Notes from: Modeling correlated, clustered responses

Chapter 9 Agresti (2007)
Saúl Sotomayor Leytón
April 2012

Setup

```
> library(MASS)
> library(gee)
> library(geepack)
> options(width=70)
```

1 Introduction

Chapter 9 and 10 presents models for handling correlated, clustered data using different approaches. But first, when does correlated data appear? Normally they appear when we take a measure from a subject many times at different intervals such as in longitudinal studies. In this type of data we expect that the data from one measure to the next be alike (*i.e.* correlated) and therefore we need a model that takes that correlation into account. Other examples might include a study about the toxicity of a compound on the development of fetuses; in this study we expect that those fetuses from a same litter (from the same mother) have a more similar response than fetuses from a different litter.
As Agresti notes, a *cluster* refers to a matched set of observations; in a longitudinal study a cluster might be all the observations for a subject during the study, while in a toxicology study, a cluster might be all the fetuses from a particular litter. The author warns that "studies that do not take into account the correlation among measures may estimate the parameters well but the standard errors can be badly biased". Finally it's important to distinguish between *marginal* and *conditional* models. As its name suggest, the former focuses on the marginal distributions and it's used to study the odds for a particular response at the level of the population. On the other hand, the latter model focuses on the individual response, that is it calculates the odds of a particular response, conditional on the subject studied. Although, as it's discussed in Agresti (chapter 10 2007), the conclusions drawn from both models do not differ substantially, they have some features that made more useful under certain circumstances while not on another.

2 Marginal models versus conditional models

Subsections 9.1.1 describe some of the basic notation used in this and the following chapter for models for correlated data, for example, Agresti uses the letter **T** for the number of observations, thus for a binary response the T success probabilities, $P(Y_1 = 1), P(Y_2 = 1), \ldots, P(Y_T = 1)$ are the marginal probabilities of a T-dimensional contingency table that cross classifies the T observations.

In section 9.1.2 the author describes a example of a longitudinal study about the effects of a new drug, compared to an old one, in treating mental depression. Subjects were classified in two groups based on an initial diagnosis and then given one of the drugs. Their responses were evaluated at weeks 1, 2 and 4 indicating whether they have a normal response or not. Table 9.1 shows the marginal counts for each of the eight possible responses throughout the study (for example a NNN repose is interpreted as a normal response in all the evaluations). But more informative is table 9.2 that shows the proportions of normal responses for all the evaluations and for all the groups. By looking at this table we can look if there is a trend (an increase or decrease in the proportions), the rate of that trend (how fast the proportions change),

all of this across the different groups. This information will help us decide which model would fit the data well, for example Agresti first describes a main effects model,

$$logit[P(Y_t = 1)] = \alpha + \beta_1 s + \beta_2 d + \beta_3 t$$

where s stands for the initial diagnosis (mild or severe), d for the drug used (standard or new) and t for the time[1] of the evaluation. Note that under this model, the time effect is the same for both drugs, something that table 9.2 doesn't suggest. To reflect this one can include an interaction term,

$$logit[P(Y_t = 1)] = \alpha + \beta_1 s + \beta_2 d + \beta_3 t + \beta_4 (dxt)$$

In this model β_3 describes the time effect for the standard drug ($d = 0$) and $\beta_3 + \beta_4$ describes the time effect for the new drug ($d = 1$).

Section 9.1.3 contrast the previous model with the equivalent one used in conditional modeling,

$$logit[P(Y_{it} = 1)] = \alpha_i + \beta_1 s + \beta_2 d + \beta_3 t + \beta_4 (dxt)$$

Note that in this case the logit refers to subject i at evaluation time t and that there is a subject-specific intercept, α_i, that indicates that the logit will vary from subject to subject, that is, this type of model accounts for heterogeneity among the data. Chapter 10 discussed further this and other differences.

