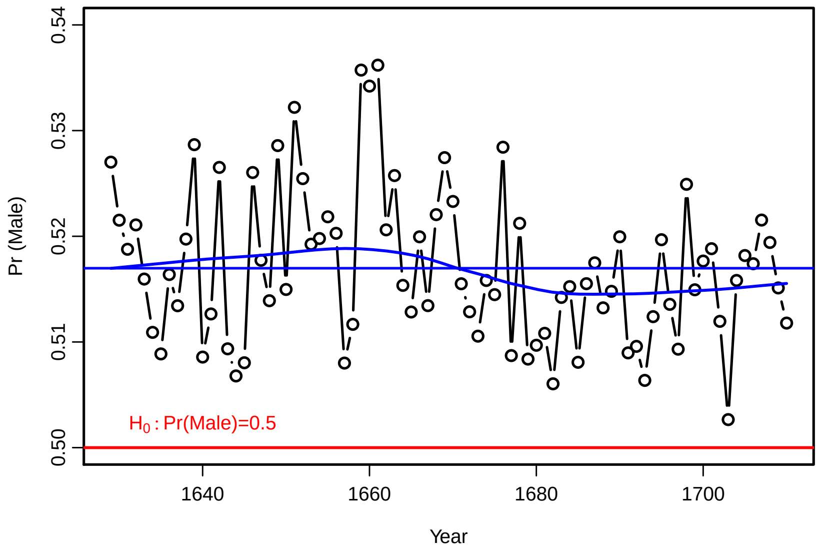


Figure 3.1: Arbuthnot's data on male/female sex ratios in London, 1629–1710, together with a (loess) smoothed curve over time and the mean Pr(Male)

We return to this data in a later chapter where we ask whether the variation around the mean can be explained by any other considerations, or should just be considered random variation. △

EXAMPLE 3.2: Families in Saxony

A related example of sex ratio data that ought to follow a binomial distribution comes from a classic study by A. Geissler (1889). Geissler listed the data on the distributions of boys and girls in families in Saxony for the period 1876–1885. In total, over four million births were recorded, and the sex distribution in the family was available because the parents had to state the the sex of all their children on the birth certificate.

The complete data, classified by number of boys and number of girls (each 0–12) appear in Edwards (1958, Table 1).[1] Lindsey (1995, Table 6.2) selected only the 6115 families with 12

[1]Edwards (1958) notes that over these 10 years, many parents will have had several children, and their family composition is therefore recorded more than once. However, in families with a given number of children, each family can appear only once.

children, and listed the frequencies by number of males. The data are shown in table form in Table 3.1 in the standard form of a complete discrete distribution. The basic outcome variable, $k = 0, 1, \ldots, 12$, is the number of male children in a family and the frequency variable, n_k is the number of families with that number of boys.

Table 3.1: Number of male children in 6115 Saxony families of size 12

Males (k)	0	1	2	3	4	5	6	7	8	9	10	11	12	Sum
Families (n_k)	3	24	104	286	670	1033	1343	1112	829	478	181	45	7	6115

Figure 3.2 shows a bar plot of the frequencies in Table 3.1. It can be seen that the distribution is quite symmetric. The questions of interest here are: (a) how close does the data follow a binomial distribution, with a constant $\Pr(\text{Male}) = p$? (b) is there evidence to reject the hypothesis that $p = 0.5$?

```
spar()
data(Saxony, package="vcd")
barplot(Saxony, xlab="Number of males", ylab="Number of families",
        col="lightblue", cex.lab=1.5)
```

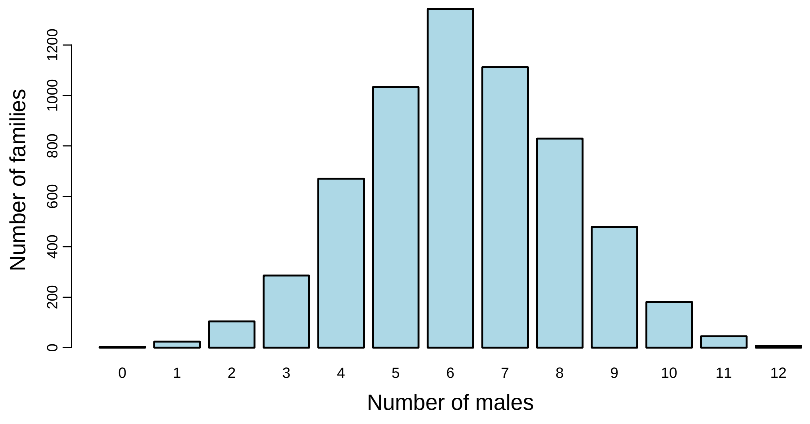


Figure 3.2: Males in Saxony families of size 12

△

Example 3.3: Weldon's dice

Common examples of binomial distributions involve tossing coins or dice, where some event outcome is considered a "success" and the number of successes (k) are are tabulated in a long series of trials to give the frequency (n_k) of each basic count, k.

Perhaps the most industrious dice-tosser of all times, W. F. Raphael Weldon, an English evolutionary biologist and joint founding editor of *Biometrika* (with Francis Galton and Karl Pearson) tallied the results of throwing 12 dice 26,306 times. For his purposes, he considered the outcome of 5 or 6 pips showing on each die to be a success to be a success, and all other outcomes as failures.

Weldon reported his results in a letter to Francis Galton dated February 2, 1894, in order "to judge whether the differences between a series of group frequencies and a theoretical law

…were more than might be attributed to the chance fluctuations of random sampling" (Kemp and Kemp, 1991). In his seminal paper, Pearson (1900) used Weldon's data to illustrate the χ^2 goodness-of-fit test, as did Kendall and Stuart (1963, Table 5.1, p. 121).

These data are shown here as Table 3.2, in terms of the number of occurrences of a 5 or 6 in the throw of 12 dice. If the dice were all identical and perfectly fair (balanced), one would expect that $p = \Pr\{5 \text{ or } 6\} = \frac{1}{3}$ and the distribution of the number of 5 or 6 would be binomial.

A peculiar feature of these data as presented by Kendall and Stuart (not uncommon in discrete distributions) is that the frequencies of 10–12 successes are lumped together.[2] This grouping must be taken into account in fitting the distribution. This dataset is available as `WeldonDice` in the vcd package. The distribution is plotted in Figure 3.3.

Table 3.2: Frequencies of 5s or 6s in throws of 12 dice

# 5s or 6s (k)	0	1	2	3	4	5	6	7	8	9	10+	Sum
Frequency (n_k)	185	1149	3265	5475	6114	5194	3067	1331	403	105	18	26306

```r
data(WeldonDice, package="vcd")
dimnames(WeldonDice)$n56[11] <- "10+"
barplot(WeldonDice, xlab="Number of 5s and 6s", ylab="Frequency",
        col="lightblue", cex.lab=1.5)
```

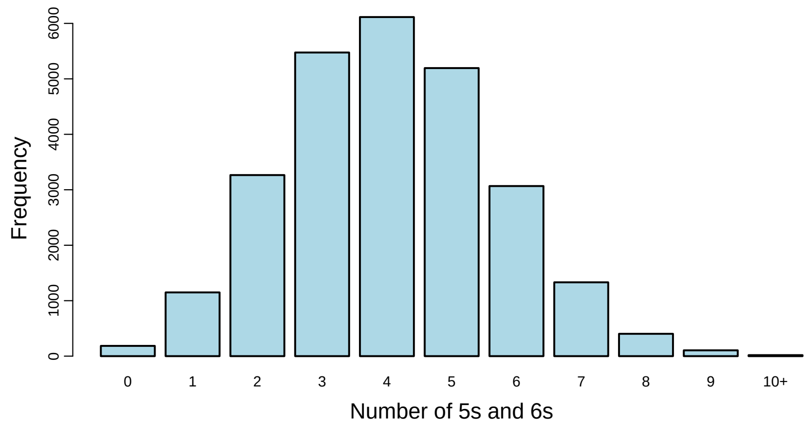


Figure 3.3: Weldon's dice data

△

3.1.2 Poisson data

Data of Poisson type arise when we observe the counts of events k within a fixed interval of time or space (length, area, volume) and tabulate their frequencies, n_k. For example, we may observe the number of radioactive particles emitted by a source per second or number of births per hour, or the number of tiger or whale sightings within some geographical regions.

[2]The unlumped entries are, for (number of 5s or 6s: frequency) — (10: 14); (11: 4), (12:0), given by Labby (2009). In this remarkable paper, Labby describes a mechanical device he constructed to repeat Weldon's experiment physically and automate the counting of outcomes. He created electronics to roll 12 dice in a physical box, and hooked that up to a webcam to capture an image of each toss and used image processing software to record the counts.

In contrast to binomial data, where the counts are bounded below and above, in Poisson data the counts k are bounded below at 0, but can take integer values with no fixed upper limit. One defining characteristic for the Poisson distribution is for rare events, which occur independently with a small and constant probability, p in small intervals, and we count the number of such occurrences.

Several examples of data of this general type are given below.

EXAMPLE 3.4: Death by horse kick

One of the oldest and best known examples of a Poisson distribution is the data from von Bortkiewicz (1898) on deaths of soldiers in the Prussian army from kicks by horses and mules, shown in Table 3.3. Ladislaus von Bortkiewicz, an economist and statistician, tabulated the number of soldiers in each of 14 army corps in the 20 years from 1875-1894 who died after being kicked by a horse (Andrews and Herzberg, 1985, p. 18). Table 3.3 shows the data used by Fisher (1925) for 10 of these army corps, summed over 20 years, giving 200 'corps-year' observations. In 109 corps-years, no deaths occurred; 65 corps-years had one death, etc.

The data set is available as `HorseKicks` in the vcd package. The distribution is plotted in Figure 3.4.

Table 3.3: von Bortkiewicz's data on deaths by horse kicks

Number of deaths (k)	0	1	2	3	4	Sum
Frequency (n_k)	109	65	22	3	1	200

```r
data(HorseKicks, package="vcd")
barplot(HorseKicks, xlab="Number of deaths", ylab="Frequency",
        col="lightblue", cex.lab=1.5)
```

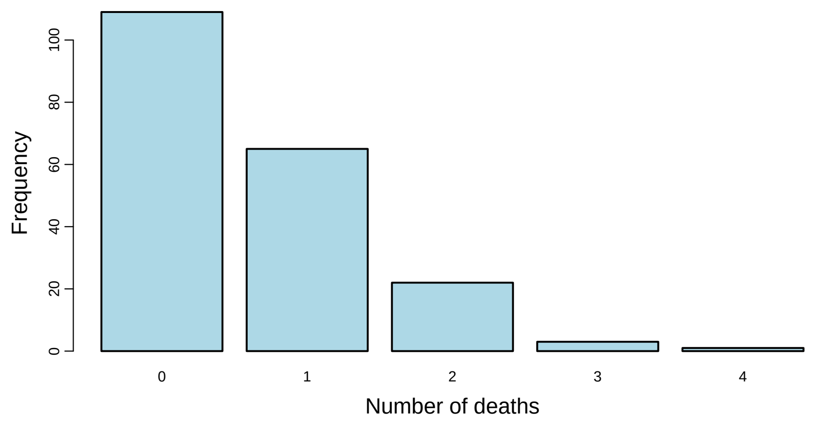


Figure 3.4: HorseKicks data

△

EXAMPLE 3.5: Federalist papers

In 1787–1788, Alexander Hamilton, John Jay, and James Madison wrote a series of newspaper essays to persuade the voters of New York State to ratify the U.S. Constitution. The essays were titled *The Federalist Papers* and all were signed with the pseudonym “Publius.” Of the 77 papers published, the author(s) of 65 are known, but *both* Hamilton and Madison later claimed sole authorship of the remaining 12. Mosteller and Wallace (1963, 1984) investigated the use of statistical methods to identify authors of disputed works based on the frequency distributions of certain key function words, and concluded that Madison had indeed authored the 12 disputed papers.[3]

Table 3.4 shows the distribution of the occurrence of one of these “marker” words, the word *may* in 262 blocks of text (each about 200 words long) from issues of the *Federalist Papers* and other essays known to be written by James Madison. Read the table as follows: in 156 blocks, the word *may* did not occur; it occurred once in 63 blocks, etc. The distribution is plotted in Figure 3.5.

Table 3.4: Number of occurrences of the word *may* in texts written by James Madison

Occurrences of *may* (k)	0	1	2	3	4	5	6	Sum
Blocks of text (n_k)	156	63	29	8	4	1	1	262

```
data(Federalist, package="vcd")
barplot(Federalist,
        xlab="Occurrences of 'may'", ylab="Number of blocks of text",
        col="lightblue", cex.lab=1.5)
```

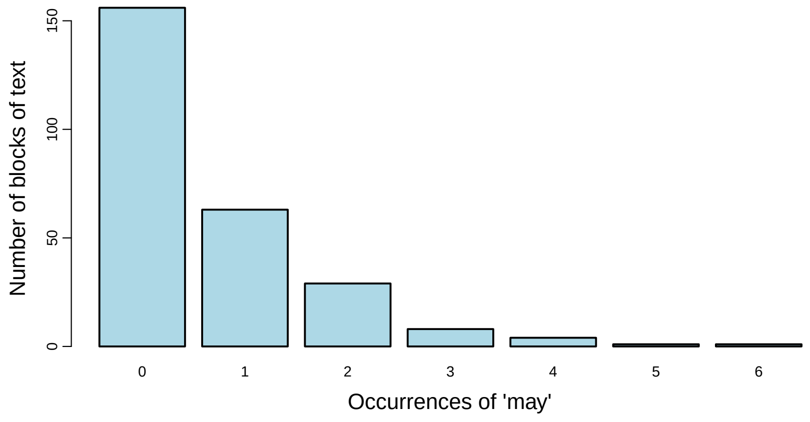


Figure 3.5: Mosteller and Wallace Federalist data

△

3.1.3 Type-token distributions

There are a variety of other types of discrete data distributions. One important class is ***type-token*** distributions, where the basic count k is the number of distinct types of some observed event, $k = 1, 2, \ldots$ and the frequency, n_k, is the number of different instances observed. For example,

[3]It should be noted that this is a landmark work in the development and application of statistical methods to the analysis of texts and cases of disputed authorthip. In addition to *may*, they considered many such marker words, such as *any*, *by*, *from*, *upon*, and so forth. Amonst these, the word *upon* was the best discriminator between the works known by Hamilton (3 per 1000 words) and Madison (1/6 per 1000 words). In this work, they pioneered the use of Bayesian discriminant analysis, and the use of cross-validation to assess the stability of estimates and their conclusions.

distinct words in a book, words that subjects list as members of the semantic category "fruit," musical notes that appear in a score, and species of animals caught in traps can be considered as types, and the occurrences of of those type comprise tokens.

This class differs from the Poisson type considered above in that the frequency for value $k = 0$ is *unobserved*. Thus, questions like (a) How many words did Shakespeare know? (b) How many words in the English language are members of the "fruit" category? (c) How many wolves remain in Canada's Northwest territories? depend on the unobserved count for $k = 0$. They cannot easily be answered without appeal to additional information or statistical theory.

EXAMPLE 3.6: **Butterfly species in Malaya**

In studies of the diversity of animal species, individuals are collected and classified by species. The distribution of the number of species (types) where $k = 1, 2, \ldots$ individuals (tokens) were collected forms a kind of type-token distribution. An early example of this kind of distribution was presented by Fisher *et al.* (1943). Table 3.5 lists the number of individuals of each of 501 species of butterfly collected in Malaya. There were thus 118 species for which just a single instance was found, 74 species for which two individuals were found, down to 3 species for which 24 individuals were collected. Fisher et-al. note however that the distribution was truncated at $k = 24$. Type-token distributions are often J-shaped, with a long upper tail, as we see in Figure 3.6.

