Abstract
This tutorial describes the creation and manipulation of frequency and contingency tables from categorical variables, along with tests of independence, measures of association, and methods for graphically displaying results. The framework is provided by the R package vcd, but other packages are used to help with various tasks. The vcdExtra package extends the graphical and statistical methods provided by vcd.

This package is now the main support package for the book Discrete Data Analysis with R: Visualizing and Modeling Techniques for Categorical and Count Data (Friendly and Meyer 2016). The web page for the book, ddar.datavis.ca, gives further details.

Contents

1 Introduction ............................ 1
2 Creating frequency tables .............. 2
  2.1 Ordered factors ..................... 4
  2.2 structable() ....................... 6
  2.3 table() and friends ............... 7
  2.4 xtabs() .......................... 7
  2.5 Collapsing over factors .......... 8
  2.6 Collapsing levels .................. 10
  2.7 Converting ........................ 11
  2.8 A complex example ............... 12
3 Tests of Independence .......... 14
  3.1 CrossTable ......................... 14
  3.2 Chi-square test ................... 15
  3.3 Fisher Exact Test ................. 15
  3.4 Mantel-Haenszel test ............. 16
  3.5 CMH tests: ordinal factors ....... 18
4 Loglinear Models ................. 22
  4.1 Fitting with loglm() ............ 24
  4.2 Fitting with glm() and gnm() .... 25
  4.3 Non-linear terms ................. 27
5 Mosaic plots ......................... 28
  5.1 Mosaics for loglinear models .... 29
  5.2 Mosaics for glm() and gnm() models 31
  5.3 Mosaic tips and techniques ...... 32
  5.3.1 Changing labels .............. 32
6 Continuous predictors ............ 33
  6.1 Spine and conditional density plots 33
  6.2 Model-based plots ................ 35

1 Introduction
This tutorial, part of the vcdExtra package, describes how to work with categorical data in the context of fitting statistical models in R and visualizing the results using the vcd and vcdExtra packages. It focuses first on methods and tools for creating and manipulating R data objects which represent frequency and contingency tables involving categorical variables.
Further sections describe some simple methods for calculating tests of independence and measures of association among categorical variables, and also methods for graphically displaying results.

There is much more to the analysis of categorical data than is described here, where the emphasis is on cross-tabulated tables of frequencies (“contingency tables”), statistical tests, associated loglinear models, and visualization of how variables are related.

A more general treatment of graphical methods for categorical data is contained in the book, *Discrete Data Analysis with R: Visualizing and Modeling Techniques for Categorical and Count Data* (Friendly and Meyer 2016). An earlier book using SAS is *Visualizing Categorical Data* (Friendly 2000), for which *vcd* is a partial R companion, covering topics not otherwise available in R. On the other hand, the implementation of graphical methods in *vcd* is more general in many respects than what I provided in SAS. Statistical models for categorical data in R have been extended considerably with the *gnm* package for generalized nonlinear models. The *vcdExtra* package extends *vcd* methods to models fit using *glm()* and *gnm()*.

A more complete theoretical description of these statistical methods is provided in Agresti’s (2002; 2013) *Categorical Data Analysis*. For this, see the Splus/R companion by Laura Thompson, http://www.stat.ufl.edu/~aa/cda/Thompson_manual.pdf and Agresti’s support web page, http://www.stat.ufl.edu/~aa/cda/cda.html.

## 2 Creating and manipulating frequency tables

R provides many methods for creating frequency and contingency tables. Several are described below. In the examples below, we use some real examples and some anonymous ones, where the variables A, B, and C represent categorical variables, and X represents an arbitrary R data object.

The first thing you need to know is that categorical data can be represented in three different forms in R, and it is sometimes necessary to convert from one form to another, for carrying out statistical tests, fitting models or visualizing the results. Once a data object exists in R, you can examine its complete structure with the *str()* function, or view the names of its components with the *names()* function.

**case form** a data frame containing individual observations, with one or more factors, used as the classifying variables. In case form, there may also be numeric covariates. The total number of observations is *nrow(X)*, and the number of variables is *ncol(X)*.

**Example:** 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.

```r
> names(Arthritis) # show the variables
[1] "ID"       "Treatment" "Sex"       "Age"       "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 63 63 ...
$ Improved : Ord.factor w/ 3 levels "None"<"Some"<...:

> head(Arthritis,5) # first 5 observations, same as Arthritis[1:5,]
```
A data frame containing one or more factors, and a frequency variable, often called `Freq` or `count`. The total number of observations is `sum(X$Freq), sum(X[,"Freq"])` or some equivalent form. The number of cells in the table is `nrow(X)`.

**Example:** 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.

```r
> # 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
```

A matrix, array or table object, whose elements are the frequencies in an n-way table. The variable names (factors) and their levels are given by `dimnames(X)`. The total number of observations is `sum(X)`. The number of dimensions of the table is `length(dimnames(X))`, and the table sizes are given by `sapply(dimnames(X), length)`.

**Example:** The HairEyeColor is stored in table form in `vcd`.

```r
> str(HairEyeColor)      # show the structure
```

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

`frequency form`
Enter frequencies in a matrix, and assign `dimnames`, giving the variable names and category labels. Note that, by default, `matrix()` uses the elements supplied by `columns` in the result, unless you specify `byrow=TRUE`.

```r
> ## 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"))
```

```r
> 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
```

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

```r
> JobSat <- as.table(JobSat)
> str(JobSat)
'table' num [1:4, 1:4] 1 2 1 0 3 3 6 1 10 10 ...
- attr(*, "dimnames")=List of 2
  ..$ income      : chr [1:4] "< 15k" "15-25k" "25-40k" "> 40k"
  ..$ satisfaction: chr [1:4] "VeryD" "LittleD" "ModerateS" "VeryS"
```

### 2.1 Ordered factors and reordered tables

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.

Yet, for analysis, there are time 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

For the HairEyeColor data, hair color and eye color are ordered arbitrarily. For visualizing the data using mosaic plots and other methods described below, 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, and it is easiest to re-order eye colors by indexing. Again str() is your friend.

> HairEyeColor <- HairEyeColor[, c(1,3,4,2),]
> str(HairEyeColor)

' table ' 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"

This is also the order for both hair color and eye color shown in the result of a correspondence analysis (Figure 6) below.

With data in case form or frequency form, when you have ordered factors represented with character values, you must ensure that they are treated as ordered in R.¹

Imagine that the Arthritis data was read from a text file. By default the Improved will be ordered alphabetically: Marked, None, Some— not what we want. In this case, the function ordered() (and others) can be useful.

> Arthritis <- read.csv("arthritis.txt",header=TRUE)
> Arthritis$Improved <- ordered(Arthritis$Improved, levels=c("None", "Some", "Marked"))

With this order of Improved, the response in this data, a mosaic display of Treatment and Improved (Figure 1) shows a clearly interpretable pattern.

Finally, there are situations where, particularly for display purposes, you want to re-order the dimensions of an n-way table, 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. We will use the following version of UCBAdmissions in Section 3.4 below.²

> UCB <- aperm(UCBAdmissions, c(2, 1, 3))
> dimnames(UCB)[[2]] <- c("Yes", "No")
> names(dimnames(UCB)) <- c("Sex", "Admit?", "Department")
> ftable(UCB)

<table>
<thead>
<tr>
<th>Department</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>512</td>
<td>353</td>
<td>120</td>
<td>138</td>
<td>53</td>
<td>22</td>
</tr>
<tr>
<td>No</td>
<td>313</td>
<td>207</td>
<td>205</td>
<td>279</td>
<td>138</td>
<td>351</td>
</tr>
<tr>
<td>Female</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>89</td>
<td>17</td>
<td>202</td>
<td>131</td>
<td>94</td>
<td>24</td>
</tr>
<tr>
<td>No</td>
<td>19</td>
<td>8</td>
<td>391</td>
<td>244</td>
<td>299</td>
<td>317</td>
</tr>
</tbody>
</table>

¹In SAS, many procedures offer the option order = data | internal | formatted to allow character values to be ordered according to (a) their order in the data set, (b) sorted internal value, or (c) sorted formatted representation provided by a SAS format.

² Changing Admit to Admit? might be useful for display purposes, but is dangerous— because it is then difficult to use that variable name in a model formula. See Section 5.3 for options labeling_args and set_labels to change variable and level names for displays in the strucplot framework.
Figure 1: Mosaic plot for the Arthritis data, showing the marginal model of independence for Treatment and Improved. Age, a covariate, and Sex are ignored here.