3 Marginal modeling: the Generalized Estimating Equations (GEE) approach

This section discusses some of the problems of fitting marginal models through maximum likelihood. Basically this is because ML focuses on the joint distribution of the clustered responses, not on the marginal. However when there are a few explanatory variables it's possible to fit the model through ML with some specialized software[2]. As an alternative one can use *quasi-likelihood* methods. For clustered data one method called Generalizing Estimating Equations (GEE) only links each marginal mean to a linear predictor and provides a guess for the variance–covariance structure $(Y_1, \ldots, Y_T)$. Unlike Generalized Linear Models (GLM), GEE do not assume a probability distribution for $E(Y) = \mu$ and $VAR(Y)$ only a relationship among them. This permits any departure from the assumptions, such as *overdispersion*. To do this, the quasi-likelihood approach assumes that the variance is a multiple of a constant ϕ that is inferred from the data.

Agresti says that after a model for each response at time t (Y_t) has been specified we must:

1. Assume a particular distribution for each Y_t which will determine how the $VAR(Y_t)$ depends on $E(Y_t) = \mu$. Then we need to,
2. Make and *educated guess* for the correlation structure among Y_t, that is the *working correlation matrix*.

The author mentions that point 2 is important, but isn't crucial to make the "right" initial assumption because the model is updated with the information in the data and even if a " wrong" guess was made, we end up with appropriate standard errors.
The possibilities for the working correlation matrix that Agresti describes are:

[1] Agresti (2007, p. 278) notes that "when the time metric reflects cumulative drug dosage, a logit scale often has an approximate linear effect for the logarithm of time", however latter he says that using the week numbers has the same effect. Obviously the latter is simple for the data input, nevertheless one should keep the idea present.

[2] For R there are set of functions developed by Joseph B. Lang from the university of Iowa (http://www.divms.uiowa.edu/~jblang/mph.fitting/index.htm)

Exchangeable structure that implies $\rho = Corr(Y_s, Y_t)$ as identical but unknown for all pairs s and t.

Autoregressive , that implies $\rho^{t-s} = Corr(Y_s, Y_t)$. This implies that the observations farther apart in time are less correlated.

Independence , that assumes $\rho = Corr(Y_s, Y_t) = 0$, which is the same as saying that observations among a cluster are independent.

Unstructured , permits the correlation matrix $Corr(Y_s, Y_t)$ to differ for each pair.

Among these, the author recommends, at least as starting point to choose the Exchangeable structure.

3.1 GEE fitting in R

For the metal depression example, Thompson describes the fitting of GEE in R. Although she imports the data directly from a text file (which is available in Agresti's website) it's important to know how to construct the data frame, which is a long data frame (see Spector 2008, section 9.4). To this end we can start by importing table 9.1, whose values are going to help us constructing the long table.

```
> tb9.1 <- data.frame(cbind(expand.grid(treat = c(0, 1), diagnose = c(0,
+     1)), matrix(c(16, 13, 9, 3, 14, 4, 15, 6, 31, 0, 6, 0, 22,
+     2, 9, 0, 2, 2, 8, 9, 9, 15, 27, 28, 7, 2, 5, 2, 31, 5, 32,
+     6), ncol = 8, byrow = TRUE)))
> colnames(tb9.1)[3:10] <- c("NNN", "NNA", "NAN", "NAA", "ANN",
+     "ANA", "AAN", "AAA")
> tb9.1
```

```
  treat diagnose NNN NNA NAN NAA ANN ANA AAN AAA
1     0        0  16  13   9   3  14   4  15   6
2     1        0  31   0   6   0  22   2   9   0
3     0        1   2   2   8   9   9  15  27  28
4     1        1   7   2   5   2  31   5  32   6
```

Then we get the row totals,

```
> margin.table(as.matrix(tb9.1[, 3:10]), 1) * 3
```

```
[1] 240 210 300 270
```

The values were multiplied by 3 because that's the number of measurements taken. Also note that `tb9.1` was converted to a matrix, selecting only the columns with numeric values (`3:10`). It would had been easier to calculate the row totals instead, however with this method there are less risk of confusion.