Table 3.5: Number of butterfly species n_k for which k individuals were collected

Individuals (k)	1	2	3	4	5	6	7	8	9	10	11	12	
Species (n_k)	118	74	44	24	29	22	20	19	20	15	12	14	
Individuals (k)	13	14	15	16	17	18	19	20	21	22	23	24	Sum
Species (n_k)	6	12	6	9	9	6	10	10	11	5	3	3	501

```
data(Butterfly, package="vcd")
barplot(Butterfly, xlab="Number of individuals", ylab="Number of species",
        cex.lab=1.5)
```

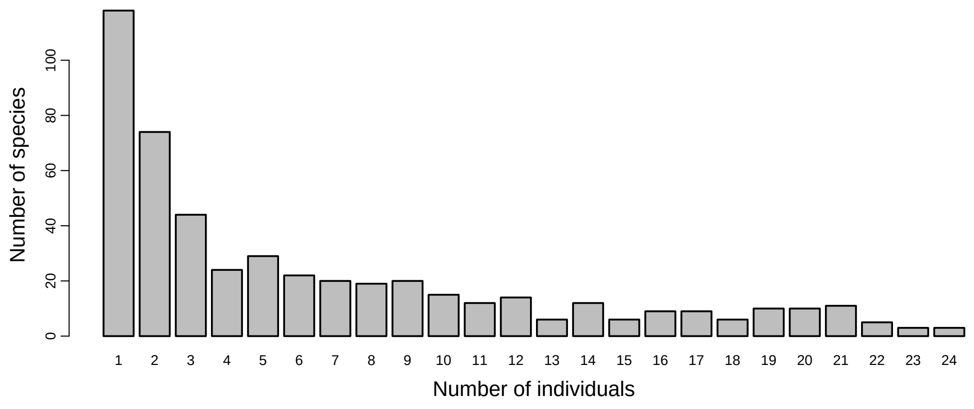


Figure 3.6: Butterfly species in Malaya

△

3.2 Characteristics of discrete distributions

This section briefly reviews the characteristics of some of the important discrete distributions encountered in practice and illustrates their use with R. An overview of these distributions is shown in Table 3.6. For more detailed information on these and other discrete distributions, Johnson *et al.* (1992) and Wimmer and Altmann (1999) present the most comprehensive treatments; Zelterman (1999, Chapter 2) gives a compact summary.

Table 3.6: Discrete probability distributions

Discrete Distribution	Probability function, $p(k)$	parameter(s)
Binomial	$\binom{n}{k} p^k (1-p)^{n-k}$	p=Pr (success); n=# trials
Poisson	$e^{-\lambda}\lambda^k/k!$	λ= mean
Negative binomial	$\binom{n+k-1}{k} p^n (1-p)^k$	p, n
Geometric	$p(1-p)^k$	p
Logarithmic series	$\theta^k/[-k\log(1-\theta)]$	θ

For each distribution, we describe properties and generating mechanisms, and show how its parameters can be estimated and how to plot the frequency distribution. R has a wealth of functions for a wide variety of distributions. For ease of reference, their names and types for the distributions covered here are shown in Table 3.7. The naming scheme is simple and easy to remember: for each distribution, there are functions, with a prefix letter, `d`, `p`, `q`, `r`, followed by the name for that class of distribution:[4]

d a density function,[5] $\Pr\{X = x\} \equiv p(x)$ for the probability that the variable X takes the value x.
p a cumulative probability function, or CDF, $F(x) = \sum_{X \leq x} p(x)$.
q a quantile function, the inverse of the CDF, $x = F^{-1}(p)$. The quantile is defined as the smallest value x such that $F(x) \geq p$.
r a random number generating function for that distribution.

In the R console, `help(Distributions)` gives an overview listing of the distribution functions available in the stats package.

Table 3.7: R functions for discrete probability distributions

Discrete distribution	Density (pmf) function	Cumulative (CDF)	Quantile CDF^{-1}	Random # generator
Binomial	`dbinom()`	`pbinom()`	`qbinom()`	`rbinom()`
Poisson	`dpois()`	`ppois()`	`qpois()`	`rpois()`
Negative binomial	`dnbinom()`	`pnbinom()`	`qnbinom()`	`rnbinom()`
Geometric	`dgeom()`	`pgeom()`	`qgeom()`	`rgeom()`

[4]The CRAN Task View on Probability Distributions, `http://cran.r-project.org/web/views/Distributions.html`, provides a general overview and lists a wide variety of contributed packages for specialized distributions, discrete and continuous.

[5]For discrete random variables this is usually called the probability mass function (pmf).

3.2.1 The binomial distribution

The binomial distribution, $\mathrm{Bin}(n, p)$, arises as the distribution of the number of events of interest which occur in n independent trials when the probability of the event on any one trial is the constant value $p = \Pr(\text{event})$. For example, if 15% of the population has red hair, the number of red-heads in randomly sampled groups of $n = 10$ might follow a binomial distribution, $\mathrm{Bin}(10, 0.15)$; in Weldon's dice data (Example 3.3), the probability of a 5 or 6 should be $\frac{1}{3}$ on any one trial, and the number of 5s or 6s in tosses of 12 dice would follow $\mathrm{Bin}(12, \frac{1}{3})$.

Over n independent trials, the number of events k may range from 0 to n; if X is a random variable with a binomial distribution, the probability that $X = k$ is given by

$$\mathrm{Bin}(n, p) : \Pr\{X = k\} \equiv p(k) = \binom{n}{k} p^k (1-p)^{n-k} \qquad k = 0, 1, \ldots, n \ , \tag{3.1}$$

where $\binom{n}{k} = n!/k!(n-k)!$ is the number of ways of choosing k out of n. The first three (central) moments of the binomial distribution are as follows (letting $q = 1 - p$),

$$\begin{aligned} \text{Mean}[X] &= np \\ \text{Var}[X] &= npq \\ \text{Skew}[X] &= npq(q-p) \ . \end{aligned}$$

It is easy to verify that the binomial distribution has its maximum variance when $p = \frac{1}{2}$. It is symmetric (Skew[X]=0) when $p = \frac{1}{2}$, and negatively (positively) skewed when $p < \frac{1}{2}$ ($p > \frac{1}{2}$).

If we are given data in the form of a discrete (binomial) distribution (and n is known), then the maximum likelihood estimator of p can be obtained as the weighted mean of the values k with weights n_k,

$$\hat{p} = \frac{\bar{x}}{n} = \frac{(\sum_k k \times n_k)/\sum_k n_k}{n} \ ,$$

and has sampling variance $Var(\hat{p}) = pq/n$.

Calculation and visualization

As indicated in Table 3.7 (but without listing the parameters of these functions), binomial probabilities can be calculated with `dbinom(x, n, p)`, where `x` is a vector of the number of successes in `n` trials and `p` is the probability of success on any one trial. Cumulative probabilities, summed up to a vector of quantiles, `Q` can be calculated with `pbinom(Q, n, p)`, and the quantiles (the smallest value x such that $F(x) \geq P$) with `pbinom(P, n, p)`. To generate `N` random observations from a binomial distribution with `n` trials and success probability `p` use `rbinom(N, n, p)`.

For example, to find and plot the binomial probabilities corresponding to Weldon's tosses of 12 dice, with $n = 0, \ldots 12$ and $p = \frac{1}{3}$, we could do the following

```r
x <- seq(0, 12)
plot(x=x, y=dbinom(x,12,1/3), type="h",
  xlab="Number of successes", ylab="Probability",
        lwd=8, lend="square")
lines(x=x, y=dbinom(x,12,1/3))
```

Note that in the call to `plot()`, `type="h"` draws histogram type lines to the bottom of the vertical axis, and `lwd=8` makes them wide. The call to `lines()` shows another way to plot the

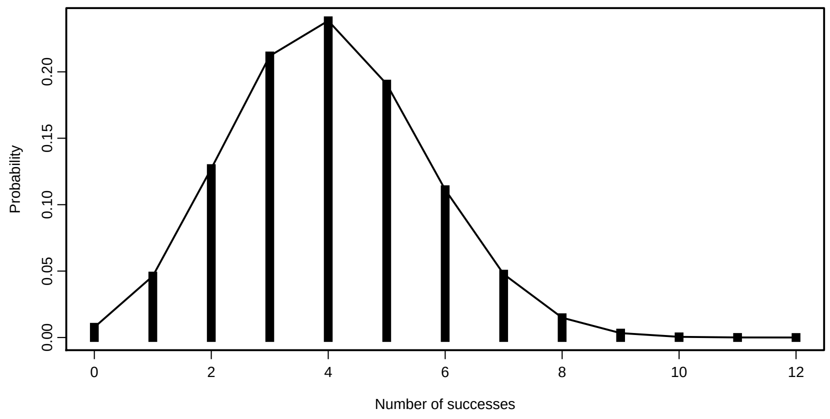


Figure 3.7: Binomial distribution for n=0–12 trials and p=1/3

data, as a probability polygon. We illustrate other styles for plotting in Section 3.2.2, Example 3.9 below.

EXAMPLE 3.7: Weldon's dice

Going a bit further, we can compare Weldon's data with the theoretical binomial distribution as shown below. Because the `WeldonDice` data collapsed the frequencies for 10–12 successes as 10+, we do the same with the binomial probabilities. The expected frequencies (`Exp`), if Weldon's dice tosses obeyed the binomial distribution are calculated as $N \times p(k)$ for $N = 26306$ tosses. The χ^2 test for goodness of fit is described later in Section 3.3, but a glance at the `Diff` column shows that these are all negative for $k = 0, \ldots 4$ and positive thereafter.

```
Weldon.df <- as.data.frame(WeldonDice)    # convert to data frame

x <- seq(0, 12)
Prob <- dbinom(x, 12, 1/3)                # binomial probabilities
Prob <- c(Prob[1:10], sum(Prob[11:13]))   # sum values for 10+
Exp= round(26306*Prob)                    # expected frequencies
Diff = Weldon.df[,"Freq"] - Exp           # raw residuals
Chisq = Diff^2 /Exp
data.frame(Weldon.df, Prob=round(Prob,5), Exp, Diff, Chisq)

##     n56 Freq    Prob  Exp Diff  Chisq
## 1     0  185 0.00771  203  -18 1.5961
## 2     1 1149 0.04624 1216  -67 3.6916
## 3     2 3265 0.12717 3345  -80 1.9133
## 4     3 5475 0.21195 5576 -101 1.8294
## 5     4 6114 0.23845 6273 -159 4.0301
## 6     5 5194 0.19076 5018  176 6.1730
## 7     6 3067 0.11127 2927  140 6.6963
## 8     7 1331 0.04769 1255   76 4.6024
## 9     8  403 0.01490  392   11 0.3087
## 10    9  105 0.00331   87   18 3.7241
## 11 10+   18 0.00054   14    4 1.1429
```

△

Finally, we can use programming features in R to calculate and plot probabilities for binomial distributions over a range of both `x` and `p` as follows, for the purposes of graphing the distributions as one or both varies. The following code uses `expand.grid()` to create a data frame `XP` containing all combinations of `x=0:12` and `p=c(1/6, 1/3, 1/2, 2/3)`. These values are then supplied as arguments to `dbinom()`. For the purpose of plotting, the decimal value of `p` is declared as a factor.

```
XP <-expand.grid(x=0:12, p=c(1/6, 1/3, 1/2, 2/3))
bin.df <- data.frame(XP, prob=dbinom(XP[,"x"], 12, XP[,"p"]))
bin.df$p <- factor(bin.df$p, labels=c("1/6", "1/3", "1/2", "2/3"))
str(bin.df)

## 'data.frame':  52 obs. of  3 variables:
##  $ x   : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ p   : Factor w/ 4 levels "1/6","1/3","1/2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ prob: num  0.1122 0.2692 0.2961 0.1974 0.0888 ...
```

This data can be plotted using `xyplot()` in lattice, using the `groups` argument to make separate curves for each value of `p`. The following code generates Figure 3.8.

```
library(lattice)
mycol <- palette()[2:5]
xyplot( prob ~ x, data=bin.df, groups=p,
  xlab=list('Number of successes', cex=1.25),
  ylab=list('Probability',  cex=1.25),
  type='b', pch=15:17, lwd=2, cex=1.25, col=mycol,
  key = list(
    title = 'Pr(success)',
    points = list(pch=15:17, col=mycol, cex=1.25),
    lines = list(lwd=2, col=mycol),
    text = list(levels(bin.df$p)),
    x=0.9, y=0.98, corner=c(x=1, y=1)
    )
  )
```

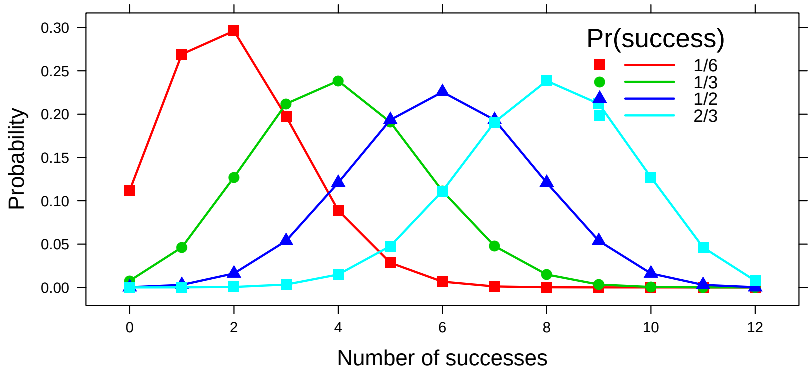


Figure 3.8: Binomial distributions for n=0–12 trials and four values of p

3.2.2 The Poisson distribution

The Poisson distribution gives the probability of an event occurring $k = 0, 1, 2, \ldots$ times over a large number of independent "trials", when the probability, p, that the event occurs on any one trial (in time or space) is small and constant. Hence, the Poisson distribution is usually applied to the study of rare events such as highway accidents at a particular location, deaths from horse kicks, or defects in a well-controlled manufacturing process. Other applications include: the number of customers contacting a call center per unit time; the number of insurance claims per unit region or unit time; number of particles emitted from a small radioactive sample.

For the Poisson distribution, the probability function is

$$\text{Pois}(\lambda) : \Pr\{X = k\} \equiv p(k) = \frac{e^{-\lambda}\,\lambda^k}{k!} \qquad k = 0, 1, \ldots \tag{3.2}$$

where the rate parameter, λ (> 0) turns out to be the mean of the distribution. The first three (central) moments of the Poisson distribution are:

$$\begin{aligned} \text{Mean}[X] &= \lambda \\ \text{Var}[X] &= \lambda \\ \text{Skew}[X] &= \lambda^{-1/2} \end{aligned}$$

So, the mean and variance of the Poisson distribution are always the same, which is sometimes used to identify a distribution as Poisson. For the binomial distribution, the mean (Np) is always greater than the variance (Npq); for other distributions (negative binomial and geometric) the mean is less than the variance. The Poisson distribution is always positvely skewed, but skewness decreases as λ increases.

The maximum likelihood estimator of the parameter λ in Eqn. (3.2) is just the mean of the distribution,

$$\hat{\lambda} = \bar{x} = \frac{\sum_k k\, n_k}{\sum_k n_k} \ . \tag{3.3}$$

Hence, the expected frequencies can be estimated by substituting the sample mean into Eqn. (3.2) and multiplying by the total sample size N.

There are many useful properties of the Poisson distribution.[6] Among these:

- Poisson variables have a nice reproductive property: if $X_1, X_2, \ldots X_m$ are independent Poisson variables with the same parameter λ, then their sum, $\sum X_i$ is a Poisson variate with parameter $m\lambda$; if the Poisson parameters differ, the sum is still Poisson with parameter $\sum \lambda_i$.
- For two or more independent Poisson variables, $X_1 \sim \text{Pois}(\lambda_1)$,X$_2$ $\sim \text{Pois}(\lambda_2), \ldots$, with rate parameters $\lambda_1, \lambda_2 \ldots$, the distribution of any X_i *conditional on their sum*, $\sum_j X_j = k$ is binomial, $\text{Bin}(k, p)$, where $p = \lambda_i / \sum_j \lambda_j$.
- As λ increases, the Poisson distribution becomes increasingly symmetric, and approaches the normal distribution $N(\lambda, \lambda)$ with mean and variance λ as $\lambda \to \infty$. The approximation is quite good with $\lambda > 20$.
- If $X \sim \text{Pois}(\lambda)$, then $\sqrt{X}$ converges much faster to a normal distribution $N(\lambda, \frac{1}{4})$, with mean $\sqrt{\lambda}$ and constant variance $\frac{1}{4}$. Hence, the sqrt transformation is often recommended as a *variance stabilizing* transformation for count data when classical methods (ANOVA, regression) assuming normality are employed.

