Tables in R – A quick practical overview
<preliminary blueprint version>

by Andri Signorell

Helsana Versicherungen AG, Health Sciences, Zurich
HWZ University of Applied Sciences in Business Administration, Zurich

andri@signorell.net

August 14, 2015

Tabulating data is both, trivial and complicated. After all it is just about counting data. But the underlying data structures are diverse and technically abstract, especially when there are more than two dimensions involved. Thus there are many functions to handle and process tables in the respective representation, which makes the situation somewhat confusing. There are some gaps in base R function list that are filled by DescTools.
This document aims to briefly summarise, how to create, handle and describe count data in tables. Some examples from the SAS-documentation FREQ are reproduced.

Note:
For all the examples in this document, `library(DescTools)` must be declared.

1 Starting Point

Data will normally be given in one of the following three data structures.

A) Single case

The raw data in form of a data.frame (or a matrix), each row contains one case, here one person:

```
head(d.col, 20)

    Hair   Eye  Sex
1  Black Brown Male
2  Black Brown Male
3  Black Brown Male
4  Black Brown Male
5  Black Brown Male
6  Black Brown Male
7  Black Brown Male
...

32 Black Brown Male
33 Brown Brown Male


...
```

B) Frequency

Unique combinations of factors extended with their counts, often called weights (column “Freq”):

```
d.weight

    Hair   Eye    Sex Freq
1  Black Brown   Male   32
2  Brown Brown   Male   53
3    Red Brown   Male   10
4  Blond Brown   Male    3
5  Black  Blue   Male   11
6  Brown  Blue   Male   50
7    Red  Blue   Male   10
8  Blond  Blue   Male   30
9  Black Hazel   Male   10
10 Brown Hazel   Male   25


...
```

C) Table

A multidimensional table (or an array, matrix):

```
tab

, , Sex = Male

       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8

, , Sex = Female

       Eye
Hair    Brown Blue Hazel Green

...
```

Either we have the raw data organised case-by-case in a data.frame (case A). Then a contingency table can be built by tabulating the data. There are several commands for this described in chapter “Tabulate”.
Or the data are given as a combination of factor levels and one count variable (typically organised as a data.frame too) (case B). The first line in this representation means, that we have 32 men with black hair and brown eyes in our sample. This corresponds to the cell [1, 1, 1] in the representation C). In representation A we have 32 rows with the same content Black/Brown/Male.
How to create such a structure is described in “Expanding”. There are functions to convert this structure to a table or to recreate the raw dataset. This is detailed in the chapter “Convert”.
When the data are given directly as a table (case C), there are again several ways how to enter that into R. This is the content of the first chapter “Create tables”.
How to process tables is described in the chapters “Reorganize”, “Aggregate”, “Append”, “Convert”.

Usually B) is the most economic representation of frequency data whereas the case-by-case form in A) is the least (provided the data set is purely categorical). The built-in data sets from the R base system that are purely categorical usually come in the form of tables (C).

2 Create Table

Some remarks about creating tables in general.

2.1 Creating from the scratch

There are several ways to enter contingency table data into R. Let's illustrate some approaches with a table concerning party affiliation by gender:

Table 2.1 Tabulating Party versus Gender, Agresti (2007) p. 39

Gender	**Party**		
	Democrat	Independent	Republican
M	762	327	468
F	484	239	477

rbind
as.table

The first approach uses the function `rbind` and builds a matrix row by row. The `as.table()` function lets R know that the matrix represents a contingency table of counts.:

```
tab <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(tab) <- list(gender = c("M", "F"),
                      party  = c("Democrat", "Independent", "Republican"))
tab
```

```
##      party
## gender Democrat Independent Republican
##      M      762         327        468
##      F      484         239        477
```

matrix

The exactly same result can be created by the second approach, using the function `matrix`. Note that, by default, `matrix()` uses the elements supplied by columns in the result, unless you specify `byrow=TRUE`.

```
as.table(matrix(c(762, 327, 468, 484, 239, 477), nrow=2, byrow=TRUE,
                dimnames=list(gender= c("M", "F"),
                              party = c("Democrat", "Independent", "Republican")))
)
```

TextToTable

The third way uses `TextToTable` to convert a text to a table. Within this function `read.table` is used to enter the data and to convert the data.frame to a table. `header=TRUE` will take the names of the variables from its first line.
The column names and row names will automatically be chosen, if the first row contains one fewer field than the number of columns. The dimension names can be provided with the specific argument:

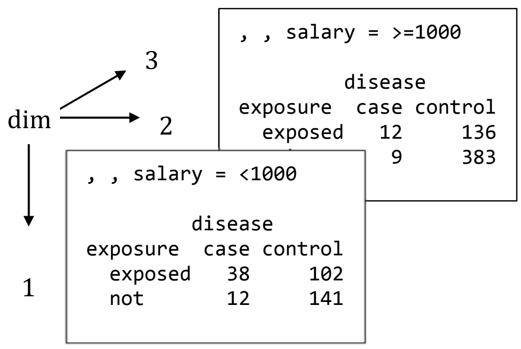
```
txt <- "
   Democrat, Independent, Republican
M, 762, 327, 468
F, 484, 239, 477"

TextToTable(txt, sep=",", dimnames=c("gender", "party"))
```

array

Higher dimensional arrays can be defined with the function `array` by using the argument `dim`:

```
salary <- array(
      c(38, 12, 102, 141, 12, 9, 136, 383),
      dim=c(2, 2, 2),
      dimnames=list(exposure = c("exposed", "not"),
                    disease  = c("case", "control"),
                    salary   = c("<1000", ">=1000"))
                    )
```



```
, , salary = >=1000

         disease
exposure  case control
  exposed   12     136
            9     383
```

```
, , salary = <1000

         disease
exposure  case control
  exposed   38     102
  not       12     141
```

Note how the dimensions are organised:
The first dimension corresponds to the rows,
the second to the columns, the third to the depth, and so on.

read.ftable

Higher dimensional tables condensed in flat tables with more than one column, resp. row variable, can be created from the appropriate text chunk by means of the base function `read.ftable`. (Beware not to insert spaces at the beginning of the lines.)

```
txt <-
"          Sex  Male                  Female
          Eye Brown Blue Hazel Green  Brown Blue Hazel Green
Hair
Black        32   11    10     3     36    9     5     2
Brown        53   50    25    15     66   34    29    14
Red          10   10     7     7     16    7     7     7
Blond         3   30     5     8      4   64     5     8
"
tab <- as.table(read.ftable(textConnection(txt)))
```