2.2 structable()

For 3-way and larger tables the structable() function in vcd provides a convenient and flexible tabular display. The variables assigned to the rows and columns of a two-way display can be specified by a model formula.

> structable(HairEyeColor)  # show the table: default

<table>
<thead>
<tr>
<th></th>
<th>Brown</th>
<th>Hazel</th>
<th>Green</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hair</td>
<td>Sex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Black</td>
<td>Male</td>
<td>32</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>36</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Brown</td>
<td>Male</td>
<td>53</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>66</td>
<td>29</td>
<td>14</td>
</tr>
<tr>
<td>Red</td>
<td>Male</td>
<td>10</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>16</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Blond</td>
<td>Male</td>
<td>3</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>4</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Brown</th>
<th>Red</th>
<th>Blond</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hair</td>
<td>Sex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Black</td>
<td>Male</td>
<td>32</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>36</td>
<td>16</td>
</tr>
<tr>
<td>Brown</td>
<td>Male</td>
<td>53</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>53</td>
<td>3</td>
</tr>
<tr>
<td>Brown</td>
<td>Male</td>
<td>66</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>66</td>
<td>3</td>
</tr>
</tbody>
</table>
Hazel  10  5  25  29  7  7  5  5  
Green  3  2  15  14  7  7  8  8  
Blue  11  9  50  34  10  7  30  64  

It also returns an object of class "structable" which may be plotted with mosaic() (not shown here).

```r
> HSE <- structable(Hair+Sex ~ Eye, HairEyeColor) # save structable object
> mosaic(HSE) # plot it
```

### 2.3 `table()` and friends

You can generate frequency tables from factor variables using the `table()` function, tables of proportions using the `prop.table()` function, and marginal frequencies using `margin.table()`.

```r
> n=500
> A <- factor(sample(c("a1","a2"), n, rep=TRUE))
> B <- factor(sample(c("b1","b2"), n, rep=TRUE))
> C <- factor(sample(c("c1","c2"), n, rep=TRUE))
> mydata <- data.frame(A,B,C)

> # 2-Way Frequency Table
> attach(mydata)
> mytable <- table(A,B) # A will be rows, B will be columns
> mytable # print table
> margin.table(mytable, 1) # A frequencies (summed over B)
> margin.table(mytable, 2) # B frequencies (summed over A)
> prop.table(mytable) # cell percentages
> prop.table(mytable, 1) # row percentages
> prop.table(mytable, 2) # column percentages
```

`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.

```r
> # 3-Way Frequency Table
> mytable <- table(A, B, C)
> ftable(mytable) # print table
```

`table()` ignores missing values by default. To include NA as a category in counts, include the table option `exclude=NULL` if the variable is a vector. If the variable is a factor you have to create a new factor using `newfactor <- factor(oldfactor, exclude=NULL)`.

### 2.4 `xtabs()`

The `xtabs()` function allows you to create crosstabulations of data using formula style input. This typically works with case-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.

```r
> # 3-Way Frequency Table
> mytable <- xtabs(~A+B+C, data=mydata)
> ftable(mytable) # print table
> summary(mytable) # chi-square test of independence
```
If a variable is included on the left side of the formula, it is assumed to be a vector of frequencies (useful if the data have already been tabulated in frequency form).

```r
gsstab <- xtabs(count ~ sex + party, data=GSS)
party
sex    dem indep rep
female 279  73  225
male   165  47  191
```

```r
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 = 7.01, df = 2, p-value = 0.03005

2.5 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:** 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 high school seniors, also classified by sex and race.

```r
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 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 ...
```

```r
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`.

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

```r
> # in 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.

```r
> # collapse over sex and race
> Dayton.ACM.tab <- apply(Dayton.tab, MARGIN=1:3, FUN=sum)
> Dayton.ACM.tab
> 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
```

9
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()** as the function to be applied to each piece.

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

### 2.6 Collapsing table levels: collapse.table()

A related problem arises when we have a table or array and for some purpose we want to reduce the number of levels of some factors by summing subsets of the frequencies. For example, we may have initially coded Age in 10-year intervals, and decide that, either for analysis or display purposes, we want to reduce Age to 20-year intervals. The **collapse.table()** function in **vcdExtra** was designed for this purpose.

**Example:** Create a 3-way table, and collapse Age from 10-year to 20-year intervals. First, we generate a $2 \times 6 \times 3$ table of random counts from a Poisson distribution with mean of 100.

```r
> # create some sample data in frequency form
> sex <- c("Male", "Female")
> education <- c("low", "med", "high")
> data <- expand.grid(sex=sex, age=age, education=education)
> counts <- rpois(36, 100)  # random Possion cell frequencies
> data <- cbind(data, counts)
> # make it into a 3-way table
> t1 <- xtabs(counts ~ sex + age + education, data=data)
> structable(t1)

```

<table>
<thead>
<tr>
<th>age</th>
<th>10-19</th>
<th>20-29</th>
<th>30-39</th>
<th>40-49</th>
<th>50-59</th>
<th>60-69</th>
</tr>
</thead>
<tbody>
<tr>
<td>sex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>low</td>
<td>98</td>
<td>105</td>
<td>104</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>97</td>
<td>105</td>
<td>101</td>
<td>88</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>99</td>
<td>101</td>
<td>109</td>
<td>88</td>
<td>99</td>
</tr>
<tr>
<td>Female</td>
<td>low</td>
<td>102</td>
<td>117</td>
<td>101</td>
<td>105</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>106</td>
<td>84</td>
<td>92</td>
<td>116</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>106</td>
<td>96</td>
<td>121</td>
<td>91</td>
<td>107</td>
</tr>
</tbody>
</table>

Now collapse **age** to 20-year intervals, and **education** to 2 levels. In the arguments, levels of **age** and **education** given the same label are summed in the resulting smaller table.

```r
> # collapse age to 3 levels, education to 2 levels
> t2 <- collapse.table(t1,
+    age=c("10-29", "10-29", "30-49", "30-49", "50-69", "50-69"),
+    education=c("<high", "<high", "high"))
> structable(t2)

```

<table>
<thead>
<tr>
<th>age</th>
<th>10-29</th>
<th>30-49</th>
<th>50-69</th>
</tr>
</thead>
<tbody>
<tr>
<td>sex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>&lt;high</td>
<td>405</td>
<td>383</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>200</td>
<td>197</td>
</tr>
<tr>
<td>Female</td>
<td>&lt;high</td>
<td>409</td>
<td>414</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>202</td>
<td>212</td>
</tr>
</tbody>
</table>

---

3 Ugh. This **plyr** function clashes with a function of the same name in **vcdExtra**. In this document I will use the explicit double-colon notation to keep them separate.
2.7 Converting among frequency tables and data frames

As we’ve seen, a given contingency table can be represented equivalently in different forms, but some R functions were designed for one particular representation. Table 1 shows some handy tools for converting from one form to another.

Table 1: Tools for converting among different forms for categorical data

<table>
<thead>
<tr>
<th>From this</th>
<th>Case form</th>
<th>Frequency form</th>
<th>Table form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case form</td>
<td>noop</td>
<td>xtabs(“A+B”)</td>
<td>table(A,B)</td>
</tr>
<tr>
<td>Frequency form</td>
<td>expand.dft(X)</td>
<td>noop</td>
<td>xtabs(count“A+B”)</td>
</tr>
<tr>
<td>Table form</td>
<td>expand.dft(X)</td>
<td>as.data.frame(X)</td>
<td>noop</td>
</tr>
</tbody>
</table>

A contingency table in table form (an object of class(table)) can be converted to a data.frame with as.data.frame(). 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. This is the inverse of xtabs().

**Example:** Convert the GSStab in table form to a data.frame in frequency form.

```
> 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
```

**Example:** Convert the Arthritis data in case form to a 3-way table of Treatment × Sex × Improved. Note the use of with() to avoid having to use Arthritis$Treatment etc. within the call to table().

