Homework 4: Data separation and multinomial regression

your name

Due: March 3rd at 11:59 PM

Problem 1: Do the following regarding the Sabermetrics dataset (bball.csv),

(a) Fit the `nnet` model and comment on the similarities and differences between the `nnet` and `VGAM` fits in the Sabermetrics example in the ordinal and multinomial regression notes. Report interesting conclusions using either implementation.

(b) Provide recommendations on how an aspiring baseball player should approach hitting. You may want to consider success metrics like hits where hits = 1B + 2B + 3B + HR, or weighted hits where weighted hits = 1B + 2×2B + 3×3B + 4×HR. Note these metrics are conditional on a ball being put into play in the context of this analysis.

Solution 1

```
library(VGAM)
library(nnet)
library(tidyverse)
#setwd('/Users/diptarka/Documents/GitHub/stat528resources/notes/5_multinomial') ## set your own WD
bball <- read.csv("../notes/5_multinomial/bball.csv")
bball$events <- as.factor(bball$events)
```

```
system.time(mod_vgam_small <- vglm(events ~ launch_speed + launch_angle + spray_angle +
                        I(launch_angle^2) ,
                  family=multinomial, data=bball))
```

(a)

```
##    user  system elapsed
##   3.169   0.297   3.479
```

```
system.time(mod_nnet_small <- multinom(events ~ launch_speed + launch_angle + spray_angle +
                I(launch_angle^2) , trace = F,
              data=bball, maxit = 1e3))
```

```
##    user  system elapsed
##   1.433   0.007   1.445
```

```
(mod_vgam_small)
```

```
##
## Call:
## vglm(formula = events ~ launch_speed + launch_angle + spray_angle +
##     I(launch_angle^2), family = multinomial, data = bball)
##
```

```
##
## Coefficients:
##       (Intercept):1       (Intercept):2       (Intercept):3       (Intercept):4
##        -0.992044856        -8.700016742       -13.901903740       -69.069422845
##      launch_speed:1      launch_speed:2      launch_speed:3      launch_speed:4
##         0.006030511         0.066921071         0.088201226         0.416300891
##      launch_angle:1      launch_angle:2      launch_angle:3      launch_angle:4
##         0.015388970         0.134974932         0.183316005         1.847503026
##       spray_angle:1       spray_angle:2       spray_angle:3       spray_angle:4
##        -0.002331842        -0.009296012         0.017281109        -0.007929848
## I(launch_angle^2):1 I(launch_angle^2):2 I(launch_angle^2):3 I(launch_angle^2):4
##        -0.001410135        -0.003499897        -0.003941694        -0.030299742
##
## Degrees of Freedom: 200000 Total; 199980 Residual
## Residual deviance: 70874.6
## Log-likelihood: -35437.3
##
## This is a multinomial logit model with 5 levels
```

```
(mod_nnet_small)
```

```
## Call:
## multinom(formula = events ~ launch_speed + launch_angle + spray_angle +
##     I(launch_angle^2), data = bball, trace = F, maxit = 1000)
##
## Coefficients:
##     (Intercept) launch_speed launch_angle  spray_angle I(launch_angle^2)
## b2   -7.7079358  0.060890419   0.11958334 -0.006963959      -0.002089705
## b3  -12.9104603  0.082174634   0.16795879  0.019612921      -0.002532400
## b4  -68.0764825  0.410266946   1.83207561 -0.005598212      -0.028888942
## out   0.9920324 -0.006030361  -0.01538899  0.002331716       0.001410122
##
## Residual Deviance: 70874.6
## AIC: 70914.6
```

```
system.time(mod_vgam <- vglm(events ~ launch_speed + launch_angle + spray_angle +
                        I(launch_angle^2) + I(spray_angle^2) + I(spray_angle^3) + I(spray_angle^4) +
                        I(spray_angle^5) + I(spray_angle^6) + I(spray_angle*launch_angle) +
                          I(spray_angle*launch_speed) + I(launch_angle*launch_speed),
                 family=multinomial, data=bball))
```

```
##    user  system elapsed
##  10.044   0.633  10.727
```

```
system.time(mod_nnet <- multinom(events ~ launch_speed + launch_angle + spray_angle +
               I(launch_angle^2) +
               I(spray_angle^2) + I(spray_angle^3) + I(spray_angle^4) +
               I(spray_angle^5) + I(spray_angle^6) +
               I(spray_angle*launch_angle) + I(spray_angle*launch_speed) +
               I(launch_angle*launch_speed),
             data=bball, maxit = 1e3, trace = F))
```

```
##    user  system elapsed
##  48.979   0.410  49.733
```

```
(mod_vgam)
```

```
##
## Call:
## vglm(formula = events ~ launch_speed + launch_angle + spray_angle +
##     I(launch_angle^2) + I(spray_angle^2) + I(spray_angle^3) +
##     I(spray_angle^4) + I(spray_angle^5) + I(spray_angle^6) +
##     I(spray_angle * launch_angle) + I(spray_angle * launch_speed) +
##     I(launch_angle * launch_speed), family = multinomial, data = bball)
##
##
## Coefficients:
##                     (Intercept):1                     (Intercept):2
##                      -1.020522e+00                      -8.758315e+00
##                     (Intercept):3                     (Intercept):4
##                      -9.471725e+00                      -8.200502e+01
##                    launch_speed:1                    launch_speed:2
##                       7.481784e-03                       5.941133e-02
##                    launch_speed:3                    launch_speed:4