2.2 Expanding

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.

expand.grid

```
tab <- data.frame(expand.grid(
   Hair  = c("Black", "Brown", "Red", "Blond"),
   Eye   = c("Brown", "Blue", "Hazel", "Green"),
   Sex   = c("Male", "Female")),
   count = c(32,53,10,3,11,50,10,30,10,25,7,5,3,15,7,8,
             36,66,16,4,9,34,7,64,5,29,7,5,2,14,7,8)
)
```

`expand.grid` will create all the interactions between the given factors. `data.frame` will bind them with the count variable, denominating the number of observations. This will be a type B representation of count data, which can be converted with `xtabs` to a table. (See Chapter “Convert”)

2.3 SAS datalines

Longstanding predominance of SAS entails, that small data tables in examples and documents are often reported in the SAS datalines format. Creating a table based on this in R is not straight forward, as there might be more than one case per row (as in the example below).
The function `ParseSASDatalines` parses the syntax and creates a table named after the *data* statement, using given column names (specified by the keyword *input*).

```
ParseSASDatalines("
  data SummerSchool;
  input Gender $ Internship $ Enrollment $ Count @@;
  datalines;
  boys  yes yes 35  boys  yes no 29
  boys   no yes 14  boys   no no 27
  girls yes yes 32  girls yes no 10
  girls  no yes 53  girls  no no 23
;")
```

ParseSAS-Datalines

The command above will create a new data object named *SummerSchool* in the GlobalEnvironment.

3 Tabulate

The built-in data set *HairEyeColor* has the class *table*. Let's take this table and turn it into a case-by-case data frame first.

Untable

```
d.col <- Untable(HairEyeColor)
head(d.col, 3)

## Hair   Eye  Sex
## 1 Black Brown Male
## 2 Black Brown Male
## 3 Black Brown Male
```

From here we can start tabulating again. The simplest case is to tabulate a single vector. The function `table` yields the absolute frequencies and `prop.table` the proportions:

table
prop.table

```
table(d.col$Hair)

## Black Brown   Red Blond
##   108   286    71   127
```

```
prop.table(table(d.col$Hair))

##    Black     Brown       Red     Blond
## 0.1824324 0.4831081 0.1199324 0.2145270
```

A combination of both extended with the cumulative sums for both, absolute and relative frequencies, can be produced by `Freq` (here ordered by decreasing frequency):

Freq

```
Freq(d.col$Hair, ord="desc")

##   level freq  perc cumfreq cumperc
## 1 Brown  286 0.483     286   0.483
## 2 Blond  127 0.215     413   0.698
## 3 Black  108 0.182     521   0.880
## 4   Red   71 0.120     592   1.000
```

By means of the `table` function we can produce multidimensional contingency tables (aka. crosstabs) as well. We use the command `with` here, so we can avoid having to qualify every column name with the name of the data.frame (which makes the code more readable).

```
with(d.col, table(Hair, Eye))

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

The first entered variable will be the row variable, the second one the column variable. Missing values are ignored by default. In order to include NA as a category in counts, use the option `useNA="always"`.

A relative frequency table can be produced using the function `prop.table`, which takes a table object as argument:

```
with(d.col, prop.table(table(Hair, Eye), margins=NULL))
```

```
##        Eye
## Hair          Brown        Blue       Hazel       Green
##   Black 0.114864865 0.033783784 0.025337838 0.008445946
##   Brown 0.201013514 0.141891892 0.091216216 0.048986486
##   Red   0.043918919 0.028716216 0.023648649 0.023648649
##   Blond 0.011824324 0.158783784 0.016891892 0.027027027
```

The function `PercTable` combines that and allows adding marginal sums in one step:

```
PercTable(Hair ~ Eye, data=d.col, rfrq="111", margins=c(1,2))
```

PercTable

```
##             Eye  Brown   Blue  Hazel  Green    Sum
## Hair
## Black freq          68     20     15      5    108
##       perc        .115   .034   .025   .008   .182
##       p.row       .630   .185   .139   .046      .
##       p.col       .309   .093   .161   .078      .
## Brown freq         119     84     54     29    286
##       perc        .201   .142   .091   .049   .483
##       p.row       .416   .294   .189   .101      .
##       p.col       .541   .391   .581   .453      .
## Red   freq          26     17     14     14     71
##       perc        .044   .029   .024   .024   .120
##       p.row       .366   .239   .197   .197      .
##       p.col       .118   .079   .151   .219      .
## Blond freq           7     94     10     16    127
##       perc        .012   .159   .017   .027   .215
##       p.row       .055   .740   .079   .126      .
##       p.col       .032   .437   .108   .250      .
## Sum   freq         220    215     93     64    592
##       perc        .372   .363   .157   .108  1.000
##       p.row          .      .      .      .      .
##       p.col          .      .      .      .      .
```

There are more options, as expected values or standard residuals, which can optionally be integrated.

The marginal tables can be produced by R base function `margin.table` or by the somewhat extended function `MarginTable` in DescTools:

```
MarginTable(tab, ord="desc")
```

MarginTable

```
## $Hair
##   level freq  perc cumfreq cumperc
## 1 Brown  286 0.483     286   0.483
## 2 Blond  127 0.215     413   0.698
## 3 Black  108 0.182     521   0.880
## 4   Red   71 0.120     592   1.000

## $Eye
##   level freq  perc cumfreq cumperc
## 1 Brown  220 0.372     220   0.372
## 2  Blue  215 0.363     435   0.735
## 3 Hazel   93 0.157     528   0.892
## 4 Green   64 0.108     592   1.000
```

`table` does not come with a formula interface, but the `xtabs` function has one. This allows us to create multidimensional crosstabulations using formula style input. The result is a contingency table in array format, whose dimensions are determined by the terms on the right side of the formula.

4 Reorganize

Say we created a three dimensional table with *Hair*, *Eye* and *Sex* as variables and typically got a 3-dim array as result. This will be displayed as:

```
(tab <- with(d.col, table(Hair, Eye, Sex)))

## , , Sex = Male
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8

##, , Sex = Female
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
##
```

ftable

To combine this multidimensional structure into a flat table while preserving all the details, there's the function `ftable`. The variables to be placed in the rows can be defined by the argument `row.vars`, which can be a vector (denoting multiple dimensions). So to put *Eye* (variable 2) and *Sex* (variable 3) in the rows and *Hair* as column variable, we write:

```
ftable(tab, row.vars = c(2,3))

