Stat 536 Final

Dongyang Wang

2022-12-06

Question 1

1

```
rm(list = ls())
df <- c(1105, 4624, 411111, 157342, 14, 497, 483, 1008)
df.array <- array(df, c(2,2,2))
df.array
```

```
## , , 1
##
##      [,1]   [,2]
## [1,] 1105 411111
## [2,] 4624 157342
##
## , , 2
##
##      [,1] [,2]
## [1,]   14  483
## [2,]  497 1008
```

```
saturated.loglin = loglin(df.array,margin=list(c(1,2,3)),
                                        fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 0
```

```
indep.loglin = loglin(df.array,margin=list(1,2,3),
                                  fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 5.820766e-11
```

```
X1indepX2X3.loglin = loglin(df.array ,margin=list(1,c(2,3)),
                                          fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 5.820766e-11
```

```
X2indepX1X3.loglin = loglin(df.array ,margin=list(2,c(1,3)),
                                              fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 0
```

```
X3indepX1X2.loglin = loglin(df.array ,margin=list(3,c(1,2)),
                                        fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 0
```

```
X2indepX3givenX1.loglin = loglin(df.array ,margin=list(c(1,2),c(1,3)),
                                 fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 0
```

```
X1indepX3givenX2.loglin = loglin(df.array ,margin=list(c(1,2),c(2,3)),
                                   fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 0
```

```
X1indepX2givenX3.loglin = loglin(df.array ,margin=list(c(1,3),c(2,3)),
                       fit=TRUE,param=TRUE)
```

```
## 2 iterations: deviation 0
```

```
no2nd.loglin = loglin(df.array ,margin=list(c(1,2),c(1,3),c(2,3)),
                                       fit=TRUE,param=TRUE)
```

```
## 5 iterations: deviation 0.08833306
```

Then we assess their fit.

```
1-pchisq(indep.loglin$lrt,indep.loglin$df)
```

```
## [1] 0
```

```
1-pchisq(X1indepX2X3.loglin$lrt,X1indepX2X3.loglin$df)
```

```
## [1] 0
```

```
1-pchisq(X2indepX1X3.loglin$lrt,X2indepX1X3.loglin$df)
```

```
## [1] 0
```

```
1-pchisq(X3indepX1X2.loglin$lrt,X3indepX1X2.loglin$df)
```

```
## [1] 0
```

```
1-pchisq(X2indepX3givenX1.loglin$lrt,X2indepX3givenX1.loglin$df)
```

```
## [1] 0
```

```
1-pchisq(X1indepX3givenX2.loglin$lrt,X1indepX3givenX2.loglin$df)
```

```
## [1] 0
```

```
1-pchisq(X1indepX2givenX3.loglin$lrt,X1indepX2givenX3.loglin$df)
```

```
## [1] 0
```

```
1-pchisq(no2nd.loglin$lrt,no2nd.loglin$df)
```

```
## [1] 0.09114565
```

Based on above results, the no second order interaction model fits data well (fail to reject the null hypothesis that the model does not fit the data). Details about the model below:

```
no2nd.loglin
```

```
## $lrt
## [1] 2.854022
##
## $pearson
## [1] 2.555765
```

```
##
## $df
## [1] 1
##
## $margin
## $margin[[1]]
## [1] 1 2
##
## $margin[[2]]
## [1] 1 3
##
## $margin[[3]]
## [1] 2 3
##
##
## $fit
## , , 1
##
##          [,1]     [,2]
## [1,] 1098.135 411117.9
## [2,] 4630.865 157335.1
##
## , , 2
##
##           [,1]      [,2]
## [1,]  20.86795  476.1291
## [2,] 490.13205 1014.8709
##
##
## $param
## $param$`(Intercept)`
## [1] 7.831966
##
## $param$`1`
## [1] -0.5489897  0.5489897
##
## $param$`2`
## [1] -1.663277  1.663277
##
## $param$`3`
## [1]  2.251693 -2.251693
##
## $param$`1.2`
##            [,1]       [,2]
## [1,] -0.5999082  0.5999082
## [2,]  0.5999082 -0.5999082
##
## $param$`1.3`
##            [,1]       [,2]
## [1,]  0.4293324 -0.4293324
## [2,] -0.4293324  0.4293324
##
## $param$`2.3`
##            [,1]       [,2]
```

```
## [1,] -0.6994481  0.6994481
## [2,]  0.6994481 -0.6994481
```

An expression for this model is $logm_{ijk} = u + u_{1(i)} + u_{2(j)} + u_{3(k)} + u_{12(ij)} + u_{13(ik)} + u_{23(jk)}$.

