Categorical Data Analysis: Partial Solution to HW5

Exercise 3: **Dysmenorrhea**

```
> myfile=file.choose()   # I typed up the contingency table in a spreadsheet
> dys <- read.csv(myfile)
> dys
```

1	1	0	0	0	0
2	1	0	0	1	2
3	1	0	1	0	2
4	1	0	1	1	9
5	1	1	0	0	0
6	1	1	0	1	0
7	1	1	1	0	1
8	1	1	1	1	1
9	2	0	0	0	2
10	2	0	0	1	0
11	2	0	1	0	0
12	2	0	1	1	9
13	2	1	0	0	1
14	2	1	0	1	0
15	2	1	1	0	0
16	2	1	1	1	4
17	3	0	0	0	0
18	3	0	0	1	1
19	3	0	1	0	1
20	3	0	1	1	8
21	3	1	0	0	1
22	3	1	0	1	3
23	3	1	1	0	0
24	3	1	1	1	1
25	4	0	0	0	0
26	4	0	0	1	1
27	4	0	1	0	1
28	4	0	1	1	8
29	4	1	0	0	1
30	4	1	0	1	0
31	4	1	1	0	0
32	4	1	1	1	1
33	5	0	0	0	3
34	5	0	0	1	0
35	5	0	1	0	0
36	5	0	1	1	7
37	5	1	0	0	0
38	5	1	0	1	1
39	5	1	1	0	2
40	5	1	1	1	1
41	6	0	0	0	1
42	6	0	0	1	5
43	6	0	1	0	0
44	6	0	1	1	4
45	6	1	0	0	0
46	6	1	0	1	3
47	6	1	1	0	1
48	6	1	1	1	0

```
> dys <- dys[dys$count>0,]
> head(dys)
  seq A B C count
2   1 0 0 1     2
3   1 0 1 0     2
4   1 0 1 1     9
7   1 1 1 0     1
8   1 1 1 1     1
9   2 0 0 0     2
> ## now dublicate rows according to information in count:
> ## this is a nice trick to do it with rownames:
> dys1 <- dys[rep(rownames(dys),dys$count),-5]
> head(dys1)
    seq A B C
2     1 0 0 1
2.1   1 0 0 1
3     1 0 1 0
3.1   1 0 1 0
4     1 0 1 1
4.1   1 0 1 1
> dim(dys1) #just checking, should have 86 rows (= # subjects)
[1] 86  4
> dys2 <- data.frame(subj=1:dim(dys1)[1], dys1) # include a subj column
> head(dys2)
    subj seq A B C
2      1   1 0 0 1
2.1    2   1 0 0 1
3      3   1 0 1 0
3.1    4   1 0 1 0
4      5   1 0 1 1
4.1    6   1 0 1 1
> ## now need to convert dataset from "wide" shape to "long" shape:
> require("reshape2")
Loading required package: reshape2
Warning message:
package reshape2 was built under R version
> dys3 <- melt(dys2, id=c("subj", "seq"))
> ## not necessary, but can order by subject:
> dys3 <- dys3[order(dys3$subj),]
> names(dys3)[3:4] <- c("treat", "resp")
> head(dys3, 15)
    subj seq treat resp
1      1   1     A    0
87     1   1     B    0
173    1   1     C    1
2      2   1     A    0
88     2   1     B    0
174    2   1     C    1
3      3   1     A    0
89     3   1     B    1
175    3   1     C    0
4      4   1     A    0
90     4   1     B    1
176    4   1     C    0
5      5   1     A    0
91     5   1     B    1
177    5   1     C    1
```

```
> tail(dys3)
    subj seq treat resp
85    85   6     A    1
171   85   6     B    0
257   85   6     C    1
86    86   6     A    1
172   86   6     B    1
258   86   6     C    0
> ## doesn't matter that treatments are not listed in the order they were applied
> ## It is only important that each treatment has the correct response next to it
> ## Treatment order doesn't matter in dataset as we are not going to use this info
> ## since using exchangeable or independence or unstructured working correlation
> ## However, if we used ar(1) correlation structure, then it would matter!