```
> Art.tab <- with(Arthritis, table(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
```

---

4 Because R is object-oriented, this is actually a short-hand for the function as.data.frame.table().
5 table() does not allow a data argument to provide an environment in which the table variables are to be found. In the examples in Section 2.3 I used attach(mydata) for this purpose, but attach() leaves the variables in the global environment, while with() just evaluates the table() expression in a temporary environment of the data.
Placebo   Female 19 7 6  
     Male    10 0 1 
Treated   Female 6 5 16 
     Male    7 2 5 

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. Marc Schwartz provided code for expand.dft() on the R-help mailing list for converting a table back into a case form data.frame. This function is included in vcdExtra.

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

```r
> 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, you’re ready for a more complicated example. The data file, tv.dat represents a 4-way table of size $5 \times 11 \times 5 \times 3$ where the table variables (unnamed in the file) are read as V1 – V4, and the cell frequency is read as V5. The file, stored in the doc/extdata directory of vcdExtra, can be read as follows:

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

For a local file, just use read.table() in this form:

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

The data tv.dat came from the initial implementation of mosaic displays in R by Jay Emerson. In turn, they came from the initial development 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.

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.
We are interested just the cell frequencies, and rely on the facts that the (a) the table is complete—there are no missing cells, so \texttt{nrow(tv.data)} = 825; (b) the observations are ordered so that \texttt{V1} varies most rapidly and \texttt{V4} most slowly. From this, we can just extract the frequency column and reshape it into an array.

```r
> TV <- array(tv.data[,5], dim=c(5,11,5,3))
> dimnames(TV) <- list(c("Monday","Tuesday","Wednesday","Thursday","Friday"),
+ 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 \texttt{xtabs()} (or \texttt{plyr::daply()}) to do the cross-tabulation, using \texttt{V5} as the frequency variable. Here’s how to do this same operation with \texttt{xtabs()}:

```r
> TV <- xtabs(V5 ~ ., data=tv.data)
> dimnames(TV) <- list(Day=c("Monday","Tuesday","Wednesday","Thursday","Friday"),
+ 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"))
```

But this 4-way table is too large and awkward to work with. Among the networks, Fox and Other occur infrequently. We can also cut it down to a 3-way table by considering only viewers who persist with the current station.\footnote{This relies on the fact that indexing an array drops dimensions of length 1 by default, using the argument \texttt{drop=TRUE}; the result is coerced to the lowest possible dimension.}

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

Finally, for some purposes, we might want to collapse the 11 times into a smaller number. Here, we use \texttt{as.data.frame.table()} to convert the table back to a data frame, \texttt{levels()} to re-assign the values of \texttt{Time}, and finally, \texttt{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)

<table>
<thead>
<tr>
<th>Day</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Network</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8:00-8:59</td>
<td>ABC</td>
<td>536</td>
<td>861</td>
<td>744</td>
<td>735</td>
</tr>
<tr>
<td></td>
<td>CBS</td>
<td>1167</td>
<td>646</td>
<td>550</td>
<td>680</td>
</tr>
<tr>
<td></td>
<td>NBC</td>
<td>858</td>
<td>1090</td>
<td>512</td>
<td>1927</td>
</tr>
<tr>
<td>9:00-9:59</td>
<td>ABC</td>
<td>1401</td>
<td>1205</td>
<td>1022</td>
<td>682</td>
</tr>
<tr>
<td></td>
<td>CBS</td>
<td>967</td>
<td>959</td>
<td>409</td>
<td>385</td>
</tr>
<tr>
<td></td>
<td>NBC</td>
<td>946</td>
<td>890</td>
<td>831</td>
<td>1858</td>
</tr>
<tr>
<td>10:00-10:44</td>
<td>ABC</td>
<td>910</td>
<td>1044</td>
<td>668</td>
<td>349</td>
</tr>
<tr>
<td></td>
<td>CBS</td>
<td>789</td>
<td>798</td>
<td>324</td>
<td>270</td>
</tr>
<tr>
<td></td>
<td>NBC</td>
<td>825</td>
<td>588</td>
<td>869</td>
<td>2101</td>
</tr>
</tbody>
</table>

Whew! See Figure 7 for a mosaic plot of the TV2 data.

3 Tests of Independence

3.1 CrossTable

OK, now we're ready to do some analyses. For tabular displays, the CrossTable() function in the gmodels package produces cross-tabulations modeled after PROC FREQ in SAS or CROSSTABS in SPSS. It has a wealth of options for the quantities that can be shown in each cell.

> # 2-Way Cross Tabulation
> library(gmodels)
> CrossTable(GSStab,prop.t=FALSE,prop.r=FALSE,prop.c=FALSE)

Cell Contents

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>Chi-square contribution</td>
</tr>
</tbody>
</table>

Total Observations in Table: 980

<table>
<thead>
<tr>
<th>sex</th>
<th>party</th>
<th>dem</th>
<th>indep</th>
<th>rep</th>
<th>Row Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>279</td>
<td>73</td>
<td>225</td>
<td>577</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.183</td>
<td>0.078</td>
<td>1.622</td>
<td></td>
<td></td>
</tr>
<tr>
<td>male</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>165</td>
<td>47</td>
<td>191</td>
<td>403</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.693</td>
<td>0.112</td>
<td>2.322</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Column Total | 444 | 120 | 416 | 980 |
There are options to report percentages (row, column, cell), specify decimal places, produce Chi-square, Fisher, and McNemar tests of independence, report expected and residual values (pearson, standardized, adjusted standardized), include missing values as valid, annotate with row and column titles, and format as SAS or SPSS style output! See help(CrossTable) for details.

### 3.2 Chi-square test

For 2-way tables you can use \texttt{chisq.test()} to test independence of the row and column variable. By default, the \( p \)-value is calculated from the asymptotic chi-squared distribution of the test statistic. Optionally, the \( p \)-value can be derived via Monte Carlo simulation.

\begin{verbatim}
> (HairEye <- margin.table(HairEyeColor, c(1, 2)))

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

> chisq.test(HairEye)

Pearson's Chi-squared test

data: HairEye
X-squared = 138.29, df = 9, p-value < 2.2e-16
\end{verbatim}

### 3.3 Fisher Exact Test

\texttt{fisher.test(X)} provides an exact test of independence. \( X \) must be a two-way contingency table in table form. Another form, \texttt{fisher.test(X, Y)} takes two categorical vectors of the same length. For tables larger than \( 2 \times 2 \) the method can be computationally intensive (or can fail) if the frequencies are not small.

\begin{verbatim}
> fisher.test(GSStab)

Fisher's Exact Test for Count Data

data: GSStab
p-value = 0.03115
alternative hypothesis: two.sided

But this does not work because HairEye data has \( n=592 \) total frequency. An exact test is unnecessary in this case.

> fisher.test(HairEye)

Error in fisher.test(HairEye) : FEXACT error 6.
LDKEY is too small for this problem.
Try increasing the size of the workspace.
\end{verbatim}
3.4 Mantel-Haenszel test and conditional association

Use the \texttt{mantelhaen.test(X)} function to perform a Cochran-Mantel-Haenszel $\chi^2$ chi test of the null hypothesis that two nominal variables are \textit{conditionally independent}, $A \perp B \mid C$, in each stratum, assuming that there is no three-way interaction. $X$ is a 3 dimensional contingency table, where the last dimension refers to the strata.

The \texttt{UCBAdmissions} serves as an example of a $2 \times 2 \times 6$ table, with \texttt{Dept} as the stratifying variable.

\begin{verbatim}
> ## UC Berkeley Student Admissions
> mantelhaen.test(UCBAdmissions)

Mantel-Haenszel chi-squared test with continuity correction

data: UCBAdmissions
Mantel-Haenszel X-squared = 1.4269, df = 1, p-value = 0.2323
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
0.7719074 1.0603298
sample estimates:
common odds ratio
0.9046968
\end{verbatim}

The results show no evidence for association between admission and gender when adjusted for department. However, we can easily see that the assumption of equal association across the strata (no 3-way association) is probably violated. For $2 \times 2 \times k$ tables, this can be examined from the odds ratios for each $2 \times 2$ table (\texttt{oddsratio()}), and tested by using \texttt{woolf_test()} in \texttt{vcd}.

\begin{verbatim}
> oddsratio(UCBAdmissions, log=FALSE)

odds ratios for Admit and Gender by Dept

A   B   C   D   E   F
0.3492120 0.8025007 1.1330596 0.9212838 1.2216312 0.8278727

> lor <- oddsratio(UCBAdmissions) # capture log odds ratios
> summary(lor)

z test of coefficients:


\begin{verbatim}

                Estimate Std. Error    z value  Pr(>|z|)
A -1.052076 0.262708  -4.0047  6.209e-05 ***
B -0.220023 0.437593  -0.5028    0.6151
C  0.124922 0.143942   0.8679    0.3855
D -0.081987 0.150208  -0.5458    0.5852
E  0.200187 0.200243   0.9997    0.3174
F -0.188896 0.305164  -0.6190    0.5359
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
\end{verbatim}

> woolf_test(UCBAdmissions)
\end{verbatim}
Woolf-test on Homogeneity of Odds Ratios (no 3-Way assoc.)

data: UCBAdmissions
X-squared = 17.902, df = 5, p-value = 0.003072

We can visualize the odds ratios of Admission for each department with fourfold displays using `fourfold()`. The cell frequencies \( n_{ij} \) of each \( 2 \times 2 \) table are shown as a quarter circle whose radius is proportional to \( \sqrt{n_{ij}} \), so that its area is proportional to the cell frequency. Confidence rings for the odds ratio allow a visual test of the null of no association; the rings for adjacent quadrants overlap iff the observed counts are consistent with the null hypothesis. In the extended version (the default), brighter colors are used where the odds ratio is significantly different from 1. The following lines produce Figure 2.