Now we can construct the data frame which will be used for the model fitting.

```
> tb9.1 <- data.frame(case = rep(1:340, each = 3), time = c(0,
+     1, 2), treat = rep(c(0, 1, 0, 1), c(240, 210, 300, 270)),
+     diagnose = rep(c(0, 1), c(450, 570)), outcome = c(rep(c(1,
+         1, 1), 16), rep(c(1, 1, 0), 13), rep(c(1, 0, 1), 9),
+         rep(c(1, 0, 0), 3), rep(c(0, 1, 1), 14), rep(c(0, 1,
+             0), 4), rep(c(0, 0, 1), 15), rep(c(0, 0, 0), 6),
```

```
+         rep(c(1, 1, 1), 31), rep(c(1, 1, 0), 0), rep(c(1, 0,
+             1), 6), rep(c(1, 0, 0), 0), rep(c(0, 1, 1), 22),
+         rep(c(0, 1, 0), 2), rep(c(0, 0, 1), 9), rep(c(0, 0, 0),
+             0), rep(c(1, 1, 1), 2), rep(c(1, 1, 0), 2), rep(c(1,
+             0, 1), 8), rep(c(1, 0, 0), 9), rep(c(0, 1, 1), 9),
+         rep(c(0, 1, 0), 15), rep(c(0, 0, 1), 27), rep(c(0, 0,
+             0), 28), rep(c(1, 1, 1), 7), rep(c(1, 1, 0), 2),
+         rep(c(1, 0, 1), 5), rep(c(1, 0, 0), 2), rep(c(0, 1, 1),
+             31), rep(c(0, 1, 0), 5), rep(c(0, 0, 1), 32), rep(c(0,
+             0, 0), 6)))
> tb9.1b <- tb9.1
> tb9.1b$diagnose <- ifelse(tb9.1b$diagnose == 0, "mild", "severe")
> tb9.1b$diagnose <- factor(tb9.1b$diagnose, levels = c("mild",
+     "severe"))
> tb9.1b$treat <- ifelse(tb9.1b$treat == 0, "standard", "new")
> tb9.1b$treat <- factor(tb9.1b$treat, levels = c("standard", "new"))
```

Remember that when constructing a data frame the first values are those that change faster; now looking at table 9.1 we see that the treatment changes faster than diagnose, so we need to reflect that in the data frame, hence the command, `treat=rep(c(0,1,0,1), c(240,210,300,270))`, then for the diagnose variable we just sum the first two values and the last two, `diagnose=rep(c(0,1),c(450,570))`. To verify that the table was constructed correctly we can calculate the proportions of normal responses, showed in table 9.2.

```
> tb11.2b <- read.table('supp_data/tb11-2b')
> temp <- xtabs(I(ifelse(tb11.2b$outcome == 0, 1, 1)) ~ treat +
+     time + diagnose, data = tb11.2b)
> round(xtabs(outcome ~ treat + time + diagnose, data = tb9.1b)/temp,
+     2)
```

```
, , diagnose = mild

          time
treat         0    1    2
  standard 0.59 0.67 0.77
  new      0.46 0.69 0.85

, , diagnose = severe

          time
treat         0    1    2
  standard 0.23 0.31 0.51
  new      0.16 0.45 0.75
```