##                       4.154721e-02                       4.736920e-01
##                    launch_angle:1                    launch_angle:2
##                       4.413206e-02                      -3.720448e-02
##                    launch_angle:3                    launch_angle:4
##                      -1.407160e-01                       2.101850e+00
##                     spray_angle:1                     spray_angle:2
##                      -1.719423e-02                      -1.982876e-02
##                     spray_angle:3                     spray_angle:4
##                       2.235206e-02                       1.609870e-02
##               I(launch_angle^2):1               I(launch_angle^2):2
##                      -1.432220e-03                      -4.205534e-03
##               I(launch_angle^2):3               I(launch_angle^2):4
##                      -4.513380e-03                      -3.648606e-02
##                I(spray_angle^2):1                I(spray_angle^2):2
##                      -1.614774e-04                      -2.375574e-03
##                I(spray_angle^2):3                I(spray_angle^2):4
##                      -3.395291e-03                       2.587399e-03
##                I(spray_angle^3):1                I(spray_angle^3):2
##                      -5.569724e-06                      -4.859610e-07
##                I(spray_angle^3):3                I(spray_angle^3):4
##                       1.242708e-06                       5.779472e-06
##                I(spray_angle^4):1                I(spray_angle^4):2
##                       1.736069e-08                       2.809633e-06
##                I(spray_angle^4):3                I(spray_angle^4):4
##                       3.142074e-06                      -1.554782e-07
##                I(spray_angle^5):1                I(spray_angle^5):2
##                       5.978194e-10                      -1.496561e-09
##                I(spray_angle^5):3                I(spray_angle^5):4
##                      -3.256414e-10                      -1.948870e-09
##                I(spray_angle^6):1                I(spray_angle^6):2
##                       2.375031e-12                      -5.001047e-10
##                I(spray_angle^6):3                I(spray_angle^6):4
##                      -5.582904e-10                      -5.527105e-11
##  I(spray_angle * launch_angle):1  I(spray_angle * launch_angle):2
##                       2.590185e-04                       3.545018e-04
##  I(spray_angle * launch_angle):3  I(spray_angle * launch_angle):4
##                      -1.616642e-04                      -1.785922e-04
```

```
##  I(spray_angle * launch_speed):1  I(spray_angle * launch_speed):2
##                     2.204672e-04                     1.171823e-04
##  I(spray_angle * launch_speed):3  I(spray_angle * launch_speed):4
##                    -6.552925e-05                    -2.097509e-04
## I(launch_angle * launch_speed):1 I(launch_angle * launch_speed):2
##                    -3.362434e-04                     2.225316e-03
## I(launch_angle * launch_speed):3 I(launch_angle * launch_speed):4
##                     3.708676e-03                     1.309787e-03
##
## Degrees of Freedom: 200000 Total; 199948 Residual
## Residual deviance: 66221.19
## Log-likelihood: -33110.59
##
## This is a multinomial logit model with 5 levels
```

```
(mod_nnet)
```

```
## Call:
## multinom(formula = events ~ launch_speed + launch_angle + spray_angle +
##     I(launch_angle^2) + I(spray_angle^2) + I(spray_angle^3) +
##     I(spray_angle^4) + I(spray_angle^5) + I(spray_angle^6) +
##     I(spray_angle * launch_angle) + I(spray_angle * launch_speed) +
##     I(launch_angle * launch_speed), data = bball, maxit = 1000,
##     trace = F)
##
## Coefficients:
##      (Intercept) launch_speed  launch_angle spray_angle I(launch_angle^2)
## b2  -0.012786946  -0.02942134 -0.1243477470  0.01220342      -0.003799177
## b3  -0.003845052  -0.10722637 -0.0384860419 -0.01969221      -0.010348101
## b4  -0.010471831  -0.33599697 -0.1617869751  0.01384867      -0.038502552
## out  0.020450927   0.00251478 -0.0008139971  0.01145874       0.001528926
##     I(spray_angle^2) I(spray_angle^3) I(spray_angle^4) I(spray_angle^5)
## b2     -0.0025318066     5.912623e-06     2.964218e-06    -2.384605e-09
## b3     -0.0030382702    -1.155508e-05     3.233370e-06     7.950473e-09
## b4      0.0020810396     1.832280e-05     4.179104e-08    -5.052417e-09
## out     0.0003445678     6.383933e-06    -9.816620e-08    -7.433864e-10
##     I(spray_angle^6) I(spray_angle * launch_angle)
## b2     -5.601701e-10                  1.728859e-04
## b3     -6.666766e-10                 -1.867436e-03
## b4     -9.077289e-11                  1.330678e-05
## out     5.825530e-12                 -2.352551e-04
##     I(spray_angle * launch_speed) I(launch_angle * launch_speed)
## b2                   -0.0002771604                   0.0034014831
## b3                    0.0008033519                   0.0066723187
## b4                   -0.0003869603                   0.0247443464
## out                  -0.0001657136                  -0.0001767814
##
## Residual Deviance: 68897.6
## AIC: 69001.6
```