>
> require(gee)
Loading required package: gee
> ##Actually, if we use "geepack", it has an anova command
> ## which is convenient for carrying out Wald tests of
> ## multiparameter hypotheses, such as if all sequence dummies equal zero
> fit_exch <- gee(resp ~ factor(seq) + treat, family=binomial, data=dys3,
corstr="exchangeable", id=subj)
Beginning Cgee S-function, (#) geeformula.q 4.13 98/01/27
running glm to get initial regression estimate
 (Intercept) factor(seq)2 factor(seq)3 factor(seq)4 factor(seq)5 factor(seq)6
  -1.0314607    0.2580769    0.1235803    0.0614507   -0.2818675   -0.5295869
      treatB       treatC
   1.9917375    2.5078182
> summary(fit_exch)

 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 = resp ~ factor(seq) + treat, id = subj, data = dys3,
    family = binomial, corstr = "exchangeable")

Summary of Residuals:
       Min               Median                  Max
-0.8497774 -0.2750477  0.1678778  0.2645932  0.8252272


Coefficients:
                Estimate Naive S.E.    Naive z Robust S.E.   Robust z
(Intercept)  -1.03221034  0.3991316 -2.5861404   0.3456853 -2.9859827
factor(seq)2  0.25750558  0.4801269  0.5363282   0.5075932  0.5073070
factor(seq)3  0.12539183  0.4844781  0.2588184   0.3539468  0.3542674
factor(seq)4  0.06304881  0.5124652  0.1230304   0.3929399  0.1604541
factor(seq)5 -0.28156587  0.4853069 -0.5801811   0.4918835 -0.5724239
factor(seq)6 -0.51996164  0.4831552 -1.0761794   0.3907339 -1.3307307
treatB        1.99139166  0.3613000  5.5117401   0.3876163  5.1375338
treatC        2.50756136  0.3867818  6.4831429   0.4141413  6.0548455
```

```
Estimated Scale Parameter:  1.040165
Number of Iterations:  2

Working Correlation
            [,1]        [,2]        [,3]
[1,]  1.00000000 -0.04403087 -0.04403087
[2,] -0.04403087  1.00000000 -0.04403087
[3,] -0.04403087 -0.04403087  1.00000000
> ## to test sequence effect, we need to test if all 5 sequence parameters are zero
> ## this can be done with a Wald test, which is a quadratic form t(beta.hat) %*% S %*% beta.hat
> ## where beta.hat refers to the vector of the fitted coefficient for the sequence terms
> ## and S is their estimated variance covariance matrix
> ## S is difficult, if not impossible to derive from the gee output
> ## All we can do is to look at robust z-statistics for each of the 5 coefficients.
> ## All are pretty small,
> ## indicating not different from zero -> Sequence does not have an effect.