```r
> col <- c("#99CCFF", "#6699CC", "#F9AFAF", "#6666A0", "#FF0000", "#000080")
> fourfold(UCB,mfrow=c(2,3), color=col)
```

![Fourfold display for the UCBAdmissions data. Where the odds ratio differs significantly from 1.0, the confidence bands do not overlap, and the circle quadrants are shaded more intensely.](image)

Figure 2: Fourfold display for the UCBAdmissions data. Where the odds ratio differs significantly from 1.0, the confidence bands do not overlap, and the circle quadrants are shaded more intensely.

Another `vcd` function, `cotabplot()`, provides a more general approach to visualizing conditional associations in contingency tables, similar to trellis-like plots produced by `coplot()` and lattice graphics. The `panel` argument supplies a function used to render each conditional subtable. The following gives a display (not shown) similar to Figure 2.

```r
> cotabplot(UCB, panel = cotab_fourfold)
```

When we want to view the conditional probabilities of a response variable (e.g., `Admit`) in relation to several factors, an alternative visualization is a `doubledecker()` plot. This plot is a specialized

---

7 The color values `col[3:4]` were modified from their default values to show a greater contrast between significant and insignificant associations here.
version of a mosaic plot, which highlights the levels of a response variable (plotted vertically) in relation to the factors (shown horizontally). The following call produces Figure 3, where we use indexing on the first factor (Admit) to make Admitted the highlighted level.

In this plot, the association between Admit and Gender is shown where the heights of the highlighted conditional probabilities do not align. The excess of females admitted in Dept A stands out here.

> doubledecker(Admit ~ Dept + Gender, data=UCBAdmissions[2:1,,])

```
Dept
   Gender
A Male  
   Female
B Male  
   Female
C Male  
   Female
D Male  
   Female
E Male  
   Female
F Male  
   Female
Admit
Rejected
Admit
```

Figure 3: Doubledecker display for the UCBAdmissions data. The heights of the highlighted bars show the conditional probabilities of Admit, given Dept and Gender.

Finally, there is a plot() method for oddsratio objects. By default, it shows the 95% confidence interval for the log odds ratio. Figure 4 is produced by:

> plot(lor, xlab="Department", ylab="Log Odds Ratio (Admit | Gender)")

3.5 Cochran-Mantel-Haenszel tests for ordinal factors

The standard \( \chi^2 \) tests for association in a two-way table treat both table factors as nominal (unordered) categories. When one or both factors of a two-way table are quantitative or ordinal, more powerful tests of association may be obtained by taking ordinality into account, using row and or column scores to test for linear trends or differences in row or column means.

More general versions of the CMH tests (Landis et al., 1978) are provided by assigning numeric scores to the row and/or column variables. For example, with two ordinal factors (assumed to be equally spaced), assigning integer scores, \( 1:R \) and \( 1:C \) tests the linear \( \times \) linear component of association. This is statistically equivalent to the Pearson correlation between the integer-scored table variables, with \( \chi^2 = (n - 1)r^2 \), with only 1 df rather than \( (R - 1) \times (C - 1) \) for the test of general association.
When only one table variable is ordinal, these general CMH tests are analogous to an ANOVA, testing whether the row mean scores or column mean scores are equal, again consuming fewer \( df \) than the test of general association.

The \texttt{CMHtest()} function in \texttt{vcdExtra} now calculates these various CMH tests for two possibly ordered factors, optionally stratified other factor(s).

\textbf{Example:} Recall the \( 4 \times 4 \) table, \texttt{JobSat} introduced in Section 2,

\begin{verbatim}
> 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
\end{verbatim}

Treating the satisfaction levels as equally spaced, but using midpoints of the income categories as row scores gives the following results:

\begin{verbatim}
> CMHtest(JobSat, rscores=c(7.5,20,32.5,60))

Cochran-Mantel-Haenszel Statistics for income by satisfaction

<table>
<thead>
<tr>
<th>AltHypothesis</th>
<th>Chisq</th>
<th>Df</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>cor</td>
<td>3.8075</td>
<td>1</td>
<td>0.051025</td>
</tr>
<tr>
<td>rmeans</td>
<td>4.4774</td>
<td>3</td>
<td>0.214318</td>
</tr>
<tr>
<td>cmeans</td>
<td>3.8404</td>
<td>3</td>
<td>0.279218</td>
</tr>
<tr>
<td>general</td>
<td>5.9034</td>
<td>9</td>
<td>0.749549</td>
</tr>
</tbody>
</table>
\end{verbatim}
Note that with the relatively small cell frequencies, the test for general give no evidence for association. However, the the cor test for linear x linear association on 1 df is nearly significant. The coin contains the functions cmh_test() and lbl_test() for CMH tests of general association and linear x linear association respectively.

### 3.6 Measures of Association

There are a variety of statistical measures of strength of association for contingency tables—similar in spirit to $r$ or $r^2$ for continuous variables. With a large sample size, even a small degree of association can show a significant $\chi^2$, as in the example below for the GSS data.

The assocstats() function in vcd calculates the $\phi$ contingency coefficient, and Cramer’s V for an $r \times c$ table. The input must be in table form, a two-way $r \times c$ table. It won’t work with GSS in frequency form, but by now you should know how to convert.

```r
> assocstats(GSStab)

          X^2 df P(> X^2)
Likelihood Ratio 7.0026 2 0.030158
Pearson 7.0095 2 0.030054

Phi-Coefficient : NA
Contingency Coeff.: 0.084
Cramer's V : 0.085
```

For tables with ordinal variables, like JobSat, some people prefer the Goodman-Kruskal $\gamma$ statistic (Agresti (2002, §2.4.3)) based on a comparison of concordant and discordant pairs of observations in the case-form equivalent of a two-way table.

```r
> GKgamma(JobSat)

gamma : 0.221
std. error : 0.117
CI : -0.009 0.451
```

A web article by Richard Darlington, [http://www.psych.cornell.edu/Darlington/crosstab/TABLE0.HTM](http://www.psych.cornell.edu/Darlington/crosstab/TABLE0.HTM) gives further description of these and other measures of association.

### 3.7 Measures of Agreement

The Kappa() function in the vcd package calculates Cohen’s $\kappa$ and weighted $\kappa$ for a square two-way table with the same row and column categories (Cohen 1960). Normal-theory $z$-tests are obtained by dividing $\kappa$ by its asymptotic standard error (ASE). A confint() method for Kappa objects provides confidence intervals.

```r
> (K <- Kappa(SexualFun))

                      value      ASE     z  Pr(>|z|)
Unweighted 0.1293 0.06860 1.885 0.059387
Weighted   0.2374 0.07832 3.031 0.002437

> confint(K)
```

---

8 Don’t confuse this with kappa() in base R that computes something entirely different (the condition number of a matrix).
A visualization of agreement, both unweighted and weighted for degree of departure from exact agreement is provided by the `agreementplot()` function. Figure 5 shows the agreementplot for the `SexualFun` data, produced as shown below. The Bangdiwala measures represent the proportion of the shaded areas of the diagonal rectangles, using weights $w_1$ for exact agreement, and $w_2$ for partial agreement one step from the main diagonal.