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

The tab, as we constructed it, has the *Hair* as rows (1), the *Eye* as columns (2), and the *Sex* as third dimension (3) defined. The dimensions and dimension names follow the defined order:

```
dimnames(tab)

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

aperm

If we would like to have the dimensions in a different order, we can use the base function `aperm`. Let's say we wanted *Eye* as row variable and *Sex* a column variable and consequently *Hair* as 3th variable, we can tell `aperm` to set dimension 2 on the first position, 3 on the second and 1 on the third position. So we get:

```
aperm(tab, c(2,3,1))

##, , Hair = Black
##
##         Sex
## Eye      Male Female
##    Brown   32     36
##    Blue    11      9
##    Hazel   10      5
##    Green    3      2
##
##, , Hair = Brown
##
##         Sex
## Eye      Male Female
##    Brown   53     66
##    Blue    50     34
##
...
```

The following would by the way not work:

```
tab["Eye", "Sex", "Hair"]
Error in tab["Eye", "Sex", "Hair"] : subscript out of bounds
```

To reorder the sequence of the levels (within a dimension) in our table, we could use `reorder.factor`. Say we would like to have the sequence *Blue*, *Green*, *Hazel*, *Brown* for the *Eye* colour. Of course, when having the raw data, we would use

```
factor(d.col$Eye, levels=c("Blue", "Green", "Hazel", "Brown"))
```

and any table afterwards would take this level order. But how can we change this in an already created table? The answer is obvious (but may yet be unexpected in this context): Use the index! This works with the level names as well as with the index positions.

```
tab[ , c("Blue", "Green", "Hazel", "Brown"), ]

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

Rev

For simply reversing the levels there’s the function `Rev`, which has a table interface implemented. The function accepts a `margins` argument, defining the dimensions whose levels should be reversed. Compare the reversed levels of *Hair* and *Sex*:

```
tab

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

```
Rev(tab, margin = c(1, 3))

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

Renaming level names can be achieved by refining the dimension names.

```
dimnames(tab)$Sex <- c("men", "women")
```

5 Aggregate

Sometimes we might want to aggregate an existing table along one or several dimensions.. Say we’d like to get rid of the *Hair* dimension, but retain all the frequency information for the other dimensions. For this we can use `apply` as we would in the case of a matrix. The function takes as well vectors for the margins.
So we just sum up all cases along the 1st dimension (*Hair*) and retain the other two (2, 3) and we get:

apply

```
apply(tab, c(2,3), sum)

##        Sex
## Eye     Male Female
##   Brown   98    122
##   Blue   101    114
##   Hazel   47     46
##   Green   33     31
```

```
apply(tab, 1, sum)

## Black Brown   Red Blond
##   108   286    71   127
```

Single margins could be calculated analogously, as demonstrated above.

xtabs

If tab was created with `xtabs`, the aggregation can be done directly by using the formula interface, which typically is clearer and more readable.

```
xtab <- xtabs(~., d.col)
xtabs(Freq ~ Eye + Sex, xtab)

##       Sex
## Eye     Male Female
##   Brown   98    122
##   Blue   101    114
##   Hazel   47     46
##   Green   33     31
```

If we wanted to consolidate some levels, we can with `CollapseTable`. Say we want to combine brown and hazel eyes to a new category *Browny*, as well as just have two groups of *Hair*, namely *Dark* and *Fair*:

CollapseTable

```
CollapseTable(tab, Eye=c("Browny","Blue","Browny","Green"),
                   Hair=c("Dark","Dark","Fair","Fair"))
```

```
## , , Sex = Male
##
##       Eye
## Hair   Browny Blue Green
##   Dark    120   61    18
##   Fair     25   40    15
##
##, , Sex = Female
##
##       Eye
## Hair   Browny Blue Green
##   Dark    136   43    16
##   Fair     32   71    15
```

6 Append

Tables can be pasted together. This can especially be useful, when two tables of the same dimension should be put together to a 3-dimensional array. But in contrast to the 2-dimensional case, where the functions `rbind` and `cbind` exist, base R does not contain a respective function for higher dimensional tables. In DescTools there's the function `Abind` included for this purpose (borrowed from the abind package).

```
a <- HairEyeColor[,,1]     # male table
b <- HairEyeColor[,,2]     # female table

Abind(Male=a, Female=b, along=3)
```

Abind

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

The first step separates the table for males from the females. `Abind` reverses this step and binds the two tables together again. This can happen along all possible dimensions.
In the example above a new dimension is introduced by setting along = 3.
`Abind(a,b,along=2)` would bind the tables by columns (as `cbind` does), whereas `Abind(a,b,along=1)` would give the same result as `rbind(a,b)`.

7 Convert

Time and again newbies wonder how to convert tables from one to the other form. Base R comprises most of the required functions, but not quite all.
Let's say we have the three forms of table given as:

```
d.col <- Untable(HairEyeColor)              # case-by-case,       A)
d.weight <- as.data.frame(HairEyeColor)     # frequency,          B)
tab <- HairEyeColor                         # table,              C)
```

The conversions can be made as follows.

A) $\longrightarrow$ B)	`1) as.data.frame(table(d.col))`
	`2) aggregate(rep(1, nrow(d.col)),`
	`             by=d.col, FUN=length)`
		This is actually A) to C) to B)! Solution 2) will yield the nonzero entries only.

A) $\longleftarrow$ B)	`Untable(d.weight)`	library(DescTools)
A) $\longrightarrow$ C)	`table(d.col)`	
A) $\longleftarrow$ C)	`Untable(tab)`	
B) $\longrightarrow$ C)	`xtabs(Freq ~ ., d.weight)`	
B) $\longleftarrow$ C)	`as.data.frame(tab)`	If tab is defined as matrix, as.data.frame has to be specified explicitly as as.data.frame.table!

as.data.frame

aggregate

Untable

xtabs

8 Print and Format

All table connected classes have their print methods which do not call for any further explanation. There are several approaches out there, how to turn tables into XML, HTML or LATEX. DescTools contains two functions for sending tables to MS-Word. `WrdTable` would create the table in Word and transfer the cell information appropriately.

