Notes from: Loglinear models for contingency tables

Chapter 7 Agresti (2007)

Saúl Sotomayor Leytón

February 2012

Setup

```
> library(MASS)
> library(vcd)
> options(contrasts=c("contr.treatment", "contr.poly"))
> options(width=70)
```

1 Concepts and commands

These models are most useful when used to analize associations among two or more categorical response variables. The reason is that these models treats all variables as responses and models the expected counts in a contingency table under a null hypothesis of independence, where the values are determined by the row and column totals: $\mu_{ij} = n\pi_{i+}\pi_{+j}$. By taking the natural logarithm (log), this relationship becomes linear, as equation 7.1 in Agresti (2007) shows:

$$log\mu_{ij} = \lambda + \lambda_i^X + \lambda_j^Y \tag{1}$$

Where λ_i^X and λ_j^Y represent the row and column effects. The fitted values that satisfy the model are, $\hat{\mu}_{ij} = n_{i+}n_{+j}/n$

These models represent a class of Generalized Linear Models, where cell counts are independent observations from some distribution, typically the *Poisson*. Note that this model "regards the observations to bel the cell counts rather than the individual classifications of the subjects"(Agresti 2007).

When working with more than two variables, different types of interactions can occur, between pairs of variables (as in the first equation, below) or also, between all the three variables (as in the second equation). Agresti (2007) mentions that the later refers to the saturated model, one that provides a perfect fit for the observed data.

$$log\mu_{ij} = \lambda + \lambda_i^X + \lambda_j^Y + \lambda_k^Z + \lambda_{ij}^{XY} + \lambda_{ik}^{XZ} + \lambda_{jk}^{YZ}$$
$$log\mu_{ij} = \lambda + \lambda_i^X + \lambda_j^Y + \lambda_k^Z + \lambda_{ij}^{XY} + \lambda_{ik}^{XZ} + \lambda_{jk}^{YZ} + \lambda_{ijk}^{XYZ}$$

Just like, in a three-way table of response variables X, Y and Z, there are different types of interactions, we can define several types of *independence* (Thompson 2007):

Mutual independence of all three variables results in *all* the interactions in the loglinear model equal zero. This means that all the joint cell probabilities equal the product of their marginal probabilities.

Joint independence of one variable Y and the combined classification of the other two (X and Z), results in a log linear model with only one non-zero interaction parameter, the one between X and Z.

Conditional independence of X and Y with Z, is when independence among X and Y holds in each partial table conditional on a given value of Z.

Something that is not mentioned in Agresti (2007) but in Thompson (2007), as well as. Agresti (2002) is that "loglinear models can be fitted using two methods for the *maximum likelihood estimation* (MLE), 1) Newton-Raphson or 2) Iterative Proportional Fitting (IPF). Both methods can be used in R through different packages/functions.

Agresti (2007, p. 205) mentions that "parameter interpretation is easier when we view one response as a function of the others, for instance in a Ix2 table the logit of the probability of Y=1 (success) equals:

$$log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = log\left(\frac{\mu_{i1}}{\mu_{i2}}\right) = log\mu_{i1} - log\mu_{i2} = \lambda + \lambda_i^X + \lambda_1^Y - (\lambda + \lambda_i^X + \lambda_2^Y) = \lambda_1^Y - \lambda_2^Y$$

As one can see, this logit does not depend on i; that is, the logit of Y does not depend on the level of X.

In this independence model, one λ_i^X and one λ_i^Y are redundant, thus most software sets the parameter for the las category equal to 0, while other set the sum of the parameters to zero. No matter what approach is taken, the difference between two main effect parameters of aparticular type is the same. Now, in R it is possible to do both ways. This is done with the following commands[1]:

```
#Sum to zero
options(contrasts=c("contr.sum", "contr.poly"))

#First category to zero
options(contrasts=c("contr.treatment", "contr.poly"))
```

Note that the second option sets the first category, instead of the last one, to zero, however, we can change the order of the factors to match the results from Agresti with one of the following commands:

```
tb$factor <- factor(tb$factor,levels=<correct order>)
tb$factor <- relevel(tb$factor,"<first level>")
fit.glm2 <- update(fit.glm, contrasts = list(alcohol = as.matrix(c(1, 0)), marijuana =
as.matrix(c(1, 0)), cigarette = as.matrix(c(1, 0))))
```

The first command (I think) works all the time, with different functions, however if one has many factors and they aren't in the correct order it may be tedious to do. The second command is described in Spector (2008, p. 70) and it is the easiest, since we only have to specify the first level. The drawback is that with some functions it does not work. Finally the third command, taken from Thompson (2007, p. 146), updates, in this case a glm, object and recodifies the levels of the factor with the as.matrix function. Altough I haven't tried it extensively, it seems to work well, but just as the first command it seems a bit tedious.