```r
> agree <- agreementplot(SexualFun, main="Is sex fun?")
> unlist(agree)

<table>
<thead>
<tr>
<th></th>
<th>Bangdiwala</th>
<th>Bangdiwala_Weighted</th>
<th>weights1</th>
<th>weights2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1464624</td>
<td>0.4981723</td>
<td>1.000000</td>
<td>0.888889</td>
</tr>
</tbody>
</table>

Figure 5: Agreement plot for the `SexualFun` data.

In other examples, the agreement plot can help to show sources of disagreement. For example, when the shaded boxes are above or below the diagonal (red) line, a lack of exact agreement can be attributed in part to different frequency of use of categories by the two raters–lack of marginal homogeneity.

### 3.8 Correspondence analysis

Use the `ca` package for correspondence analysis for visually exploring relationships between rows and columns in contingency tables. For an $r \times c$ table, the method provides a breakdown of the Pearson $\chi^2$ for association in up to $M = \min(r-1,c-1)$ dimensions, and finds scores for the row ($x_{im}$) and column ($y_{jm}$) categories such that the observations have the maximum possible correlations.\(^9\)

Here, we carry out a simple correspondence analysis of the `HairEye` data. The printed results show that nearly 99% of the association between hair color and eye color can be accounted for in 2 dimensions, of which the first dimension accounts for 90%.

\(^9\) Related methods are the non-parametric CMH tests using assumed row/column scores (Section 3.5), the analogous `glm()` model-based methods (Section 3.5), and the more general RC models which can be fit using `gamm()`. Correspondence analysis differs in that it is a primarily descriptive/exploratory method (no significance tests), but is directly tied to informative graphic displays of the row/column categories.
> library(ca)
> ca(HairEye)

Principal inertias (eigenvalues):

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.208773</td>
<td>0.022227</td>
<td>0.002598</td>
</tr>
<tr>
<td>Percentage</td>
<td>89.37%</td>
<td>9.52%</td>
<td>1.11%</td>
</tr>
</tbody>
</table>

Rows:

<table>
<thead>
<tr>
<th></th>
<th>Black</th>
<th>Brown</th>
<th>Red</th>
<th>Blond</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass</td>
<td>0.182432</td>
<td>0.483108</td>
<td>0.119932</td>
<td>0.214527</td>
</tr>
<tr>
<td>ChiDist</td>
<td>0.551192</td>
<td>0.159461</td>
<td>0.354770</td>
<td>0.838397</td>
</tr>
<tr>
<td>Inertia</td>
<td>0.055425</td>
<td>0.012284</td>
<td>0.015095</td>
<td>0.150793</td>
</tr>
<tr>
<td>Dim. 1</td>
<td>-1.104277</td>
<td>-0.324463</td>
<td>-0.283473</td>
<td>1.828229</td>
</tr>
<tr>
<td>Dim. 2</td>
<td>1.440917</td>
<td>-0.219111</td>
<td>-2.144015</td>
<td>0.466706</td>
</tr>
</tbody>
</table>

Columns:

<table>
<thead>
<tr>
<th></th>
<th>Brown</th>
<th>Hazel</th>
<th>Green</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass</td>
<td>0.371622</td>
<td>0.157095</td>
<td>0.108108</td>
<td>0.363176</td>
</tr>
<tr>
<td>ChiDist</td>
<td>0.500487</td>
<td>0.288654</td>
<td>0.385727</td>
<td>0.553684</td>
</tr>
<tr>
<td>Inertia</td>
<td>0.093086</td>
<td>0.013089</td>
<td>0.016085</td>
<td>0.111337</td>
</tr>
<tr>
<td>Dim. 1</td>
<td>-1.077128</td>
<td>-0.465286</td>
<td>0.354011</td>
<td>1.198061</td>
</tr>
<tr>
<td>Dim. 2</td>
<td>0.592420</td>
<td>-1.122783</td>
<td>-2.274122</td>
<td>0.556419</td>
</tr>
</tbody>
</table>

The resulting \texttt{ca} object can be plotted just by running the \texttt{plot()} method on the \texttt{ca} object, giving the result in Figure 6. \texttt{plot.ca()} does not allow labels for dimensions; these can be added with \texttt{title()}. It can be seen that most of the association is accounted for by the ordering of both hair color and eye color along Dimension 1, a dark to light dimension.

> plot(ca(HairEye), main="Hair Color and Eye Color")
> title(xlab="Dim 1 (89.4%)", ylab="Dim 2 (9.5%)")

4 Loglinear Models

You can use the \texttt{loglm()} function in the \texttt{MASS} package to fit log-linear models. Equivalent models can also be fit (from a different perspective) as generalized linear models with the \texttt{glm()} function using the \texttt{family=’poisson’} argument, and the \texttt{gnm} package provides a wider range of generalized \textit{nonlinear} models, particularly for testing structured associations. The visualization methods for these models were originally developed for models fit using \texttt{loglm()}, so this approach is emphasized here. Some extensions of these methods for models fit using \texttt{glm()} and \texttt{gnm()} are contained in the \texttt{vcdExtra} package and illustrated in Section 4.2.

Assume we have a 3-way contingency table based on variables A, B, and C. The possible different forms of loglinear models for a 3-way table are shown in Table 2. The \textbf{Model formula} column shows how to express each model for \texttt{loglm()} in \texttt{R}.\footnote{In the \texttt{Interpretation} column, the symbol “⊥” is to be read as “is independent of,” and “|” means “conditional on,” or “adjusting for,” or just “given”.} In the \textbf{Interpretation} column, the symbol “⊥” is to be read as “is independent of,” and “|” means “conditional on,” or “adjusting for,” or just “given”.

\footnote{For \texttt{glm()}, or \texttt{gnm()}, with the data in the form of a frequency data.frame, the same model is specified in the form \texttt{glm(Freq ~ ..., family=’poisson’)}, where \texttt{Freq} is the name of the cell frequency variable and ... specifies the \textbf{Model formula}.}
Figure 6: Correspondence analysis plot for the HairEye data.

For example, the formula \( \sim A + B + C \) specifies the model of mutual independence with no associations among the three factors. In standard notation for the expected frequencies \( m_{ijk} \), this corresponds to

\[
\log(m_{ijk}) = \mu + \lambda_i^A + \lambda_j^B + \lambda_k^C \equiv A + B + C
\]

The parameters \( \lambda_i^A, \lambda_j^B \) and \( \lambda_k^C \) pertain to the differences among the one-way marginal frequencies for the factors A, B and C.

Similarly, the model of joint independence, \( (A \ B) \perp C \), allows an association between A and B, but specifies that C is independent of both of these and their combinations,

\[
\log(m_{ijk}) = \mu + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ij}^{AB} \equiv A \ast B + C
\]

where the parameters \( \lambda_{ij}^{AB} \) pertain to the overall association between A and B (collapsing over C).

<table>
<thead>
<tr>
<th>Model</th>
<th>Model formula</th>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual independence</td>
<td>( \sim A + B + C )</td>
<td>( [A][B][C] )</td>
<td>( A \perp B \perp C )</td>
</tr>
<tr>
<td>Joint independence</td>
<td>( \sim A*B + C )</td>
<td>( [AB][C] )</td>
<td>( (A \ B) \perp C )</td>
</tr>
<tr>
<td>Conditional independence</td>
<td>( \sim (A*B)*C )</td>
<td>( [AC][BC] )</td>
<td>( (A \perp B) \perp C )</td>
</tr>
<tr>
<td>All two-way associations</td>
<td>( \sim A<em>B + A</em>C + B*C )</td>
<td>( [AB][AC][BC] )</td>
<td>homogeneous association</td>
</tr>
<tr>
<td>Saturated model</td>
<td>( \sim A<em>B</em>C )</td>
<td>( [ABC] )</td>
<td>3-way association</td>
</tr>
</tbody>
</table>
In the literature or text books, you will often find these models expressed in shorthand symbolic notation, using brackets, [ ] to enclose the high-order terms in the model. Thus, the joint independence model can be denoted \([AB][C]\), as shown in the Symbol column in Table 2.

Models of conditional independence allow (and fit) two of the three possible two-way associations. There are three such models, depending on which variable is conditioned upon. For a given conditional independence model, e.g., \([AB][AC]\), the given variable is the one common to all terms, so this example has the interpretation \((B \perp C)\mid A\).