Since the table is correct, we can now fit the gee,

```
> ## library(gee)
> fit.gee <- gee(outcome ~ diagnose + treat * time, id = case,
+     family = binomial, corstr = "exchangeable", data = tb9.1b)
```

```
   (Intercept) diagnosesevere       treatnew           time
   -0.02798843    -1.31391092    -0.05960381     0.48241209
 treatnew:time
    1.01744498
```

```
> summary(fit.gee)
```

```
 GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
 gee S-function, version 4.13 modified 98/01/27 (1998)

Model:
 Link:                      Logit
 Variance to Mean Relation: Binomial
 Correlation Structure:     Exchangeable

Call:
gee(formula = outcome ~ diagnose + treat * time, id = case, data = tb9.1b,
    family = binomial, corstr = "exchangeable")

Summary of Residuals:
        Min          1Q      Median          3Q         Max
-0.94843397 -0.40683122  0.05156603  0.38832332  0.80238627


Coefficients:
                  Estimate Naive S.E.    Naive z Robust S.E.
(Intercept)    -0.02809866  0.1625499 -0.1728617   0.1741791
diagnosesevere -1.31391033  0.1448627 -9.0700418   0.1459630
treatnew       -0.05926689  0.2205340 -0.2687427   0.2285569
time            0.48246420  0.1141154  4.2278625   0.1199383
treatnew:time   1.01719312  0.1877051  5.4191018   0.1877014
                 Robust z
(Intercept)    -0.1613205
diagnosesevere -9.0016667
treatnew       -0.2593091
time            4.0226037
treatnew:time   5.4192084

Estimated Scale Parameter:  0.985392
Number of Iterations:  2

Working Correlation
             [,1]         [,2]         [,3]
[1,]  1.000000000 -0.003432732 -0.003432732
[2,] -0.003432732  1.000000000 -0.003432732
[3,] -0.003432732 -0.003432732  1.000000000
```

The results are the same as those reported in table 9.3; note that the common correlation value is the one in the secondary diagonal of the working correlation. As Agresti notes this indicates a very low correlation among data. Regarding the estimates, one should be aware that the estimate for the treatment

effect (-0.06) only applies for week 0, for subsequent weeks the effect of the new drug is, $-0.06 + 1.02t$. Finally note that the initial diagnosis has also an important effect (a negative one) in the log odds of a normal response.

Section 9.2.4 shows another example where rats on iron–deficient diets where assigned to four groups. Group 1 was the control and groups 2 to 4 received an iron supplement at different times. Later the rats where made pregnant and sacrificed at week 3. For each fetus in each rat's litter, the response was whether the fetus was dead. Note that in this case each litter is a cluster. For comparison, Agresti first fits a logistic model that assumes independence among the subjects (fetuses),

$$logit(\pi_{it}) = \alpha + \beta_2 z_{i2} + \beta_3 z_{i3} + \beta_4 z_{i4}$$

where β_i is an indicator variable that compares group i with the placebo group. Next, the author fits the GEE model that treats each litter as a cluster and permits correlations among subjects within a cluster.

In R we first import the data from Agresti's web page. This data frame, however, is not subject specific, thus we need to convert it to one where each row represents one fetus. Also we need to include a `case` variable. Note that in the following commands we repeat the rows in two occasions, first for the positive results, which is indicated by the `num.dead` variable, and then for the negative results, which corresponds to the difference between the `litter.size` and `num.dead` variables. These steps are done in two data frames that latter are merged together. Finally we order the data according to case number.

```
> tb9.4 <- read.table("supp_data/tb9-4")
> colnames(tb9.4) <- c("group", "litter.size", "num.dead")
> tb9.4$group <- factor(tb9.4$group)
> tb9.4a <- tb9.4
> tb9.4a <- within(tb9.4a, {
+     litter.size <- litter.size - num.dead
+ })
> tb9.4a$case <- c(1:58)
> tb9.4b <- tb9.4a
> tb9.4b <- tb9.4b[rep(1:58, tb9.4b$num.dead), ]
> tb9.4b$yes <- 1
> tb9.4c <- tb9.4a
> tb9.4c <- tb9.4c[rep(1:58, tb9.4c$litter.size), ]
> tb9.4c$no <- 0
> colnames(tb9.4b) <- c("group", "litter.size", "num.dead", "case",
+     "outcome")
> colnames(tb9.4c) <- c("group", "litter.size", "num.dead", "case",
+     "outcome")
> tb9.4b <- tb9.4b[, c(4, 1, 5)]
> tb9.4c <- tb9.4c[, c(4, 1, 5)]
> tb9.4long <- rbind(tb9.4b, tb9.4c)
> tb9.4long$group <- factor(tb9.4long$group)
> tb9.4long <- tb9.4long[order(tb9.4long[, "case"]), ]
> rownames(tb9.4long) <- c(1:nrow(tb9.4long))
```

