Chapter 3
Categorical data

3.1 Introduction

In this chapter, we introduce R commands for organizing, summarizing, and displaying categorical data. We will see that categorical data is conveniently expressed by a special R object called a factor. The `table` function is useful in constructing frequency tables and the `plot` and `barplot` functions are useful in displaying tabulated output. The chi-square goodness-of-fit test for assessing if a vector of counts follows a specified discrete distribution is implemented in the `chisq.test` function. The `cut` function is helpful in dividing a numerical value into a categorical variable using a vector of dividing values. The `table` function with several variables can be used to construct a two-way frequency table and the `prop.table` function can be used to compute conditional proportions to explore the association pattern in the table. Side-by-side and segmented bar charts of conditional probabilities are constructed by the `barplot` function. The hypothesis of independence in a two-way table can be tested by the `chisq.test` function. A special graphical display `mosaicplot` can be used to display the counts in a two-way frequency table and, in addition, show the pattern of residuals from a fit of independence.

3.1.1 Tabulating and plotting categorical data

Example 3.1 (Flipping a coin).

To begin, suppose we flip a coin 20 times and observe the sequence

$$H, T, H, H, T, H, H, T, H, H, T, T, H, T, T, T, H, H, H, T.$$

We are interested in tabulating these outcomes, finding the proportions of heads and tails, and graphing the proportions.

J. Albert and M. Rizzo, *R by Example*, Use R, DOI 10.1007/978-1-4614-1365-3_3,

A convenient way of entering these data in the R console is with the `scan` function. One indicates by the `what=character` argument that character-type data will be entered. By default, this function assumes that "white space" will be separating the individual entries. We complete entering the outcomes by pressing the Enter key on a blank line. The character data is placed in the vector `tosses`.

```
> tosses = scan(what="character")
1: H T H H T H H T H H T T
13: H T T T
17: H H H T
21:
Read 20 items
```

We can tabulate this coin flipping data using the `table` function. The output is a table of frequencies of the different outcomes, H and T.

```
> table(tosses)
tosses
 H  T
11  9
```

We see that 11 heads and 9 tails were flipped. To summarize these counts, one typically computes proportions or relative frequencies. One can obtain these proportions by simply dividing the table frequencies by the number of flips using the `length` function.

```
> table(tosses) / length(tosses)
tosses
   H    T
0.55 0.45
```

There are several ways of displaying these data. First, we save the relative frequency output in the variable `prop.tosses`:

```
> prop.tosses = table(tosses) / length(tosses)
```

Using the `plot` method, we obtain a line graph displayed in Figure 3.1(a).

```
> plot(prop.tosses)
```

Alternately, one can display the proportions by a bar graph using the `barplot` function shown in Figure 3.1(b).

```
> barplot(prop.tosses)
```

The line graph and bar graph both give the same general impression that we obtained similar counts of heads and tails in this coin-tossing experiment.

3.1.2 Character vectors and factors

In the previous section, we illustrated the construction of a *character vector* which is a vector of string values such as "H" and "T." When one has a vector

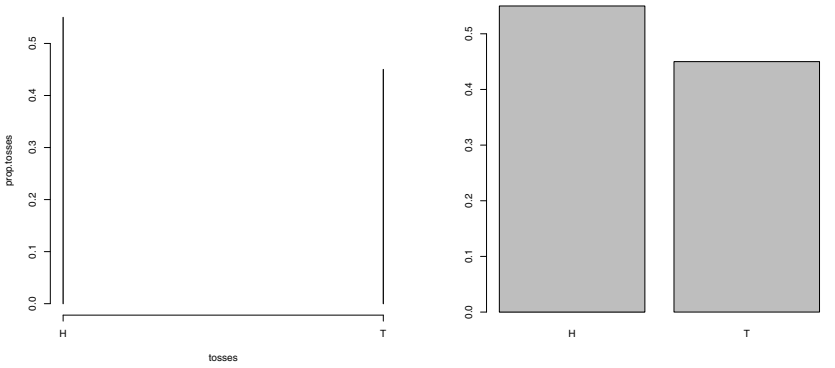


Fig. 3.1 Two displays of the proportions of heads and tails from 20 flips of a coin.

consisting of a small number of distinct values, either numerical or character, a *factor* is a useful way of representing this vector. To define a factor, we start with a vector of values, a second vector that gives the collection of possible values, and a third vector that gives labels to the possible values.

Example 3.2 (Rolling a die).

As a simple example, suppose we wish to collect several rolls of a die. We observe seven rolls of the die and save the rolls in the vector `y`.

```
> y = c(1, 4, 3, 5, 4, 2, 4)
```

For die rolls, we know that the possible rolls are 1 through 6 and we store these in the vector `possible.rolls`:

```
> possible.rolls = c(1, 2, 3, 4, 5, 6)
```

We wish to label the rolls by the words "one", ..., "six" – we place these labels in the vector `labels.rolls`:

```
> labels.rolls = c("one", "two", "three", "four", "five", "six")
```

We now are ready to construct the factor variable `fy` using the function `factor`:

```
> fy = factor(y, levels=possible.rolls, labels=labels.rolls)
```

By displaying the vector `fy`, we see the difference between a character vector and a factor.

```
> fy
[1] one   four  three five  four  two   four
Levels: one two three four five six
```

Note that the numerical roll values in `y` have been replaced by the factor labels in `fy`. Also, note that the display of the factor variable shows the *levels*, the possible values of the die roll. Suppose we construct a frequency table of the factor.

```
> table(fy)
fy
  one   two three  four  five   six
    1     1     1     3     1     0
```

Note that frequencies of all possible rolls of the die are displayed. In many situations, we wish to display the frequencies of categories such as "six" that are possible but have not been observed.