As stated previously, loglinear models can be fitted either with the Newton-Raphson or the IPF algorithm. Thompson (2007, p. 143–145) describes both, but I find the former easier since it uses the glm function.

```
> ## options(contrasts=c("contr.treatment", "contr.poly"))
> tb7.3 <- data.frame(expand.grid(cigarrette=c('yes', 'no'),
+                                 alcohol=c('yes', 'no'),  marijuana=c('yes', 'no')),
+                     count=c(911, 44, 3, 2, 538, 456, 43, 279))
> fit.glm <- glm(count ~ .^2, data = tb7.3, family = poisson)
```

[1]From previous test it seems that the second way (*i.e.* First category to zero) is the one set by default

In the previous commands count represents a vector of the observed cell values and $.^2$ represents all the two-way interactions.

In a similar way we fit the nested models to, in a following step, compare them and see which one is the best fit for the data. Note that in the following commands the function update is used on a glm object to simplify the input.

```
> fit.ACM <- glm(count ~ alcohol * cigarrette * marijuana, data = tb7.3,
+     family = poisson)
> fit.AC.AM.CM <- update(fit.ACM, . ~ . - alcohol:cigarrete:marijuana)
> fit.AM.CM <- update(fit.AC.AM.CM, . ~ . - alcohol:cigarrette)
> fit.AC.M <- update(fit.AC.AM.CM, . ~ . - alcohol:marijuana -
+     cigarrette:marijuana)
> fit.A.C.M <- update(fit.AC.M, . ~ . - alcohol:cigarrette)
```

The notation used for the R objects (*e.g.* fit.AM.CM) was done according to what is mentioned in Agresti (2007), namely, that the symbols AC or ACM represent the highest order interactions in the model. For a three-way table, the symbols ACM would represent an interaction among all its member, thus a perfect fit. Contrary, a model represented with the symbols A.C.M would represent one with no interaction among its members, *i.e.* a *mutual independence* model.

Once we have fitted all the models of interest we can create a data frame of fitted values in a way similar to what is described in Thompson (2007, p. 143).

```
> fitted.values <- data.frame(tb7.3[, -4], ACM = c(round(fitted(fit.ACM),
+                                            2)), AC.AM.CM. = c(round(fitted(fit.AC.AM
+                            AM.CM. = c(round(fitted(fit.AM.CM), 2)),
+                            AC.M. = c(round(fitted(fit.AC.M), 2)),
+                            A.C.M. = c(round(fitted(fit.A.C.M), 2)))
> fitted.values
```

```
  cigarrette alcohol marijuana ACM AC.AM.CM. AM.CM. AC.M. A.C.M.
1        yes     yes       yes 911       911    911   911    911
2         no     yes       yes  44        44     44    44     44
3        yes      no       yes   3         3      3     3      3
4         no      no       yes   2         2      2     2      2
5        yes     yes        no 538       538    538   538    538
6         no     yes        no 456       456    456   456    456
7        yes      no        no  43        43     43    43     43
8         no      no        no 279       279    279   279    279
```

Comparing with the command in Thompson (2007), the function aperm was not used, maybe because the data frame was constructed with glm objects instead of loglm. More important, the data frame was stored under the variable, fitted.values, which will be used to calculate conditional as well as marginal odds ratio using the functions xtabs and oddsratio; the later from the vcd package.