2

$$\frac{P(X_3 = 2|X_1 = i, X_2 = j)}{P(X_3 = 1|X_1 = i, X_2 = j)} = exp((\hat{u}_{3(2)} - \hat{u}_{3(1)}) + (\hat{u}_{13(i2)} - \hat{u}_{13(i1)}) + (\hat{u}_{23(j2)} - \hat{u}_{23(j1)}))$$

which is equivalent to

For i = 1, j = 1

```
exp((-2.251693 - ( 2.251693)) + ( -0.4293324 - 0.4293324) + ( 0.6994481 - (- 0.6994481))  )
```

```
## [1] 0.01900307
```

For i = 1, j = 2

```
exp(( -2.251693 - (2.251693)) + (-0.4293324 - 0.4293324) + (- 0.6994481 -0.6994481 )  )
```

```
## [1] 0.001158132
```

For i = 2, j = 1

```
exp((-2.251693 - (+ 2.251693)) + (0.4293324 - (-0.4293324)) + (0.6994481 - (-0.6994481))  )
```

```
## [1] 0.1058402
```

For i = 2, j = 2

```
exp((-2.251693 - (2.251693)) + (0.4293324 - (-0.4293324)) + ( -0.6994481 - 0.6994481)  )
```

```
## [1] 0.006450372
```

Based on the above results, we learn that the odds of fatality is small regardless of the conditions.

3

In the following reconstruction of the dataset, x1 is seat belt and is 1 if yes; x2 is ejection and is 1 if yes; x3 is injury and 2 if fatal.

```
mydata = matrix(c(rep(c(1,1,1),1105),rep(c(2,1,1),4624),rep(c(1,2,1),411111),
                  rep(c(2,2,1),157342),rep(c(1,1,2),14),rep(c(2,1,2),497),
                  rep(c(1,2,2),483),rep(c(2,2,2),1008)),ncol = 3, byrow = TRUE)

mylogit = glm(factor(mydata[,3])~1,
              family=binomial(link=logit))
mylogit_1 = glm(factor(mydata[,3])~factor(mydata[,1]),
                family=binomial(link=logit))
mylogit_2= glm(factor(mydata[,3])~factor(mydata[,2]),
               family=binomial(link=logit))
mylogit_all= glm(factor(mydata[,3])~factor(mydata[,1])+factor(mydata[,2]),
                 family=binomial(link=logit))

AIC(mylogit)
```

```
## [1] 26670.81
```

```
AIC(mylogit_1)
```

```
## [1] 24785.5
```

```
AIC(mylogit_2)
```

```
## [1] 24249.72
```

```
AIC(mylogit_all)
```

```
## [1] 23109.94
```

Based on the AIC, we select the last comprehensive logistic regression model.

```
summary(mylogit_all)
```

```
##
## Call:
## glm(formula = factor(mydata[, 3]) ~ factor(mydata[, 1]) + factor(mydata[,
##     2]), family = binomial(link = logit))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.4486  -0.1134  -0.0481  -0.0481   3.6775
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -3.96315    0.06944  -57.07   <2e-16 ***
## factor(mydata[, 1])2   1.71732    0.05401   31.79   <2e-16 ***
## factor(mydata[, 2])2 -2.79779    0.05526  -50.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26669  on 576183  degrees of freedom
## Residual deviance: 23104  on 576181  degrees of freedom
## AIC: 23110
##
## Number of Fisher Scoring iterations: 9
```

4

Based on the results from above three sections, we know that seat belts and whether they are ejected both matter in terms of whether car injuries are fatal. Furthermore, There is a negative correlation between ejection and seat belt, with the correlation being -0.1394401 as shown below. Based on the coefficient from part 3, whether a person is ejected from the car is more relevant for fatality.

```
cor(mydata[,2], mydata[,3])
```

```
## [1] -0.1394401
```

Question 2

```
rm(list = ls())
library(Rgraphviz)
```

```
## Warning: package 'Rgraphviz' was built under R version 4.2.1
```

```
## Loading required package: graph
```

```
## Warning: package 'graph' was built under R version 4.2.1
## Loading required package: BiocGenerics
## Warning: package 'BiocGenerics' was built under R version 4.2.1
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
## Loading required package: grid
```

```
library(gRbase)
library(graph)
library(gRim)
```

```
# data set
#data(gRbase::reinis)
getdata <- function(...)
{
    e <- new.env()
    name <- data(..., envir = e)[1]
    e[[name]]
}
reinis1 = getdata(reinis)
str(reinis1)
```

```
##  'table' num [1:2, 1:2, 1:2, 1:2, 1:2, 1:2] 44 40 112 67 129 145 12 23 35 12 ...
##  - attr(*, "dimnames")=List of 6
##   ..$ smoke  : chr [1:2] "y" "n"
##   ..$ mental : chr [1:2] "y" "n"
##   ..$ phys   : chr [1:2] "y" "n"
##   ..$ systol : chr [1:2] "y" "n"
##   ..$ protein: chr [1:2] "y" "n"
##   ..$ family : chr [1:2] "y" "n"
```