Similarity:

- `vgam` and `nnet` are just different implementations of the the same underlying theoretical model. So they should give the same fitted coefficients and deviance and everything, which is the case for the smaller models fitted above, `mod_vgam_small` and `mod_nnet_small`.

Differences:

- Two implementations of models pick different baseline category. For model fitted by `vgam`, the chosen baseline category is `out`, while in model fitted by `nnet`, the chosen baseline category is `b1`. Considering this difference, we can notice that two smaller models actually give identical coefficients.
- However, the fitting results of the larger model by two implementations completely disagree. Also, the estimated standard errors of coefficients of smaller models are slightly different. This is probably caused by different optimization algorithms. `vgam` uses IRLS, while `nnet` uses BFGS. We can see this from the number of iterations of two models.
- The fitting time of `nnet` is much longer than `vgam` for large model, while for smaller model `nnet` is faster. This on the other hand may be caused by different underlying structure of two packages. `nnet` uses neural network to represent the model, of which the number of learnable parameters will grow at a faster speed as the model gets larger.

Interesting findings(according to summary table of `mod_vgam`):

- Note that the estimated coefficients of `launch_speed` for HR has considerably greater magnitude than that of 1B, 2B, 3B. This probably means that launch speed is more crucial for a player to hit a HR. That is to say, good spray angle and launch angle may be sufficient to hit 1B, 2B and 3B, but for a HR, great launch speed is a must.
- For higher order(greater than 2) polynomials of `spray_angle`, only few of them are significant. In particular, for HR, none of the higher order polynomials of `spary_angle` is significant for HR. On the other hand, all coefficients of even order polynomials of `spray_angle` are significant for 2B and 3B. Our conclusion from last bullet point is supported by these findings: good angles are important for 2B and 3B, but for HR, launch speed is the one that makes difference.
- More interestingly, as mentioned in the last point, basically only even order polynomials of `spray_angle` display significance. Does this mean that actually only the magnitude is needed for `spray angle`? The reason we include `spray_angle` up to order of 6 is that we want to account for the positions of opponent catchers. But this model is probably saying that this is not necessary.