Let's create an artificial table, with one cell being 0 and one being NA. Then we format the counts with a big.mark and set 0 digits. The zero values should be expressed as "-" and the NAs as "missing". Finally all is to be aligned to the right.

```
(tab <- as.table(matrix(c(2000, 0, 34, NA), nrow=2)))
##       A    B
## A 2000   34
## B    0

tab[] <- Format(tab, big.mark = "'", digits=0, zero.form="-", na.form="Missing")
tab[] <- StrAlign(tab, "\\$")
tab

##         A       B
## A   2'000      34
## B       - Missing
```

The counts and percentages in `PercTable` can be formatted by setting the options `fmt.abs` and `fmt.per`. The percentages are formatted as .000 and the counts with a space for big.mark.

```
options(fmt.abs=structure(list(digits=1, big.mark=" "), class="fmt"))
options(fmt.per=structure(list(digits=3, leading="drop"), class="fmt"))
PercTable(tab)

##               A       B
##
## A freq  2 000.0    34.0
##   perc     .720    .012
##
## B freq      0.0   745.0
##   perc     .000    .268
```

Note that by applying formats to the cells, the numeric values turn to strings and cannot be subsequently used for further calculating.

`FixToTab` is trying to chop the fixed font output of a table given as text to a tab delimited table.

9 Plot

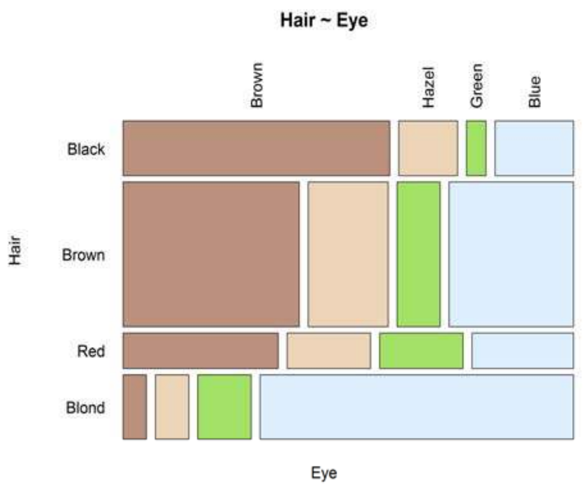
The usual representation of a table is a mosaicplot. Such a plot will display the conditional frequencies in two directions.

```
tab <- as.table(apply(HairEyeColor, c(1,2), sum))
tab <- tab[,c("Brown","Hazel","Green","Blue")]
cols <- SetAlpha(c("sienna4", "burlywood", "chartreuse3", "slategray1"), 0.6)

PlotMosaic(tab, col=cols, main = "Hair ~ Eye")
```

PlotMosaic

This will display the following fact:

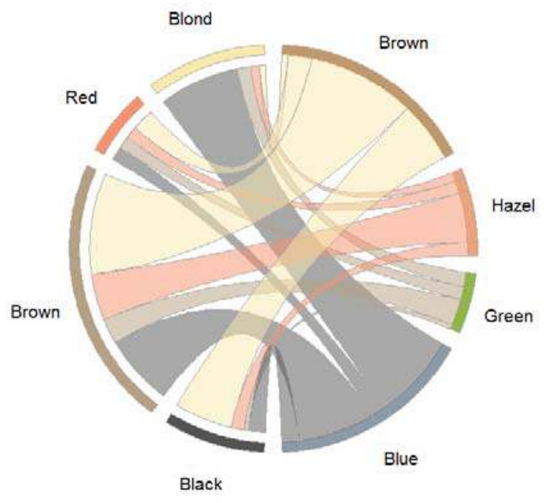


```
PercTable(tab, freq=FALSE, rfrq="010")

##       Eye
##         Brown Hazel Green  Blue
## Hair
## Black  63.0% 13.9%  4.6% 18.5%
## Brown  41.6% 18.9% 10.1% 29.4%
## Red    36.6% 19.7% 19.7% 23.9%
## Blond   5.5%  7.9% 12.6% 74.0%

prop.table(margin.table(tab, 1))
## Hair
##  Black   Brown     Red   Blond
## 0.1824  0.4831  0.1199  0.2145
```

The idea to describe proportions in circles is rather new. It emphasises the association structure of the data. The left side of the circle represents the rows and the right one the columns. We see both margins in the plot.



```
cols <- c(
  rgb(249,234,174,maxColorValue = 255),
  rgb(250,146,109,maxColorValue = 255),
  rgb(183,161,130,maxColorValue = 255),
  rgb(83,83,83,maxColorValue = 255),
  rgb(rev(c("sienna4", "burlywood",
        "chartreuse3", "slategray1")))

PlotCirc(t(tab), acol=cols)
```

PlotCirc

10 Descriptions, Statistics and Tests

Let's create a 2-dimensional table and describe it with some bells and whistles. The argument `verbose = high` will maximize the volume of output:

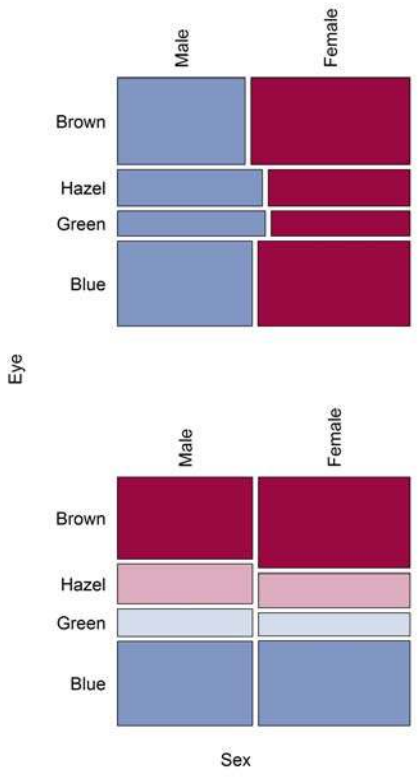
Desc

```
# aggregate 3-d table to Eye and Hair colour only:
tab <- as.table(apply(HairEyeColor, c(2,3), sum))

# order the levels along colours:
tab <- tab[c("Brown","Hazel","Green","Blue"),]

# describe the table
Desc(tab, verbose="high")
```

```
## Summary:
## n: 592, rows: 4, columns: 2
##
## Pearson's Chi-squared test:
##   X-squared = 1.5298, df = 3, p-value = 0.6754
## Pearson's Chi-squared test (cont. adj):
##   X-squared = 1.5298, df = 3, p-value = 0.6754
## Likelihood Ratio:
##   X-squared = 1.5294, df = 3, p-value = 0.6755
## Mantel-Haenszel Chi-squared:
##   X-squared = 0.2438, df = 1, p-value = 0.6214
##
##                          estimate  lwr.ci  upr.ci
## Phi Coeff.                 0.0508       -       -
## Contingency Coeff.         0.0508       -       -
## Cramer V                   0.0508  0.0000  0.1076
## Goodman Kruskal Gamma     -0.0343 -0.1610  0.0924
## Kendall Tau-b             -0.0202 -0.0949  0.0544
## Stuart Tau-c              -0.0238 -0.1115  0.0640
## Somers D C|R              -0.0171 -0.0804  0.0461
## Somers D R|C              -0.0238 -0.1119  0.0642
## Pearson Correlation       -0.0203 -0.1007  0.0604
## Spearman Correlation      -0.0218 -0.1022  0.0589
## Lambda C|R                 0.0108  0.0000  0.0983
## Lambda R|C                 0.0081  0.0000  0.0821
## Lambda sym                 0.0092  0.0000  0.0658
## Uncertainty Coeff. C|R     0.0019 -0.0041  0.0078
## Uncertainty Coeff. R|C     0.0010 -0.0022  0.0042
## Uncertainty Coeff. sym     0.0013 -0.0029  0.0055
## Mutual Information         0.0019       -       -
##               Sex
##                 Male Female     Sum
## Eye
## Brown freq        98    122     220
##       perc     16.6%  20.6%   37.2%
##       p.row    44.5%  55.5%       .
##       p.col    35.1%  39.0%       .
##
## Hazel freq        47     46      93
##       perc      7.9%   7.8%   15.7%
##       p.row    50.5%  49.5%       .
##       p.col    16.8%  14.7%       .
##
## Green freq        33     31      64
##       perc      5.6%   5.2%   10.8%
##       p.row    51.6%  48.4%       .
##       p.col    11.8%   9.9%       .
##
## Blue  freq       101    114     215
##       perc     17.1%  19.3%   36.3%
##       p.row    47.0%  53.0%       .
##       p.col    36.2%  36.4%       .
##
## Sum   freq       279    313     592
##       perc     47.1%  52.9%  100.0%
##       p.row        .      .       .
##       p.col        .      .       .
```