In the example to follow, a datafile is read that contains a character variable. When a data frame is created in R (say, using the `read.table` function), by default all variables consisting of character values are automatically converted to factors.

3.2 Chi-square Goodness-of-Fit Test

Example 3.3 (Weldon's dice).

"Weldon's Dice" is a famous data set published in Karl Pearson's 1900 paper [39] that introduced the chi-square goodness-of-fit test. At that time (before electronic computers) the English biologist Walter F. R. Weldon used dice to generate random data, recording the results of 26,306 rolls of 12 dice. See "Weldon's Dice, Automated" [29] for more details and results of an automated version of the experiment.

In the results, Weldon considered five or six dots showing among the 12 dice to be "successes" and other results "failures." If a single die is fair, then each face is equally likely to occur, so that the probability of a success (five or six) is 1/3. The total number of successes among 12 fair dice is a binomial random variable with success probability 1/3. The binomial probabilities are given by the `dbinom` function:

```
> k = 0:12
> p = dbinom(k, size=12, prob=1/3)
```

In 26,306 rolls of 12 fair dice, the expected outcomes (rounded to the nearest integer) would be:

```
> Binom = round(26306 * p)
> names(Binom) = k
```

Labels for the binomial counts were applied with the `names` function. The "Weldon's Dice" data are entered below.

```
> Weldon = c(185, 1149, 3265, 5475, 6114, 5194, 3067,
+   1331, 403, 105, 14, 4, 0)
> names(Weldon) = k
```

The binomial counts, data, and deviations between the binomial and observed counts can be summarized for display several ways; here we use the `data.frame` function. To combine data in a data frame we simply list the data vectors separated by commas. Here we also assigned a name "Diff" for the difference between the observed (Weldon) and expected (binomial) counts.

```
> data.frame(Binom, Weldon, Diff=Weldon - Binom)
   Binom Weldon Diff
0    203    185  -18
1   1216   1149  -67
2   3345   3265  -80
3   5576   5475 -101
4   6273   6114 -159
5   5018   5194  176
6   2927   3067  140
7   1255   1331   76
8    392    403   11
9     87    105   18
10    13     14    1
11     1      4    3
12     0      0    0
```

A visual comparison of the observed and expected counts can be made in several ways. To display the two bar plots of frequencies side by side we combine our data into a matrix using `cbind` and specify `beside=TRUE` in the `barplot` function.

```
> counts = cbind(Bin, Weldon)
> barplot(counts, beside=TRUE)
```

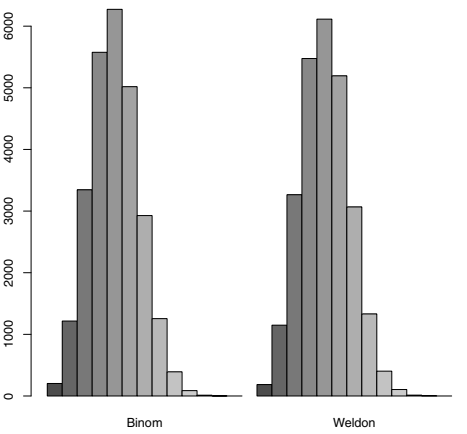


Fig. 3.2 Bar plots of expected and observed frequencies in Weldon dice example.

The barplots in Figure 3.2 appear to agree approximately. For a comparison of observed and fitted data, however, it is somewhat easier to interpret the data on a single plot. Another possible plot for comparing the observed and expected counts is produced by:

```
> plot(k, Binom, type="h", lwd=2, lty=1, ylab="Count")
> lines(k + .2, Weldon, type="h", lwd=2, lty=2)
> legend(8, 5000, legend=c("Binomial", "Weldon"),
+    lty=c(1,2), lwd=c(2,2))
```

The extra arguments for plotting the binomial counts are `type="h"` (plot vertical lines like a barplot), `lwd=2` (line width is doubled), and `lty=1` (choose a solid line type). The `lines` function is used to overlay vertical lines for the corresponding observed counts. These lines are drawn slightly to the right of the expected count lines by adding a small value of 0.2 to the variable `k` and drawn using the alternative line type `lty=2`. This plot is shown in Figure 3.3.