Model comparison, which test the significance of interactions is done with the anova function, which performs model comparison based on a χ^2 approximation.

```
> anova(fit.A.C.M, fit.AC.M, fit.AM.CM, fit.AC.AM.CM, fit.ACM,
+     test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: count ~ alcohol + cigarrette + marijuana + alcohol:cigarrette:marijuana
Model 2: count ~ alcohol + cigarrette + marijuana + alcohol:cigarrette +
    alcohol:cigarrette:marijuana
Model 3: count ~ alcohol + cigarrette + marijuana + alcohol:marijuana +
    cigarrette:marijuana + alcohol:cigarrette:marijuana
Model 4: count ~ alcohol + cigarrette + marijuana + alcohol:cigarrette +
    alcohol:marijuana + cigarrette:marijuana + alcohol:cigarrette:marijuana
Model 5: count ~ alcohol * cigarrette * marijuana
  Resid. Df  Resid. Dev Df    Deviance Pr(>Chi)
1         0 -2.4758e-13
2         0 -4.2411e-14  0 -2.0517e-13
3         0 -1.9806e-13  0  1.5565e-13
4         0 -1.8652e-13  0 -1.1546e-14
5         0 -1.8652e-13  0  0.0000e+00
```

Note that for easy interpretation the simpler models where introduced first. Still, interpreting the results may be a bit tricky. The Deviance column shows the result of comparing the residual deviance from the previous line with the one from the current; this value provides evidence for the null hypothesis (H_0) that the parameters in the more complex model, that aren't in the current one (in this case in the previous line) equal zero. As we can see, the only model for wich the deviance value yields a non-significant result is the one that allows conditional independence (*i.e.* interactions among all pairs of variables, fit.AC.AM.CM).

Having chosen a particular model, we can now see the strenght of association, among pairs of variables, by comparing *conditional* and *marginal* odds ratios. To calculate them Thompson (2007, p. 143-144) describes a method that, first, defines a function to calculate odds ratios and applies it to the fitted values for the model chosen (fit.AC.AM.CM), however there is a simpler way with the functions xtabs and oddsratio.
But first I have constructed a new data frame of fitted vales without rounding up the values.

```
> fitted.values2 <- data.frame(tb7.3[, -4], ACM = c(fitted(fit.ACM)),
+     AC.AM.CM. = c(fitted(fit.AC.AM.CM)), AM.CM. = c(fitted(fit.AM.CM)),
+     AC.M. = c(fitted(fit.AC.M)), A.C.M. = c(fitted(fit.A.C.M)))
> oddsratio(xtabs(AC.AM.CM. ~ cigarrette + marijuana + alcohol,
+     data = fitted.values2), log = FALSE)
```

```
 odds ratios for cigarrette and marijuana by alcohol 

      yes        no 
17.548834  9.732558 
```

```
> oddsratio(xtabs(AC.AM.CM. ~ cigarrette + marijuana, data = fitted.values2),
+     log = FALSE)
```

```
 odds ratios for cigarrette and marijuana 

[1] 25.1362
```

This was done to ensure that the odds ratios will be the same across the levels of the third variable (see the results from the first command), which is something that we expect given that we have chosen a

conditional independence model.
The interpretation of the results is as following, for the first one, conditional on the use of alcohol, the odds of a person who smokes cigarrettes to use marijuana is 17 times the odds of a person who doesn't smoke cigarettes to use marijuana. For the second one, *i.e.* the marginal association, the odds of a person who smokes cigarrettes to use marijuana is 25 times the odds of a person who doesn't smoke to use marijuana.

Also helpful when choosing a particular model is to look at *cell residuals*. As stated by Agresti (2007, p. 213), "sometimes they indicate that only certain cell display a lack of fit in an otherwise good-fitting model". As it is explained in previous chapters, there are more than one type of residuals (*e.g.* Pearson residuals or Standardized Pearson Residuals), the later being the most useful, since we know that values higher than 2 (when working with a moderate number of cells) or 3 (when working with many cells) indicate lack of fit. In R we calculate standardized pearson residuals with the following command (for the AC.AM.CM. and AM.CM models)

```
> res.AC.AM.CM <- resid(fit.AC.AM.CM, type = "pearson")/sqrt(1 -
+     lm.influence(fit.AC.AM.CM)$hat)
> res.AM.CM <- resid(fit.AM.CM, type = "pearson")/
+     sqrt(1 - lm.influence(fit.AM.CM)$hat)
```