Once the data is constructed we can fit the two models, and extract the coefficients along with their standard errors

```
> glm.9.4 <- glm(I(num.dead/litter.size) ~ group, weights = litter.size,
+     family = binomial, data = tb9.4)
> summary(glm.9.4)$coefficients[, 1:2]
```

```
             Estimate Std. Error
(Intercept)  1.143981  0.1291917
group2      -3.322513  0.3308440
group3      -4.476185  0.7311275
group4      -4.129663  0.4762261
```

```
> gee.9.4 <- gee(outcome ~ group, id = case, family = binomial,
+     corstr = "exchangeable", data = tb9.4long)
```

```
(Intercept)      group2      group3      group4
   1.143981   -3.322513   -4.476184   -4.129663
```

```
> summary(gee.9.4)$coefficients[, c(1, 4)]
```

```
             Estimate Robust S.E.
(Intercept)  1.211492   0.2695606
group2      -3.369165   0.4304211
group3      -4.583701   0.6235402
group4      -4.247410   0.6047894
```

These values are the same as those in table 9.5.

Agresti notes that when there is a positive correlation within cluster, then the standard errors for between–cluster effects and standard errors of estimate means tend to be larger than when we consider observations as independent. Contrary, within–cluster effects tend to be smaller than those when we treat observations as independent.

3.2 Limitations of GEE compared with ML

Although most software readily fits GEEs there are some limitations with this approach; the most important is that, because GEEs do not specify a multivariate distribution there is no likelihood function. This implies that there are no likelihood ratio methods to check the fit, compare models or conduct inference about parameters, in consequence one must rely on large–sample approximations like the Wald test, which may not work well with small data samples, however some software "improves" the Wald–type of inference by using the data information.

4 Extending GEE: Multinomial responses

This section deals with responses that have more than two categories, *i.e.* multinomial responses. Depending on, whether the response is nominal or ordinal, generalizations of GEEs use base line category logits or cumulative logits.

4.1 Multinomial GEEs in R

For the insomnia example, Thompson uses the `ordgee` function from the `geepack` package, however the results obtained are quite different from those reported in Agresti (2007, p. 286). Constructing the data frame from the data provided in table 9.6 (not importing from Agresti's web site) gives the same result as in Thompson (2007, pp. 219–220). The code is shown below,

```
> tb9.6a <- data.frame(expand.grid(outcome = c(1, 1, 1, 2, 1, 3,
+     1, 4, 2, 1, 2, 2, 2, 3, 2, 4, 3, 1, 3, 2, 3, 3, 3, 4, 4,
+     1, 4, 2, 4, 3, 4, 4), treat = c(0, 1)), occasion = c(0, 1),
+     count = rep(c(7, 4, 2, 1, 14, 5, 1, 0, 6, 9, 18, 2, 4, 11,
+         14, 22, 7, 4, 1, 0, 11, 5, 2, 2, 13, 23, 3, 1, 9, 17,
+         13, 8), each = 2))
> tb9.6a <- tb9.6a[rep(1:nrow(tb9.6a), tb9.6a$count), ]
> row.names(tb9.6a) <- 1:nrow(tb9.6a)
> tb9.6a$case <- rep(1:sum(tb9.6a$occasion), each = 2)
> tb9.6a$outcome2 <- ifelse(tb9.6a$outcome == 1, 10, ifelse(tb9.6a$outcome ==
+     2, 25, ifelse(tb9.6a$outcome == 3, 45, 75)))
> tb9.6b <- tb9.6a[order(tb9.6a$case, tb9.6a$occasion), ]
> tb9.6b$outcome <- ordered(tb9.6b$outcome, levels = 1:4)
```