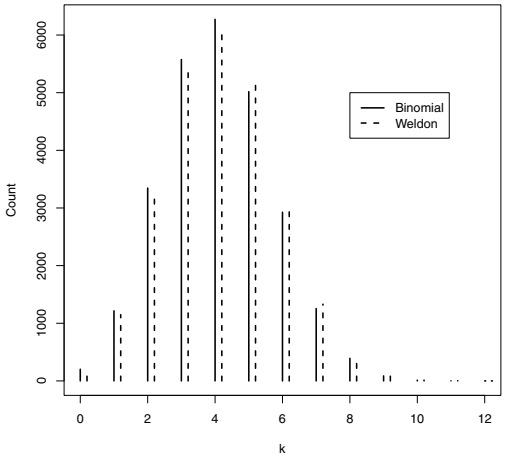


Fig. 3.3 Line plot of expected and observed frequencies in Weldon dice example.

A chi-square goodness-of-fit test can be applied to test whether the data are consistent with the 'fair dice' binomial model. However, as we learn in basic statistics, the expected cell counts should be large — say, at least 5 for all cells. For a chi-square test we should collapse the categories corresponding to 10, 11, and 12 successes into a single category. This is accomplished with the following code.

```
>  cWeldon = c(Weldon[1:10], sum(Weldon[11:13]))
>  cWeldon
   0    1    2    3    4    5    6    7    8    9
 185 1149 3265 5475 6114 5194 3067 1331  403  105   18
```

One can now apply a chi-square goodness-of-fit test. We find the binomial probabilities for the first nine categories using the vector of probabilities stored in `p`. The sum of probabilities must equal 1, and this determines the probability for the final category. We use the `chisq.test` function to test the null hypothesis that the true model is binomial.

```
>  probs = c(p[1:10], 1 - sum(p[1:10]))
>  chisq.test(cWeldon, p=probs)
        Chi-squared test for given probabilities

data:  cWeldon
X-squared = 35.4943, df = 10, p-value = 0.0001028
```

The test computes a p-value of less than 0.001. Therefore, we conclude the experimental results that Weldon obtained are not consistent with the binomial model.

To better understand why the test rejects the binomial model, it is helpful to examine the Pearson residuals defined by

$$residual = \frac{observed - expected}{\sqrt{expected}},$$

where *observed* and *expected* denote respectively the observed and expected counts in a particular cell. To obtain these residuals, we save the results of the chi-square test in the variable `test` and the component `residuals` contains the vector of residuals. Using the `plot` function, we display the residuals as a function of `k` and overlay (using the `abline` function) a horizontal line at zero.

```
> test = chisq.test(cWeldon, p=probs)
> plot(0:10, test$residuals,
+   xlab="k", ylab="Residual")
> abline(h=0)
```

We see from Figure 3.4 that the observed Weldon counts are smaller than the binomial expected for values of k four or smaller, and larger than the expected counts for values of k larger than 4. One possible explanation for this conclusion is that the dice are not perfectly balanced.

3.3 Relating Two Categorical Variables

3.3.1 Introduction

Example 3.4 (The twins dataset).

Ashenfelter and Krueger [3] describe an interesting study to address the question "how much will an additional year of schooling raise one's income?" There are several difficulties in learning about the relationship be-

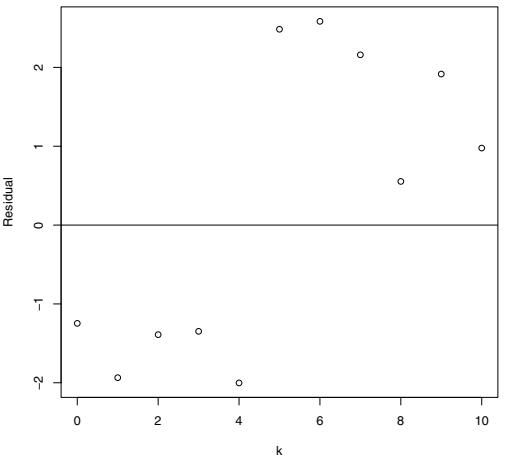


Fig. 3.4 Graph of the Pearson residuals from the chi-square test for the Weldon example.

tween schooling and income. First, there are many variables besides schooling that relate to a person's income such as gender, socioeconomic status, and intelligence, and it is difficult to control for these other variables in this observational study. Second, it can be difficult to obtain truthful information about a person's schooling; people are more likely to report a higher level than they actually attain. The errors in obtaining actual educational levels can lead to biased estimates of the relationship between schooling and income. To address these concerns, these researchers interviewed twins, collecting information about income, education, and other background variables. Monozygotic twins (twins from one egg) have identical family backgrounds and they provide a good control for confounding variables. Also information about a person's education was obtained from a twin (self-reported) and also from his/her twin (cross-reported). By having two education measurements, one is able to estimate the bias from not getting a truthful response.