The first line reports the total n in the table and the dimension, so we have 592 Persons in a table with 4 rows and two columns. Then several Chi-Square-tests are calculated. The null hypothesis is that the eye colour is not associated with the sex. The small value of the chi-square statistic, 1.5298, and the p-value of 0.6754 indicate that the null hypothesis can't be rejected at the 0.05 level of significance. Thus we would conclude that the observation does not indicate an association between eye colour and sex of the person.
The Pearson Chi-Square statistic involves the differences between the observed cell frequencies and the expected deviation-frequencies. Following a rule of thumb the expected frequency in every cell of the table should not be less than 5. If it is a message will be printed.

The Continuity-Adjusted Chi-Square test statistic consists of the Pearson Chi-Square modified with an adjustment for continuity. As the sample size increases, the difference between the continuity-adjusted and Pearson Chi-Square decreases. Thus in very large samples as we have here the two statistics are almost the same.
This test statistic is also an alternative to Pearson's if any of the expected values in a 2x2 table are less than 5 (Cody and Smith, 1997). Some prefer to use the continuity-adjusted Chi-Square statistic when the sample size is small regardless of the expected values.

The expected frequencies can be obtained by using the `expected` option on the Desc command (`Desc(tab, verbose="high", expected=TRUE)`). Additionally, the difference between the observed cell count and the expected cell count will be reported when using the `residuals=TRUE` and `stdres=TRUE` option for the standardized residuals (amount that each cell contributes to the value of the test statistic).

```
options(fmt.num=structure(list(digits=3), class="fmt"))
PercTable(tab, freq=TRUE, rfrq="000",
+          expected=TRUE, residuals=TRUE)
```

PercTable, ExpFreq

```
##              Sex
##                  Male  Female
## Eye
## Brown freq         98     122
##       exp     103.682 116.318
##       res      -0.558   0.527
##
## Blue  freq        101     114
##       exp     101.326 113.674
##       res      -0.032   0.031
##
## Hazel freq         47      46
##       exp      43.829  49.171
##       res       0.479  -0.452
##
## Green freq         33      31
##       exp      30.162  33.838
##       res       0.517  -0.488
```

This output shows the observed frequencies (`freq`), the expected values (`exp`) and the Pearson residuals (`res`), whose squared values are each cell's contribution to the Chi-Square statistic. None of the expected values are less than 5, so we feel comfortable with the result of the Chi-Square test above.

The Likelihood Ratio Chi-Square is asymptotically equivalent to the Pearson Chi-Square (and Mantel-Haenszel Chi-Square) but not usually used when analyzing 2x2 tables. It is used in logistic regression and loglinear modeling which involves contingency tables.

The Mantel-Haenszel Chi-Square is related to the Pearson Chi- Square and, in the 2x2 case, as the sample size gets large these statistics converge. In the case of 2xC or Rx2 tables, if the variable with more than 2 categories is ordinal, the Mantel-Haenszel Chi-square is a test for trend while the Pearson Chi-square remains a general test for association.

When verbose option is set as high, several statistics that describe the nominal and ordinal association between the two variables of the contingency table will be computed.
The Phi coefficient is a measure of the degree of association between two categorical variables and is interpretable as a correlation coefficient. It is derived from the Chi-Square statistic, but is free of the influence of the total sample size (Fleiss, 1981). Being independent of the sample size is a desirable quality because the Chi-Square statistic itself is sensitive to sample size. As the sample size increases, the Chi-Square value will increase even if the cell proportions remain unchanged.
Pearson's contingency coefficient and Cramer's V are also derived from the chi-square and in the 2x2 table they are identical to the Phi coefficient (and similar to the Phi coefficient in interpretation). These three measures of degree of association are well suited for nominal variables in which the order of the levels is meaningless.

The following are measures of ordinal association that consider whether the variable Y tends to increase as X increases: Gamma, Kendall's tau-b, Stuart's tau-c, and Somers' D. These measures are appropriate for ordinal variables, and they classify pairs of observations as concordant or discordant. A pair is concordant if the observation with the larger value of X also has the larger value of Y. A pair is discordant if the observation with the larger value of X has the smaller value of Y. Refer to Agresti (1996) and the other references cited in the discussion of each measure of association.
The Pearson correlation coefficient and the Spearman rank correlation coefficient are also appropriate for ordinal variables. The Pearson correlation describes the strength of the

linear association between the row and column variables, and it is computed using the row and column scores specified. The Spearman correlation is computed with rank scores.
The polychoric correlation is not reported, but can be calculated with the function `CorPolychor`. It also requires ordinal variables and assumes that the variables have an underlying bivariate normal distribution.
The following measures of association do not require ordinal variables, but they are appropriate for nominal variables: lambda asymmetric, lambda symmetric, and uncertainty coefficients.
Asymptotic confidence limits for all statistics are computed. The confidence coefficient is determined according to the value of the `conf.level` option, which by default equals 0.95 and produces 95% confidence limits.