### 4.1 Fitting with loglm()

For example, we can fit the model of mutual independence among hair color, eye color and sex in HairEyeColor as

```r
> library(MASS)
> ## Independence model of hair and eye color and sex.
> hec.1 <- loglm(~Hair+Eye+Sex, data=HairEyeColor)
> hec.1
```

Call:
```r
loglm(formula = ~Hair + Eye + Sex, data = HairEyeColor)
```

Statistics:

<table>
<thead>
<tr>
<th></th>
<th>X^2</th>
<th>df</th>
<th>P(&gt; X^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>166.3001</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>Pearson</td>
<td>164.9247</td>
<td>24</td>
<td>0</td>
</tr>
</tbody>
</table>

Similarly, the models of conditional independence and joint independence are specified as

```r
> ## Conditional independence
> hec.2 <- loglm(~(Hair + Eye) * Sex, data=HairEyeColor)
> hec.2
```

Call:
```r
loglm(formula = ~(Hair + Eye) * Sex, data = HairEyeColor)
```

Statistics:

<table>
<thead>
<tr>
<th></th>
<th>X^2</th>
<th>df</th>
<th>P(&gt; X^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>156.6779</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>Pearson</td>
<td>147.9440</td>
<td>18</td>
<td>0</td>
</tr>
</tbody>
</table>

```r
> ## Joint independence model.
> hec.3 <- loglm(~Hair*Eye + Sex, data=HairEyeColor)
> hec.3
```

Call:
```r
loglm(formula = ~Hair * Eye + Sex, data = HairEyeColor)
```

Statistics:

<table>
<thead>
<tr>
<th></th>
<th>X^2</th>
<th>df</th>
<th>P(&gt; X^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>19.85666</td>
<td>15</td>
<td>0.1775045</td>
</tr>
<tr>
<td>Pearson</td>
<td>19.56712</td>
<td>15</td>
<td>0.1891745</td>
</tr>
</tbody>
</table>
```

24
Note that printing the model gives a brief summary of the goodness of fit. A set of models can be compared using the `anova()` function.

```r
> anova(hec.1, hec.2, hec.3)
```

LR tests for hierarchical log-linear models

Model 1:
```
~Hair + Eye + Sex
```
Model 2:
```
~(Hair + Eye) * Sex
```
Model 3:
```
~Hair * Eye + Sex
```

<table>
<thead>
<tr>
<th>Deviance df</th>
<th>Delta(Dev)</th>
<th>Delta(df)</th>
<th>P(&gt; Delta(Dev))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>166.30014</td>
<td>14</td>
<td>24</td>
</tr>
<tr>
<td>Model 2</td>
<td>156.67789</td>
<td>9.62225</td>
<td>6</td>
</tr>
<tr>
<td>Model 3</td>
<td>19.85656</td>
<td>136.82133</td>
<td>3</td>
</tr>
<tr>
<td>Saturated</td>
<td>0.00000</td>
<td>19.85656</td>
<td>15</td>
</tr>
</tbody>
</table>

4.2 Fitting with `glm()` and `gnm()`

The `glm()` approach, and extensions of this in the `gnm` package allows a much wider class of models for frequency data to be fit than can be handled by `loglm()`. Of particular importance are models for ordinal factors and for square tables, where we can test more structured hypotheses about the patterns of association than are provided in the tests of general association under `loglm()`. These are similar in spirit to the non-parametric CMH tests described in Section 3.5.

**Example:** The data `Mental` in the `vcdExtra` package gives a two-way table in frequency form classifying young people by their mental health status and parents’ socioeconomic status (SES), where both of these variables are ordered factors.

```r
> str(Mental)
'data.frame': 24 obs. of 3 variables:
$ ses : Ord.factor w/ 6 levels "1"<"2"<"3"<"4"<"5"<"6": 1 1 1 1 2 2 2 2 3 3 ...
$ mental: Ord.factor w/ 4 levels "Well"<"Mild"<"Moderate"<"Impaired": 1 2 3 4 1 2 3 4 1 2 ...
$ Freq : int 64 94 58 46 57 94 54 40 57 105 ...
```

```r
> xtabs(Freq ~ mental+ses, data=Mental) # display the frequency table
```

<table>
<thead>
<tr>
<th>ses</th>
<th>mental</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Well</td>
</tr>
<tr>
<td>2</td>
<td>Mild</td>
</tr>
<tr>
<td>3</td>
<td>Moderate</td>
</tr>
<tr>
<td>4</td>
<td>Impaired</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>94</td>
</tr>
</tbody>
</table>

Simple ways of handling ordinal variables involve assigning scores to the table categories, and the simplest cases are to use integer scores, either for the row variable (“column effects” model), the column variable (“row effects” model), or both (“uniform association” model).

```r
> indep <- glm(Freq ~ mental + ses, family = poisson, data = Mental) # independence model
```
To fit more parsimonious models than general association, we can define numeric scores for the row and column categories

> # Use integer scores for rows/cols
> Cscore <- as.numeric(Mental$ses)
> Rscore <- as.numeric(Mental$mental)

Then, the row effects model, the column effects model, and the uniform association model can be fit as follows:

> # column effects model (ses)
> coleff <- glm(Freq ~ mental + ses + Rscore:ses, family = poisson, data = Mental)
> # row effects model (mental)
> roweff <- glm(Freq ~ mental + ses + mental:Cscore, family = poisson, data = Mental)
> # linear x linear association
> linlin <- glm(Freq ~ mental + ses + Rscore:Cscore, family = poisson, data = Mental)

The `Summarize()` in `vcdExtra` provides a nice, compact summary of the fit statistics for a set of models, collected into a "glmlist" object. Smaller is better for AIC and BIC.

> # compare models using AIC, BIC, etc
> vcdExtra::LRstats(glmlist(indep, roweff, coleff, linlin))

Likelihood summary table:

<table>
<thead>
<tr>
<th>Model</th>
<th>AIC</th>
<th>BIC</th>
<th>LR</th>
<th>Df</th>
<th>Pr(&gt;Chi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>indep</td>
<td>209.59</td>
<td>220.19</td>
<td>47.418</td>
<td>15</td>
<td>3.155e-05 ***</td>
</tr>
<tr>
<td>roweff</td>
<td>174.45</td>
<td>188.59</td>
<td>6.281</td>
<td>12</td>
<td>0.9013</td>
</tr>
<tr>
<td>coleff</td>
<td>179.00</td>
<td>195.50</td>
<td>6.829</td>
<td>10</td>
<td>0.7415</td>
</tr>
<tr>
<td>linlin</td>
<td>174.07</td>
<td>185.85</td>
<td>9.895</td>
<td>14</td>
<td>0.7698</td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

For specific model comparisons, we can also carry out tests of nested models with `anova()` when those models are listed from smallest to largest. Here, there are two separate paths from the most restrictive (independence) model through the model of uniform association, to those that allow only one of row effects or column effects.

> anova(indep, linlin, coleff, test="Chisq")

Analysis of Deviance Table

Model 1: Freq ~ mental + ses
Model 2: Freq ~ mental + ses + Rscore:Cscore
Model 3: Freq ~ mental + ses + Rscore:ses
   Resid. Df Resid. Dev Deviance Pr(>Chi)
1       15       47.418
2       14       9.895   1  37.523 9.035e-10 ***
3       10       6.829   4  3.066  0.5469

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(indep, linlin, roweff, test="Chisq")
Analysis of Deviance Table

Model 1: Freq ~ mental + ses
Model 2: Freq ~ mental + ses + Rscore:Cscore
Model 3: Freq ~ mental + ses + mental:Cscore

Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1 15 47.418
2 14 9.895 1 37.523 9.035e-10 ***
3 12 6.281 2 3.614 0.1641

---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

The model of linear by linear association seems best on all accounts. For comparison, one might try the CMH tests on these data:

> CMHtest(xtabs(Freq~ses+mental, data=Mental))

Cochran-Mantel-Haenszel Statistics for ses by mental

<table>
<thead>
<tr>
<th>AltHypothesis</th>
<th>Chisq</th>
<th>Df</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>cor</td>
<td>37.156</td>
<td>1</td>
<td>1.0907e-09</td>
</tr>
<tr>
<td>rmeans</td>
<td>40.297</td>
<td>5</td>
<td>1.3012e-07</td>
</tr>
<tr>
<td>cmeans</td>
<td>40.666</td>
<td>3</td>
<td>7.6971e-09</td>
</tr>
<tr>
<td>general</td>
<td>45.958</td>
<td>15</td>
<td>5.4003e-05</td>
</tr>
</tbody>
</table>

4.3 Non-linear terms

The strength of the gnm package is that it handles a wide variety of models that handle non-linear terms, where the parameters enter the model beyond a simple linear function. The simplest example is the Goodman RC(1) model, which allows a multiplicative term to account for the association of the table variables. In the notation of generalized linear models with a log link, this can be expressed as

$$\log \mu_{ij} = \alpha_i + \beta_j + \gamma_i \delta_j$$

where the row-multiplicative effect parameters $\gamma_i$ and corresponding column parameters $\delta_j$ are estimated from the data.\(^{11}\) Similarly, the RC(2) model adds two multiplicative terms to the independence model,

$$\log \mu_{ij} = \alpha_i + \beta_j + \gamma_{i1}\delta_{j1} + \gamma_{i2}\delta_{j2}$$

In the gnm package, these models may be fit using the Mult() to specify the multiplicative term, and instances() to specify several such terms.

**Example:** For the Mental data, we fit the RC(1) and RC(2) models, and compare these with the independence model.

> RC1 <- gnm(Freq ~ mental + ses + Mult(mental,ses), data=Mental, + family=poisson, , verbose=FALSE)
> RC2 <- gnm(Freq ~ mental+ses + instances(Mult(mental,ses),2), data=Mental, + family=poisson, verbose=FALSE)
> anova(indep, RC1, RC2, test="Chisq")

\(^{11}\) This is similar in spirit to a correspondence analysis with a single dimension, but as a statistical model.
Analysis of Deviance Table

Model 1: Freq ~ mental + ses
Model 2: Freq ~ mental + ses + Mult(mental, ses)
Model 3: Freq ~ mental + ses + Mult(mental, ses, inst = 1) + Mult(mental, ses, inst = 2)

Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1 15 47.418
2 9 40.230 6 7.188 0.3038
3 3 0.523 6 39.707 5.2e-07 ***

---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

5 Mosaic plots

Mosaic plots provide an ideal method both for visualizing contingency tables and for visualizing the fit—or more importantly—lack of fit of a loglinear model. For a two-way table, mosaic() fits a model of independence, [A][B] or ~A*B as an R formula. For n-way tables, mosaic() can fit any loglinear model, and can also be used to plot a model fit with loglm(). See Friendly (1994, 1999) for the statistical ideas behind these uses of mosaic displays in connection with loglinear models.

The essential idea is to recursively sub-divide a unit square into rectangular “tiles” for the cells of the table, such that the area of each tile is proportional to the cell frequency. For a given loglinear model, the tiles can then be shaded in various ways to reflect the residuals (lack of fit) for a given model. The pattern of residuals can then be used to suggest a better model or understand where a given model fits or does not fit.

mosaic() provides a wide range of options for the directions of splitting, the specification of shading, labeling, spacing, legend and many other details. It is actually implemented as a special case of a more general class of displays for n-way tables called strucplot, including sieve diagrams, association plots, double-decker plots as well as mosaic plots. For details, see help(strucplot) and the “See also” links, and also Meyer, Zeileis, and Hornik (2006), which is available as an R vignette via vignette("strucplot", package="vcd").

Figure 1, showing the association between Treatment and Improved was produced with the following call to mosaic().