EXAMPLE 3.8: UK Soccer scores

Table 3.8 gives the distributions of goals scored by the 20 teams in the 1995/96 season of the Premier League of the UK Football Association as presented originally by Lee (1997), and now

Table 3.8: Goals scored by home and away teams in 380 games in the Premier Football League, 1995/96 season

Home Team Goals	Away Team Goals 0	1	2	3	4+	Total
0	27	29	10	8	2	76
1	59	53	14	12	4	142
2	28	32	14	12	4	90
3	19	14	7	4	1	45
4+	7	8	10	2	0	27
Total	140	136	55	38	11	380

available as the two-way table `UKSoccer` in the **vcd** package. Over a season each team plays each other team exactly once, so there are a total of $20 \times 19 = 380$ games. Because there may be an advantage for the home team, the goals scored have been classified as "home team" goals and "away team" goals in the table. Of interest for this example is whether the number of goals scores by home teams and away teams follow Poisson distributions, and how this relates to the distribution of the total number of goals scored.

If we assume that in any small interval of time there is a small, constant probability that the home team or the away team may score a goal, the distributions of the goals scored by home teams (the row totals in Table 3.8) may be modeled as Pois(λ_H) and the distribution of the goals scored by away teams (the column totals) may be modeled as Pois(λ_A).

If the number of goals scored by the home and away teams are independent[7], we would expect that the total number of goals scored in any game would be distributed as Pois($\lambda_H + \lambda_A$). These totals are shown in Table 3.9.

Table 3.9: Total goals scored in 380 games in the Premier Football League, 1995/95 season

Total goals	0	1	2	3	4	5	6	7
Number of games	27	88	91	73	49	31	18	3

As preliminary check of the distributions for the home and away goals, we can determine if the means and variances are reasonably close to each other. If so, then the total goals variable should also have a mean and variance equal to the sum of those statistics for the home and away goals.

In the R code below, we first convert the two-way frequency table `UKSoccer` to a data frame in frequency form. We use `within()` to convert *`Home`* and *`Away`* to numeric variables, and calculate *`Total`* as their sum.

[6]See: `http://en.wikipedia.org/wiki/Poisson_distribution`

[7]This question is examined visually in Chapter **??** (Example **??**) and Chapter **??** (Example **??**), where we find that the answer is "basically, yes".

```
data(UKSoccer, package="vcd")

soccer.df <- as.data.frame(UKSoccer, stringsAsFactors=FALSE)
soccer.df <- within(soccer.df,
  {
  Home <- as.numeric(Home)          # make numeric
  Away <- as.numeric(Away)          # make numeric
  Total <- Home + Away              # total goals
  })
str(soccer.df)

## 'data.frame': 25 obs. of  4 variables:
##  $ Home : num  0 1 2 3 4 0 1 2 3 4 ...
##  $ Away : num  0 0 0 0 0 1 1 1 1 1 ...
##  $ Freq : num  27 59 28 19 7 29 53 32 14 8 ...
##  $ Total: num  0 1 2 3 4 1 2 3 4 5 ...
```

To calculate the mean and variance of these variables, first expand the data frame to 380 individual observations using `expand.dft()`. Then use `apply()` over the rows to calculate the mean and variance in each column.

```
soccer.df <- expand.dft(soccer.df)   # expand to ungrouped form
apply(soccer.df, 2, FUN=function(x) c(mean=mean(x), var=var(x)))

##       Home  Away Total
## mean 1.487 1.063 2.550
## var  1.316 1.173 2.618
```

The means are all approximately equal to the corresponding variances. More to the point, the variance of the `Total` score is approximately equal to the sum of the individual variances. Note also there does appear to be an advantage for the home team, of nearly half a goal.

△

Calculation and visualization

For the Poisson distribution, you can generate probabilities using `dpois(x, lambda)` for the numbers of events in `x` with rate parameter `lambda`. As we did earlier for the binomial distribution, we can calculate these for a collection of values of `lambda` by using `expand.grid()` to create all combinations of with the values of `x` we wish to plot.

EXAMPLE 3.9: Plotting styles for discrete distributions

In this example, we illustrate some additional styles for plotting discrete distributions, using both lattice `xyplot()` and the ggplot2 package. The goal here is to visualize a collection of Poisson distributions for varying values of λ.

We first create the 63 combinations of `x=0:20` for three values of λ, `lambda=c(1, 4, 10)`, and use these columns as arguments to `dpois()`. Again, `lambda` is a numeric variable, but the plotting methods are easier if this variable is converted to a factor.

```
XL <-expand.grid(x=0:20, lambda=c(1, 4, 10))
pois.df <- data.frame(XL, prob=dpois(XL[,"x"], XL[,"lambda"]))
pois.df$lambda = factor(pois.df$lambda)
str(pois.df)
```

```
## 'data.frame': 63 obs. of  3 variables:
##  $ x     : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ lambda: Factor w/ 3 levels "1","4","10": 1 1 1 1 1 1 1 1 1 1 ...
##  $ prob  : num  0.3679 0.3679 0.1839 0.0613 0.0153 ...
```

Discrete distributions are often plotted as bar charts or in histogram-like form, as we did for the examples in Section 3.1, rather than the line-graph form used for the binomial distribution in Figure 3.8. With `xyplot()`, the plot style is controled by the `type` argument, and the code below uses `type=c("h", "p")` to get *both* histogram-like lines to the origin and points. As well, the plot formula, `prob ~ x | lambda` instructs `xyplot()` to produce a multi-panel plot, conditioned on values of `lambda`. These lines produce Figure 3.9.

```
xyplot( prob ~ x | lambda, data=pois.df,
  type=c("h", "p"), pch=16, lwd=4, cex=1.25, layout=c(3,1),
  xlab=list("Number of events (k)", cex=1.25),
  ylab=list("Probability",  cex=1.25))
```

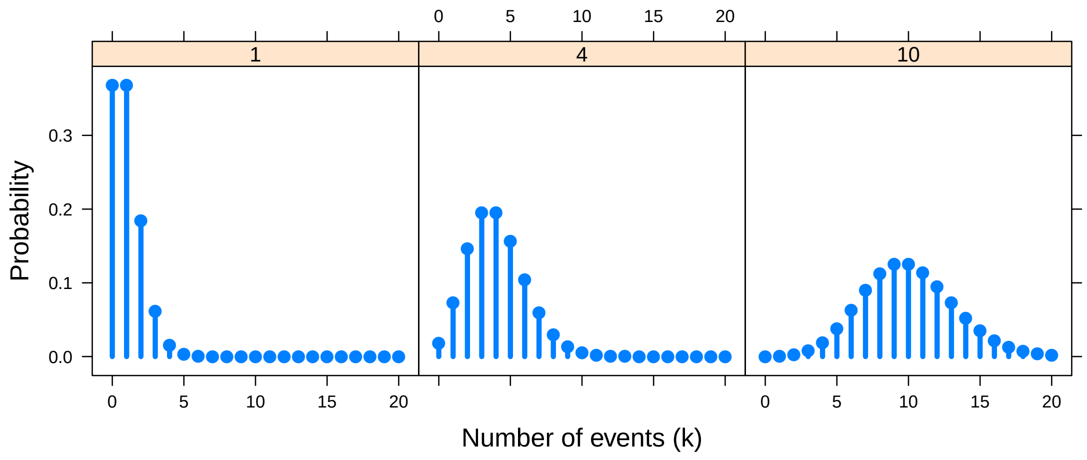


Figure 3.9: Poisson distributions for $\lambda = 1, 4, 10$, in a multi-panel display

The line-graph plot style of Figure 3.8 has the advantage that it is easier to compare the separate distributions in a single plot (using the `groups` argument) than across multiple panels (using a conditioning formula). It has the disadvantages that (a) a proper legend is difficult to contruct with lattice, and (b) is difficult to read, because you have to visually coordinate the curves in the plot with the values shown in the legend. Figure 3.10 solves both problems using the directlabels package.

```
mycol <- palette()[2:4]
plt <- xyplot( prob ~ x, data=pois.df, groups=lambda,
  type="b", pch=15:17, lwd=2, cex=1.25, col=mycol,
  xlab=list("Number of events (k)", cex=1.25),
        ylab=list("Probability",  cex=1.25))

library(directlabels)
direct.label(plt, list("top.points", cex=1.5, dl.trans(y=y+0.1)))
```

Note that the plot constructed by `xyplot()` is saved as a ("trellis") object, `plt`. The function `direct.label()` massages this to add the labels directly to each curve. In the second argument above, `"top.points"` says to locate these at the maximum value on each curve.

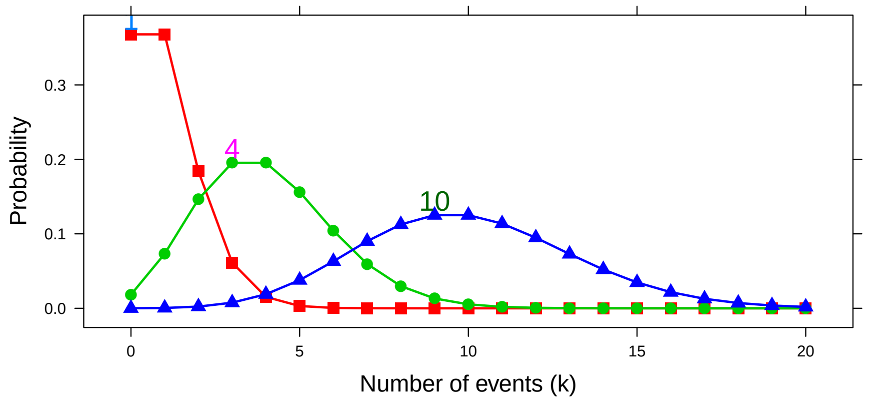


Figure 3.10: Poisson distributions for λ = 1, 4, 10, using direct labels

Finally, we illustrate the use of ggplot2 to produce a single-panel, multi-line plot of these distributions. The basic plot uses `aes(x=x, y=prob, ...)` to produce a plot of `prob` vs. `x`, assigning color and shape attributes to the values of `lambda`.

```
library(ggplot2)
gplt <- ggplot(pois.df, aes(x=x, y=prob, colour=lambda, shape=lambda)) +
  geom_line(size=1) + geom_point(size=3) +
        xlab("Number of events (k)") +
        ylab("Probability")
```

ggplot2 allows most details of the plot to be modified using `theme()`. Here we use this to move the legend inside the plot, and enlarge the axis labels and titles.

```
gplt + theme(legend.position=c(0.8,0.8)) +  # manually move legend
       theme(axis.text=element_text(size=12),
             axis.title=element_text(size=14,face="bold"))
```

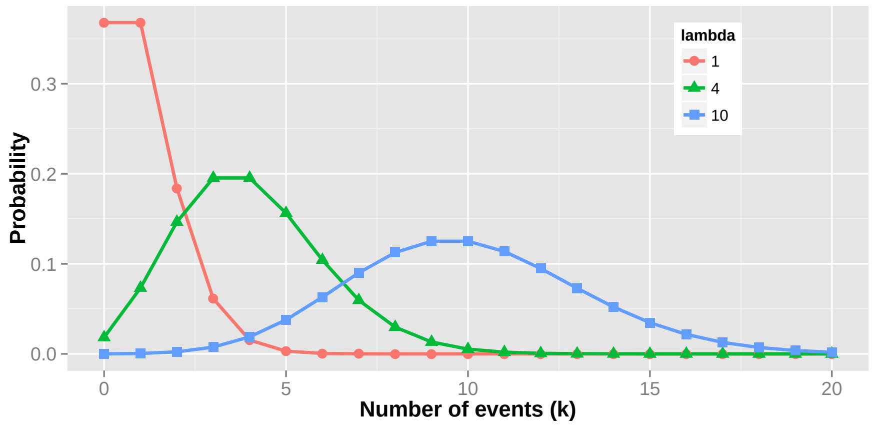


Figure 3.11: Poisson distributions for λ = 1, 4, 10, using ggplot

△

3.2.3 The negative binomial distribution

The negative binomial distribution is a type of waiting-time distribution, but also arises in statistical applications as a generalization of the Poisson distribution, allowing for ***overdispersion*** (variance > mean). See Hilbe (2011) for a comprehensive treatment of negative binomial statistical models with many applications in R.

One form of the negative binomial distribution (also called the ***Pascal distribution***) arises when a series of independent Bernoulli trials is observed with constant probability p of some event, and we ask how many non-events (failures), k, it takes to observe n successful events. For example, in tossing one die repeatedly, we may consider the outcome "1" as a "success" (with $p = \frac{1}{6}$) and ask about the probability of observing $k = 0, 1, 2, \ldots$ failures before getting $n = 3$ 1s.

The probability function with parameters n (a positive integer, $0 < n < \infty$) and p $(0 < p < 1)$ gives the probability that k non-events (failures) are observed before the n-th event (success), and can be written[8]

$$\text{NBin}(n, p) : \Pr\{X = k\} \equiv p(k) = \binom{n+k-1}{k} p^n (1-p)^k \qquad k = 0, 1, \ldots, \infty \tag{3.4}$$

This formulation makes clear that a given sequence of events involves a total of $n + k$ trials of which there are n successes, with probability p^n, and k are failures, with probability $(1 - p)^k$. The binomial coefficient, $\binom{n+k-1}{k}$ gives the number of ways to choose the k successes from the remaining $n + k - 1$ trials preceeding the last success.

The first three central moments of the negative binomial distribution are:

$$\begin{aligned} \text{Mean}[X] &= nq/p = \mu \\ \text{Var}[X] &= nq/p^2 \\ \text{Skew}[X] &= \frac{2-p}{\sqrt{nq}} \ , \end{aligned}$$

where $q = 1 - p$. The variance of X is therefore greater than the mean, and the distribution is always positively skewed.

A more general form of the negative binomial distribution (the ***Polya distribution***) allows n to take non-integer values and to be an unknown parameter. In this case, the combinatorial coefficient, $\binom{n+k-1}{k}$ in Eqn. (3.4) is calculated using the gamma function, $\Gamma(\bullet)$, a generalization of the factorial for non-integer values, defined so that $\Gamma(x + 1) = x!$ when x is an integer.

Then the probability function Eqn. (3.4) becomes

$$\Pr\{X = k\} \equiv p(k) = \frac{\Gamma(n+k)}{\Gamma(n)\Gamma(k+1)} p^n (1-p)^k \qquad k = 0, 1, \ldots, \infty \ . \tag{3.5}$$

Greenwood and Yule (1920) developed the negative binomial distribution as a model for accident proneness or susceptibility of individuals to repeated attacks of disease. They assumed that for any individual, i, the number of accidents or disease occurrences has a Poisson distribution

[8] There are a variety of other parameterizations of the negative binomial distribution, but all of these can be converted to the form shown here, which is relatively standard, and consistent with R. They differ in whether the parameter n relates to the the number of successes or the total number of trials, and whether the stopping criterion is defined in terms of failures or successes. See: `http://en.wikipedia.org/wiki/Negative_binomial_distribution` for details on these variations.

with parameter λ_i. If individuals vary in proneness, so that the λ_i have a gamma distribution, the resulting distribution is the negative binomial.

In this form, the negative binomial distribution is frequently used as an alternative to the Poisson distribution when the assumptions of the Poisson (constant probability and independence) are not satisfied, or when the variance of the distribution is greater than the mean (overdispersion). This gives rise to an alternative parameterization in terms of the mean (μ) of the distribution and its relation to the variance. From the relation of the mean and variance to the parameters n, p given above,

$$\text{Mean}[X] = \mu = \frac{n(1-p)}{p} \quad \Longrightarrow \quad p = \frac{n}{n+\mu} \tag{3.6}$$

$$\text{Var}[X] = \frac{n(1-p)}{p^2} \quad \Longrightarrow \quad \text{Var}[X] = \mu + \frac{\mu^2}{n} \tag{3.7}$$

This formulation allows the variance of the distribution to exceed the mean, and in these terms, the "size" parameter n is called the the ***dispersion parameter***.[9] Increasing this parameter corresponds to less heterogeneity, variance closer to the mean, and therefore greater applicability of the Possion distribution.