3.3.2 Frequency tables and graphs

We illustrate several statistical methods for categorical variable as a preliminary exploration of this twins dataset. We begin by reading in the dataset `twins.dat.txt` and storing it in the data frame `twn`:

```
> twn = read.table("twins.dat.txt", header=TRUE,
+    sep=",", na.strings=".")
```

There were 183 pairs of twins who were interviewed in this study. In each pair of twins, one twin was randomly assigned to "twin 1" and the other is called "twin 2." The variables EDUCL and EDUCH give the self-reported education (in years) for twin 1 and twin 2. (In the variable definition, the last letter of "L" refers to twin 1 and "H" refers to twin 2.) We can obtain frequency tables of the education years of the twins by two applications of the `table` function.

```
> table(twn$EDUCL)
 8 10 11 12 13 14 15 16 17 18 19 20
 1  4  1 61 21 30 11 37  1 10  3  3
> table(twn$EDUCH)
 8  9 10 11 12 13 14 15 16 17 18 19 20
 2  1  2  1 65 22 22 15 33  2 11  2  5
```

The education years for both twins show much variation and the year values with large frequencies are 12, corresponding to a high school degree, and 16, corresponding to a college degree.

Since there are so many educational year values, it is useful to categorize this variable into a smaller number of more meaningful levels. Suppose we say that a person's educational level is "high school" if he/she has 12 years of education, "some college" if the years are between 13 and 15, "college degree" for 16 years, and "graduate school" if the years are greater than 16. The `cut` function is very helpful for creating these new categories. In `cut`, the first argument is the variable to be changed, the argument `breaks` is a vector defining the breakpoints for the categories, and the argument `labels` is a character vector with the labels for the new categories. We use this function twice, once for twin 1's educational years and again for twin 2's educational years.

```
> c.EDUCL = cut(twn$EDUCL, breaks=c(0, 12, 15, 16, 24),
+   labels=c("High School", "Some College", "College Degree",
+   "Graduate School"))
> c.EDUCH = cut(twn$EDUCH, breaks=c(0, 12, 15, 16, 24),
+   labels=c("High School", "Some College", "College Degree",
+   "Graduate School"))
```

We tabulate the the educational levels for twin 1 using the `table` function, find relative frequencies by the `prop.table` function, and construct a bar graph of the relative frequencies using the `barplot` function. The resulting graph is shown in Figure 3.5. We see that approximately 70% of the first twins fall within the high school or some college levels.

```
> table(c.EDUCL)
c.EDUCL
    High School    Some College  College Degree Graduate School
             67              62              37              17
> prop.table(table(c.EDUCL))
c.EDUCL
    High School    Some College  College Degree Graduate School
     0.36612022      0.33879781      0.20218579      0.09289617
> barplot(prop.table(table(c.EDUCL)))
```

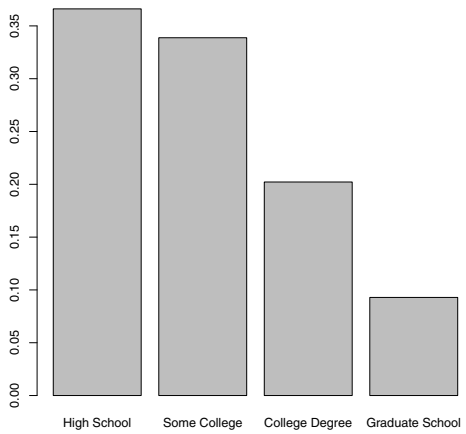


Fig. 3.5 Bar graph of the educational levels for the first twin in the twins study

An alternative graphical display of table counts, a *mosaic plot*, is constructed using the function `mosaicplot`. In this display (shown in Figure 3.6), the total count, represented by a solid rectangle, is divided into vertical regions corresponding to the counts for the different educational levels. (Friendly [17] gives some history of this display.) Both the mosaic plot and the bar graph tell the same story – most of the first twins are in the high school and some college categories – but we will shortly see that the mosaic plot is especially helpful when we classify people with respect to two categorical variables.

```
> mosaicplot(table(c.EDUCL))
```

3.3.3 Contingency tables

The exploration of years of schooling in the previous section raises an interesting question. Is the educational level of twin 1 related to the educational level for twin 2? One can answer this question by constructing a contingency table of the educational levels for the two twins. This is easily constructed by the `table` function with the two variables `c.EDUCL` and `c.EDUCH` as arguments.

```
> table(c.EDUCL, c.EDUCH)
                c.EDUCH
c.EDUCL          High School Some College College Degree Graduate School
  High School              47           16              2                2
```

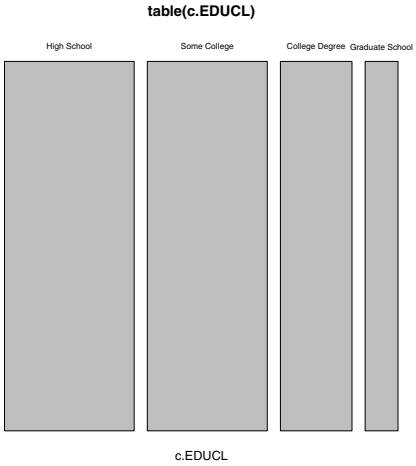


Fig. 3.6 Mosaic plot of the educational levels for the first twin in the twins study

Some College	18	32	8	4
College Degree	5	10	18	4
Graduate School	1	1	5	10

Note from the contingency table that there are large counts along the diagonal of the table where the twins have the same self-reported educational levels. What proportion of twins have the same level? We store the table in the variable `T1` and extract the diagonal counts using the `diag` function:

```
> T1=table(c.EDUCL, c.EDUCH)
> diag(T1)
    High School    Some College  College Degree Graduate School
             47              32              18              10
```

We can compute the proportion of "same educational level" twins by two applications of the `sum` function, one on the vector of diagonal elements, and a second on the entire table.

```
> sum(diag(T1)) / sum(T1)
[1] 0.5846995
```

We see that about 58% of the twins have the same educational level.