To obtain the relevant log linear models, one approach is to replicate the paper discussed in class, namely starting from the saturated model and deleting edges and adding edges by searches. However, a more convenient approach is use the stepwise approach and calculate the model that achieves the lowest AIC or BIC. Therefore, we only need one relevant model, that is the saturated model as listed below.

```
# saturated model
m<-dmod(~.^.,data=reinis1)
m
```

```
## Model: A dModel with 6 variables
##  -2logL    :       13286.27 mdim :   63 aic :     13412.27
##  ideviance :         843.96 idf  :   57 bic :     13759.91
```

```
##  deviance  :           0.00 df   :    0
```

We then do the model selection by AIC and BIC, and visualize the results.

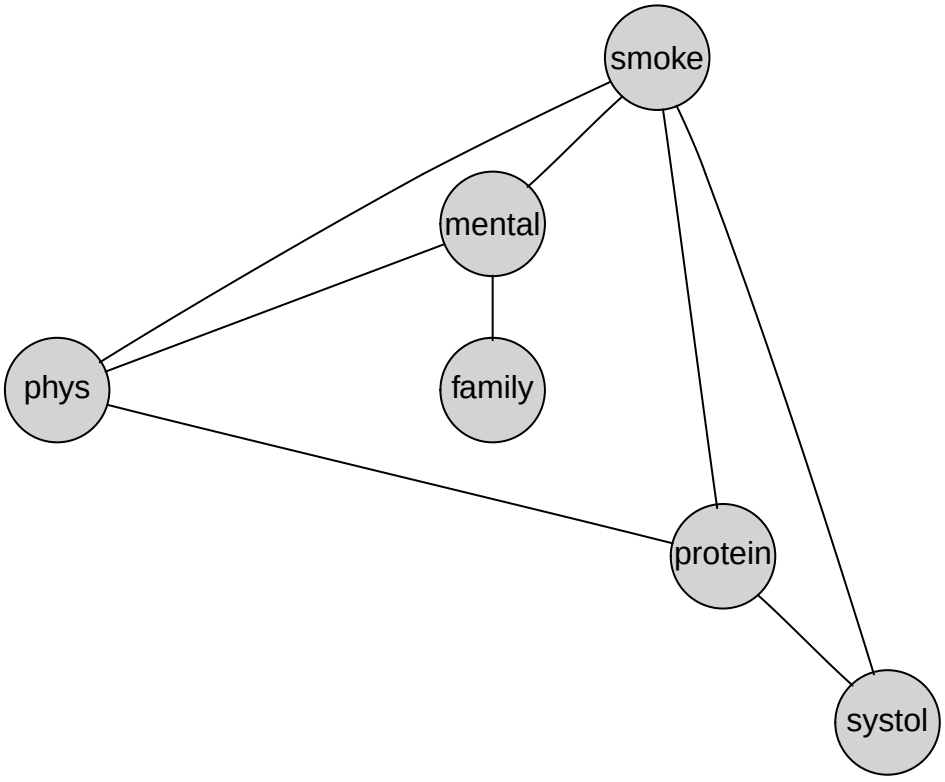
```
# AIC for model selection
step_m <- stepwise(m)
step_m
```

```
## Model: A dModel with 6 variables
##  -2logL    :       13337.63 mdim :   17 aic :     13371.63
##  ideviance :         792.60 idf  :   11 bic :     13465.43
##  deviance  :          51.36 df   :   46
```