```r
> mosaic(art, gp = shading_max, split_vertical = TRUE,
+   main="Arthritis: [Treatment] [Improved]")
```

Note that the residuals for the independence model were not large (as shown in the legend), yet the association between Treatment and Improved is highly significant.

```r
> summary(art)
```

Call: xtabs(formula = ~Treatment + Improved, data = Arthritis)
Number of cases in table: 84
Number of factors: 2
Test for independence of all factors:
  Chisq = 13.055, df = 2, p-value = 0.001463

In contrast, one of the other shading schemes, from Friendly (1994) (use: gp = shading_Friendly), uses fixed cutoffs of ±2, ±4, to shade cells which are individually significant at approximately α = 0.05 and α = 0.001 levels, respectively. The right panel below uses gp = shading_Friendly.
5.1 Mosaics for loglinear models

When you have fit a loglinear model using `loglm()`, and saved the result (as a `loglm` object) the simplest way to display the results is to use the `plot()` method for the `loglm` object. Calling `mosaic(loglm.object)` has the same result. In Section 4.1 above, we fit several different models to the `HairEyeColor` data. We can produce mosaic displays of each just by plotting them:

```r
> # mosaic plots, using plot.loglm() method
> plot(hec.1, main="model: [Hair][Eye][Sex]")
> plot(hec.2, main="model: [HairSex][EyeSex]")
> plot(hec.3, main="model: [HairEye][Sex]")
```

Alternatively, you can supply the model formula to `mosaic()` with the `expected` argument. This is passed to `loglm()`, which fits the model, and returns residuals used for shading in the plot.

For example, here we examine the `TV2` constructed in Section 2.8 above. The goal is to see how Network choice depends on (varies with) Day and Time. To do this:

- We fit a model of joint independence of `Network` on the combinations of `Day` and `Time`, with the model formula `~Day:Time + Network`.
- To make the display more easily read, we place `Day` and `Time` on the vertical axis and `Network` on the horizontal,
- The `Time` values overlap on the right vertical axis, so we use `level()` to abbreviate them.

`mosaic()` also supports a more sophisticated set of labeling functions. Instead of changing the
data table, we could have used `labeling_args = list(abbreviate = c(Time = 2))` for a similar effect.

The following call to `mosaic()` produces Figure 7:

```r
> 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)
```

![Mosaic plot](image)

Figure 7: Mosaic plot for the TV data showing model of joint independence, `Day:Time + Network`.

From this, it is easy to read from the display 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.

In interpreting this mosaic and other plots, it is important to understand that associations included in the model—here, that between day and time—are not shown in the shading of the cells, because they have been fitted (taken into account) in the loglinear model.

For comparison, you might want to try fitting the model of homogeneous association. This allows all pairs of factors to be associated, but asserts that each pairwise association is the same across the levels of the remaining factor. The resulting plot displays the contributions to a 3-way association, but is not shown here.
> mosaic(~ Day + Network + Time, data=TV2,
+       expected=~Day:Time + Day:Network + Time:Network,
+       legend=FALSE, gp=shading_Friendly)

5.2 Mosaics for glm() and gnm() models

The vcdExtra package provides an additional method, mosaic.glm() for models fit with glm() and gnm(). These are not restricted to the Poisson family, but only apply to cases where the response variable is non-negative.

**Example:** Here, we plot the independence and the linear-by-linear association model for the Mental health data from Section 4.2. These examples illustrate some of the options for labeling (variable names and residuals printed in cells). Note that the formula supplied to mosaic() for "glm" objects refers to the order of factors displayed in the plot, not the model.

```r
> long.labels <- list(set_varnames = c(mental="Mental Health Status", ses="Parent SES"))
> mosaic(indep, ~ses+mental, residuals_type="rstandard",
+        labeling_args = long.labels, labeling=labeling_residuals,
+        main="Mental health data: Independence")
> mosaic(linlin, ~ses+mental, residuals_type="rstandard",
+        labeling_args = long.labels, labeling=labeling_residuals, suppress=1,
+        gp=shading_Friendly, main="Mental health data: Linear x Linear")
```

The gnm package also fits a wide variety of models with nonlinear terms or terms for structured associations of table variables. In the following, we fit the RC(1) model

\[
\log(m_{ij}) = \mu + \lambda^A_i + \lambda^B_j + \phi \mu_i \nu_j
\]

This is similar to the linear by linear model, except that the row effect parameters \((\mu_i)\) and column parameters \((\nu_j)\) are estimated from the data rather than given assigned equally-spaced values. The multiplicative terms are specified by the Mult().

Models fit with gnm() are of class = c("gnm", "glm", "lm"), so all *.glm methods apply, unless overridden in the gnm package.
Mental$mental <- C(Mental$mental, treatment)
Mental$ses <- C(Mental$ses, treatment)
RC1model <- gnm(Freq ~ mental + ses + Mult(mental, ses),
  family = poisson, data = Mental)
mosaic(RC1model, residuals_type="rstandard", labeling_args = long.labels,
  labeling=labeling_residuals, suppress=1, gp=shading_Friendly,
  main="Mental health data: RC(1) model")

Other forms of nonlinear terms are provided for the inverse of a predictor (Inv()) and the exponential of a predictor (Exp()). You should read vignette("gnmOverview", package="gnm") for further details.

### 5.3 Mosaic tips and techniques

The vcd package implements an extremely general collection of graphical methods for n-way frequency tables within the strucplot framework, which includes mosaic plots (mosaic()), as well as association plots (assoc()), sieve diagrams (sieve()), as well as tabular displays (structable())

The graphical methods in vcd support a wide of options that control almost all of the details of the plots, but it is often difficult to determine what arguments you need to supply to achieve a given effect from the help(). As a first step, you should read the vignette("strucplot") in vcd to understand the overall structure of these plot methods. The notes below describe a few useful things that may not be obvious, or can be done in different ways.

#### 5.3.1 Changing the labels for variables and levels

With data in contingency table form or as a frequency data frame, it often happens that the variable names and/or the level values of the factors, while suitable for analysis, are less than adequate when used in mosaic plots and other strucplot displays.

For example, we might prefer that a variable named ses appear as "Socioeconomic Status", or a factor with levels c("M", "F") be labeled using c("Male", "Female") in a plot. Or, sometimes we start with a factor whose levels are fully spelled out (e.g., c("strongly disagree", "disagree", "neutral", "agree", "strongly agree")), only to find that the level labels overlap in graphic displays.

The strucplot framework in vcd provides an extremely large variety of functions and options for controlling almost all details of text labels in mosaics and other plots. See help(labelings) for an overview.

For example, in Section 2.1 we showed how to rearrange the dimensions of the UCBAdmissions table, change the names of the table variables, and relabel the levels of one of the table variables. The code below changes the actual table for plotting purposes, but we pointed out that these changes can create other problems in analysis.