One can graphically display the table by a mosaic plot constructed using the `plot` method for a table. (See Figure 3.7.)

```
> plot(T1)
```

This display is constructed in a two-step process. As in the first example, one first partitions a grey square into vertical bars where the widths of the bars correspond to the educational level counts for twin 1. Then each of the

vertical bars is divided into pieces where the heights of the pieces correspond to the education counts for the second twin. The areas of the regions in the mosaic plot correspond to the counts in the contingency table. Observe from Figure 3.7 that the two largest areas correspond to the "high school, high school" and "some college, some college" counts. This means that most of the twins either both had a high school or both had "some college" background.

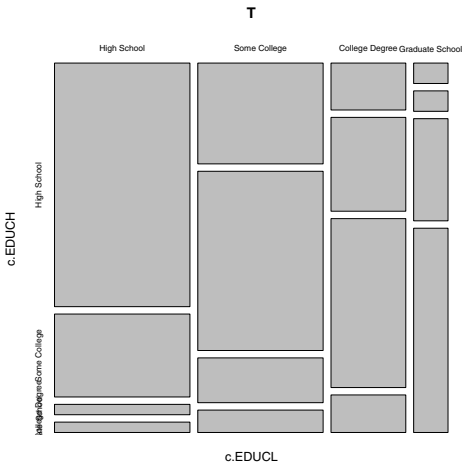


Fig. 3.7 Mosaic plot of the educational levels for the twins in the twins study.

3.4 Association Patterns in Contingency Tables

3.4.1 Constructing a contingency table

The purpose of the twins study was to explore the relationship of educational level with salary. We do some initial exploration by focusing on the data for the first twin. The variable `HRWAGEL` contains the hourly wage (in dollars). If one graphs the hourly wages by say, a histogram, one will observe the shape of the wages is right-skewed. We divide the wages into four groups using the `cut` function with break points 0, 7, 13, 20, and 150. We chose these break points so we would have roughly the same number of twins in each class interval. We assign the categorized wage to the variable `c.wage`.

```
> c.wage = cut(twn$HRWAGEL, c(0, 7, 13, 20, 150))
```

We construct a frequency table of the categorized wage by the `table` function.

```
> table(c.wage)
c.wage
   (0,7]   (7,13]  (13,20] (20,150]
      47       58       38       19
```

There were 21 twins who did not respond to the wage question, so there were $183 - 21 = 162$ recorded wages.

To investigate the relationship of education with salary, we construct a two-way contingency table by another application of `table`; the first variable will appear as rows in the table and the second variable as columns.

```
> table(c.EDUCL, c.wage)
                 c.wage
c.EDUCL           (0,7] (7,13] (13,20] (20,150]
  High School        23     21      10         1
  Some College       15     23      12         5
  College Degree      7     12      14         3
  Graduate School     2      2       2        10
```

We see that there were 23 people with a high school educational level and who are earning \$7 or less per week, there were 21 people with a high school level and who are earning between \$7 and \$13 per week, and so on.

To quantify the relationship between education and salary, one can compute the proportion of different wage categories (column) for each educational level (row). This can be done using the `prop.table` function. The arguments are the table and the margin; if we use `margin = 1`, the proportions of each row of the table will be computed, and `margin = 2` the proportions of each column will be computed. Since we wish to compute proportions of different wages for each educational level, we first save the table in the variable `T2`, and use `prop.table` with arguments `T2` and `margin = 1`.

```
> T2 = table(c.EDUCL, c.wage)
> prop.table(T2, margin=1)
                 c.wage
c.EDUCL                (0,7]     (7,13]    (13,20]   (20,150]
  High School     0.41818182 0.38181818 0.18181818 0.01818182
  Some College    0.27272727 0.41818182 0.21818182 0.09090909
  College Degree  0.19444444 0.33333333 0.38888889 0.08333333
  Graduate School 0.12500000 0.12500000 0.12500000 0.62500000
```

Of the high school students, we see from the table that 42% earned between 0 and \$7, 32% earned between \$7 and \$13, and 18% earned between \$13 and \$20. Likewise, the table shows the proportions of students earning the different wage categories for each of the "Some College," "College Degree," and "Graduate School" educational levels.