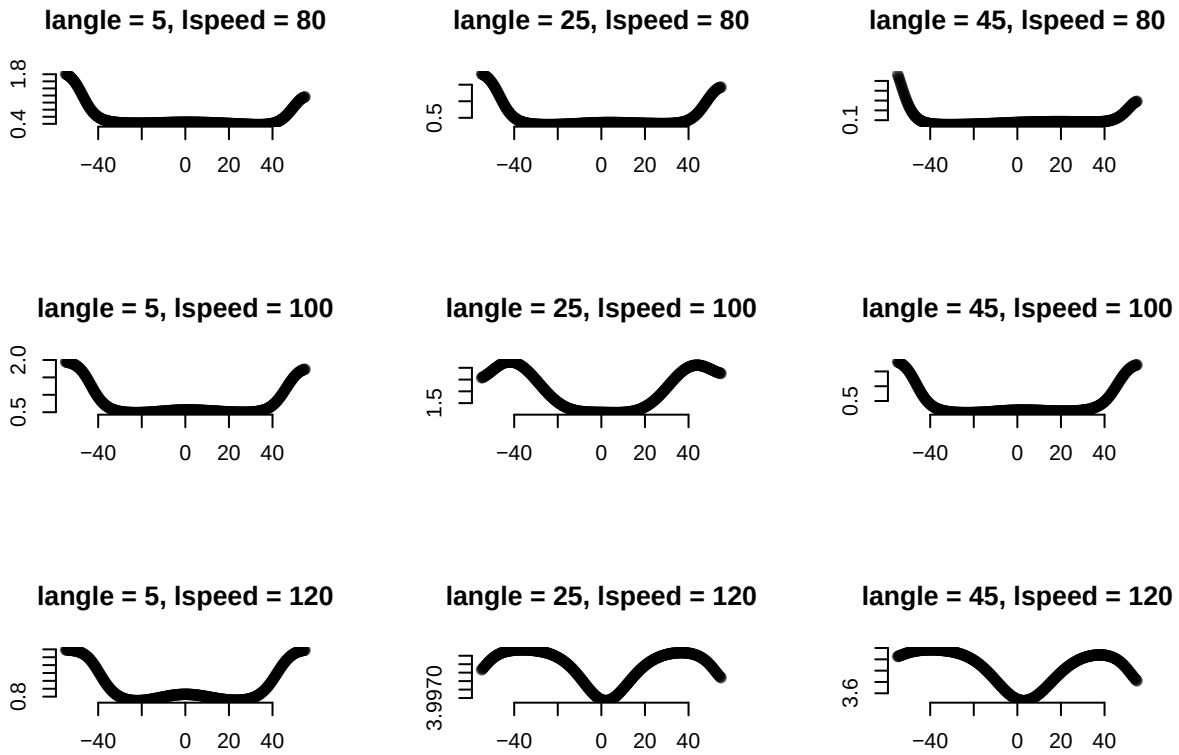
(b) Calculate the weighted hits as success metric.

```
weighted_hits <- function(lspeed, langle) {
  ## obtain predictions
  new_data = data.frame(spray_angle = seq(-55,55, by = 0.1),
             launch_speed = lspeed,
             launch_angle = langle)
  pred <- predict(mod_vgam, newdata = new_data, type = 'response')
  weighted_hits <- pred[,1] + 2*pred[,2] + 3*pred[,3] + 4*pred[,4]

  ## Make plot
  plot.new()
  title(paste0( "langle = ",langle, ", lspeed = ", lspeed))
  plot.window(xlim = c(-55,55), ylim = c(min(weighted_hits), max(weighted_hits)))
  points(new_data$spray_angle, weighted_hits, pch = 19, col = rgb(0,0,0,alpha=0.2))
  axis(1)
  axis(2)
}
```

Plot weighted hits against spray angle, under different combinations of launch speed and launch angle.

```
par(mfrow = c(3,3))
weighted_hits(80, 5)
weighted_hits(80, 25)
weighted_hits(80, 45)
```

```
weighted_hits(100, 5)
weighted_hits(100, 25)
weighted_hits(100, 45)
weighted_hits(120, 5)
weighted_hits(120, 25)
weighted_hits(120, 45)
```



According to plots above, we recommend a player to - hit the ball as hard as possible(so launch speed will be fast), - keep the launch angle in the positive middle range, i.e. 20-30 degrees - and try to hit the ball around side line.

Problem 2: Comment on the differences between the `vglm` and `polr` implementations in the happiness and trauma example.