CochranTest
Sensitivity/Spec

11 Cases

11.1 Eye colour - Binomial Proportions for One-Way Frequency Tables

The binomial proportions are computed as the proportion of observations for all the levels of the variable. The following statements compute the proportion of children with brown eyes (from the data set in Example 28.1 on page 1335) and test this value against the hypothesis that the proportion is 50%. Also, these statements test whether the proportion of children with fair hair is 28%.

```
tab <- as.table(apply(HairEyeColor, 2, sum)[c("Brown","Hazel","Green","Blue")])
Desc(tab)
```

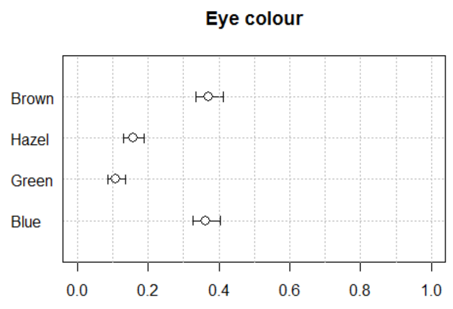
```
## ------------------------------------------------------------------------------
## tab (table)
##
## Summary:
## n: 592, rows: 4
##
## Pearson's Chi-squared test (1-dim uniform):
##   X-squared = 133.47, df = 3, p-value < 2.2e-16
##
##    level  freq   perc  cumfreq  cumperc
## 1  Brown   220  37.2%      220    37.2%
## 2  Hazel    93  15.7%      313    52.9%
## 3  Green    64  10.8%      377    63.7%
## 4   Blue   215  36.3%      592   100.0%
```

```
xci <- BinomCI(tab, sum(tab))
rownames(xci) <- rownames(tab)
print(xci, digits=3)
```

```
##         est lwr.ci upr.ci
## Brown 0.372 0.3336  0.411
## Hazel 0.157 0.1300  0.189
## Green 0.108 0.0856  0.136
## Blue  0.363 0.3254  0.403
```

Let's produce a plot of that:

PlotDot



```
PlotDot(xci[,1], main="Eye colour", pch=NA,
        args.errbars = list(
          from=xci[,2], to=xci[,3], mid=xci[,1],
          pch=21, cex=1.4),
        xlim=c(0,1))

abline(v=seq(0,1,0.1), col="grey", lty="dotted")
```

11.2 Heart – 2x2-Table

This example computes chi-square tests and Fisher's exact test to compare the probability of coronary heart disease for two types of diet. It also estimates the relative risks and computes exact confidence limits for the odds ratio.
The data set contains hypothetical data for a case-control study of high fat diet and the risk of coronary heart disease. The data can be entered as:

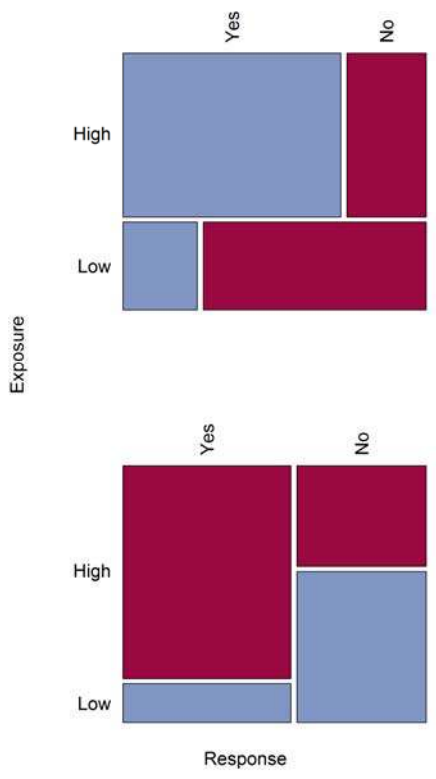
```
heart <- as.table(matrix(c(11, 2, 4, 6), nrow=2,
                         dimnames = list(Exposure = c("High", "Low"),
                                         Response = c("Yes", "No"))))
Label(heart) <- "Table of Exposure by Response"
```

The data is sorted in descending order by the variables Exposure and Response, so that the first cell of the 2x2-table contains the frequency of positive exposure and positive response.

```
Desc(heart, main="Case-Control Study of High Fat/Cholesterol Diet")
```

will produce the following result:

```
## Case-Control Study of High Fat/Cholesterol Diet
##   Table of Exposure by Response
##
##
## Summary:
## n: 23, rows: 2, columns: 2
##
## Pearson's Chi-squared test (cont. adj):
##   X-squared = 3.1879, df = 1, p-value = 0.0741
## Fisher's exact test p-value = 0.03931
## McNemar's chi-squared = 0.16667, df = 1,
##   p-value = 0.6831
##
## Warning message:
##   Exp. counts < 5: Chi-squared approx. may
##   be incorrect!!
##
##                      estimate lwr.ci upr.ci
##
## odds ratio              8.250  1.154 59.003
## rel. risk (col1)        2.933  0.850 10.120
## rel. risk (col2)        0.356  0.140  0.901
##
## Phi-Coefficient         0.464
## Contingency Coeff.      0.421
## Cramer's V              0.464
##
##                  Response
##                     Yes      No     Sum
## Exposure
## High      freq       11       4      15
##           perc    47.8%   17.4%   65.2%
##           p.row   73.3%   26.7%       .
##           p.col   84.6%   40.0%       .
##
## Low       freq        2       6       8
##           perc     8.7%   26.1%   34.8%
##           p.row   25.0%   75.0%       .
##           p.col   15.4%   60.0%       .
##
## Sum       freq       13      10      23
##           perc    56.5%   43.5%  100.0%
##           p.row       .       .       .
##           p.col       .       .       .
##
```



If the expected value of one or more cells is less than 5, the Chi-Square test may not be valid. In this case, Fisher's Exact Test is an alternative test which does not depend on the expected values. A criticism of this test is that it fixes the row and column margin totals, which in effect makes an assumption about the distribution of the variables in the population being studied. Since the expected counts in some of the cells are small, a warning that the asymptotic chi-square tests may not be appropriate is displayed. In this case, the Fisher's exact test is appropriate. It analyses whether the probability of heart disease in the high fat group differs from the one in the low fat group; since this p-value is small ($p < 0.05$), the alternative hypothesis is supported. Note that only the one-sided test will be reported.