Note that the outcome is codified from 1 to 4 and it's enter in "pairs", for example the values `1,1` represent the same outcome in the two evaluations; also note that these are entered row–wise so they match the order of the `count` variable. In the next command the rows are repeated count–times so each row represents an individual. The fourth command creates the `case` vector. The value of the `sum(tb9.6a$occasion)` command is half the number of rows in the data frame; it would be equally valid if it would we had been entered, `nrow(tb9.6)/2`. Finally, Thompson notes that it's important that the `case`, `time` and `outcome` vectors to be ordered.
Once the data is constructed we can fit the model

```
> ## library(geepack)
> fit.ordgee9.6 <- ordgee(outcome ~ treat * occasion, id = case,
+     data = tb9.6b, corstr = "independence", rev = TRUE, control = geese.control(maxit = 100))
```

The `rev=TRUE` argument indicates that we want the cumulative probabilities $P(y_{it} \leq j)$.
Although the package help page indicates that other correlation structures can be indicated, with this data the only one that seems to work is the independence correlation matrix.

Coding and model fitting in R At the end of section 9.3.2 Agresti mentions that an advantage of treating observations as correlated is that the standard errors are smaller than those obtained by considering the observations as independent. The author mentions that by doing the latter we obtain the same estimates but with higher SE. Interesting, fitting this model in `R` gives the same results as in Agresti, but with different sign! Thompson (2007, p. 220) notes this for the sign of the interaction estimate.

```
> ## library(MASS)
> polr(outcome ~ treat * occasion, data = tb9.6b)
```

```
Call:
polr(formula = outcome ~ treat * occasion, data = tb9.6b)
```

```
Coefficients:
         treat       occasion treat:occasion
   -0.03360977    -1.03808139    -0.70775209

Intercepts:
       1|2        2|3        3|4
-2.2670867 -0.9514647  0.3517109

Residual Deviance: 1241.988
AIC: 1253.988
```

4.2 Limitations in GEEs for categorical data

Agresti warns that for categorical data the correlation among observation can not take any value between [-1, +1], instead they are limited to a narrower interval defined by the marginal probabilities. As an alternative, some software uses the *iterative alternating logistic regression algorithm.*

5 Transitional modeling, given the past

This section describes another type of modeling that is useful when there is an interest in the relationship between a response at time t and previous responses y-k; such models that include past responses are called *transitional models.* Discrete–time *Markov Chains* discrete time stochastic processes with a discrete stat space, that is that the random variable may changes states at discrete time points, and the states come from a set of discrete possible states. First–order Markov chains is a transitional model for which, for all t, the conditional distribution of Y_t, given $Y_1, \ldots, Y_{t-1}$ is assumed identical to the conditional distribution of Y_t given Y_{t-1} alone; that is, given Y_{t-1} Y_t is conditionally independent of $Y_1, \ldots, Y_{t-2}$. In words this means that knowing the most recent observation, information about about previous observations, before that, don't help with predicting the next observation.

Agresti (2007) illustrates these models with an study about respiratory illness on children. These children were evaluated from age 7 through age 10 on whether they present a respiratory illness (1) or not (0); another variable recorded was whether their mothers smoke at the beginning of the study. The first–order Markov chain model for this study would be,

$$logit[P(Y_t = 1)] = \alpha + \beta y_{t-1} + \beta_1 s + \beta_2 t$$

where $t = 8, 9, 10$ s equals 1 for mothers who smoke regularly and 0 for those who don't.

This model treats, given the predictors in the model, past observations as independent; and it's called *regressive logistic model.* From that name we can infer that this type of model can be fitted with ordinary logistic regression software. Because of this we can make inferences based on likelihood ratio test (see section 3.2).

5.1 Transitional models in R

In order to fit these models, Thompson describes two methods, the first one allows higher order models, *i.e.* models that include more than the previous response; while the second allows the inclusion of other explanatory variables beside past observation. The latter is the one illustrated in Agresti (2007).
For the respiratory illness example we need to construct two data frames, the first with the data in table