3.4.2 Graphing patterns of association

There are several useful graphs for displaying the conditional proportions in this contingency table. One way of displaying a proportion vector is a *segmented bar chart* where one divides a single bar into regions where the region areas correspond to the proportions. The `barplot` function, when applied to a matrix, will construct segmented bar charts for the column vectors of the matrix. If `P` denotes the variable containing our proportion matrix, we are interested in graphing the row (not column) vectors of `P`. So we first take the transpose of `P` (using the `t` function) and then apply the `barplot` function. We add several arguments in this function: we add a label of "PROPORTION" along the y-axis, and add a legend that indicates the color of each of the four wage categories. The resulting display is shown in Figure 3.8.

```
> P = prop.table(T2, 1)
> barplot(t(P), ylim=c(0, 1.3), ylab="PROPORTION",
+   legend.text=dimnames(P)$c.wage,
+  args.legend=list(x = "top"))
```

Note the areas of the lighter colored regions, corresponding to higher wages, get larger from left to right, indicating that higher educational levels have higher wages.

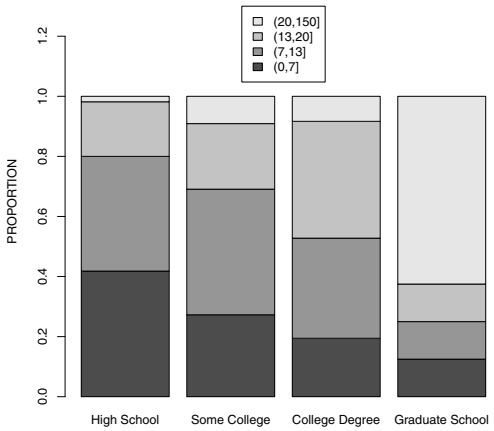


Fig. 3.8 Segmented bar chart of the wage categories for people in four educational levels.

Another useful display are side-by-side barplots, where the proportions for a single educational level are displayed as a bar chart. This display is constructed using the `barplot` using the `beside=TRUE` argument; see Figure

3.9. As in the previous graph, we add a legend that indicates the color of the bars corresponding to the four wage categories. Lower wages are colored using darker bars. We see that the darkest bars, corresponding to the lowest wage category, are predominant for the High School and Some College categories, and are unlikely for the Graduate College category.

```
> barplot(t(P), beside=T,  legend.text=dimnames(P)$c.wage,
+   args.legend=list(x="topleft"), ylab="PROPORTION")
```

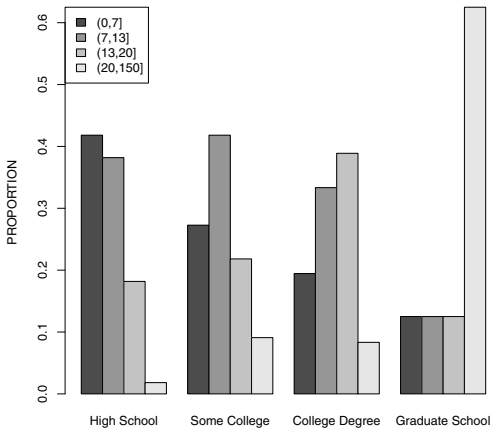


Fig. 3.9 Side-by-side bar charts of the wage categories for people in four educational levels.

3.5 Testing Independence by a Chi-square Test

In the previous section, several methods for exploring the relationship between educational level and wage were illustrated. A more formal way of investigating the relationship between the two categorical variables is by a test of independence. If the variables educational level and wage are *independent*, this means that the probabilities of a twin earning the four wage categories will not depend on his/her education background. Based on our exploratory work, we strongly suspect that educational level and wage are *not* independent – a higher educational background appears to be associated with higher wages – but we will see that this statistical test will give new insight on how education and income are related.

The traditional test of independence is based on Pearson's chi-square statistic. If we test the hypothesis

$$H : \text{education background and wage are independent,}$$

we compute estimated expected counts under the assumption that H is true. If we let *observed* denote the table of counts and *expected* denote the table of expected counts, then the Pearson statistic is defined by

$$X^2 = \sum_{\text{all cells}} \frac{(observed - expected)^2}{expected}.$$

The Pearson statistic measures the deviation of the observed counts from the expected counts and one rejects the hypothesis of independence for large values of the statistic X^2. If the hypothesis of independence is true, then X^2 will have, for large samples, an approximate chi-square distribution with degrees of freedom given by $df =$ (number of rows $-$ 1) $\times$ (number of columns $-$ 1). Suppose that the computed value of X^2 for our data is equal to X^2_{obs}. The p-value is the probability of observing X^2 at least as extreme as X^2_{obs}; applying the chi-square approximation, this p-value is the probability that a chi-square(df) random variable exceeds X^2_{obs}:

$$p - value = Prob(\chi^2_{df} \geq X^2_{obs}).$$

In R, recall we have the first twin's educational level stored in the variable `c.EDUCL` and the wage category stored in `c.wage`. The contingency table classifying twins by educational level and wage is stored in the variable `T2`:

```
> T2 = table(c.EDUCL, c.wage)
```

We perform a test of independence using the `chisq.test` function with the table `T2` as the sole argument. We save the test calculations in the variable `S`.

```
> S  = chisq.test(T2)
 Warning message:
In chisq.test(T2) : Chi-squared approximation may be incorrect
```

The warning tells that the accuracy of the chi-square approximation is in doubt due to a few small expected counts in the table. The results of the test are obtained by simply printing this variable.

```
> print(S)
        Pearson's Chi-squared test

data:  T2
X-squared = 54.5776, df = 9, p-value = 1.466e-08
```