PlotDot

The odds ratio is simply a ratio of odds. Recall that the odds of an event occurring is the ratio of p/q where p is the probability of the event occurring and q is the probability of the event not occurring. The odds ratio

$$OR = \frac{\frac{n_{11}}{n_{12}}}{\frac{n_{21}}{n_{22}}} = \frac{n_{11} \cdot n_{22}}{n_{12} \cdot n_{21}} = \frac{11 \cdot 6}{4 \cdot 2} = 8.25$$

provides in fact an estimate of the relative risk when an event is rare. This estimate indicates that the odds of heart disease is 8.25 times higher in the high fat diet group; however, the wide confidence limits (1.154, 59.003) indicate that this estimate has low precision. The odds ratio and the 95% confidence interval is included in the default description of 2x2 tables in DescTools.

The column 1 relative risk is the ratio of the column 1 risks for row 1 to row 2. The column 1 risk for row 1 is the proportion of the row 1 observations classified in column 1:

$$RR_1 = \frac{\frac{n_{11}}{n_{1.}}}{\frac{n_{21}}{n_{2.}}} = \frac{n_{11} \cdot n_{2.}}{n_{1.} \cdot n_{21}} = \frac{11 \cdot 8}{15 \cdot 2} = 2.933$$

A relative risk greater than 1 indicates that the probability of positive response is greater in row 1 than in row 2. Similarly, a relative risk less than 1 indicates that the probability of positive response is less in row 1 than in row 2. The strength of association increases with the deviation from 1.
Relative Risk is the ratio of the incidence of an outcome given one treatment or exposure level to the risk of the outcome in the other level of treatment or exposure. Recall an incidence rate is the proportion of new cases (outcomes) occurring over a period of any one time. Therefore the risk of an outcome makes sense in the context of prospective cohort studies where the outcome has not occurred in any case at the start of the study. In such a context, referring to, figure 2, the risk of outcome 1 is R1=a/r1 for exposure level 1 and R2=c/r2 for exposure level 2. The relative risk of outcome 1 is R1/R2 for subjects with exposure 1 compared to those with exposure 2.

While the Relative Risk is a measure which is appropriate for prospective cohort studies, the Odds Ratio can be used for crosssectional case-control studies as well as prospective studies. The estimates in the output are even labeled under 'Type of Study'. In both cases, a value of 1 indicates no difference between groups. Finally, the reader should verify that interchanging the row and column variables or modifying the table order will result in different values of odds ratio and relative risks. The interpretations should however remain consistent.

11.3 Skin - Agreement Study

Medical researchers are interested in evaluating the efficacy of a new treatment for a skin condition. Dermatologists from participating clinics were trained to conduct the study and to evaluate the condition. After the training, two dermatologists examined patients with the skin condition from a pilot study and rated the same patients. The possible evaluations are terrible, poor, marginal, and clear.
In order to evaluate the agreement of the diagnoses (a possible contribution to measurement error in the study), the kappa coefficient is computed.

```
ParseSASDatalines("
  data d.SkinCondition;
  input Derm1 $ Derm2 $ Count;
  datalines;
  terrible terrible 10  terrible poor 4   terrible marginal 1   terrible clear 0
  poor terrible 5       poor poor 10      poor marginal 12      poor clear 2
  marginal terrible 2   marginal poor 4   marginal marginal 12  marginal clear 5
  clear terrible 0      clear poor 2      clear marginal 6      clear clear 13
;")
skin <- xtabs(Count ~ ., d.SkinCondition)
```

The function `Agree` computes raw simple percentage agreement among raters.

```
Agree(Untable(skin))

## [1] 0.5113636
## attr(,"subjects")
## [1] 88
## attr(,"raters")
## [1] 2
```

We learn that 51.1% of the ratings were the same between the two researchers. A less coarse approach to measure agreement is Cohen's kappa.

CohenKappa

```
CohenKappa(skin, conf.level=0.95)

##     kappa    lwr.ci    upr.ci
## 0.3448753 0.2048513 0.4848994


CohenKappa(skin, conf.level=0.95, weights="Fleiss-Cohen")
##     kappa    lwr.ci    upr.ci
## 0.6607229 0.4207465 0.9006993
```

The kappa coefficient has the value 0.3449, which indicates slight agreement between the dermatologists. The conclusion to reject the null hypothesis of no agreement is supported by the confidence interval of (0.2030, 0.4868), which suggests that the true kappa is greater than zero. The weighted kappa coefficient can be calculated by defining the `weights` argument. Its value is even larger (0.6607) than the unweighted kappa.
The Bowker's test for symmetry (reported by `mcnemar.test`) is not defined here (because of the zeros in the table).

11.4 Migraine - Statistics for a Stratified 2x2-Table

The data set Migraine contains hypothetical data for a clinical trial of migraine treatment. Subjects of both genders receive either a new drug therapy or a placebo. Their response to treatment is coded as 'Better' or 'Same'. The data are recorded as cell counts, and the number of subjects for each treatment and response combination is recorded in the variable Count. The following statements create a three-way table stratified by Gender, where Treatment forms the rows and Response forms the columns.

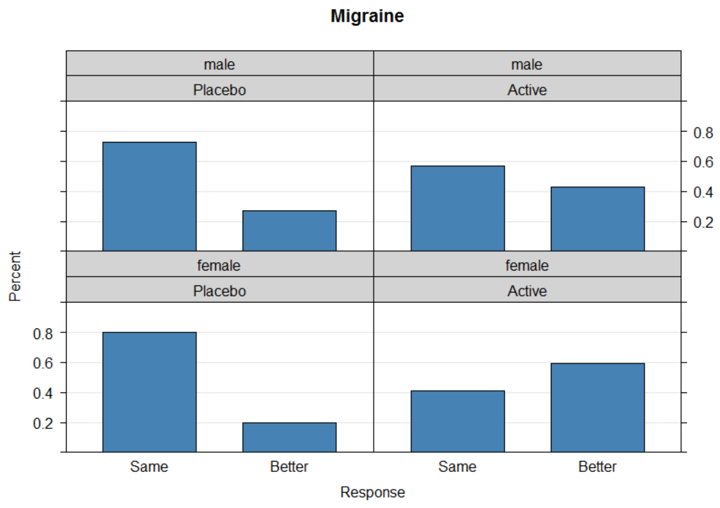
```
ParseSASDatalines("
  data d.Migraine;
  input Gender $ Treatment $ Response $ Count @@;
  datalines;
  female Active Better 16 female Active Same 11
  female Placebo Better 5 female Placebo Same 20
  male Active Better 12 male Active Same 16
  male Placebo Better 7 male Placebo Same 19
;
")
migraine <- xtabs(Count ~ Treatment + Response + Gender, d.Migraine)
```

