Package ltable 2.0.3. Part 1

Ocheredko Oleksandr

Content:

FUNCTIONALITY

1. Constructs tables of counts and proportions out of data sets.
2. Inserts table into Excel and Word documents using clipboard, into LaTeX, HTML, Markdown and reStructuredText documents by the knitr::kable agency.
3. Molds table into acceptable for log-linear modeling data.frame, co.
4. Performs log-linear modeling.
5. Performs power analysis.

- This version is coded in R language exclusively to support across-systems portability.
- Log-linear and power analyses are enhanced with ability to model risks (rates) and relative risks (standardized ratios). Modelling survival data with interval censoring is also supported.

Construction of tables of counts and proportions out of data sets

Use function table_f():

table_f(data, datavars, type = 1, digits = 2, extended = FALSE, MV = FALSE, cb = FALSE)

Examples:

```
data(sdata, package="ltable")
sdata
```

```
       a  b      c    d
1   TRUE NA   male    A
2     NA  1   male    B
3  FALSE  1   male    A
4   TRUE  1   male <NA>
5   TRUE  1   male    A
6   TRUE  2 female    B
7  FALSE  2 female    A
8  FALSE  2 female    B
9   TRUE  2 female    A
10 FALSE  2 female    B
11    NA NA   <NA> <NA>
12  TRUE  1   male    A
13 FALSE  1   male    B
14 FALSE  1   male    A
15  TRUE  1   male    B
16  TRUE  1   male    A
17  TRUE  2 female    B
18 FALSE  2 female    A
19 FALSE  2 female    B
20  TRUE  2 female    A
21 FALSE  2 female    B
22    NA NA   <NA> <NA>
```

```
lapply(sdata,class)
```

```
$a
[1] "logical"

$b
[1] "numeric"

$c
[1] "factor"

$d
[1] "character"
```

I built data.frame *sdata* with fields of different basic classes just for demonstration. No other meaning applies. Let's build a simple table across levels of *a*:

```
ltable::table_f(sdata, "a")
```

```
  a:FALSE a:TRUE Total, N
1       9     10       19
```

One might have interest in *NA* values for there may be quite informative pattern across levels or levels combinations. Use *MV=TRUE*. It's a part of data exploration:

```
ltable::table_f(sdata, "a", MV=TRUE, ext=TRUE)
```

```
  a:FALSE a:TRUE NA Total, N
1       9     10  3       22
```

Unrelated option *extended=TRUE* is used just to demonstrate that abundant args have no effect. If one wants to tabulate numerous factors it's important to arrange them properly in sequence of presentation delimited with comma ",". Sorted levels of all but last variable are rolled out vertically in indicated sequence, the last has its sorted levels spread by columns.

```
ltable::table_f(sdata, "b,c")
```

```
          b c:female c:male Total, N
1         1        0      9        9
2         2       10      0       10
sum Total, N       10      9       19
```

One can also obtain the table of frequencies by choosing arg *type* values { *2*, *3*, *4* } as shown below:

```
ltable::table_f(sdata, "a,c",
                type=2, digits=3)
```

```
          a c:female c:male Total, p
1     FALSE    0.667  0.333        1
2      TRUE      0.4    0.6        1
sum Total, p    0.534  0.466        1
```

```
ltable::table_f(sdata, "a,c",
                type=3, digits=2)
```

```
          a c:female c:male Total, p
1     FALSE      0.6   0.33     0.47
2      TRUE      0.4   0.67     0.53
sum Total, p       1      1        1
```

```
ltable::table_f(sdata, "a,c",
                type=4, digits=3)
```

```
          a c:female c:male Total, p
1     FALSE    0.316  0.158    0.474
2      TRUE    0.211  0.316    0.527
sum Total, p    0.527  0.474    1.001
```

One can include number of fields (variables):

```
options(width=40)
ltable::table_f(sdata, "a,b,c,d",
        type=2, digits=3)
```

```
          a        b        c   d:A
1     FALSE        1   female     0
2     FALSE        1     male 0.667
3     FALSE        2   female 0.333
4     FALSE        2     male     0
5      TRUE        1   female     0
6      TRUE        1     male  0.75
7      TRUE        2   female   0.5
8      TRUE        2     male     0
sum Total, p Total, p Total, p 0.562
     d:B Total, p
1      0        0
2  0.333        1
3  0.667        1
4      0        0
5      0        0
6   0.25        1
7    0.5        1
8      0        0
sum 0.438        1
```

arg value *extended=TRUE* adds margins of counts, applied only for proportions and frequencies, value is *FALSE* by default. In last two examples options(width) was used to accommodate tables:

```
options(width=40)
ltable::table_f(sdata, "b,c,a,d", type=2,
                digits=3, extended=TRUE)
```

```
          b        c        a   d:A
1         1   female    FALSE     0
2         1   female     TRUE     0
3         1     male    FALSE 0.667
4         1     male     TRUE  0.75
5         2   female    FALSE 0.333
6         2   female     TRUE   0.5
7         2     male    FALSE     0
8         2     male     TRUE     0
sum Total, p Total, p Total, p 0.562
    Total, N Total, N Total, N     9
```

```
      d:B Total, p Total, N
1       0         0         0
2       0         0         0
3   0.333         1         3
4    0.25         1         4
5   0.667         1         6
6     0.5         1         4
7       0         0         0
8       0         0         0
sum 0.438         1        17
        8        17        17
```