More over, we can easily attach these residuals to create a table of fitted values together with their residuals

```
> data.frame(fitted.values[, c(1:3)], AC.AM.CM = fitted.values[,
+     5], Residuals = round(res.AC.AM.CM, 3), AM.CM = fitted.values[,
+     6], Residuals = round(res.AM.CM, 3))
```

```
  cigarrette alcohol marijuana AC.AM.CM Residuals AM.CM Residuals.1
1        yes     yes       yes      911       Inf   911         Inf
2         no     yes       yes       44      -Inf    44         Inf
3        yes      no       yes        3       Inf     3        -Inf
4         no      no       yes        2      -Inf     2        -Inf
5        yes     yes        no      538       Inf   538         Inf
6         no     yes        no      456       Inf   456         Inf
7        yes      no        no       43       Inf    43        -Inf
8         no      no        no      279       Inf   279         Inf
```

Note that residuals for the marginal independece model are all below 1 (in absolute value), while those corresponding to the simpler model AC.CM have values higher than 3. Interesting, Agresti (2007) note that χ^2 relates to the two nonredundant residuals by $\chi^2 = (-3.69)^2 + (-12.80)^2 = 177.6$.

When we work with more than three variables, the number of possible interactions grows considerably and it may be difficult not only to fit all the possible models for further comparison but to interpret any three-way interaction. To ilustrate this Agresti (2007) as well as Thompson (2007) uses the data of car accidents described by four variables: gender, location, use of seat belt and injury, all with two possible values. Interesting, Agresti (2007) only addresses the latter problem (*i.e.* the interpretation of a three-way interaction) but doesn't disscuss how he end-up with the final model, as Thompson (2007) and Agresti (2002) do.
First we construct the table of observed counts.

```
> tb.accident <- data.frame(expand.grid(Seat = c(0, 1), Location = c("urban",
+     "rural"), Gender = c("female", "male"), Injury = c(0, 1)), Count = c(7287,
+     11587, 3246, 6134, 10381, 10969, 6123, 6693, 996, 759, 973,
+     757, 812, 380, 1084, 513))
```

Then we fit a model with all the three-way interactions, which will serve as a base from where we will start removing all the "non-significant" interactions.

```
> fit.3w <- glm(Count ~ .^3, data = tb.accident, family = poisson)
> fit.accident <- stepAIC(fit.3w, scope = list(lower = Count ~
+     Seat + Gender + Location + Injury), direction = "backward")
```

```
Start:  AIC=184.78
Count ~ (Seat + Location + Gender + Injury)^3

                         Df Deviance    AIC
- Seat:Gender:Injury      1   1.3670 182.83
<none>                        1.3253 184.78
- Location:Gender:Injury  1   3.5624 185.02
- Seat:Location:Injury    1   4.3720 185.83
- Seat:Location:Gender    1  16.1391 197.60

Step:  AIC=182.83
Count ~ Seat + Location + Gender + Injury + Seat:Location + Seat:Gender +
    Seat:Injury + Location:Gender + Location:Injury + Gender:Injury +
    Seat:Location:Gender + Seat:Location:Injury + Location:Gender:Injury

                         Df Deviance    AIC
<none>                        1.3670 182.83
- Location:Gender:Injury  1   3.5914 183.05
- Seat:Location:Injury    1   4.4909 183.95
- Seat:Location:Gender    1  16.5610 196.02
```