It's always a good idea to have a plot of the situation:

```
d.frm <- as.data.frame(prop.table(migraine, c(2,3)))
d.frm$Treatment <- reorder.factor(d.frm$Treatment, new.order =
c("Placebo","Active"))
d.frm$Response <- reorder.factor(d.frm$Response, new.order = c("Same","Better"))

library(lattice)
barchart(Freq ~ Response | Treatment + Gender, data=d.frm,
         col="steelblue",
         panel = function(x, ...) {
           panel.grid(h=-1, v=0)
           panel.barchart(x, ...)
         },
         par.settings = list(strip.background=list(col="lightgrey"),
                             layout.heights=list(strip=1.45)),
         par.strip.text = list(col="black"),
         layout=c(2,2), cex.axis=2, ylim=c(0,1), xlab="Response", ylab="Percent",
         scales=list(tck=c(0.8,0.8), col="black", x=list(cex=1), y=list(cex=1)),
         main="Migraine")
```

This code yields:



The percentages are calculated so, that every panel has a total of 100%:

```
ptab <- prop.table(migraine, c(2,3))
ptab[] <- Format(ptab, digits=1, fmt="%")
ptab

## , , Treatment = Active

##          Gender
## Response female male
##   Better 59.3%  42.9%
##   Same   40.7%  57.1%
##
##, , Treatment = Placebo
##
##          Gender
## Response female male
##   Better 20.0%  26.9%
##   Same   80.0%  73.1%
```

Apparently the treatment seems to have a obvious effect. But the plot seems as well to indicate a gender effect, as the treatment is more pronounced for women than for men.

The function `mantelhaen.test` produces the Cochran-Mantel-Haenszel statistics. For this stratified 2x2 table, an estimate of the common odds ratio including its confidence interval is also displayed. (Note that the function expects the third dimension to be the strata, here gender.)

```
mantelhaen.test(migraine,  alternative = "two.sided", correct = FALSE)
```

mantelhaen.test

```
##      Mantel-Haenszel chi-squared test without continuity correction
##
## data:  migraine
## Mantel-Haenszel X-squared = 8.3052, df = 1, p-value = 0.003953
## alternative hypothesis: true common odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.445613 7.593375
## sample estimates:
## common odds ratio
##          3.313168
```

The significant p-value (0.004) indicates that the association between treatment and response remains strong after adjusting for gender.

****************************** CHECKME **************
For this stratified 2x2 table, estimates of the **common relative risk** are also displayed.
The CMH option also produces a table of relative risks, as shown in Output 28.6.2. Because this is a prospective study, the relative risk estimate assesses the effectiveness of the new drug; the "Cohort (Col1 Risk)" values are the appropriate estimates for the first column, or the risk of improvement. The probability of migraine improvement with the new drug is just over two times the probability of improvement with the placebo.

The Breslow-Day test for homogeneity of the odds ratios can be calculated with the eponymous function. It tests the null hypothesis that the odds ratios for the q strata are all equal.

```
BreslowDayTest(migraine)
```

BreslowDayTest

```
##      Breslow-Day Test on Homogeneity of the Odds Ratios
##
## data:  migraine
## X-squared = 1.4965, df = 1, p-value = 0.2212
```

The large p-value (0.2212) indicates no significant gender difference in the odds ratios.
Had the test for homogeneity of the odds ratios been statistically significant, a closer examination of each 2x2 table at each strata of the stratification variable would be required before making any further interpretations or conclusions.
Caution: Unlike the Cochran-Mantel-Haenszel statistics, the Breslow-Day test requires a large sample size within each stratum, and this limits its usefulness. In addition, the validity of the CMH tests does not depend on any assumption of homogeneity of the odds ratios; therefore, the Breslow-Day test should never be used as such an indicator of validity. (Refer to Breslow and Day (1994).)

Homogeneity could also be assessed using Woolf's test.

```
WoolfTest(migraine)
```

```
##      Woolf-test on Homogeneity of Odds Ratios (no 3-Way assoc.)
##
## data:  migraine
## X-squared = 1.4808, df = 1, p-value = 0.2236
```

Here the Woolf gives almost equivalent results to the BreslowDay test for consistency for the odds ratio.

The odds ratio for the treatment is

```
tab <- t(apply(migraine, c(1,2), sum))
OddsRatio(tab, conf.level = 0.95)
```

```
## odds ratio     lwr.ci     upr.ci
##   3.370370   1.461559   7.772108
```

Now, let's create logistic regression models on the raw data, first using just the two covariates Treatment and Gender:

```
r.glm <- glm(Response ~ Treatment + Gender, data=d.mig, family="binomial")
summary(r.glm)
```

```
## Call:
## glm(formula = Response ~ Treatment + Gender, family = "binomial",
##     data = d.mig)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2455  -1.0502  -0.6943   1.1108   1.7556
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.0602     0.3864  -2.744  0.00607 **
## TreatmentActive   1.2188     0.4271   2.853  0.00433 **
## Gendermale       -0.2398     0.4186  -0.573  0.56674
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 140.50  on 105  degrees of freedom
## Residual deviance: 131.55  on 103  degrees of freedom
## AIC: 137.55
##
## Number of Fisher Scoring iterations: 4
```

The estimates of the odds ratio are:

```
exp(coef(r.glm))
```

```
##     (Intercept) TreatmentActive      Gendermale
##       0.3463854       3.3830977       0.7867777
```

We learn that the treatment is significantly effective. Persons with treatment are 3.3 times as likely to report a positive response. The gender is not significant. By the way, also an interaction term would not become significant (not shown here).

12 References

http://cran.r-project.org/web/packages/vcdExtra/vignettes/vcd-tutorial.pdf

http://www.stattutorials.com/SAS/TUTORIAL-PROC-FREQ-1.htm

http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf

http://www.stat.ufl.edu/~presnell/Courses/sta4504-2011sp/Notes/icda-notes-3x2.pdf

Agresti, A. (2002) Categorical Data Analysis. John Wiley & Sons.

Dalgaard, P. (2008) Introductory Statistics with R (2. Aufl.), London, UK: Springer.