Transporting table into documents

One can paste table into clipboard by using arg *cb=TRUE*. To insert table into Word document one should first open Excel, choose left high corner of placement by mouse click and use copy and paste key combinations or click on the Copy and Paste icons (the clipboard), then open Word document, use Copy icon to place the table. $table\_f(sdata,\ "a,c",\ type = 2,\ digits = 3,\ cb = TRUE)$

Use knitr::kable() to import table to other available formats through .Rmd or other engines:

$t <- table\_f(sdata,\ "a,c",\ type = 2,\ digits = 3)$

$knitr :: kable(t)$

	a	c:female	c:male	Total, p
1	FALSE	0.667	0.333	1
2	TRUE	0.4	0.6	1
sum	Total, p	0.534	0.466	1

Transforming table into acceptable for log-linear modelling data.frame.

Use function tableToData():

$tableToData(\ tname,\ numerictype = "",\ orderedtype = ""\ )$

Example:

```
data(sdata, package="ltable")
stab<-ltable::table_f(sdata, "a,b,c")
```

```
sdat<-ltable::tableToData(stab,
                          numerictype ="b",
                          orderedtype="a,c")
sdat
```

```
      a b      c Counts
1 FALSE 1 female      0
2 FALSE 2 female      6
3  TRUE 1 female      0
4  TRUE 2 female      4
5 FALSE 1   male      3
6 FALSE 2   male      0
7  TRUE 1   male      5
8  TRUE 2   male      0
```

Arg *tname* is the name of table created by function *table_f()*. In both next args *numerictype* and *orderedtype* variable names separated by comma to be transformed to numeric or ordered factor classes. Variable "Counts" shouldn't be listed in both.

One can use functions tableToData() and tableToData() in sequence to transform **long to wide** data form, which is burdensome to do otherwise.

In previous example data.frame *sdata* is of *long* form while *sdat* is of *wide*.

Wide form of data has unique profiles as rows, one variable (usually last) indicating frequency of each profile's occurrence in the data. Wide form is commonly used to include salient points of data into presentation or publication.

Log-linear modeling

Use function MCLogLin():

$MCLogLin(formula,\ data,\ offset,\ contrasts = NULL,\ XLB = -100,\ XUB = 100,\ a = 0.1,\ b = 0.1,\ DIC = FALSE,\ pcov = FALSE,\ draw = 10000,\ burnin = 3000)$

Log-linear analysis features some advantages against $glm\{stats\}$, first of all due to stability of GSL IWLS algorithms that insures distinctly less biased covariances estimates, the pivot issue for implemented power analysis. In some instances hypothesis testing of higher order effects disagrees with that of *glm* on account of larger GSL estimated errors. Another though related enhancement is distinct better fit assessed by sum of squared differences between observed and expected counts.

Example

Let's begin with historical example of log-linear modeling with Tromboembolism Data. This case-control data first considered by Worcester, J. (1971). The data y[ijk] cross-classify thromboembolism and control patients (i=1 and 2 respectively) by two risk factors: oral contraceptive user (j =1 for user, j = 2 for non-user) and smoking (k=1 for smokers, k =2 for non-smokers). Test quantifies boosting effect of contraceptive on odds of thromboembolism using log-linear analysis. Reproduced grouped data frame with 8 rows of factors' levels combinations is given below. Factors are: smoking status (Yes, No), contraceptive usage (Yes, No), thromboembolism status (Trombol, Control).

```
data(tdata, package="ltable")
tdata
```

```
  smoker contraceptive   tromb Counts
1    Yes           Yes Trombol     14
2    Yes           Yes Control      2
3    Yes            No Trombol      7
4    Yes            No Control     22
5     No           Yes Trombol     12
6     No           Yes Control      8
7     No            No Trombol     25
8     No            No Control     84
```

Data has been used in subsequent model choice studies, such as Spiegelhalter and Smith (1982), Pettit and Young (1990), Congdon (2005).