9.8 and with this a second table representing the 12 possible combinations of the values for the predictor variables, *i.e.* 2 smoking status times 3 evaluations, at years 8, 9, 10, and 2 respiratory status.

```
> #contingency table
> tb9.8 <- data.frame(expand.grid(Y10 = c(0, 1), smoking = c(0,
+     1), Y9 = c(0, 1), Y8 = c(0, 1), Y7 = c(0, 1)), count = c(237,
+     10, 118, 6, 15, 4, 8, 2, 16, 2, 11, 1, 7, 3, 6, 4, 24, 3,
+     7, 3, 3, 2, 3, 1, 6, 2, 4, 2, 5, 11, 4, 7))
> # outcome table
> tb9.8b <- data.frame(expand.grid(previous = c(0, 1), t = 8:10,
+     smoking = c(0, 1)))
> count.yes <- c(sum(tb9.8[tb9.8$Y8 == 1 & tb9.8$smoking == 0 &
+     tb9.8$Y7 == 0, "count"]), sum(tb9.8[tb9.8$Y8 == 1 & tb9.8$smoking ==
+     0 & tb9.8$Y7 == 1, "count"]), sum(tb9.8[tb9.8$Y9 == 1 & tb9.8$smoking ==
+     0 & tb9.8$Y8 == 0, "count"]), sum(tb9.8[tb9.8$Y9 == 1 & tb9.8$smoking ==
+     0 & tb9.8$Y8 == 1, "count"]), sum(tb9.8[tb9.8$Y10 == 1 &
+     tb9.8$smoking == 0 & tb9.8$Y9 == 0, "count"]), sum(tb9.8[tb9.8$Y10 ==
+     1 & tb9.8$smoking == 0 & tb9.8$Y9 == 1, "count"]), sum(tb9.8[tb9.8$Y8 ==
+     1 & tb9.8$smoking == 1 & tb9.8$Y7 == 0, "count"]), sum(tb9.8[tb9.8$Y8 ==
+     1 & tb9.8$smoking == 1 & tb9.8$Y7 == 1, "count"]), sum(tb9.8[tb9.8$Y9 ==
+     1 & tb9.8$smoking == 1 & tb9.8$Y8 == 0, "count"]), sum(tb9.8[tb9.8$Y9 ==
+     1 & tb9.8$smoking == 1 & tb9.8$Y8 == 1, "count"]), sum(tb9.8[tb9.8$Y10 ==
+     1 & tb9.8$smoking == 1 & tb9.8$Y9 == 0, "count"]), sum(tb9.8[tb9.8$Y10 ==
+     1 & tb9.8$smoking == 1 & tb9.8$Y9 == 1, "count"]))
> count.no <- c(sum(tb9.8[tb9.8$Y8 == 0 & tb9.8$smoking == 0 &
+     tb9.8$Y7 == 0, "count"]), sum(tb9.8[tb9.8$Y8 == 0 & tb9.8$smoking ==
+     0 & tb9.8$Y7 == 1, "count"]), sum(tb9.8[tb9.8$Y9 == 0 & tb9.8$smoking ==
+     0 & tb9.8$Y8 == 0, "count"]), sum(tb9.8[tb9.8$Y9 == 0 & tb9.8$smoking ==
+     0 & tb9.8$Y8 == 1, "count"]), sum(tb9.8[tb9.8$Y10 == 0 &
+     tb9.8$smoking == 0 & tb9.8$Y9 == 0, "count"]), sum(tb9.8[tb9.8$Y10 ==
+     0 & tb9.8$smoking == 0 & tb9.8$Y9 == 1, "count"]), sum(tb9.8[tb9.8$Y8 ==
+     0 & tb9.8$smoking == 1 & tb9.8$Y7 == 0, "count"]), sum(tb9.8[tb9.8$Y8 ==
+     0 & tb9.8$smoking == 1 & tb9.8$Y7 == 1, "count"]), sum(tb9.8[tb9.8$Y9 ==
+     0 & tb9.8$smoking == 1 & tb9.8$Y8 == 0, "count"]), sum(tb9.8[tb9.8$Y9 ==
+     0 & tb9.8$smoking == 1 & tb9.8$Y8 == 1, "count"]), sum(tb9.8[tb9.8$Y10 ==
+     0 & tb9.8$smoking == 1 & tb9.8$Y9 == 0, "count"]), sum(tb9.8[tb9.8$Y10 ==
+     0 & tb9.8$smoking == 1 & tb9.8$Y9 == 1, "count"]))
> tb9.8b$prop <- count.yes/(total <- count.yes + count.no)
```