Calculation and visualization

In R, the density (pmf), distribution (CDF), quantile and random number functions for the negative binomial distribution are a bit special, in that the parameterization can be specified using either (n, p) or (n, μ) forms, where $\mu = n(1-p)/p$. In our notation, probabilities can be calculated using `dnbinom()` using the call `dbinom(k, n, p)` or the call `dbinom(k, n, mu=)`, as illustrated below:

```
k = 2
n = 2:4
p = .2
dnbinom( k, n,  p)

## [1] 0.07680 0.03072 0.01024

mu = n*(1-p)/p
mu

## [1]  8 12 16

dnbinom( k, n, mu=mu)

## [1] 0.07680 0.03072 0.01024
```

Thus, for the distribution with `k=2` failures and `n=2:4` successes with probability `p=0.2`, the values `n=2:4` correspond to means $\mu = 8, 12, 16$ as shown above.

As before, we can calculate these probabilities for a range of the combinations of arguments using `expand.grid()`. In the example below, we allow three values for each of `n` and `p` and

[9] Other terms are "shape parameter," with reference to the mixing distribution of Poissons with varying λ, "heterogeneity parameter," or "aggregation parameter."

calculate all probabilities for all values of `k` from 0 to 20. The result, `nbin.df` is like a 3-way, $21 \times 3 \times 3$ array of `prob` values, but in data frame format.

```
XN <-expand.grid(k=0:20, n=c(2, 4, 6), p=c(0.2, 0.3, 0.4))
nbin.df <- data.frame(XN, prob=dnbinom(XN[,"k"], XN[,"n"], XN[,"p"]))
nbin.df$n = factor(nbin.df$n)
nbin.df$p = factor(nbin.df$p)
str(nbin.df)

## 'data.frame': 189 obs. of  4 variables:
##  $ k   : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ n   : Factor w/ 3 levels "2","4","6": 1 1 1 1 1 1 1 1 1 1 ...
##  $ p   : Factor w/ 3 levels "0.2","0.3","0.4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ prob: num  0.04 0.064 0.0768 0.0819 0.0819 ...
```

With 9 combinations of the parameters, it is most convenient to plot these in separate panels, in a 3×3 display. The formula `prob ~ k | n + p` in the call to `xyplot()` constructs plots of `prob` vs. `k` conditioned on the combinations of `n` and `p`.

```
xyplot( prob ~ k | n + p, data=nbin.df,
  xlab=list('Number of failures (k)', cex=1.25),
  ylab=list('Probability',  cex=1.25),
  type=c('h', 'p'), pch=16, lwd=2,
  strip = strip.custom(strip.names=TRUE)
        )
```

ToDo: Modify Figure 3.12 to show the mean and standard deviation

It can be readily seen that the mean increases from left to right with `n`, and increases from top to bottom with decreasing `p`. For these distributions, we can also calculate the theory-implied means, μ, across the entire distributions, $k = 0, 1, \ldots \infty$, as shown below.

```
NP <- expand.grid(n=c(2, 4, 6), p=c(0.2, 0.3, 0.4))
NP <- within(NP, { mu = n*(1-p)/p })
# show as matrix
matrix(NP$mu, 3, 3, dimnames=list(n=c(2,4,6), p=(2:4)/10))

##    p
## n   0.2    0.3 0.4
##   2   8  4.667   3
##   4  16  9.333   6
##   6  24 14.000   9
```

3.2.4 The geometric distribution

The special case of the negative binomial distribution when $n = 1$ is a geometric distribution. We observe a series of independent trials and count the number of non-events (failures) preceding the first successful event. The probability that there will be k failures before the first success is given by

$$\text{Geom}(p) : \Pr\{X = k\} \equiv p(k) = p(1-p)^k \qquad k = 0, 1, \ldots \ . \tag{3.8}$$

For this distribution,

$$\begin{aligned} \text{Mean}[X] &= 1/p \\ \text{Var}[X] &= (1-p)/p^2 \\ \text{Skew}[X] &= (2-p)/\sqrt{1-p} \end{aligned}$$

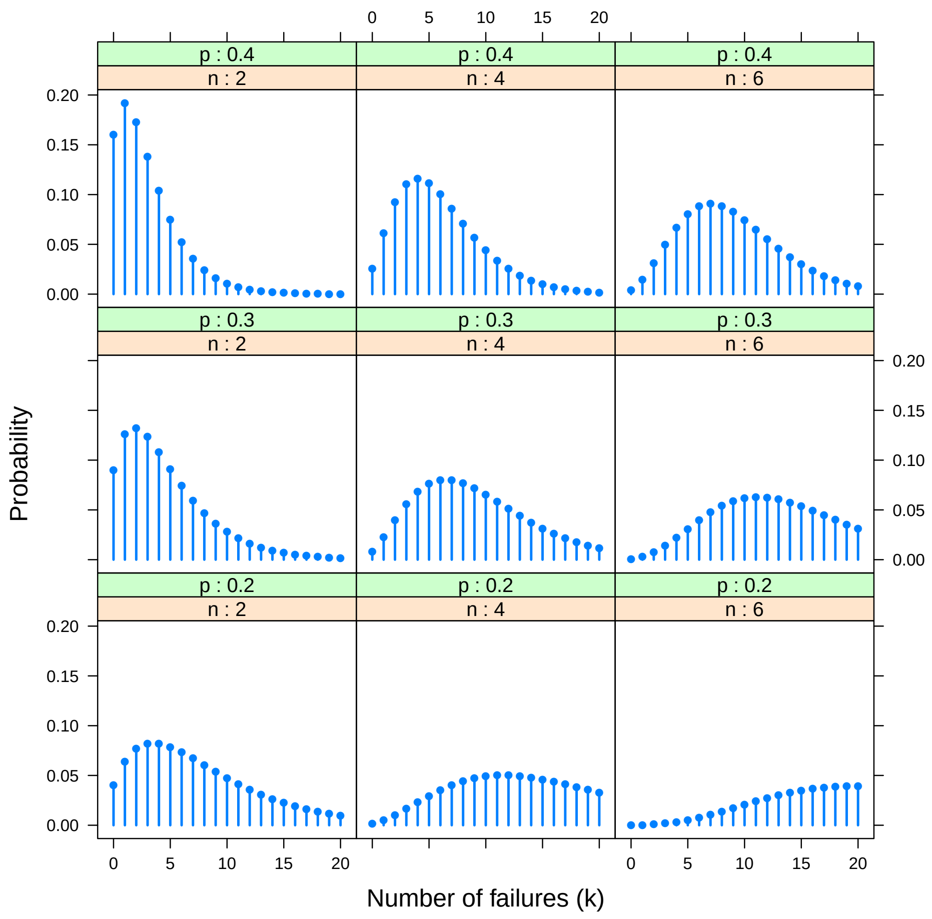


Figure 3.12: Negative binomial distributions for $n = 2, 4, 6$ and $p = 0.2, 0.3, 0.4$, using xyplot

Note that estimation of the parameter p for the geometric distribution can be handled as the special case of the negative binomial by fixing $n = 1$, so no special software is needed. Going the other way, if $X_1, X_2, \dots X_n$ are independent geometrically distributed as $\mathrm{Geom}(p)$, then their sum, $Y = \sum_j^n X_j$ is distributed as $\mathrm{NBin}(p,n)$.

3.2.5 The logarithmic series distribution

The logarithmic series distribution is a long-tailed distribution introduced by Fisher *et al.* (1943) in connection with data on the abundance of individuals classified by species of the type shown for the distribution of butterfly species in Table 3.5.

The probability distribution function with parameter θ is given by

$$\mathrm{LogSer}(\theta) : \Pr\{X = k\} \equiv p(k) = \frac{\theta^k}{-(k \log(1-\theta))} = \alpha\theta^k/k \qquad k = 1, 2, \dots, \infty \ , \tag{3.9}$$

where $\alpha = -1/\log(1-\theta)$ and $0 < \theta < 1$. Fisher derived the logarithmic series distribution by assuming that for a given species the number of individuals trapped has a Poisson distribution with parameter $\lambda = \gamma t$, where γ is a parameter of the species (susceptibility to entrapment) and t is a parameter of the trap. If different species vary so that the parameter γ has a gamma distribution, then the number of representatives of each species trapped will have a negative binomial distribution. However, the observed distribution is necessarily truncated on the left, because one cannot observe the number of species never caught (where $k = 0$). The logarithmic series distribution thus arises as a limiting form of the zero-truncated negative binomial.

ToDo: Revise this section– use p not θ ...; implement `plogseries()`, `dlogseries()`, etc., analogous to other distributions.

3.2.6 Power series family

We mentioned earlier that the Poisson distribution was unique among all discrete (one parameter) distributions, in that it is the only one whose mean and variance are equal (Kosambi, 1949). The relation between mean and variance of discrete distributions also provides the basis for integrating them into a general family. All of the discrete distributions described in this section are in fact special cases of a family of discrete distributions called the power series distributions by Noack (1950) and defined by

$$p(k) = a(k)\theta^k / f(\theta) \qquad k = 0, 1, \dots \ ,$$

with parameter $\theta > 0$, where $a(k)$ is a coefficient function depending only on k and $f(\theta) = \sum_k a(k)\theta^k$ is called the series function. The definitions of these functions are shown in Table 3.10.

These relations among the discrete distribution provide the basis for graphical techniques for diagnosing the form of discrete data described later in this chapter (Section 3.5.5).

3.3 Fitting discrete distributions

In applications to discrete data such as the examples in Section 3.1, interest is often focused on how closely such data follow a particular distribution, such as the Poisson, binomial, or geometric distribution. A close fit provides for interpretation in terms of the underlying mechanism for the

Table 3.10: The Power Series family of discrete distributions

Discrete Distributiion	Probability function, $p(k)$	Series parameter, θ	Series function, $f(\theta)$	Series coefficient, $a(k)$
Poisson	$e^{-\lambda}\lambda^k/k!$	$\theta = \lambda$	e^θ	$1/k!$
Binomial	$\binom{n}{k}p^k(1-p)^{n-k}$	$\theta = p/(1-p)$	$(1+\theta)^n$	$\binom{n}{k}$
Negative binomial	$\binom{n+k-1}{k}p^n(1-p)^k$	$\theta = (1-p)$	$(1-\theta)^{-k}$	$\binom{n+k-1}{k}$
Geometric	$p(1-p)^k$	$\theta = (1-p)$	$(1-\theta)^{-k}$	1
Logarithmic series	$\theta^k/[-k\log(1-\theta)]$	$\theta = \theta$	$-\log(1-\theta)$	$1/k$

distribution; conversely, a bad fit can suggest the possibility for improvement by relaxing one or more of the assumptions. We examine more detailed and nuanced methods for diagnosing and testing discrete distributions in Section 3.4 and Section 3.5 below.

Fitting a discrete distribution involves three basic steps:

1. Estimating the parameter(s) of the distribution from the data, for example, p for the binomial, λ for the Poisson, n and p for the negative binomial. Typically, this is carried out by maximum likelihood methods, or a simpler method of moments, which equates sample moments (mean, variance, skewness) to those of the theoretical distribution, and solves for the parameter estimates. These methods are illustrated in Section 3.3.1.
2. From this, we can calculate the fitted probabilities, $\hat{p}_k$ that apply for the given distribution, or equivalently, the model expected frequencies, $N\hat{p}_k$, where N is the total sample size.
3. Finally, we can calculate goodness-of-fit test measuring the departure between the observed and fitted frequencies.

Often goodness-of-fit is examined with a classical (Pearson) ***goodness-of-fit*** (GOF) chi-square test,

$$\chi^2 = \sum_{k=1}^{K} \frac{(n_k - N\hat{p}_k)^2}{N\hat{p}_k} \sim \chi^2_{(K-s-1)} \ , \tag{3.10}$$

where there are K frequency classes, s parameters have been estimated from the data and $\hat{p}_k$ is the estimated probability of each basic count, under the null hypothesis that the data follows the chosen distribution.

An alternative test statistic is the likelihood-ratio G^2 statistic,

$$G^2 = \sum_{k=1}^{K} n_k \log(n_k/N\hat{p}_k) \ , \tag{3.11}$$

when the $\hat{p}_k$ are estimated by maximum likelihood, which also has an asymptotic $\chi^2_{(K-s-1)}$ distribution. "Asymptotic" means that these are *large sample tests*, meaning that the test statistic follows the χ^2 distribution increasingly well as $N \to \infty$. A common rule of thumb is that all expected frequencies should exceed one and that fewer than 20% should be less than 5.

EXAMPLE 3.10: Death by horse kick

We illustrate the basic ideas of goodness-of fit tests with the `HorseKick` data, where we expect a Poisson distribution with parameter λ = mean number of deaths. As shown in Eqn. (3.3), this is calculated as the frequency (n_k) weighted mean of the k values, here, number of deaths.

In R, such one-way frequency distributions should be converted to data frames with numeric variables. The calculation below uses `weighted.mean()` with the frequencies as weights, and finds $\lambda = 0.61$ as the mean number of deaths per corps-year.

```
# goodness-of-fit test
tab <- as.data.frame(HorseKicks, stringsAsFactors=FALSE)
colnames(tab) <- c("nDeaths", "Freq")
str(tab)

## 'data.frame': 5 obs. of  2 variables:
##  $ nDeaths: chr  "0" "1" "2" "3" ...
##  $ Freq   : int  109 65 22 3 1

(lambda <- weighted.mean(as.numeric(tab$nDeaths), w=tab$Freq))

## [1] 0.61
```

From this, we can calculate the probabilities (`phat`) of `k=0:4` deaths, and hence the expected (`exp`) frequencies in a Poisson distribution.

```
phat <- dpois(0:4, lambda=lambda)
exp <- sum(tab[,"Freq"]) * phat
chisq <- (tab$Freq - exp)^2 / exp

GOF <- data.frame(tab, phat, exp, chisq)
GOF

##   nDeaths Freq     phat      exp    chisq
## 1       0  109 0.543351 108.6702 0.001001
## 2       1   65 0.331444  66.2888 0.025057
## 3       2   22 0.101090  20.2181 0.157048
## 4       3    3 0.020555   4.1110 0.300253
## 5       4    1 0.003135   0.6269 0.222006
```

Finally, the Pearson χ^2 is just the sum of the `chisq` values and `pchisq()` is used to calculate the the p-value of this test statistic.

```
sum(chisq)  # chi-square value

## [1] 0.7054

pchisq(sum(chisq), df=nrow(tab)-2, lower.tail=FALSE)

## [1] 0.8719
```

The result, $\chi^2_3 = 0.70537$ shows an extremely good fit of these data to the Poisson distribution, perhaps exceptionally so.[10]

△

[10] An exceptionally good fit occurs when the p-value for the test χ^2 statistic is so high, as to suggest that that something unreasonable under random sampling might have occurred. The classic example of this is the controversy over Gregor Mendel's experiments of cross-breeding garden peas with various observed (phenotype) characteristics, where R. A. Fisher 1936 suggested that observed frequencies of combinations like (smooth/wrinkled), (green/yellow) in a $2^n d$ generation were uncomfortably too close to the $3:1$ ratio predicted by genetic theory.

3.3.1 R tools for discrete distributions

In R, the function `fitdistr()` in the MASS is a basic work horse for fitting a variety of distributions by maximum likelihood and other methods, giving parameter estimates and standard errors. Among discrete distributions, the binomial, Poisson and geometric distributions have closed-form maximum likelihood estimates; the negative binomial distribution, (parameterized by (n, μ) is estimated iteratively by direct optimization.

These basic calculations are extended and enhanced for one-way discrete distributions in the vcd function `goodfit()`, which computes the fitted values of a discrete distribution (either Poisson, binomial or negative binomial) to the count data. If the parameters are not specified they are estimated either by ML or Minimum Chi-squared. `print()` and `summary()` methods for the "goodfit" objects give, respectively a table of observed and fitted frequencies, and the Pearson and/or likelihood ratio goodness-of-fit statistics. Plotting methods for visualizing the discrepancies between observed and fitted frequencies are described and illustrated in Section 3.3.2.

EXAMPLE 3.11: Families in Saxony

This example uses `goodfit()` to fit the binomial to the distribution of the number of male children in families of size 12 in Saxony. Note that for the binomial, both n and p are considered as parameters, and by default n is taken as the maximum count.

```
data(Saxony, package="vcd")
Sax.fit <- goodfit(Saxony, type="binomial")
Sax.fit$par          # estimated parameters

## $prob
## [1] 0.5192
##
## $size
## [1] 12
```

So, we estimate the probability of a male in these families to be $p = 0.519$, a value that is quite close to the value found in Arbuthnot's data ($p = 0.517$).