It is instructive to confirm the calculations of this statistical test. One first computes the estimated expected counts of the table under the independence

assumption – these expected counts are stored in the component `expected` of `S` that we display.

```
 > S$expected
                 c.wage
c.EDUCL               (0,7]    (7,13]   (13,20] (20,1e+03]
  High School     15.956790 19.691358 12.901235   6.450617
  Some College    15.956790 19.691358 12.901235   6.450617
  College Degree  10.444444 12.888889  8.444444   4.222222
  Graduate School  4.641975  5.728395  3.753086   1.876543
```

The observed counts are stored in the table `T2`. We can compute the test statistic by perfoming the operation "observed minus expected squared divided by expected" for all counts and summing over all cells.

```
 > sum((T2 - S$expected)^2 / S$expected)
 [1] 54.57759
```

Our answer, 54.57759, agrees with the displayed value of `X-squared` from the `chisq.test` output. Also we can check the computation of the p-value. The function `pchisq` computes the cdf of a chi-square random variable. Here the number of rows and number of columns of the table are both 4, and the degrees of freedom is equal to $df = (4-1)(4-1) = 9$. Since the distribution of the test statistic X^2 has approximately a chi-square(9) distribution under independence, the p-value is (approximately) the probability a $\chi^2(9)$ variate exceeds 54.57759 which is given by

```
> 1 - pchisq(54.57759, df=9)
[1] 1.465839e-08
```

This also agrees with the p-value given in the `chisq.test` output. This p-value is very small, so clearly the hypothesis of independence of educational level and wage category is rejected.

All of the calculations related to this chi-square test are stored in the variable `S`. One can view all components of `S` using the `names` function.

```
> names(S)
[1] "statistic" "parameter" "p.value"   "method"    "data.name" "observed"
[7] "expected"  "residuals"
```

One useful component is `residuals` – this contains the table of Pearson residuals, where a particular residual is defined by

$$residual = \frac{observed - expected}{\sqrt{expected}},$$

and *observed* and *expected* are, respectively, the count and the estimated expected count in that cell. By displaying the table of Pearson residuals, we see where the counts deviate from the independence model.

```
> S$residuals
                 c.wage
c.EDUCL               (0,7]     (7,13]    (13,20]   (20,150]
```

```
High School      1.7631849  0.2949056 -0.8077318 -2.1460758
Some College    -0.2395212  0.7456104 -0.2509124 -0.5711527
College Degree  -1.0658020 -0.2475938  1.9117978 -0.5948119
Graduate School -1.2262453 -1.5577776 -0.9049176  5.9300942
```

Informally, any residual larger than 2 in absolute value indicates a "significant" deviation from independence. It is interesting that using this criterion, there are two "large" residuals given in the rightmost column of the table. The residual of -2.14 indicates that there are fewer High School people earning wages over \$20 than anticipated by the independence model. In addition, the residual of 5.93 indicate there are more Graduate School people earning over \$20 that we would expect for independent variables. We can summarize the association by saying that educational level matters most in the highest wage category.

One can display the significant residuals by means of a mosaic plot. The `mosaicplot` function is first applied with the `shade=FALSE` (default) argument and the areas of the rectangles in the display in Figure 3.10(a) correspond to the counts in the table classifying twins by educational level and wage category.

```
> mosaicplot(T2, shade=FALSE)
```

If the `shade=TRUE` argument is used, one obtains an *extended* mosaic plot displayed in Figure 3.10(b). The border type and the shading of the rectangles relate to the sizes of the Pearson residuals. The two shaded rectangles correspond to the same two large residuals that we found by inspection of the table of residuals.

```
> mosaicplot(T2, shade=TRUE)
```

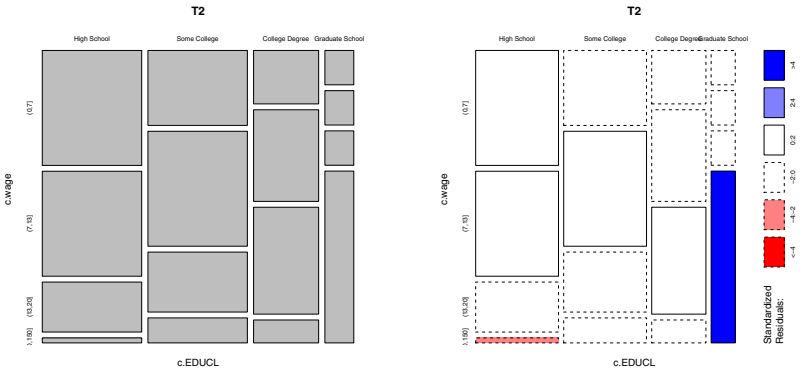
Exercises

3.1 (Fast food eating preference). Fifteen students in a statistics class were asked to state their preference among the three restaurtants Wendys, McDonalds, and Subway. The responses for the students are presented below.

```
Wendys    McDonalds Subway    Subway    Subway    Wendys
Wendys    Subway    Wendys    Subway    Subway    Subway
Subway    Subway    Subway
```

a. Use the `scan` function to read these data into the R command window.
b. Use the `table` function to find the frequencies of students who prefer the three restaurants.
c. Compute the proportions of students in each category.
d. Construct two different graphical displays of the proportions.