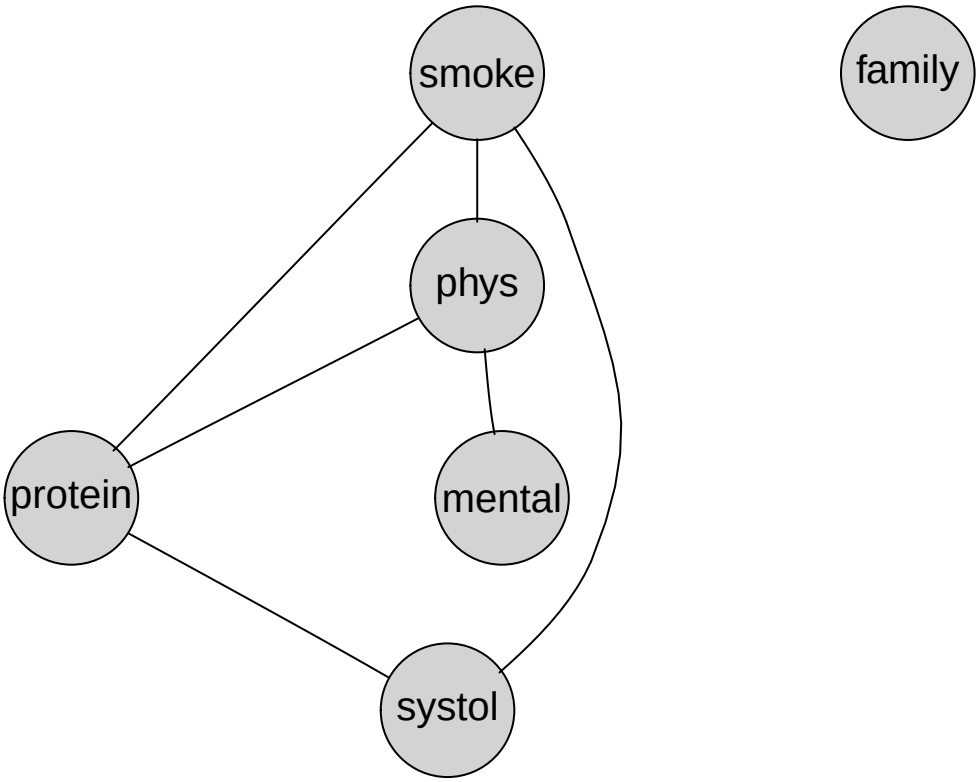
```
# BIC for model selection
step_m2 <- stepwise(m,k=log(sum(reinis1)))
step_m2
```

```
## Model: A dModel with 6 variables
##  -2logL    :       13348.35 mdim :   14 aic :     13376.35
##  ideviance :         781.88 idf  :    8 bic :     13453.60
##  deviance  :          62.08 df   :   49
```

```
# visualize the models
plot(step_m)
```




```
plot.new()
plot(step_m2)
```



We can, for example, choose the model with lowest BIC. The formula is listed below.

```
formula(step_m2)
```

```
## ~smoke * phys * protein + smoke * systol * protein + mental *
##     phys + family
```

```
as(step_m2, "matrix")
```

```
##         smoke phys protein systol mental family
## smoke       0    1       1      1      0      0
## phys        1    0       1      0      1      0
## protein     1    1       0      1      0      0
## systol      1    0       1      0      0      0
## mental      0    1       0      0      0      0
## family      0    0       0      0      0      0
```

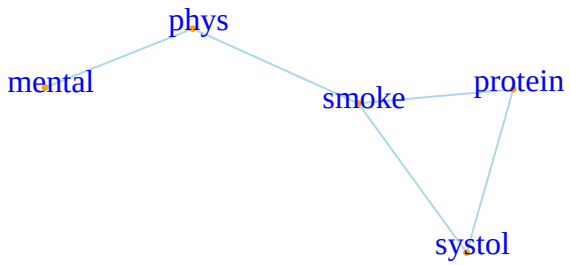
We also attempt the Graph estimation in GGMs using the birth-death MCMC.

```
library(BDgraph)
reinis2 = getdata(reinis)
sample <- bdgraph.mpl(data=reinis2, method = "dgm-binary",
                      algorithm = "bdmcmc", iter = 10000,
                      burnin = 6000)
```

```
##   This OS does not support multi-threading for the BDgraph package
## 10000 MCMC sampling ... in progress:
## 10%->20%->30%->40%->50%->60%->70%->80%->90%-> done
```

```
select(sample, cut=0.5,vis = TRUE)
```

Graph with links posterior probabilities > 0.5



family

```
##      smoke mental phys systol protein family
## [1,]     0      0    1      1       1      0
## [2,]     0      0    1      0       0      0
## [3,]     0      0    0      0       0      0
## [4,]     0      0    0      0       1      0
## [5,]     0      0    0      0       0      0
## [6,]     0      0    0      0       0      0
```

Therefore, based on model selection of BIC and Graph estimation in GGMs using the birth-death MCMC we obtain the result that family is not related with other variables. For the logistic regression, we would need include any variable of ABCDE, and only include the constant term based on our results above.