It is useful to know that `goodfit()` returns a list structure of named components which are used by method functions for class "goodfit" objects. The `print.goodfit()` method prints the table of observed and fitted frequencies. `summary.goodfit()` calculates and prints the likelihood ratio χ^2 GOF test when the ML estimation method is used.

```
names(Sax.fit)       # components of "goodfit" objects

## [1] "observed" "count"    "fitted"   "type"     "method"
## [6] "df"       "par"

Sax.fit              # print method

##
## Observed and fitted values for binomial distribution
## with parameters estimated by `ML'
##
##  count observed   fitted
##      0        3   0.9328
##      1       24  12.0888
##      2      104  71.8032
##      3      286 258.4751
##       4      670  628.0550
##       5     1033 1085.2107
##       6     1343 1367.2794
##       7     1112 1265.6303
##       8      829  854.2466
##       9      478  410.0126
##      10      181  132.8357
##      11       45   26.0825
##      12        7    2.3473

summary(Sax.fit)   # summary method

##
##   Goodness-of-fit test for binomial distribution
##
##                     X^2 df  P(> X^2)
## Likelihood Ratio 97.01 11 6.978e-16
```

Note that the GOF test gives a highly significant p-value, indicating significant lack of fit to the binomial distribution.[11] Some further analysis of this result is explored in examples below. △

EXAMPLE 3.12: Weldon's dice

Weldon's dice data, explored in Example 3.3, are also expected to follow a binomial distribution, here with $p = \frac{1}{3}$. However, as given in the data set `WeldonDice`, the frequencies for counts 10–12 were grouped as "10+". In this case, it it necessary to supply the correct value of $n = 12$ as the value of the `size` parameter in the call to `goodfit()`.

```
data(WeldonDice, package="vcd")
dice.fit <- goodfit(WeldonDice, type="binomial", par=list(size=12))
unlist(dice.fit$par)

##    prob    size
##  0.3377 12.0000
```

The probability of a success (a 5 or 6) is estimated as $p = 0.3377$, not far from the theoretical value, $p = 1/3$.

ToDo: Fix infelicity in vcd:::print.goodfit to provide control of number of digits in the `fitted` column. – `print(dice.fit)` uses E notation.

```
summary(dice.fit)

##
##   Goodness-of-fit test for binomial distribution
##
##                     X^2 df P(> X^2)
## Likelihood Ratio 11.51  9   0.2426
```

Here, we find an acceptable fit for the binomial distribution. △

[11] A handy rule-of-thumb is to think of the ratio of χ^2/df, because, under the null hypothesis of acceptable fit, $\mathcal{E}(\chi^2/df) = 1$, so ratios exceeding ≈ 2.5 are troubling. Here, the ratio is $97/11 = 8.8$, so the lack of fit is substantial.

EXAMPLE 3.13: Death by horse kick

This example reproduces the calculations done "manually" in Example **??** above. We fit the Poisson distribution to the `HorseKicks` data by specifying `type="poisson"` (actually, that is the default for `goodfit()`).

```
data("HorseKicks", package="vcd")
HK.fit <- goodfit(HorseKicks, type="poisson")
HK.fit$par

## $lambda
## [1] 0.61

HK.fit

##
## Observed and fitted values for poisson distribution
## with parameters estimated by `ML'
##
##   count observed   fitted
##       0      109 108.6702
##       1       65  66.2888
##       2       22  20.2181
##       3        3   4.1110
##       4        1   0.6269
```

The `summary` method uses the LR test by default, so the `X^2` value reported below differs slightly from the Pearson χ^2 value shown earlier.

```
summary(HK.fit)

##
##    Goodness-of-fit test for poisson distribution
##
##                       X^2 df P(> X^2)
## Likelihood Ratio 0.8682  3   0.8331
```

△

EXAMPLE 3.14: Federalist papers

In Example 3.5 we examined the distribution of the marker word "may" in blocks of text in the *Federalist Papers* written by James Madison. A naive hypothesis is that these occurrences might follow a Poisson distribution, that is, as independent occurrences with constant probability across the 262 blocks of text. Using the same methods as above, we fit these data to the Poisson distribution

```
data("Federalist", package="vcd")
Fed.fit0 <- goodfit(Federalist, type="poisson")
unlist(Fed.fit0$par)

## lambda
## 0.6565

Fed.fit0
```

```
##
## Observed and fitted values for poisson distribution
## with parameters estimated by `ML'
##
##  count observed    fitted
##      0      156 135.89139
##      1       63  89.21114
##      2       29  29.28305
##      3        8   6.40799
##      4        4   1.05169
##      5        1   0.13808
##      6        1   0.01511
```

The GOF test below shows a substantial lack of fit, rejecting the assumptions of the Poisson model.

```
summary(Fed.fit0)

##
##   Goodness-of-fit test for poisson distribution
##
##                      X^2 df  P(> X^2)
## Likelihood Ratio 25.24  5 0.0001251
```

Mosteller and Wallace (1963) determined that the negative binomial distribution provided a better fit to these data than the Poisson. We can verify this as follows:

```
Fed.fit1 <- goodfit(Federalist, type = "nbinomial")
unlist(Fed.fit1$par)

##   size   prob
## 1.1863 0.6438

summary(Fed.fit1)

##
##   Goodness-of-fit test for nbinomial distribution
##
##                      X^2 df P(> X^2)
## Likelihood Ratio 1.964  4   0.7424
```

Recall that the Poisson assumes that the probability of a word like *may* appearing in a block of text is small and constant and that for the Poisson, $\mathcal{E}(x) = \mathcal{V}(x) = \lambda$. One interpretation of the better fit of the negative binomial is that the use of a given word occurs with Poisson frequencies, but Madison varied its rate λ_i from one block of text to another according to a gamma distribution, allowing greater the variance to be greater than the mean.

$\triangle$

3.3.2 Plots of observed and fitted frequencies

In the examples of the last section, we saw cases where the GOF tests showed close agreement between the observed and model-fitted frequencies, and cases where they diverged significantly, to cause rejection of a hypothesis that the data followed the specified distribution.

What is missing from such numerical summaries is any appreciation of the *details* of this statistical comparison. Plots of the observed and fitted frequencies can help to show both the shape of the theoretical distribution we have fitted and the pattern of any deviations between our data and theory.

In this section we illustrate some simple plotting tools for these purposes, using the `plot.goodfit()` method for "gootfit" objects.[12] The left panel of Figure 3.13 shows the fit of the Poisson distribution to the Federalist papers data, using one common form of plot that is sometimes used for this purpose. In this plot, observed frequencies are shown by bars and fitted frequencies are shown by points, connected by a smooth (spline) curve.

Such a plot, however, is dominated by the largest frequencies, making it hard to assess the deviations among the smaller frequencies. To make the smaller frequencies more visible, Tukey (1977) suggest plotting the frequencies on a square-root scale, which he calls a ***rootogram***. This plot is shown in the right panel of Figure 3.13.

```
plot(Fed.fit0, scale="raw", type="standing")
plot(Fed.fit0, type="standing")
```

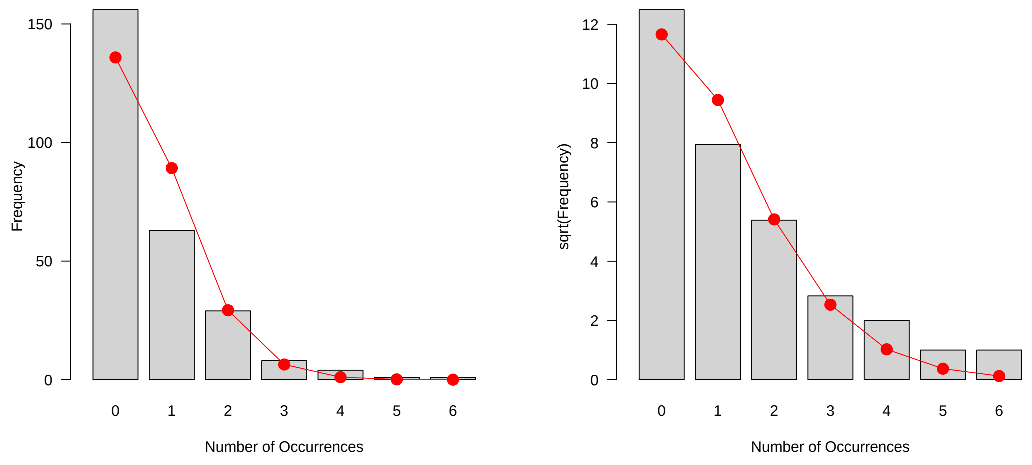


Figure 3.13: Plots for the Federalist Papers data, fitting the Poisson model. Each panel shows the observed frequencies as bars and the fitted frequencies as a smooth curve. Left: raw frequencies; right: plotted on a square root scale to emphasize smaller frequencies.

Additional improvements over the standard plot on the scale of raw frequencies are shown in Figure 3.14, both of which use the square root scale. The left panel movs the rootogram bars so their tops are at the expected frequencies (giving a ***hanging rootogram***). This has the advantage that we can more easily judge the pattern of departures against the horizontal reference line at 0, than against the curve.

```
plot(Fed.fit0, type="hanging")
plot(Fed.fit0, type="deviation")
```

[12]Quantile-quantile (QQ) plots are a common alternative for for the goal of comparing observed and expected values under some distribution. These plots are useful for unstructured samples, but less so when we want to also see the shape of a distribution, as is the case here.

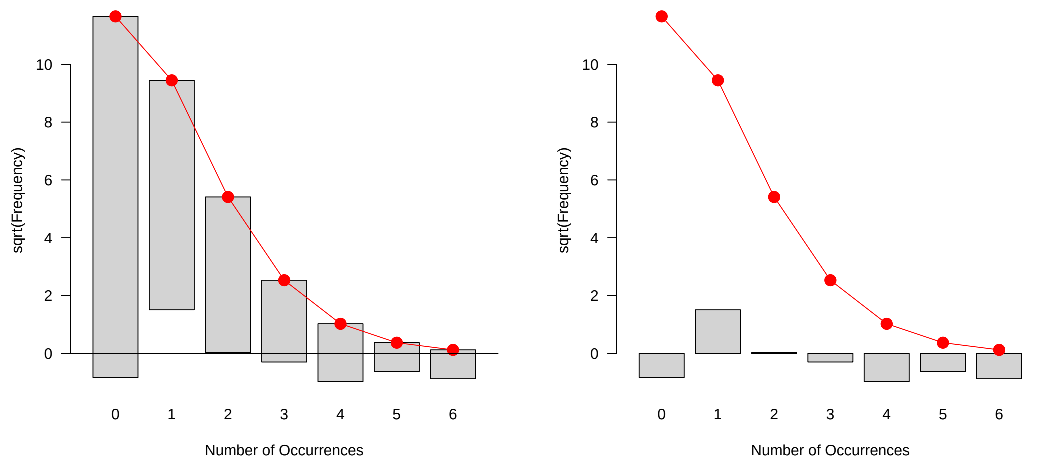


Figure 3.14: Plots for the Federalist Papers data, fitting the Poisson model. Left: hanging rootogram; right: deviation rootogram.

A final variation is to emphasize the differences between the observed and fitted frequencies by drawing the bars to show the gaps between the 0 line and the (observed-expected) difference (Figure 3.14, right).

All of these plots are actually produced by the `rootogram()` function in vcd. The default is `type="hanging"`, and there are many options to control the plot details.

The plots in Figure 3.13 and Figure 3.14 used the ill-fitting Poisson model on purpose to highlight how these plots show the departure between the observed and fitted frequencies. Figure 3.15 compares this with the negative binomial model, `Fed.fit1` which we saw has a much better, and acceptable fit.

```
plot(Fed.fit0, main="Poisson")
plot(Fed.fit1, main="Negative binomial")
```

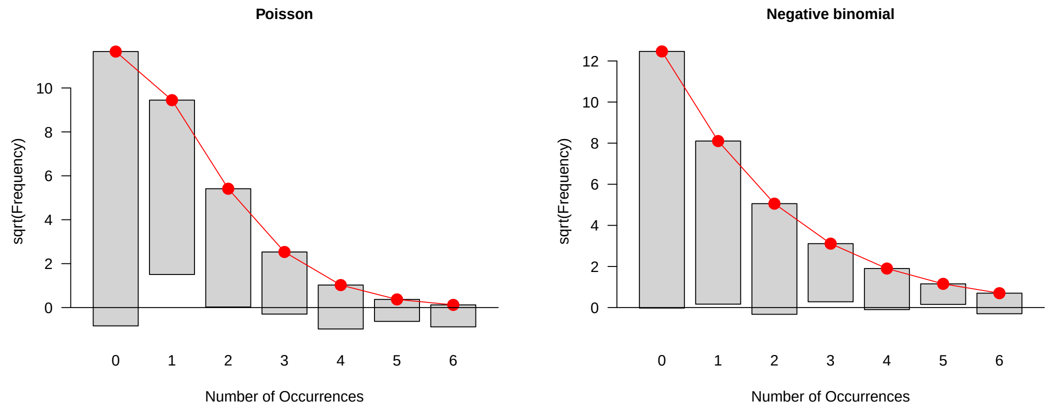


Figure 3.15: Hanging rootograms for the Federalist Papers data, comparing the Poisson and negative binomial models.

Comparing the two plots in Figure 3.15, we can see that the Poisson model underestimates the frequencies of 0 counts and the larger counts for 4-6 occurrences. The deviations for the negative binomial are small and unsystematic.

Finally, Figure 3.16 shows hanging rootograms for two attrociously bad models for the data on butterfly species in Malaya considered in Example 3.6. As we will see in Section 3.4, this long-tailed distribution is better approximated by the logarithmic series distribution, but this distribution is presently not handled by `goodfit()`.

```
data(Butterfly, package="vcd")
But.fit1 <- goodfit(Butterfly, type="poisson")
But.fit2 <- goodfit(Butterfly, type="nbinomial")
plot(But.fit1, main="Poisson")
plot(But.fit2, main="Negative binomial")
```

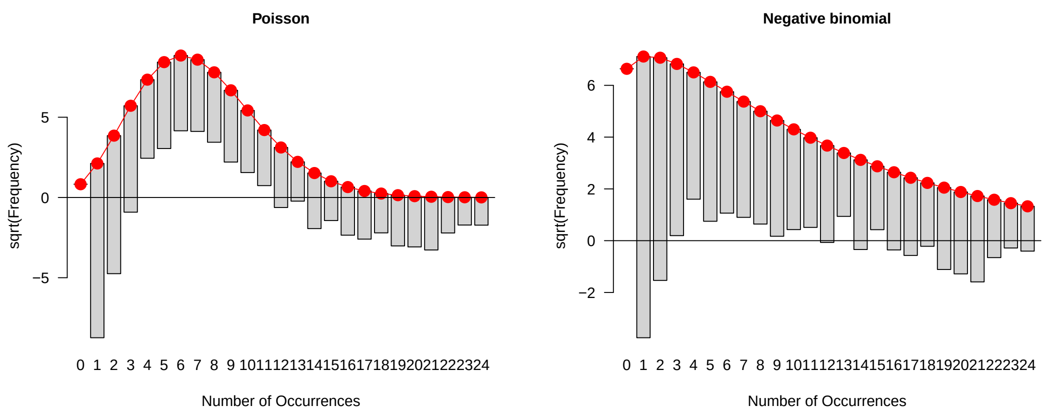


Figure 3.16: Hanging rootograms for the Butterfly data, comparing the Poisson and negative binomial models. The lack of fit for both is readily apparent.

ToDo: Old sections here described the general ideas behind maximum likelihood estimation, and methods for fitting discrete distributions as loglinear models. What should be included in this revision?

3.4 Diagnosing discrete distributions: Ord plots

Ideally, the general form chosen for a discrete distribution should be dictated by substantive knowledge of a plausible mechanism for generating the data. When such knowledge is lacking, however, we may not know which distribution is most appropriate for some particular set of data. In these cases, the question is often turned around, so that we seek a distribution that fits well, and then try to understand the mechanism in terms of aspects of the underlying probability theory (independent trials, rare events, waiting-time to an occurrence, and so forth).