Note that the contingency table is the easiest to build, however (and as usual) one must make sure that the count vector matches the rest of the columns.The second data frame consist of three columns, representing the parameters in the model, namely, the previous outcome (2), the number of evaluations (3), and the smoking status of the mother (2). Then for each of the possible outcome (that has a previous observation) we construct vectors of positive and negative outcomes, for example the command, `sum(tb9.8[tb9.8\$Y8==1 \& tb9.8\$smoking==0 \& tb9.8\$Y7==0,"count"])` "counts" the number of positive (`tb9.8\$Y8==1`) outcomes that match the first line of the data frame,

```
  previous t smoking      prop
1        0 8       0 0.0952381
```

namely, a positive outcome at year 8, considering that the previous outcome was negative and that the smoking status of the mother was negative.
Then, we fit the model,

```
> fit9.8 <- glm(prop ~ previous + t + smoking, data = tb9.8b, family = binomial,
+     weight = total)
> summary(fit9.8)$coefficients
```

```
             Estimate Std. Error    z value     Pr(>|z|)
(Intercept) -0.2925632 0.84603077 -0.3458068 7.294879e-01
previous     2.2110745 0.15819280 13.9770870 2.151139e-44
t           -0.2428077 0.09465651 -2.5651452 1.031326e-02
smoking      0.2959619 0.15633938  1.8930733 5.834812e-02
```

Thompson reminds that as with any other GLM model the hypothesis that any term is non-significant can be tested by fitting a model that lacks that term and comparing to the more complex one; in R with the `anova` function.

Section 9.4.3 explains how this model can be used for the insomnia example of section 9.3.2. Because the response variable is an ordinal multicategory variable, we can use the `polr` function from the `MASS` package (see Thompson 2007, chapter 7). This function needs un-grouped data and an `ordered` response, so,

```
> #grouped data
> tb9.6x <- data.frame(expand.grid(outcome2 = 1:4, outcome1 = 1:4,
+     treat = c(1, 0)), count = c(7, 4, 2, 1, 14, 5, 1, 0, 6, 9,
+     18, 2, 4, 11, 14, 22, 7, 4, 1, 0, 11, 5, 2, 2, 13, 23, 3,
+     1, 9, 17, 13, 8))
> # ungrouped data
> tb9.6x$outcome1 <- ifelse(tb9.6x$outcome1 == 1, 10, ifelse(tb9.6x$outcome1 ==
+     2, 25, ifelse(tb9.6x$outcome1 == 3, 45, 75)))
> tb9.6y <- tb9.6x[rep(1:nrow(tb9.6x), tb9.6x$count), ]
> tb9.6y$outcome2 <- ordered(tb9.6y$outcome2, levels = 1:4)
```

Note that the previous response was re-codified to 10,25,45 and 75 and, as noted before, the response variable was ordered. Now we can fit the model

```
> fit.polr96 <- polr(outcome2 ~ outcome1 + treat, data = tb9.6y)
> summary(fit.polr96)$coefficients
```

```
             Value Std. Error   t value
outcome1 0.0421060 0.00589377  7.144154
treat    0.8846839 0.24654644  3.588305
1|2      1.4658397 0.33405296  4.388046
2|3      3.1621172 0.38504118  8.212413
3|4      4.6356348 0.44263601 10.472792
```

These values correspond to the model,

$$logit[P(Y_2 \leq j)] = \alpha_j + \beta_1 x + \beta_2 y_1$$

where x represents the treatment effect and y_1 the outcome at the first observation. The values match those of Agresti (2007, p. 290).

The author ends the chapter by highlighting that in this transitional models, the estimate for the previous response is, usually, the one with the strongest effect, affecting (reducing) that of the other explanatory variables.

References

Agresti, Alan (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)*.