Which results in a model with 3 out of four of the possible 3 way interactions (*i.e.* only the interaction Seat:Gender:Injury was removed).

```
> summary(fit.accident)
```

```
Call:
glm(formula = Count ~ Seat + Location + Gender + Injury + Seat:Location +
    Seat:Gender + Seat:Injury + Location:Gender + Location:Injury +
    Gender:Injury + Seat:Location:Gender + Seat:Location:Injury +
    Location:Gender:Injury, family = poisson, data = tb.accident)

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                      8.89220    0.01160 766.542  < 2e-16
Seat                             0.46648    0.01470  31.738  < 2e-16
Locationrural                   -0.80445    0.02064 -38.968  < 2e-16
Gendermale                       0.35669    0.01501  23.762  < 2e-16
Injury                          -1.97646    0.03057 -64.654  < 2e-16
Seat:Locationrural               0.16601    0.02549   6.512 7.40e-11
```

```
Seat:Gendermale                  -0.41363    0.01963 -21.072  < 2e-16
Seat:Injury                      -0.76616    0.03946 -19.416  < 2e-16
Locationrural:Gendermale          0.27400    0.02568  10.671  < 2e-16
Locationrural:Injury              0.76046    0.04448  17.096  < 2e-16
Gendermale:Injury                -0.58782    0.03923 -14.984  < 2e-16
Seat:Locationrural:Gendermale    -0.12723    0.03266  -3.895 9.81e-05
Seat:Locationrural:Injury        -0.09773    0.05529  -1.768   0.0771
Locationrural:Gendermale:Injury   0.08148    0.05466   1.491   0.1360

(Intercept)                      ***
Seat                             ***
Locationrural                    ***
Gendermale                       ***
Injury                           ***
Seat:Locationrural               ***
Seat:Gendermale                  ***
Seat:Injury                      ***
Locationrural:Gendermale         ***
Locationrural:Injury             ***
Gendermale:Injury                ***
Seat:Locationrural:Gendermale    ***
Seat:Locationrural:Injury        .
Locationrural:Gendermale:Injury
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 61709.521  on 15  degrees of freedom
Residual deviance:     1.367  on  2  degrees of freedom
AIC: 182.83

Number of Fisher Scoring iterations: 3
```

This is not the same model that is described in Agresti (2007, p. 215–217), however if we look at the p-values for all the remaining three-way interactions we will note that the only one that has a value lower than 0.05 is Seat:Location:Gender. Thus if we base on this values to further eliminate other interactions we will end with a model similar to the one described in Agresti (2007). These differences may arise from the way the selection was performed, in this case based on the AIC value.

Agresti (2007) explains that the three-way interaction implies that the odds ratio of two of the variables that composes this interaction varies across the levels of the remaining variable, as it is show in table 7.11. Now, in R we can calculate these odd ratios from a table of fitted values as it was, previously, described. First we fit the models of interest, which are the model with all the two-way interactions and the one with the interaction Seat:Location:Gender plus all the lower order terms.

```
> fit4.accident <- glm(Count ~ .^2, family = poisson, data = tb.accident)
> fit3.accident <- glm(Count ~ Seat + Location + Gender + Injury +
+     Seat:Injury + Seat:Gender + Location:Injury + Gender:Injury +
+     Location:Gender + Seat:Location + Seat:Location:Gender, family = poisson,
+     data = tb.accident)
```

Then we can create a table of fitted values. Note that I have included the values from the model selected by the fitAIC function.

```
> (tb.fit.accid <- cbind(tb.accident, GI.GL.GS.IL.IS.LS = fitted(fit4.accident),
+     GLS.GI.IL.IS = fitted(fit3.accident), GLS.GLI.SLI = fitted(fit.accident)))
```

```
   Seat Location Gender Injury Count GI.GL.GS.IL.IS.LS GLS.GI.IL.IS
1     0    urban female      0  7287         7166.3688    7273.2140
2     1    urban female      0 11587        11748.3087   11632.6221
3     0    rural female      0  3246         3353.8294    3254.6618
4     1    rural female      0  6134         5985.4930    6093.5021
5     0    urban   male      0 10381        10471.4955   10358.9317
6     1    urban   male      0 10969        10837.8269   10959.2323
7     0    rural   male      0  6123         6045.3062    6150.1925
8     1    rural   male      0  6693         6811.3714    6697.6436
9     0    urban female      1   996          993.0169    1009.7860
10    1    urban female      1   759          721.3055     713.3779
11    0    rural female      1   973          988.7848     964.3382
12    1    rural female      1   757          781.8927     797.4979
13    0    urban   male      1   812          845.1187     834.0683
14    1    urban   male      1   380          387.5588     389.7677
15    0    rural   male      1  1084         1038.0796    1056.8075
16    1    rural   male      1   513          518.2429     508.3564
   GLS.GLI.SLI
1    7274.9904
2   11599.0096
3    3254.3635
4    6125.6365
5   10393.0096
6   10956.9904
7    6114.6365
8    6701.3635
9    1008.0096
10    746.9904
11    964.6365
12    765.3635
13    799.9904
14    392.0096
15   1092.3635
16    504.6365
```