Although it is possible to fit each of several possibilities, the summary goodness-of-fit statistics can easily be influenced by one or two disparate cells, or additional (ignored or unknown) factors. One simple alternative is a plot suggested by Ord (1967) which may be used to diagnose the form of the discrete distribution.

Ord showed that a linear relationship of the form,

$$\frac{k\,p(k)}{p(k-1)} \equiv \frac{k\,n_k}{n_{k-1}} = a + b\,k \ , \tag{3.12}$$

holds for each of the Poisson, binomial, negative binomial, and logarithmic series distributions, and these distributions are distinguished by the signs of the intercept, a, and slope, b, as shown in Table 3.11.

Table 3.11: Diagnostic slope and intercept for four discrete distributions. The ratios kn_k/n_{k-1} plotted against k should appear as a straight line, whose slope and intercept determine the particular distribution.

Slope (b)	Intercept (a)	Distribution (parameter)	Parameter estimate
0	$+$	Poisson (λ)	$\lambda = a$
$-$	$+$	Binomial (n, p)	$p = b/(b-1)$
$+$	$+$	Negative binomial (n,p)	$p = 1 - b$
$+$	$-$	Log. series (θ)	$\theta = b$
			$\theta = -a$

The slope, b, in Eqn. (3.12) is zero for the Poisson, negative for the binomial, and positive for the negative binomial and logarithmic series distributions; the latter two are distinguished by their intercepts. In practical applications of this idea, the details are important: how to fit the line, and how to determine if the pattern of signs are sufficient to reasonably provide a diagnosis of the distribution type.

One difficulty in applying this technique is that the number of points (distinct values of k) in the Ord plot is often small, and the sampling variances of $k\,n_k/n_{k-1}$ can vary enormously. A little reflection indicates that points where n_k is small should be given less weight in determining the slope of the line (and hence determining the form of the distribution). In applications it has been found that using a weighted least squares fit of $k\,n_k/n_{k-1}$ on k, using weights of $w_k = \sqrt{n_k - 1}$ produces reasonably good automatic diagnosis of the form of a probability distribution. Moreover, to judge whether a coefficient is positive or negative, a small tolerance is used; if none of the distributions can be classified, no parameters are estimated. Caution is advised in accepting the conclusion, because it is based on these simple heuristics.

In the vcd package this method is implemented in the `Ord_plot()` function. The essential ideas are illustrated using the `Butterfly` data below, which produces Figure 3.17. Note that the function returns (invisibly) the values of the intercept and slope in the weighted LS regression.

```
ord <- Ord_plot(Butterfly,
                main = "Butterfly species collected in Malaya",
                gp=gpar(cex=1), pch=16)
ord

## Intercept     Slope
##    -0.709     1.061
```

In this plot, the black line shows the usual OLS regression fit of frequency, n_k on number of occurrences, k; the red line shows the weighted least squares fit, using weights of $\sqrt{n_k - 1}$.

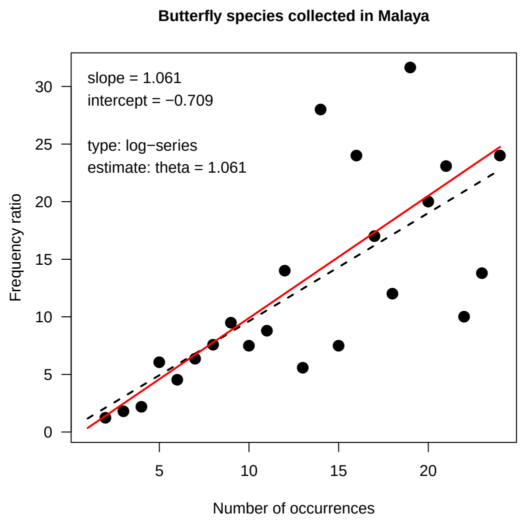


Figure 3.17: Ord plot for the Butterfly data. The slope and intercept in the plot correctly diagnoses the log-series distribution.

In this case, the two lines are fairly close together, as regards their intercepts and slopes. The positive slope and negative intercept diagnoses this as a log-series distribution.

In other cases, the number of distinct points (values of k) is small, and the sampling variances of the ratios $k\, n_k / n_{k-1}$ can vary enormously. The following examples illustrate some other distributions and some of the details of the heuristics.

Done: In `vcd_1.3-2`, fixed `Ord_plot()` to provide control of `lwd`, `lty`, etc.

Ord plot examples

EXAMPLE 3.15: Death by horse kick

The results below show the calculations for the horse kicks data, with the frequency ratio $k\, n_k / n_{k-1}$ labeled `y`.

```
data(HorseKicks, package="vcd")
nk <- as.vector(HorseKicks)
k <- as.numeric(names(HorseKicks))
nk1 <- c(NA, nk[-length(nk)])
y <- k * nk/nk1
weight = sqrt(pmax(nk, 1) - 1)
(ord.df <- data.frame(k, nk, nk1, y, weight))

##   k  nk nk1      y weight
## 1 0 109  NA     NA 10.392
## 2 1  65 109 0.5963  8.000
## 3 2  22  65 0.6769  4.583
## 4 3   3  22 0.4091  1.414
## 5 4   1   3 1.3333  0.000

coef(lm(y ~ k, weights=weight, data=ord.df))

## (Intercept)           k
##     0.65602    -0.03414
```

The weighted least squares line, with weights w_k, has a slope (-0.03) close to zero, indicating the Poisson distribution.[13] The estimate $\lambda = a = .656$ compares favorably with the MLE, $\lambda = 0.610$ and the value from the Poissonness plot, shown in the following section. The call to `Ord_plot()` below produces Figure 3.18.

```r
Ord_plot(HorseKicks,
         main = "Death by horse kicks", gp=gpar(cex=1), pch=16)
```

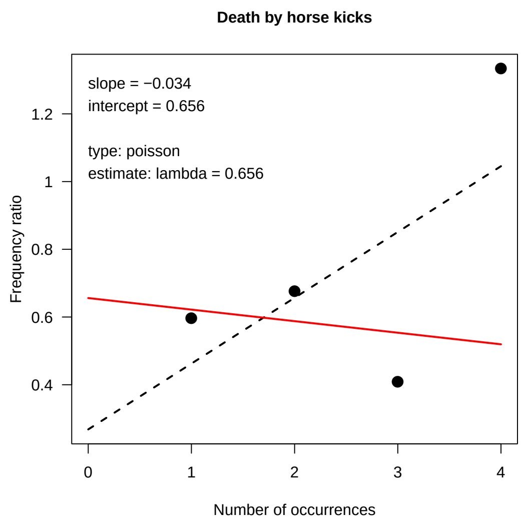


Figure 3.18: Ord plot for the HorseKicks data. The plot correctly diagnoses the Poisson distribution.

△

EXAMPLE 3.16: Federalist papers

Figure 3.19 (left) shows the Ord plot for the `Federalist` data. The slope is positive, so either the negative binomial or log series are possible, according to Table 3.11. The intercept is essentially zero, which is ambiguous. However, the logarithmic series requires $b \approx -a$, so the negative binomial is a better choice. Mosteller and Wallace (1963, 1984) did in fact find a reasonably good fit to this distribution. Note that there is one apparent outlier, at $k = 6$, whose effect on the OLS line is to increase the slope and decrease the intercept. △

```r
Ord_plot(Federalist, main = "Instances of 'may' in Federalist papers",
         gp=gpar(cex=1), pch=16)
Ord_plot(WomenQueue, main = "Women in queues of length 10",
         gp=gpar(cex=1), pch=16)
```

EXAMPLE 3.17: Women in queues

Jinkinson and Slater (1981), Hoaglin and Tukey (1985) give the frequency distribution of the number of females observed in 100 queues of length 10 in a London Underground station, recorded in the data set `WomenQueue` in vcd.

[13] The heuristic adopted in `Ord_plot()` uses a tolerance of 0.1 to decide if a coefficient is negative, zero, or positive.

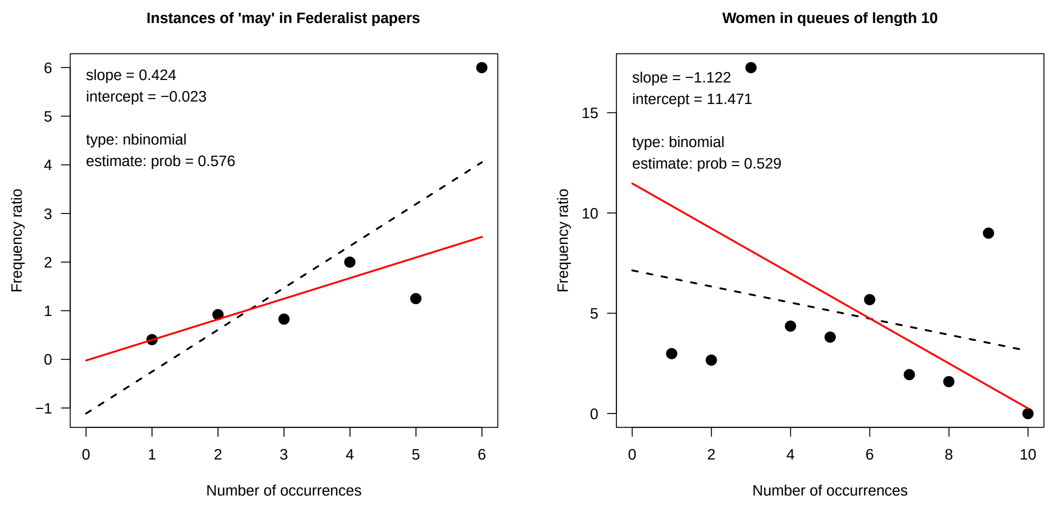


Figure 3.19: Ord plots for the Federalist (left) and WomenQueue (right) data sets.

```
data(WomenQueue, package="vcd")
WomenQueue

## nWomen
##   0   1   2   3   4   5   6   7   8   9  10
##   1   3   4  23  25  19  18   5   1   1   0
```

If it is assumed that people line up independently, and that men and women are equally likely to be found in a queue (not necessarily reasonable assumptions), then the number of women out of 10 would have a (symmetric) binomial distribution with parameters $n = 10$ and $p = \frac{1}{2}$. However, there is no real reason to expect that males and females are equally likely to be found in queues in the London underground, so we may be interested in estimating p from the data and determining if a binomial distribution fits.

Figure 3.19 (right) shows the Ord plot for these data. The negative slope and positive intercept clearly diagnose this distribution as binomial. The rough estimate of $\hat{p} = b/(1-b) = 0.53$ indicates that women are slightly more prevalent than men in these data for the London underground.

$\triangle$

Limitations of Ord plots

Using a single simple diaganostic plot to determine one of four common discrete distributions is advantageous, but your enthusiasm should be dampened by several weaknesses:

- The Ord plot lacks resistance, since a single discrepant frequency affects the points n_k/n_{k-1} for both k and $k+1$.
- The sampling variance of $k\, n_k/n_{k-1}$ fluctuates widely (Hoaglin and Tukey, 1985, Jinkinson and Slater, 1981). The use of weights w_k helps, but is purely a heuristic device. The `Ord_plot()` function explicitly shows both the OLS line and the WLS line, which provides some indication of the effect of the points on the estimation of slope and intercept.

3.5 Poissonness plots and generalzed distribution plots

The ***Poissonness plot*** (Hoaglin, 1980) is a robust plot to sensitively determine how well a one-way table of frequencies follows a Poisson distribution. It plots a quantity called a count metameter against k, designed so that the result will be points along a straight line when the data follow a Poisson distribution. When the data deviate from a Poisson, the points will be curved. Hoaglin and Tukey (1985) develop similar plots for other discrete distributions, including the binomial, negative binomial, and logarithmic series distributions. We first describe the features and construction of these plots for the Poisson distribution and then (Section **??**) the extension to other distributions.

3.5.1 Features of the Poissonness plot

The Poissonness plot has the following desirable features:

- ***Resistance***: a single discrepant value of n_k affects only the point at value k. (In the Ord plot it affects each of its neighbors.)
- ***Comparison standard***: An approximate confidence interval can be found for each point, indicating its inherent variability and helping to judge whether each point is discrepant.
- ***Influence***: Extensions of the method result in plots which show the effect of each point on the estimate of the main parameter of the distribution (λ in the Poisson).

3.5.2 Plot construction

Assume, for some fixed λ, each observed frequency, n_k equals the expected frequency, $m_k = Np_k$. Then, setting $n_k = Np_k = Ne^{-\lambda}\,\lambda^k/k!$, and taking logs of both sides gives

$$\log(n_k) = \log\, N - \lambda + k\, \log\, \lambda - \log\, k! \ .$$

This can be rearranged to a linear equation in k,

$$\phi\,(n_k) \equiv \log\left(\frac{k!\, n_k}{N}\right) = -\lambda + (\log\, \lambda)\, k \ . \tag{3.13}$$

The left side of Eqn. (3.13) is called the ***count metameter***, and denoted $\phi\,(n_k)$. Hence, plotting $\phi(n_k)$ against k should give a striaght line of the form $\phi(n_k) = a + bk$ with

- slope = $\log\, \lambda$
- intercept = $-\lambda$

when the observed frequencies follow a Poisson distribution. If the points in this plot are close enough to a straight line, then an estimate of λ may be obtained from the slope b of the line, $\hat{\lambda} = e^b$ should be reasonably close in value to the MLE of λ, $\hat{\lambda} = \bar{x}$. In this case, we might as well use the MLE as our estimate.

Leveled plot

If we have a preliminary estimate λ_0 of λ, we can use this to give a new plot where the reference line is horizontal, making comparison of the points with the line easier. In this leveled plot the vertical coordinate $\phi(n_k)$ is modified to

$$\phi'(n_k) = \phi(n_k) + \lambda_0 - k \log \lambda_0 \ . \tag{3.14}$$

When the data follow a Poisson distribution with parameter λ, the modified plot will have

- slope = $\log \lambda - \log \lambda_0 = \log(\lambda / \lambda_0)$
- intercept = $\lambda_0 - \lambda$

In the ideal case, where our estimate of λ_0 is close to the true λ, the line will be approximately horizontal at $\phi(n_k)' = 0$. The modified plot is particularly useful in conjunction with the confidence intervals for individual points described below.

Confidence intervals

The goal of the Poissonness plot is to determine whether the points are "sufficiently linear" to conclude that the Poisson distribution is adequate for the data. Confidence intervals for the points can help you decide, and also show the relative precision of the points in these plots.

For example, when one or two points deviate from an otherwise nearly linear relation, it is helpful to determine whether the discrepancy is consistent with chance variation. As well, we must recognize that classes with small frequencies n_k are less precise than classes with large frequencies.

Hoaglin and Tukey (1985) develop approximate confidence intervals for $\log(m_k)$ for each point in the Poissonness plot. These are calculated as

$$\phi\left(n_k^*\right) \pm h_k \tag{3.15}$$

where the count metameter function is calculated using a modified frequency n_k^*, defined as

$$n_k^* = \begin{cases} n_k - .8 n_k - .67 & n \geq 2 \\ 1/e & n = 1 \\ \text{undefined} & n = 0 \end{cases}$$

and h_k is the half-width of the 95% confidence interval,

$$h_k = 1.96 \frac{\sqrt{1 - \widehat{p}_k}}{[n_k - (.25 \widehat{p}_k + .47)\sqrt{n_k}]^{1/2}}$$

and $\hat{p}_k = n_k / N$.

3.5.3 The `distplot()` function

Poissonness plots (and versions for other distributions) are produced by the function `distplot()` in vcd. As with `Ord_plot()`, the first argument is either a vector of counts, a one-way table of frequencies of counts or a data frame or matrix with frequencies in the first column and the corresponding counts in the second column. Nearly all of the examples in this chapter use one-way tables of counts.

The `type` argument specifies the type of distribution. For `type = "poisson"`, specifying a value for `lambda` $= \lambda_0$ gives the leveled version of the plot.