Under the potentially informative priors used, the Bayes factor estimate was B_{21} = 23.8, quite strongly in favour of the smaller model with single interaction effect *contraceptive*thromboembolism* that was opted for consideration in example. The fact that the reduced model gives a close fit implies that the use of oral contraceptives indeed instigates the odds of thromboembolism, effect significance supported by classical and MCMC based log-linear estimates. Further inclusion of third order interaction indicated that the use of oral contraceptives particularly among those who smoke, is a risk for thromboembolism, but for smokers who do not take the pill there is no excess risk.

Let's check hypothesis by compare output of $MCLogLin\{ltable\}$ function with that of $glm\{stats\}$ function:

```
resglm<-glm(Counts~ smoker +contraceptive +
         tromb + contraceptive*tromb,
         family="poisson",
         data=tdata)
```

Results of MCLogLin {ltable} modeling

```
resMCLogLin<-ltable::MCLogLin(Counts~smoker +
      contraceptive +tromb +contraceptive*tromb,
      data=tdata)
```

```
Call:
ltable::MCLogLin(formula = Counts ~ smoker + contraceptive +
    tromb + contraceptive * tromb, data = tdata)


Coefficients:
                                 Estimate  Std.Error  |z-score|   Pr(>|z|)
(Intercept)                     4.415e+00  3.841e-01  1.150e+01  1.396e-30
smokerYes                      -9.614e-01  3.860e-01  2.491e+00  1.276e-02
contraceptiveYes               -2.426e+00  5.781e-01  4.197e+00  2.704e-05
trombTrombol                   -1.225e+00  5.073e-01  2.416e+00  1.571e-02
contraceptiveYes:trombTrombol   2.498e+00  7.871e-01  3.173e+00  1.508e-03
phi                             4.912e+00  7.612e-01  6.453e+00  1.099e-10


Model fit:
MCMC fitting
Samplers : Gibbs for expected counts, Slice for regr. coeff. and inv.var.par. phi
Language: R
Jacobian reciprocal condition number =  0.05064114
chisq/n =  0.02268587
Deviance=  0.226366
NULL Deviance=  9.243558
Log.likelihood=  -26.62244
AIC(1) =  63.24489
AIC(n) =  7.905611
BIC =  63.6421


Residuals report (Y denotes Counts):
Row Ovserved Y Predicted Y Raw Residual Pearson Residual Anscombe Residual
 1          14      12.432        1.568            0.237            0.894
 2           2       2.420       -0.420           -0.221           -0.561
 3           7       7.471       -0.471           -0.109           -0.363
 4          22      22.782       -0.782           -0.069           -0.320
 5          12      13.594       -1.594           -0.223           -0.893
 6           8       7.608        0.392            0.089            0.294
 7          25      24.421        0.579            0.048            0.225
 8          84      83.456        0.544            0.014            0.098
```

Output also conveys info:

Jacobian reciprocal condition number measures the inverse sensitivity of the solution to small perturbations in the input data. It tends to zero as J tends to singularity indicating solution instability.")

The value of ch-squared per number of counts (chisq/n) approximately 1 indicates a good fit.) If chisq/n » 1 the error estimates obtained from the covariance matrix will be too small and should be multiplied by square root of chisq/dof .

Poor fit will result from the use of an inappropriate model, and the scaled error estimates may then be outside the range of validity for Gaussian errors.

BEWARE: Poor fit jeopardizes the validity of power analysis.

Results of glm {stats} modelling

```
options(width=80)
summary(resglm)$coefficients
```

```
                              Estimate Std. Error   z value     Pr(>|z|)
(Intercept)                   4.364196  0.1069522 40.805108 0.000000e+00
smokerYes                    -1.053150  0.1731305 -6.082984 1.179660e-09
contraceptiveYes             -2.360854  0.3308080 -7.136628 9.564814e-13
trombTrombol                 -1.197703  0.2017027 -5.937964 2.885828e-09
contraceptiveYes:trombTrombol 2.153215  0.4232558  5.087265 3.632639e-07
```

```
cat("Predicted Y:", sprintf("%.2f", resglm$fitted.values), "\n
Deviance residuals:", sprintf("%.2f", summary(resglm)$deviance.resid))
```

```
Predicted Y: 6.72 2.59 8.28 27.41 19.28 7.41 23.72 78.59

Deviance residuals: 2.45 -0.38 -0.46 -1.07 -1.78 0.21 0.26 0.60
```

Juxtaposing two results we have the same conclusion on effects, specifically on hypothesized second order interaction term *contraceptive*tromb*, though differences are conspicuous on a part of error terms, higher order effect in particular. Checking with other data sets the regularity holds, that is higher order effects estimates feature larger errors against glm $\{stats\}$ counterparts due to handling overdispersion. The same rests with *chisq/n* statistic, predicted counts, and residuals (*deviance residuals* are larger in glm $\{stats\}$ than *Anscombe Residuals*). Repercussion on power analysis is about to be demonstrated (Part 2).