Note that the more complex model differs slightly with that with only one triple interaction and that with only two-way interactions (see below). Now we can calculate the odds ratio for the different combinations of variables, just as it is done in table 7.11 in Agresti (2007).

```
> #Odds ratio LS Agresti's results 1.17 and 1.03
> oddsratio(xtabs(GLS.GLI.SLI ~ Location + Seat + Gender, data = tb.fit.accid),
+     log = FALSE)
```

```
 odds ratios for Location and Seat by Gender

   female      male 
1.0958071 0.9861174 
```

```
> # Odds ratio LS Agresti's results 1.09 and 1.09
> oddsratio(xtabs(GI.GL.GS.IL.IS.LS ~ Location + Seat + Gender,
+     data = tb.fit.accid), log = FALSE)
```

```
 odds ratios for Location and Seat by Gender

  female     male 
1.019704 1.043171 
```

It's curious that, despite the fitted values obtained are the same as those from Agresti (2007, table 7.9, p. 216) the calculated odds ratio are different. I did a separate calculation but the results were the same as the above.

Update Revising the data I've found that the results presented in Agresti (2007) are obtained when all the variables are included in the xtabs function; for the previous examples:

```
> #Odds ratio LS Agresti's results 1.17 and 1.03
> oddsratio(xtabs(GLS.GI.IL.IS ~ Location + Seat + Gender + Injury,
+     data = tb.fit.accid), log = FALSE)
```

```
 odds ratios for Location and Seat by Gender, Injury

        Injury
Gender          0        1
  female 1.170604 1.170604
  male   1.029362 1.029362
```

```
> # Odds ratio LS Agresti's results 1.09 and 1.09
> oddsratio(xtabs(GI.GL.GS.IL.IS.LS ~ Location + Seat + Gender +
+     Injury, data = tb.fit.accid), log = FALSE)
```

```
 odds ratios for Location and Seat by Gender, Injury

        Injury
Gender          0        1
  female 1.088636 1.088636
  male   1.088636 1.088636
```

Note that, for the first command, the row values repeat in each column, which represent the independence of the variables.The same happens for the second command but also for the row values.

In section 7.2.8 Agresti (2007) discusses the differences among the chosen models in fitting the observed data. We have seen that R selected a complex model with three out of four of the triple interaction terms

yet its fitted values differ slightly from a simpler model with no three-way interaction terms. Agresti (2007) ascribes this difference to the high sample size; he mentions that "we are more likely to detect an effect as the sample size increases, eventhought those effects may be weak and unimportant".
As an alternative (or complement) to the model selection, Agresti (2007) describes a measure that does not take into account the sample size, the *dissimilatity index*.

$$D = \sum |n_i - \hat{\mu}_i|/2n = \sum |p_i - \hat{\pi}_i|/2$$

"This index takes values between 0 and 1, where smaller values represent a better fit. It represents the prportion of sample cases that must move to different cells for the model to achieve a perfect fit"(Agresti 2007). In R this is easily calculated with

```
> #Dissimilary indes for the simpler model
> sum(abs(tb.fit.accid$Count - tb.fit.accid$GI.GL.GS.IL.IS.LS))/(2 *
+     sum(tb.fit.accid$Count))
```

```
[1] 0.008219103
```

```
> # Dissimilary index for the more complex model
> sum(abs(tb.fit.accid$Count - tb.fit.accid$GLS.GLI.SLI))/(2 *
+     sum(tb.fit.accid$Count))
```

```
[1] 0.001186309
```

Comparing the two indexes we can see that the differences are small, moving less than 1% of the data yields a perfect fit. In this sense, Agresti (2007) mentions that the small D values for the simpler model suggest that, despite the large value of G^2, in practical terms, the model provides a decen fit.