>
> ## When fitting GEE models with package "geepack", we can get this Wald test
> ## and give a more precise analysis:
> require("geepack")
Loading required package: geepack
> fit1_exch <- geeglm(resp ~ factor(seq) + treat, family=binomial, data=dys3,
corstr="exchangeable", id=subj)
> summary(fit1_exch)

Call:
geeglm(formula = resp ~ factor(seq) + treat, family = binomial,
    data = dys3, id = subj, corstr = "exchangeable")

 Coefficients:
             Estimate  Std.err   Wald Pr(>|W|)
(Intercept)  -1.03221  0.34569  8.916  0.00283 **
factor(seq)2  0.25751  0.50759  0.257  0.61194
factor(seq)3  0.12539  0.35395  0.126  0.72314
factor(seq)4  0.06305  0.39294  0.026  0.87252
factor(seq)5 -0.28157  0.49188  0.328  0.56703
factor(seq)6 -0.51996  0.39073  1.771  0.18328
treatB        1.99139  0.38762 26.394 2.78e-07 ***
treatC        2.50756  0.41414 36.661 1.41e-09 ***
---
Estimated Correlation Parameters:
      Estimate Std.err
alpha -0.04403 0.06366
Number of clusters:   86   Maximum cluster size: 3

> anova(fit1_exch)
Analysis of 'Wald statistic' Table
Model: binomial, link: logit
Response: resp
Terms added sequentially (first to last)


            Df   X2 P(>|Chi|)
factor(seq)  5  4.2      0.52
treat        2 38.4   4.6e-09 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
>
```

```
> ## Can fit model without sequence effect:
> fit2_exch <- geeglm(resp ~ treat, family=binomial, data=dys3, corstr="exchangeable",
id=subj)
> exp(-1.068)/(1+exp(-1.069)) # fitted success prob for drug A
[1] 0.256
> exp(-1.068+1.960)/(1+exp(-1.068+1.960)) # fitted success prob for drug B
[1] 0.709
> exp(-1.068+2.469)/(1+exp(-1.068+2.469)) # fitted success prob for drug C
[1] 0.802
> ## If you wantto get 95% confidence intervals for the success probability for each
drug,
> ## you can get if for drug A:
> intA <- -1.068+cbind(-1,1)*qnorm(1-0.05/2)*0.247
> intA
      [,1]   [,2]
[1,] -1.55 -0.584
> exp(intA)/(1+exp(intA))  #95% confidence interval for success with A
      [,1]  [,2]
[1,] 0.175 0.358
> ## However, for the interval for B and C, we would need the covariance matrix between
> ## the intercept and treatB or treatC estimates
> ## With glm, one has the vcov() command for this, but this doesn't work with geeglm
> ## One solution is to fit the model without the intercept, which then results in
three
> ## parameters that estimate the odds ratio (and hence the probability) directly:
> fit3_exch <- geeglm(resp ~ -1 + treat, family=binomial, data=dys3,
corstr="exchangeable", id=subj)
> summary(fit3_exch)

Coefficients:
       Estimate Std.err Wald Pr(>|W|)
treatA   -1.068   0.247 18.7  1.6e-05 ***
treatB    0.892   0.237 14.1  0.00017 ***
treatC    1.401   0.271 26.8  2.3e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Estimated Scale Parameters:
            Estimate Std.err
(Intercept)        1   0.123

Correlation: Structure = exchangeable  Link = identity

Estimated Correlation Parameters:
      Estimate Std.err
alpha  -0.0427  0.0691
Number of clusters:   86   Maximum cluster size: 3
> intB <- 0.892+cbind(-1,1)*1.96*0.237
> exp(intB)/(1+exp(intB)) #95% confidence interval for success with B
      [,1]  [,2]
[1,] 0.605 0.795
> intC <- 1.401+cbind(-1,1)*1.96*0.271
> exp(intC)/(1+exp(intC)) #95% confidence interval for success with C
      [,1]  [,2]
[1,] 0.705 0.873
```

```
> ## If you want that the error in all three confidence intervals combined is no larger
> ## than 0.05, you need to use a Bonferroni multiplicity adjustment for the critical
value:
> intA <- -1.068+cbind(-1,1)*qnorm(1-0.05/(2*3))*0.247
> intA
      [,1]   [,2]
[1,] -1.66 -0.477
> exp(intA)/(1+exp(intA))  # Simultaneous 95% confidence interval for success with A
     [,1]  [,2]
[1,] 0.16 0.383
> intB <- 0.892+cbind(-1,1)*qnorm(1-0.05/(2*3))*0.237
> exp(intB)/(1+exp(intB)) # Simultaneous 95% confidence interval for success with B
     [,1]  [,2]
[1,] 0.58 0.811
> intC <- 1.401+cbind(-1,1)*qnorm(1-0.05/(2*3))*0.271
> exp(intC)/(1+exp(intC)) # Simultaneous 95% confidence interval for success with C
     [,1]  [,2]
[1,] 0.68 0.886
>
```