```r
> UCB <- aperm(UCBAdmissions, c(2, 1, 3))
> names(dimnames(UCB)) <- c("Sex", "Admit?", "Department")
> dimnames(UCB)[[2]] <- c("Yes", "No")
```

The same effects can be achieved without modifying the data using the set_varnames and set_labels options in mosaic() as follows:

```r
> vnames <- list(set_varnames = c(Admit="Admission", Gender="Sex", Dept="Department"))
> lnames <- list(Admit = c("Yes", "No"),
  Gender = c("Males", "Females"),
  Dept = LETTERS[1:6])
> mosaic(UCBAdmissions, labeling_args=vlnames, set_labels=lnames)
```
In some cases, it may be sufficient to abbreviate (or clip, or rotate) level names to avoid overlap. For example, the statements below produce another version of Figure 7 with days of the week abbreviated to their first three letters. Section 4 in the vignette("strucplot") provides many other examples.

```r
> 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,
+ labeling_args=list(abbreviate=c(Day=3)) )
```

6 Continuous predictors

When continuous predictors are available—and potentially important—in explaining a categorical outcome, models for that outcome include: logistic regression (binary response), the proportional odds model (ordered polytomous response), multinomial (generalized) logistic regression. Many of these are special cases of the generalized linear model using the "poisson" or "binomial" family and their relatives.

6.1 Spine and conditional density plots

I don’t go into fitting such models here, but I would be remiss not to illustrate some visualizations in vcd that are helpful here. The first of these is the spine plot or spinogram (Hummel 1996) (produced with spine()). These are special cases of mosaic plots with specific spacing and shading to show how a categorical response varies with a continuous or categorical predictor.

They are also a generalization of stacked bar plots where not the heights but the widths of the bars corresponds to the relative frequencies of x. The heights of the bars then correspond to the conditional relative frequencies of y in every x group.

**Example:** For the Arthritis data, we can see how Improved varies with Age as follows. spine() takes a formula of the form y ~ x with a single dependent factor and a single explanatory variable x (a numeric variable or a factor). The range of a numeric variable is divided into intervals based on the breaks argument, and stacked bars are drawn to show the distribution of y as x varies. As shown below, the discrete table that is visualized is returned by the function.

```r
> (spine(Improved ~ Age, data = Arthritis, breaks = 3))

<table>
<thead>
<tr>
<th>Improved</th>
<th>Age</th>
<th>None</th>
<th>Some</th>
<th>Marked</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20,40]</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(40,60]</td>
<td>21</td>
<td>3</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>(60,80]</td>
<td>11</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

> (spine(Improved ~ Age, data = Arthritis, breaks = "Scott"))

<table>
<thead>
<tr>
<th>Improved</th>
<th>Age</th>
<th>None</th>
<th>Some</th>
<th>Marked</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20,30]</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(30,40]</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(40,50]</td>
<td>9</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>(50,60]</td>
<td>12</td>
<td>3</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>(60,70]</td>
<td>11</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>(70,80]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```
The conditional density plot (Hofmann and Theus 2005) is a further generalization. This visualization technique is similar to spinograms, but uses a smoothing approach rather than discretizing the explanatory variable. As well, it uses the original x axis and not a distorted one.

> cdplot(Improved ~ Age, data = Arthritis)
> with(Arthritis, rug(jitter(Age), col="white", quiet=TRUE))

Figure 8: Conditional density plot for the `Arthritis` data showing the variation of Improved with Age.
In such plots, it is useful to also see the distribution of the observations across the horizontal axis, e.g., with a `rug()` plot. Figure 8 uses `cdplot()` from the `graphics` package rather than `cd_plot()` from `vcd`, and is produced with

```r
> cdplot(Improved ~ Age, data = Arthritis)
> with(Arthritis, rug(jitter(Age), col="white", quiet=TRUE))
```

From Figure 8 it can be easily seen that the proportion of patients reporting Some or Marked improvement increases with Age, but there are some peculiar bumps in the distribution. These may be real or artifactual, but they would be hard to see with most other visualization methods. When we switch from non-parametric data exploration to parametric statistical models, such effects are easily missed.

### 6.2 Model-based plots: effect plots and `ggplot2` plots

The nonparametric conditional density plot uses smoothing methods to convey the distributions of the response variable, but displays that are simpler to interpret can often be obtained by plotting the predicted response from a parametric model.

For complex `glm()` models with interaction effects, the `effects` package provides the most useful displays, plotting the predicted values for a given term, averaging over other predictors not included in that term. I don’t illustrate this here, but see ?? and `help(package="effects").`

Here I just briefly illustrate the capabilities of the `ggplot2` package for model-smoothed plots of categorical responses in `glm()` models.

**Example:** The Donner data frame in `vcdExtra` gives details on the survival of 90 members of the Donner party, a group of people who attempted to migrate to California in 1846. They were trapped by an early blizzard on the eastern side of the Sierra Nevada mountains, and before they could be rescued, nearly half of the party had died. What factors affected who lived and who died?

```r
> data(Donner, package="vcdExtra")
> str(Donner)

'data.frame': 90 obs. of 5 variables:
$ family : Factor w/ 10 levels "Breen","Donner",...:
$ age : int 23 13 15 14 40 51 9 3 8 ...
$ sex : Factor w/ 2 levels "Female","Male":
$ survived: int 0 1 1 1 1 1 1 1 1 1 ...
$ death : POSIXct, format: "1846-12-29" NA ...
```

A potential model of interest is the logistic regression model for $Pr(survived)$, allowing separate fits for males and females as a function of `age`. The key to this is the `stat_smooth()` function, using `method = "glm", method.args = list(family = binomial)`. The formula = `y ~ x` specifies a linear fit on the logit scale (Figure 9, left)

```r
> # separate linear fits on age for M/F
> ggplot(Donner, aes(age, survived, color = sex)) +
  +  geom_point(position = position_jitter(height = 0.02, width = 0)) +
  +  stat_smooth(method = "glm", method.args = list(family = binomial), formula = y ~ x,
                  + alpha = 0.2, size=2, aes(fill = sex))
```

Alternatively, we can allow a quadratic relation with `age` by specifying `formula = y ~ poly(x,2)` (Figure 9, right).
> # separate quadratics
> ggplot(Donner, aes(age, survived, color = sex)) +
>     geom_point(position = position_jitter(height = 0.02, width = 0)) +
>     stat_smooth(method = "glm", method.args = list(family = binomial), formula = y ~ poly(x, 2),
>                  alpha = 0.2, size = 2, aes(fill = sex))

Figure 9: Logistic regression plots for the Donner data showing survival vs. age, by sex. Left: linear logistic model; right: quadratic model

These plots very nicely show (a) the fitted $Pr(survived)$ for males and females; (b) confidence bands around the smoothed model fits and (c) the individual observations by jittered points at 0 and 1 for those who died and survived, respectively.

References