In the next section (7.3) Agresti (2007) discusses the similarities between loglinear models and logistic regresion models. Though, both models assume different distributions they result in the same conditional odds ratio estimations as ilustrated with a three-way table where variable Y is binary:

$$\begin{aligned}
logitP(Y=1) = &\quad log\left[\frac{P(Y=1)}{1-P(Y=1)}\right] = log\left[\frac{P(Y=1|X=i,Z=k)}{P(Y=2|X=i,Z=k)}\right] \\
= &\quad log\left(\frac{\mu_{i1k}}{\mu_{i2k}}\right) = log(\mu_{i1k}) - log(\mu_{i2k}) \\
= &\quad (\lambda + \lambda_i^X + \lambda_1^Y + \lambda_k^Z + \lambda_{i1}^{XY} + \lambda_{ik}^{XZ} + \lambda_{1k}^{YZ}) - (\lambda + \lambda_i^X + \lambda_2^Y + \lambda_k^Z + \lambda_{i2}^{XY} + \lambda_{ik}^{XZ} + \lambda_{2k}^{YZ}) \\
= &\quad (\lambda_1^Y - \lambda_2^Y) + (\lambda_{i1}^{XY} - \lambda_{i2}^{XY}) + (\lambda_{1k}^{YZ} - \lambda_{2k}^{YZ})
\end{aligned}$$

Which is equivalent to the logit model:

$$logitP(Y=1) = \alpha + \beta_i^X + \beta_k^Z$$

All this process can be summarized as following, "in logit calculation, all the terms in the loglinear model not having the response variable index in the subscript cancel".

Interesting, in section 7.3.3, Agresti (2007) notes that the logit model resulting from the loglinear model (XY,XZ,YZ) is the same as the one resulting from the model (XY,YZ). This is because logistic models doesn't describe relationships among response variables, so it assumes nothing about their association structure.

Regarding these disctintion among response variables and explanatory ones, the author mentions that when a clear disctintion happens or when some marginal totals are fixed by design, the model should contain the term for that margin; for example in the car accident example if the gender and location factors are treated as a explanatory variables, then the term GL must be in the model, thus it should be at least as complex as (GL,S,I).

Next, in section 7.4, the author describes conditions to collapse three-way or higher order tables. These conditions assure that the marginal odds ratios and the conditional odds ratios would be the same. In order to know which variable can be collapsed Agresti (2007) describes *independence graphs*. As its name suggest, these are graphical representations of loglinear models where each variable is represented by a letter and interactions among variables are represented by a line between them. Variables that does not have a connecting line represent conditional independence.
Having described independence graphs we can define the condition for collapsibility as: “For three-way tables, XY marginal and conditional odds ratios are identical if either Z and X are conditionally independen or if Z and Y are conditionally independent” this is equivalent to the following independece graphs:

X —— Y —— Z or Y —— X —— Z

For higher order tables the same basic graph can be used (see Agresti 2007, sections 7.4.4 and 7.5.5) where letters represent sets of variables instead of individual variables.
The conditions for collapsibility explain why in the above example of car accidents different values for the odds ratios were obtained, namely this was because the initial model collapsed the data over the injury levels, when infact this wasn’t possible. Later as it was shown, by including this variable the values matched those from Agresti (2007).

Agresti (2007) ends the chapter describing *Linear-by-Linear* association models, which are used to model ordinal responses. These are very similar to independence models, except that they have an extra estimate for the correlation between row and column scores.

$$log \mu_{ij} = \lambda + \lambda_i^X + \lambda_j^Y + \beta u_i v_j \tag{2}$$

where u_i and v_j represents row and column scores, respectively.

The author notes that β specifies the direction and strenght of association. It also allow us to calculate the odds ratio for any $2x2$ table resulting from the selection of two rows (a and c) and columns (b and d).

$$\frac{\mu_{ab}\mu_{cd}}{\mu_{ad}\mu_{cb}} = exp[\beta(u_c - u_a)(u_d - u_b)] \tag{3}$$

By looking at Thompson (2007, p.156) we noticed that fitting these models in R is easy; we just have to create a data frame with the observed counts for each row and column combination as well as columns for the factors and their codes. For the example described in the book, as it’s noted in Thompson (2007), it’s necessary to do a few additional steps to create the data frame

```
> tb7.15 <- read.table("supp_data/tb7-15", header = TRUE)
> tb7.15$u1 <- tb7.15$prem
> tb7.15$v1 <- tb7.15$birth
> tb7.15$birth <- factor(tb7.15$birth, levels = 4:1)
> tb7.15$prem <- factor(tb7.15$prem, levels = 4:1)
```