EXAMPLE 3.18: Death by horse kick

The calculations for the Poissonness plot, including confidence intervals, are shown below for the `HorseKicks` data. The call to `distplot()` produces the plot in the left panel of Figure 3.20.

```
data("HorseKicks", package="vcd")
dp <- distplot(HorseKicks, type = "poisson",
  xlab="Number of deaths", main="Poissonness plot: HorseKicks data")
print(dp, digits=4)

##   Counts Freq Metameter CI.center CI.width CI.lower CI.upper
## 1      0  109    -0.607   -0.6131   0.1305  -0.7436  -0.4827
## 2      1   65    -1.124   -1.1343   0.2069  -1.3412  -0.9274
## 3      2   22    -1.514   -1.5451   0.4169  -1.9620  -1.1281
## 4      3    3    -2.408   -2.6607   1.3176  -3.9783  -1.3431
## 5      4    1    -2.120   -3.1203   2.6887  -5.8089  -0.4316
```

In this plot, the open circles show the calculated observed values of the count `Metameter` = $\phi(n_k)$. The smaller filled points show the centers of the confidence intervals, `CI.center` = $\phi(n_k^*)$ (Eqn. (3.15)), and the dashed lines show the extent of the confidence intervals.

The fitted least squares line has a slope of -0.431, which would indicate $\lambda = e^{-0.431} = 0.65$. This compares well with the MLE, $\lambda = \bar{x} = 0.61$.

Using `lambda = 0.61` as below gives the leveled version shown in the right panel of Figure 3.20.

```
# leveled version, specifying lambda
distplot(HorseKicks, type = "poisson", lambda = 0.61,
  xlab="Number of deaths", main="Leveled Poissonness plot")
```

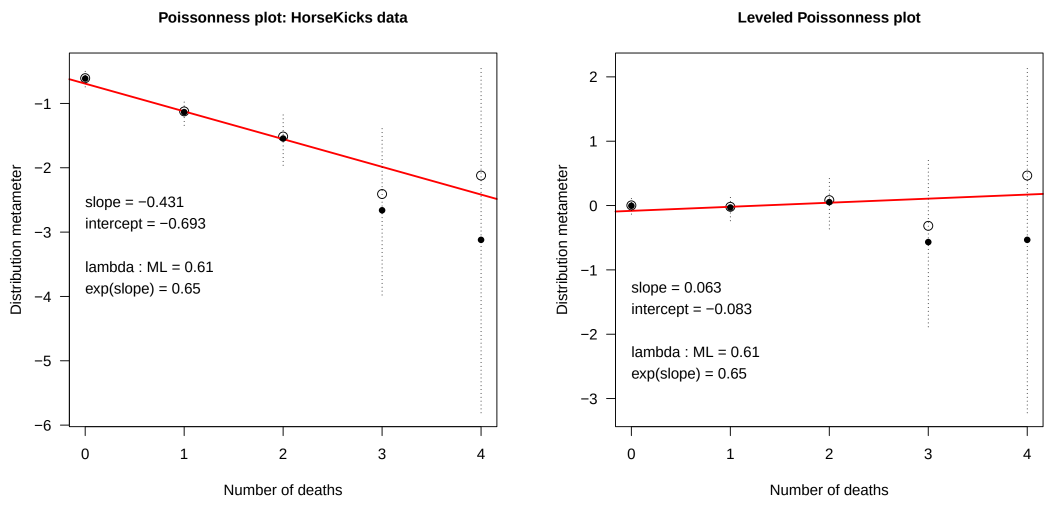


Figure 3.20: Poissonness plots for the HorseKick data. Left: standard plot; right: leveled plot.

In both plots the fitted line is within the confidence intervals, indicating the adequacy of the Poisson model for these data. The widths of the intervals for $k > 2$ are graphic reminders that these observations have decreasingly low precision where the counts n_k are small.

△

3.5.4 Leverage and influence

ToDo: This subsection should be omittted, unless we extend the calculation and plotting methods.

3.5.5 Plots for other distributions

As described in Section 3.2.6, the binomial, Poisson, negative binomial, geometric, and logseries distributions are all members of the general power series family of discrete distributions. For this family, Hoaglin and Tukey (1985) develop similar plots of a count metameter against k which appear as a straight line when a data distribution follows a given family member.

The distributions which can be analyzed in this way are shown in Table 3.12, with the interpretation given to the slope and intercept in each case. For example, for the Binomial distribution, a “binomialness” plot is constructed by plotting $\log n_k^*/N\binom{n}{k}$ against k. If the points in this plot approximate a straight line, the slope is interpreted as $\log(p/(1-p))$, so the binomial parameter p may be estimated as $p = e^b/(1+e^b)$.

Table 3.12: Plot parameters for five discrete distributions. In each case the count metameter, $\phi(n_k^*)$ is plotted against k, yielding a straight line when the data follow the given distribution.

Distributiion	Probability function, $p(k)$	Count metameter, $\phi(n_k^*)$	Theoretical Slope (b)	Theoretical Intercept (a)
Poisson	$e^{-\lambda}\lambda^k/k!$	$\log(k!n_k^*/N)$	$\log(\lambda)$	-λ
Binomial	$\binom{n}{k}p^k(1-p)^{n-k}$	$\log\left(n_k^*/N\binom{n}{k}\right)$	$\log\left(\frac{p}{1-p}\right)$	$n\log(1-p)$
Negative binomial	$\binom{n+k-1}{k}p^n(1-p)^k$	$\log\left(n_k^*/N\binom{n+k-1}{k}\right)$	$\log(1-p)$	$n\log(p)$
Geometric	$p(1-p)^k$	$\log\left(n_k^*/N\right)$	$\log(1-p)$	$\log(p)$
Logarithmic series	$\theta^k/[-k\log(1-\theta)]$	$\log\left(kn_k^*/N\right)$	$\log(\theta)$	$-\log\left(-\log(1-\theta)\right)$

Source: adapted from Hoaglin and Tukey (1985), Table 9-15.

Unlike the Ord plot, a different plot is required for each distribution, because the count metameter, $\phi(n_k)$, differs from distribution to distribution. Moreover, systematic deviation from a linear relationship does not indicate which distribution provides a better fit. However, the attention to robustness, and the availability of confidence intervals and influence diagnostics make this a highly useful tool for visualizing discrete distributions.

EXAMPLE 3.19: Families in Saxony

Our analysis in Example 3.2 and Example 3.11 of the Saxony data showed that the distribution of male children had slightly heavier tails than the binomial, meaning the observed distribution is overdispersed. We can see this in the `goodfit()` plot shown in Figure 3.21 (left), and even more clearly in the distribution diagnostic plot produced by `distplot()` in the right panel of Figure 3.21. For a binomial distribution, we call this distribution plot a "binomialness plot".

```
data("Saxony", package="vcd")
plot(goodfit(Saxony, type="binomial", par=list(size=12)))
distplot(Saxony, type = "binomial", size = 12,
  xlab="Number of males")
```

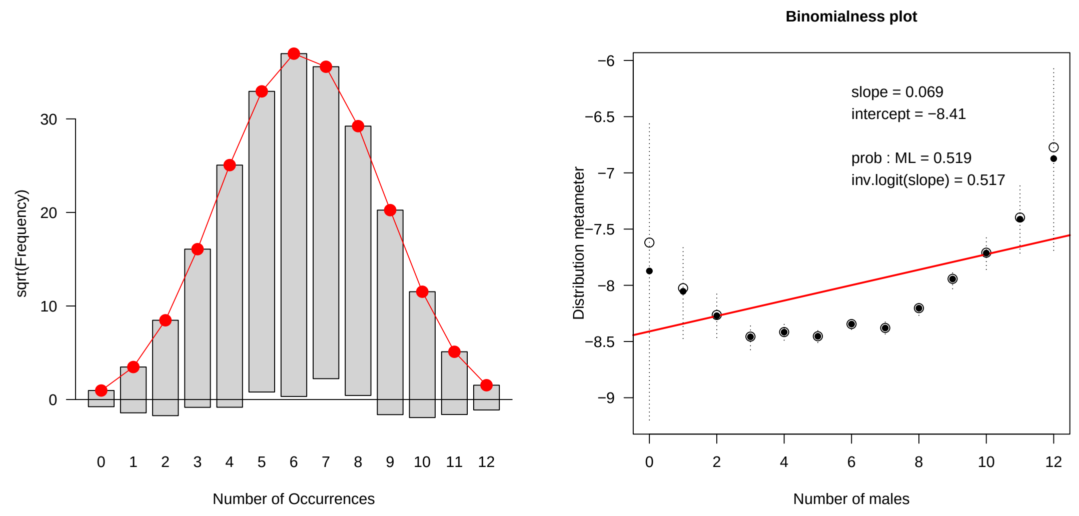


Figure 3.21: Diagnostic plots for males in Saxony families. Left: `goodfit()` plot; right: `distplot()` plot. Both plots show heavier tails than in a binomial distribution.

The weight of evidence is thus that, as simple as the binomial might be, it is inadequate to fully explain the distribution of sex ratios in this large sample of families of 12 children. To understand this data better, it is necessary to question the assumptions of the binomial (births of males are independent Bernoulli trials with constant probability p) as a model for this birth distribution and/or find a more adequate model.[14] △

Example 3.20: Federalist papers

In Example 3.14 we carried out GOF tests for the Poisson and negative binomial models with the Federalist papers data; Figure 3.15 showed the corresponding rootogram plots. Figure 3.22 compares these two using the diagnostic plots of this section. Again the Poisson shows systematic departure from the linear relation required in the Poissonness plot, while the negative binomial model provides an acceptable fit to these data.

[14]On these questions, Edwards (1958) reviews numerous other studes of these Geissler's data, and fits a so-called ***β-binomial*** model proposed by Skellam (1948), where p varies among families according to a β distribution. He concludes that there is evidence that p varies between families of the same size. One suggested explanation is that family decisions to have a further child is influenced by the balance of boys and girls among their earlier children.

```
distplot(Federalist, type = "poisson", xlab="Occurrences of 'may'")
distplot(Federalist, type = "nbinomial", xlab="Occurrences of 'may'")
```

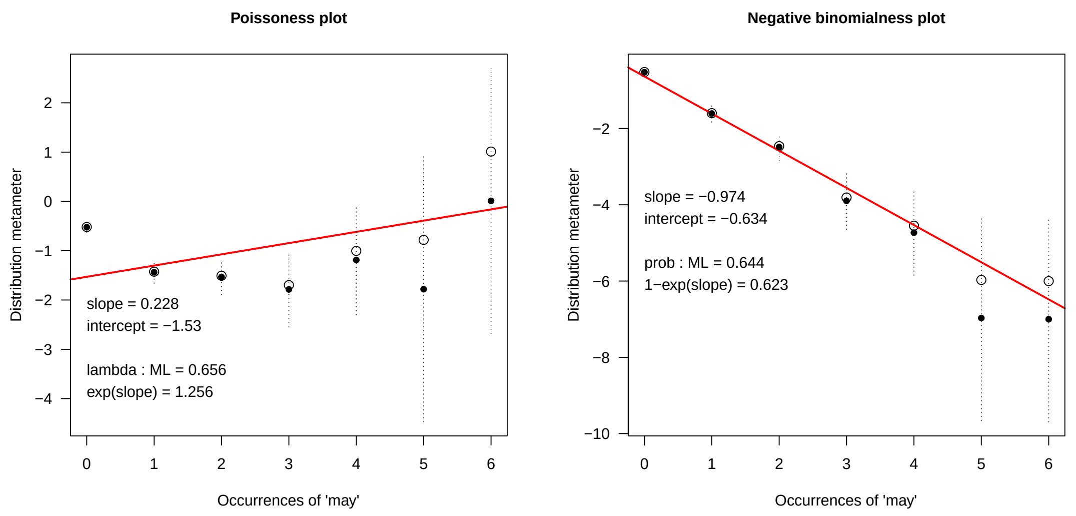


Figure 3.22: Diagnostic plots for the Federalist papers data. Left: Poissonness plot; right: negative binomialness plot.

△

3.6 Fitting discrete distributions as generalized linear models

In Section 3.2.6, we described how the common discrete distributions are all members of the general power series family. This provides the basis for the generalized distribution plots described in Section 3.5.5. Another general family of distributions—the ***exponential family***—includes most of the common continuous distributions: the normal, gamma, exponential, and others, and is the basis of the class of generalized linear models fit by `glm()`.

A clever approach by Lindsey and Mersch (1992), Lindsey (1995, §6.1) shows how various discrete (and continuous) distributions can be fit to frequency data using generalized linear models for log frequency (which are equivalent to Poisson loglinear models). The uniform, geometric, binomial, and the Poisson distributions may all be fit easily in this way, but the idea extends to some other distributions, such as the ***double binomial*** distribution, that allows a separate parameter for overdispersion relative to the binomial. A clear advantage is that this method gives estimated standard errors for the distribution parameters as well as estimated confidence intervals for fitted probabilities.

The essential idea is that, for frequency data, any distribution in the exponential family may be represented by a linear model for the logarithm of the cell frequency, with a Poisson distribution for errors, otherwise known as a "Poisson loglinear regression model". These have the form

$$\log(N\pi_k) = \text{offset} + \beta_0 + \boldsymbol{\beta}^\mathsf{T}\boldsymbol{S}(k) \ ,$$

where N is the total frequency, π_k is the modeled probability of count k, $\boldsymbol{S}(k)$ is a vector of zero or more sufficient statistics for the canonical parameters of the exponential family distribution, and the offset term is a value which does not depend on the parameters.

Table 3.13 shows the sufficient statistics and offsets for several discrete distributions. See Lindsey and Mersch (1992) for further details, and definitions for the double-binomial distribution,[15] and Lindsey (1995, pp. 130–133) for his analysis of the `Saxony` data using this distribution. Lindsey and Altham (1998) provide an analysis of the complete Geissler data using several different models to handle overdispersion.

Table 3.13: Poisson loglinear representations for some discrete distributions

Distribution	Sufficient statistics	Offset
Geometric	k	
Poisson	k	$-\log(k!)$
Binomial	k	$\log\binom{n}{k}$
Double binomial	$k, k\log(k) + (n-k)\log(n-k)$	$\log\binom{n}{k}$

EXAMPLE 3.21: Families in Saxony

The binomial distribution and the double binomial can both be fit to frequency data as a Poisson regression via `glm()` using $\log\binom{n}{k}$ as an offset. First, we convert `Saxony` into a numeric data frame for use with `glm()`.

```r
data(Saxony, package="vcd")
Males <- as.numeric(names(Saxony))
Families <- as.vector(Saxony)
Sax.df <- data.frame(Males, Families)
```

To calculate the offset for `glm()` in R, note that `choose(12,0:12)` returns the binomial coefficients, and `lchoose(12,0:12)` returns their logs.

```r
# fit binomial (12, p) as a glm
Sax.bin <- glm(Families ~ Males, offset=lchoose(12,0:12),
               family=poisson, data=Sax.df)

# brief model summaries
summarise(Sax.bin)

## Model Summary:
##          LR Chisq Df Pr(>Chisq) AIC BIC
## Sax.bin        97 11   6.98e-16  75 1.1

coef(Sax.bin)

## (Intercept)       Males
##    -0.06952     0.07690
```

[15] In R, the double binomial distribution is implemented in the rmutil package, providing the standard complement of density function (`ddoublebinom()`), CDF (`pdoublebinom()`), quantiles (`qdoublebinom()`) and random generation (`rdoublebinom()`).

As we have seen, this model fits badly. The parameter estimate for Males, $\beta_1 = 0.0769$ is actually estimating the logit of p, $\log p/(1-p)$, so the inverse transformation gives $\hat{p} = \frac{\exp(\beta_1)}{1+\exp(\beta_1)} = 0.5192$, as we had before.