3.2 (Dice rolls). Suppose you roll a pair of dice 1000 times.



(a) (b)

Fig. 3.10 Mosaic plots of the table categorizing twin 1 by educational level and wage category. The left plot displays a basic mosaic plot and the right plot shows an extended mosaic plot where the shaded rectangles in the lower left and lower right sections of the graph correspond to large values of the corresponding Pearson residuals.

a. One can simulate 1000 rolls of a fair die using the R function `sample(6, 1000, replace=TRUE)`. Using this function twice, store 1000 simulated rolls of the first die in the variable `die1` and 1000 simulated rolls of the second die in the variable `die2`.
b. For each pair of rolls, compute the sum of rolls, and store the sums in the variable `die.sum`.
c. Use the `table` function to tabulate the values of the sum of die rolls. Compute the proportions for each sum value and compare these proportions with the exact probabilities of the sum of two die rolls.

3.3 (Does baseball hitting data follow a binomial distribution?). Albert Pujols is a baseball player who has n opportunities to hit in a single game. If y denotes the number of hits for a game, then it is reasonable to assume that y has a binomial distribution with sample size n and probability of success $p = 0.312$, where 0.312 is Pujols' batting average (success rate) for the 2010 baseball season.

a. In 70 games Pujols had exactly $n = 4$ opportunities to hit and the number of hits y in these 70 games is tabulated in the following table. Use the `dbinom` function to compute the expected counts and the `chisq.test` function to test if the counts follow a binomial(4, 0.312) distribution.
b. In 25 games Pujols had exactly $n = 5$ opportunities to hit and the number of hits y in these 25 games is shown in the table below. Use the `chisq.test` function to test if the counts follow a binomial(5, 0.312) distribution.

Number of hits	0	1	2	3 or more
Frequency	17	31	17	5

Number of hits	0	1	2	3 or more
Frequency	5	5	4	11

3.4 (Categorizing ages in the twins dataset). The variable `AGE` gives the age (in years) of twin 1.

a. Use the `cut` function on `AGE` with the breakpoints 30, 40, and 50 to create a categorized version of the twin's age.
b. Use the `table` function to find the frequencies in the four age categories.
c. Construct a graph of the proportions in the four age categories.

3.5 (Relating age and wage in the twins dataset). The variables `AGE` and `HRWAGEL` contain the age (in years) and hourly wage (in dollars) of twin 1.

a. Using two applications of the `cut` function, create a categorized version of `AGE` using the breakpoints 30, 40, and 50, and a categorized version of `HRWAGEL` using the same breakpoints as in Section 3.3.
b. Using the categorized versions of `AGE` and `HRWAGEL`, construct a contingency table of the two variables using the function `table`.
c. Use the `prop.table` function to find the proportions of twins in each age class that have the different wage groups.
d. Construct a suitable graph to show how the wage distribution depends on the age of the twin.
e. Use the conditional proportions in part (c) and the graph in part (d) to explain the relationship between age and wage of the twins.

3.6 (Relating age and wage in the twins dataset, continued).

a. Using the contingency table of the categorized version of `AGE` and `HRWAGEL` and the function `chisq.test`, perform a test of independence of age and wage. Based on this test, is there significant evidence to conclude that age and wage are dependent?
b. Compute and display the Pearson residuals from the test of independence. Find the residuals that exceed 2 in absolute value.
c. Use the function `mosaicplot` with the argument `shade=TRUE` to construct a mosaic plot of the table counts showing the extreme residuals.
d. Use the numerical and graphical work from parts (b) and (c) to explain how the table of age and wages differs from an independence structure.

3.7 (Dice rolls, continued). Suppose you roll a pair of dice 1000 times and you are interested in the relationship between the maximum of the two rolls and the sum of the rolls.

a. Using the `sample` function twice, simulate 1000 rolls of two dice and store the simulated rolls in the variables `die1` and `die2`.
b. The `pmax` function will return the parallel maximum value of two vectors. Using this function, compute the maximum for each of the 1000 pair of rolls and store the results in the vector `max.rolls`. Similarly, store the sum for each pair of rolls and store the sums in the vector `sum.rolls`.
c. Using the `table` function, construct a contingency table of the maximum roll and the sum of rolls.
d. By the computation of conditional proportions, explore the relationship between the maximum roll and the sum of rolls.

3.8 (Are the digits of π random?). The National Institute of Standards and Technology has a web page that lists the first 5000 digits of the irrational number π. One can read these digits into R by means of the script

```
pidigits =
read.table("http://www.itl.nist.gov/div898/strd/univ/data/PiDigits.dat",
  skip=60)
```

a. Use the `table` function to construct a frequency table of the digits 1 through 9.
b. Construct a bar plot of the frequencies found in part (a).
c. Use the chi-square test, as implemented in the `chisq.test` function, to test the hypothesis that the digits 1 through 9 are equally probable in the digits of π.