After that we can fit the model.

```
> (fit.lbl <- glm(count ~ prem + birth + u1:v1, family = poisson,
+     data = tb7.15))
```

```
Call:  glm(formula = count ~ prem + birth + u1:v1, family = poisson,
    data = tb7.15)

Coefficients:
(Intercept)        prem3        prem2        prem1       birth3
    0.47349     -0.01634      0.10772      1.75369      1.15514
     birth2       birth1        u1:v1
    1.41556      1.87966      0.28584

Degrees of Freedom: 15 Total (i.e. Null);  8 Residual
Null Deviance:           431.1
Residual Deviance: 11.53        AIC: 118.2
```

With the data stored in the variable, then, we calculate the value of β with its corresponding 95% Wald confidence interval

```
> exp(coef(fit.lbl)["u1:v1"])
```

```
   u1:v1
1.330873
```

```
> exp(summary(fit.lbl)$coef["u1:v1", 1] + 1.96 * c(-1, 1) * summary(fit.lbl)$coef["u1:v1",
+     2])
```

```
[1] 1.259216 1.406609
```

Then, for the fitted values, first we append them to the data frame (*i.e.* tb7.15), next we extract these values with the xtabs function. Note that because we inverted the factor values (in order to match the results presented in the book) we need to perform additional steps to present the fitted values in the correct order.

```
> tb7.15$fitted <- fitted(fit.lbl)
> matrix(rev(xtabs(fitted ~ prem + birth, data = tb7.15)), nrow = 4,
+     dimnames = list(PreMar = c("Always wrong", "Almost always wrong",
+         "Wrong sometimes", "Not wrong"), TeenBirth = c("S. Disagree",
+         "Disagree", "Agree", "S. Agree")))
```

```
                     TeenBirth
PreMar                S. Disagree Disagree     Agree  S. Agree
  Always wrong           80.85658 67.65406  69.39574  29.09363
  Almost always wrong    20.75004 23.10650  31.54350  17.59996
  Wrong sometimes        24.39370 36.15178  65.68137  48.77315
  Not wrong              32.99969 65.08766 157.37940 155.53326
```

In the same way we can create a table of residuals, namely, first we add a column of residuals and then we create the table.

```
> tb7.15$residuals <- round(resid(fit.lbl, type = "pearson")/sqrt(1 -
+     lm.influence(fit.lbl)$hat), 2)
> matrix(rev(xtabs(residuals ~ prem + birth, data = tb7.15)), nrow = 4,
+     dimnames = list(PreMar = c("Always wrong", "Almost always wrong",
+         "Wrong sometimes", "Not wrong"), TeenBirth = c("S. Disagree",
+         "Disagree", "Agree", "S. Agree")))
```

```
                     TeenBirth
PreMar                S. Disagree Disagree Agree S. Agree
  Always wrong               0.03     0.06 -1.65     2.42
  Almost always wrong        0.87     0.74 -0.59    -1.02
  Wrong sometimes           -1.56     1.01  1.46    -1.28
  Not wrong                  0.81    -1.47  0.52     0.27
```

Finally, we can test the null hypothesis that $\beta = 0$. For this end, we have to fit a model without that term and then compare that with the previous with the anova function.

```
> fit.null <- update(fit.lbl, ~. - u1:v1)
> anova(fit.null, fit.lbl, test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: count ~ prem + birth
Model 2: count ~ prem + birth + u1:v1
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1         9    127.653
2         8     11.534  1   116.12 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The small P-value indicate a strong correlation.

To end these notes it is important to mention that Agresti's more advanced book Agresti (2002) describes other association models, thus it is worth looking at them after reading this book.

References

Agresti, Alan (2002). *Categorical Data Analysis*. 2nd ed. New York: Wiley-Interscience. ISBN: 978-0-471-36093-3.

— (2007). *An Introduction to Categorical Data Analysis*. 2nd ed. Hoboken NJ: Wiley-Interscience. ISBN: 978-0-471-22618-5.

Spector, Phil (2008). *Data Manipulation with R*. New York: Springer Verlag. ISBN: 978-0-387-74730-9.

Thompson, Laura A. (2007). *S-plus (and R) Manual to Accompany Agresti's "Categorical Data Analysis" (2002)*.