The double binomial model can be fitted as follows. The term YlogitY calculates $k\log(k) + (n-k)\log(n-k)$, the second sufficient statistic for the double binomial (see Table 3.13) fitted via glm(). **ToDo**: Fix Table 3.13 entry here

```
# double binomial, (12, p, psi)
Sax.df$YlogitY <-
  Males        * log(ifelse(Males==0, 1, Males)) +
        (12-Males) * log(ifelse(12-Males==0, 1, 12-Males))

Sax.dbin <- glm(Families ~ Males + YlogitY, offset=lchoose(12,0:12),
        family=poisson, data=Sax.df)
coef(Sax.dbin)

## (Intercept)        Males      YlogitY
##    -3.09692      0.06598      0.14021

summarise(glmlist(Sax.bin, Sax.dbin))

## Model Summary:
##            LR Chisq Df Pr(>Chisq)   AIC    BIC
## Sax.bin       97.0 11       0.00 75.0    1.1
## Sax.dbin      13.1 10       0.22 -6.9  -74.1
```

From the above, we can see that the double binomial model Sax.dbin with one more parameter is significantly better than the simple binomial and represents an adequate fit to the data. The table below displays the fitted values and standardized residuals for both models.

```
results <- data.frame(Sax.df,
           fit.bin=fitted(Sax.bin), res.bin=rstandard(Sax.bin),
           fit.dbin=fitted(Sax.dbin), res.dbin=rstandard(Sax.dbin))
print(results, digits=2)

##    Males Families YlogitY fit.bin res.bin fit.dbin res.dbin
## 1      0        3      30    0.93    1.70      3.0    0.026
## 2      1       24      26   12.09    3.05     23.4    0.136
## 3      2      104      24   71.80    3.71    104.3   -0.036
## 4      3      286      23  258.48    1.87    307.8   -1.492
## 5      4      670      22  628.06    1.94    652.9    0.778
## 6      5     1033      22 1085.21   -1.87   1038.5   -0.202
## 7      6     1343      22 1367.28   -0.75   1264.2    2.635
## 8      7     1112      22 1265.63   -5.09   1185.0   -2.550
## 9      8      829      22  854.25   -1.03    850.1   -0.846
## 10     9      478      23  410.01    3.75    457.2    1.144
## 11    10      181      24  132.84    4.23    176.8    0.371
## 12    11       45      26   26.08    3.42     45.2   -0.039
## 13    12        7      30    2.35    2.45      6.5    0.192
```

Finally, Figure 3.23 shows the rootogram for the double binomial, which can be compared with that for the binomial model shown in Figure 3.21. We can see that the fit is now quite good, particularly in the tails, where the term YlogitY gives additional weight.

```
with(results, rootogram(Families, fit.dbin, names=Males,
                        xlab="Number of males"))
```

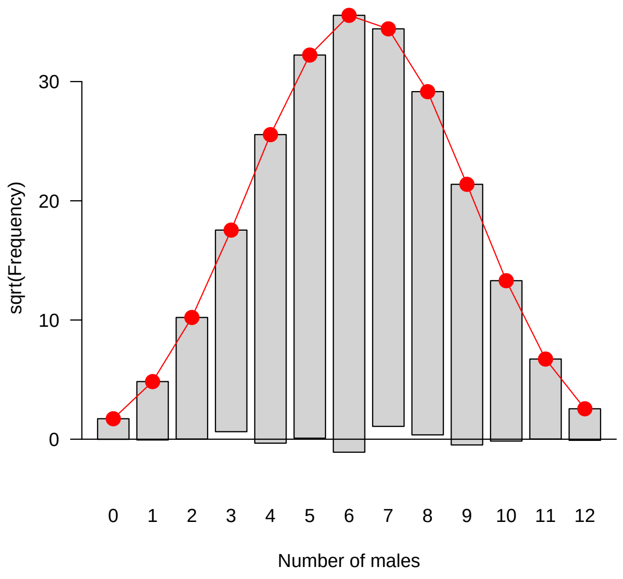


Figure 3.23: Rootogram for the double binomial model for the Saxony data.

ToDo: Interpret the coefficient for YlogitY in terms of dispersion??

△

3.7 Chapter summary

- Discrete distributions typically involve basic *counts* of occurrences of some event occurring with varying *frequency*. The ideas and methods for one-way tables described in this chapter are building blocks for analysis of more complex data.
- The most commonly used discrete distributions include the binomial, Poisson, negative binomial, geometric, and logarithmic series distributions. Happily, these are all members of a family called the power series distributions. Methods of fitting an observed data set to any of these distributions are described, and implemented in the `goodfit()`.
- After fitting an observed distribution it is useful to plot the observed and fitted frequencies. Several ways of making these plots are described, and implemented in the `rootogram()`.
- A heuristic graphical method for identifying which discrete distribution is most appropriate for a given set of data involves plotting ratios kn_k/n_{k-1} against k. These plots are constructed by the function `Ord_plot()`.
- A more robust plot for a Poisson distribution involves plotting a count metameter, $\phi(n_k)$ against k, which gives a straight line (whose slope estimates the Poisson parameter) when the data follows a Poisson distribution. This plot provides robust confidence intervals for individual points and provides a means to assess the influence of individual points on the Poisson parameter. These plots are provided by the function `distplot()`.

- The ideas behind the Poissonness plot can be applied to the other discrete distributions.

3.8 Further reading

3.9 Lab exercises

1. Use the graphical methods illustrated in Section 3.2 to plot a collection of geometric distributions for $p = 0.2, 0.4, 0.6, 0.8$, over a range of values of $k = 0, 1, \ldots 10$.

   (a) With `xyplot()`, try the different plot formats using points connected with lines, as in Figure 3.8, or using points and lines down to the origin, as in the panels of Figure 3.9.

   (b) Also with `xyplot()`, produce one version of a multi-line plot in a single panel that you think shows well how these distributions change with the probability p of success.

   (c) Do the same in a multi-panel version, conditional on p.

2. Use the data set `WomenQueue` to:

   (a) produce plots analogous to those shown in sec:discrete-intro (some sort of bar graph of frequencies)

   (b) check for goodness-of-fit to the binomial distribution using the `goodfit()` methods described in Section 3.3.2.

   (c)

3. Continue Example 3.11 on the distribution of male chilren in families in Saxony by fitting a binomial distribution, $\text{Bin}(n = 12, p = \frac{1}{2})$, specifying equal probability for boys and girls. [Hint: you need to specify both `size` and `prob` values for `goodfit()`.]

   (a) Carry out the GOF test for this fixed binomial distribution. What is the ratio of χ^2/df? What do you conclude?

   (b) Test the additional lack of fit for the model $\text{Bin}(n = 12, p = \frac{1}{2})$ compared to the model $\text{Bin}(n = 12, p = \hat{p})$ where $\hat{p}$ is estimated from the data.

   (c) Use the `plot.gootfit()` method to visualize these two models.

4. For the `Federalist` data, the examples in Section 3.3.1 and Section 3.3.2 showed the negative binomial to provide an acceptable fit. Compare this with the simpler special case of geometric distribution, corresponding to $n = 1$.

   (a) Use `goodfit()` to fit the geometric distribution. [Hint: use `type="nbinomial"`, but specify `size=1` as a parameter.]

   (b) Compare the negative binomial and the geometric models statistically, by a likelihood-ratio test of the difference between these two models.

   (c) Compare the negative binomial and the geometric models visually by hanging rootograms or other methods.

5. Mosteller and Wallace (1963, Table 2.4) give the frequencies, n_k of counts $k = 0, 1, \ldots$ of other selected marker words in 247 blocks of text known to have been written by Alexander Hamilton. The data below show the occurrences of the word *upon*, that Hamilton used much more than did James Madison.

```
count <- 0:5
Freq <- c(129, 83, 20, 9, 5, 1)
```

(a) Read these data into R and construct a 1-way table of frequencies of counts or a matrix or data frame with frequencies in the first column and the corresponding counts in the second column, suitable for use with `goodfit()`.
(b) Fit and plot the Poisson model for these frequencies.
(c) Fit and plot the negative binomial model for these frequencies.
(d) What do you conclude?

6. The data frame `Geissler` in the vcdExtra package contains the complete data from Geissler's (1889) tabulation of family sex composition in Saxony. The table below gives the number of boys in families of size 11.

boys	0	1	2	3	4	5	6	7	8	9	10	11
Freq	8	72	275	837	1540	2161	2310	1801	1077	492	93	24

(a) Read these data into R
(b) Following Example 3.11, use `goodfit()` to fit the binomial model and plot the results. Is there an indication that the binomial does not fit these data?
(c) Diagnose the form of the distribution using the methods described in Section 3.4.
(d) Try fitting the negative binomial distribution, and use the `distplot()` to diagnose whether the negative binomial is a reasonable fit.

ToDo: Cleanup local variables

```
ls()

##  [1] "Arbuthnot"    "Arthritis"    "bin.df"
##  [4] "But.fit1"     "But.fit2"     "Butterfly"
##  [7] "chapters"     "chisq"        "Chisq"
## [10] "count"        "DaytonSurvey" "dice.fit"
## [13] "Diff"         "dp"           "exp"
## [16] "Exp"          "Families"     "Fed.fit0"
## [19] "Fed.fit1"     "Federalist"   "fm"
## [22] "Freq"         "Geissler"     "GOF"
## [25] "gplt"         "GSS"          "GSStab"
## [28] "HairEye"      "HairEyeColor" "HEC"
## [31] "HK.fit"       "HorseKicks"   "horsetab"
## [34] "HSE"          "includeonly"  "includes"
## [37] "JobSat"       "k"            "knitrSet"
## [40] "lambda"       "logit2p"      "Males"
## [43] "mu"           "mycol"        "n"
## [46] "nbin.df"      "nk"           "nk1"
## [49] "NP"           "op"           "ord"
## [52] "ord.df"       "p"            "phat"
## [55] "plt"          "pois.df"      "pred"
## [58] "predicted"    "Prob"         "results"
## [61] "Sax.bin"      "Sax.dbin"     "Sax.df"
## [64] "Sax.fit"      "Saxony"       "Saxony11"
## [67] "soccer.df"    "SpaceShuttle" "spar"
## [70] "tab"          "TV"           "UCB"
## [73] "UKSoccer"     "weight"       "Weldon.df"
## [76] "WeldonDice"   "WomenQueue"   "x"
## [79] "XL"           "XN"           "XP"
## [82] "y"
```

Chapter 4
Two-way contingency tables

The analysis of two-way frequency tables concerns the association between two variables. A variety of specialized graphical displays help to visualize the pattern of association, using area of some region to represent the frequency in a cell. Some of these methods are focused on visualizing an odds ratio (for 2×2 tables), or the general pattern of association, or the agreement between row and column categories.

4.1 Introduction

4.2 Tests of association for two-way tables

4.3 Stratified analysis

4.4 Fourfold display for 2 x 2 tables

4.5 Sieve diagrams

4.6 Association plots

4.7 Observer agreement

4.8 Trilinear plots

4.9 Chapter summary

4.10 Further reading

4.11 Lab exercises

References

Agresti, A. (2002). *Categorical data analysis*. Wiley Series in Probability and Statistics. New York: Wiley-Interscience [John Wiley & Sons], 2nd edn.

Andrews, D. F. and Herzberg, A. M. (1985). *Data: A Collection of Problems from Many Fields for the Student and Research Worker*. New York, NY: Springer-Verlag.

Arbuthnot, J. (1710). An argument for devine providence, taken from the constant regularity observ'd in the births of both sexes. *Philosophical Transactions*, 27, 186–190. Published in 1711.

Bertin, J. (1983). *Semiology of Graphics*. Madison, WI: University of Wisconsin Press. (trans. W. Berg).

Chambers, J. M., Cleveland, W. S., Kleiner, B., and Tukey, P. A. (1983). *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.

Cleveland, W. S. (1993). *Visualizing Data*. Summit, NJ: Hobart Press.

Edwards, A. W. F. (1958). An analysis of geissler's data on the human sex ratio. *Annals of Human Genetics*, 23(1), 6–15.

Fisher, R. A. (1925). *Statistical Methods for Research Workers*. London: Oliver & Boyd.

Fisher, R. A. (1936). Has Mendel's work been rediscovered? *Annals of Science*, 1, 115–âĂŞ137.

Fisher, R. A., Corbet, A. S., and Williams, C. B. (1943). The relation between the number of species and the number of individuals. *Journal of Animal Ecology*, 12, 42.

Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression*. Thousand Oaks CA: Sage, 2nd edn.

Geissler, A. (1889). Beitrage zur frage des geschlechts verhaltnisses der geborenen. *Z. K. Sachsischen Statistischen Bureaus*, 35(1), n.p.

Greenwood, M. and Yule, G. U. (1920). An inquiry into the nature of frequency distributions of multiple happenings, with particular reference to the occurrence of multiple attacks of disease or repeated accidents. *Journal of the Royal Statistical Society, Series A*, 83, 255–279.

Hartigan, J. A. and Kleiner, B. (1984). A mosaic of television ratings. *The American Statistician*, 38, 32–35.

Hilbe, J. (2011). *Negative Binomial Regression*. Cambridge University Press, 2nd edn.

Hoaglin, D. C. (1980). A poissonness plot. *The American Statistician*, 34, 146–149.

Hoaglin, D. C. and Tukey, J. W. (1985). Checking the shape of discrete distributions. In D. C. Hoaglin, F. Mosteller, and J. W. Tukey, eds., *Exploring Data Tables, Trends and Shapes*, chap. 9. New York: John Wiley and Sons.

Jinkinson, R. A. and Slater, M. (1981). Critical discussion of a graphical method for identifying discrete distributions. *The Statistician*, 30, 239–248.

Johnson, N. L., Kotz, S., and Kemp, A. W. (1992). *Univariate Discrete Distributions*. New York, NY: John Wiley and Sons, 2nd edn.

Kemp, A. W. and Kemp, C. D. (1991). Weldon's dice data revisited. *The American Statistician*, 45, 216–222.

Kendall, M. G. and Stuart, A. (1963). *The Advanced Theory of Statistics*, vol. 1. London: Griffin.

Koch, G. and Edwards, S. (1988). Clinical efficiency trials with categorical data. In K. E. Peace, ed., *Biopharmaceutical Statistics for Drug Development*, (pp. 403–451). New York: Marcel Dekker.

Kosambi, D. D. (1949). Characteristic properties of series distributions. *Proceedings of the National Institute of Science of India*, 15, 109–113.

Labby, Z. (2009). Weldon's dice, automated. *Chance*, 22(4), 6–13.

Lee, A. J. (1997). Modelling scores in the Premier League: Is Manchester United really the best? *Chance*, 10(1), 15–19.

Leifeld, P. (2013). texreg: Conversion of statistical model output in r to latex and html tables. *Journal of Statistical Software*, 55(8), 1–24.

Lindsey, J. K. (1995). *Modelling Frequency and Count Data*. Oxford, UK: Oxford University Press.

Lindsey, J. K. and Altham, P. M. E. (1998). Analysis of the human sex ratio by using overdispersion models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(1), 149–157.

Lindsey, J. K. and Mersch, G. (1992). Fitting and comparing probability distributions with log linear models. *Computational Statistics and Data Analysis*, 13, 373–384.

Mosteller, F. and Wallace, D. L. (1963). Inference in an authorship problem. *Journal of the American Statistical Association*, 58(302), 275–309.

Mosteller, F. and Wallace, D. L. (1984). *Applied Bayesian and Classical Inference: The Case of the Federalist Papers*. New York, NY: Springer-Verlag.

Noack, A. (1950). A class of random variables with discrete distributions. *Annals of Mathematical Statistics*, 21, 127–132.

Ord, J. K. (1967). Graphical methods for a class of discrete distributions. *Journal of the Royal Statistical Society, Series A*, 130, 232–238.

Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen by random sampling. *Philosophical Magazine*, 50(5th Series), 157–175.

Skellam, J. G. (1948). A probability distribution derived from the binomial distribution by regarding the probability of success as variable between the sets of trials. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2), 257–261.

Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.

Tufte, E. R. (1990). *Envisioning Information*. Cheshire, CT: Graphics Press.

Tufte, E. R. (1997). *Visual Explanations: Images and Quantities, Evidence and Narrative*. Cheshire, CT: Graphics Press.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Reading, MA: Addison Wesley.

von Bortkiewicz, L. (1898). *Das Gesetz der Kleinen Zahlen*. Leipzig: Teubner.

Wickham, H. (2009). *ggplot2: elegant graphics for data analysis*. Springer New York.

Wickham, H. and Chang, W. (2013). *ggplot2: An implementation of the Grammar of Graphics*. R package version 0.9.3.1.

Wilkinson, L. (2005). *The Grammar of Graphics*. New York: Springer, 2nd edn.

Wimmer, G. and Altmann, G. (1999). *Thesaurus of univariate discrete probability distributions*. Stamm.

Zelterman, D. (1999). *Models for Discrete Data*. New York: Oxford University Press.