We first load in the happiness data.

```
#gsetwd('/Users/diptarka/Documents/GitHub/stat528resources/notes/5_multinomial') ## set your own WD
happiness <- read.table("../notes/5_multinomial/happiness.txt", header=TRUE)
head(happiness)
```

```
##   race trauma happy
## 1    0      0     1
## 2    0      0     1
## 3    0      0     1
## 4    0      0     1
## 5    0      0     1
## 6    0      0     1
```

We first use the `vglm` function to fit a proportional odds model (cumulative logit):

```
library(VGAM)
mod <- vglm(happy ~ trauma + race, family=propodds(reverse=FALSE),
data=happiness)
summary(mod)
```

```
##
## Call:
## vglm(formula = happy ~ trauma + race, family = propodds(reverse = FALSE),
##     data = happiness)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept):1  -0.5181     0.3382  -1.532  0.12552
## (Intercept):2   3.4006     0.5648   6.021 1.74e-09 ***
## trauma         -0.4056     0.1809  -2.242  0.02493 *
## race           -2.0361     0.6911  -2.946  0.00322 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: logitlink(P[Y<=1]), logitlink(P[Y<=2])
##
## Residual deviance: 148.407 on 190 degrees of freedom
##
## Log-likelihood: -74.2035 on 190 degrees of freedom
##
## Number of Fisher scoring iterations: 5
##
## No Hauck-Donner effect found in any of the estimates
##
##
## Exponentiated coefficients:
##    trauma      race
## 0.6665934 0.1305338
```

We now do the same using the `polr` function

```
library(MASS)
mod2 <- polr(factor(happy) ~ trauma + race, data=happiness)
summary(mod2)
```

```
## Call:
## polr(formula = factor(happy) ~ trauma + race, data = happiness)
##
## Coefficients:
##         Value Std. Error t value
## trauma 0.4056     0.1830   2.216
## race   2.0361     0.6859   2.968
##
## Intercepts:
##     Value   Std. Error t value
## 1|2 -0.5181  0.3400    -1.5238
## 2|3  3.4006  0.5680     5.9872
##
## Residual Deviance: 148.407
```

```
## AIC: 156.407
```

Comparing the two functions:

Again we can note that the `vglm` function gives more information. The estimates for the intercepts and their standard deviations are exactly the same from both models. Note that the estimates for the coefficients have their signs flipped in the two models while their values are exactly the same. This is because the `vglm` function models the data as $G^{-1}(P(Y \leq j|x)) = \alpha_j + x^T\beta$

whereas the `polr` function considers the model $G^{-1}(P(Y \leq j|x)) = \alpha_j - x^T\beta$

and thus the signs of the β estimates are flipped. Also note that the former function gives the z-values whereas the latter gives the t values. They serve the same purpose of testing for significance. The values of the residual deviance are also the same from both the models. Note that if we use the log-likelihood value that is given in the summary of the `vglm` function to obtain the AIC we would get the value of AIC in the `polr` function. Finally again `vglm` enjoys the advantage of printing out the names of the linear predictors which makes it easier for the user to keep track of the model the function is using.

Problem 3: A study of factors affecting alcohol consumption measures the response variable with the scale (abstinence, a drink a day or less, more than one drink a day). For a comparison of two groups while adjusting for relevant covariates, the researchers hypothesize that the two groups will have about the same prevalence of abstinence, but that one group will have a considerably higher proportion who have more than one drink a day. Even though the response variable is ordinal, explain why a cumulative logit model with proportional odds structure may be inappropriate for this study.

Solution 3:

The study has two groups let us assume they are G_1 and G_2.

Now the researchers hypothesize that the two groups have about the same prevalance of abstinence which means that

$$\pi_1(G_1) \approx \pi_1(G_2)$$

Also they hypothesize that one group will have a considerably higher proportion who have more than one drink a day i.e.

$$\pi_3(G_1) > \pi_3(G_2)$$

Now in the cumulative logit model with proportional odds structure we have the property that

$$logit(P(Y \leq j|G_1)) - logit(P(Y \leq j|G_2)) \text{ does not depend on j}$$

For j = 1 we have

$$logit(P(Y \leq 1|G_1)) - logit(P(Y \leq 1|G_2)) = log\left(\frac{\pi_1(G_1)}{1-\pi_1(G_1)}\right) - log\left(\frac{\pi_1(G_2)}{1-\pi_1(G_2)}\right) \approx 0$$

But for j = 3

$$logit(P(Y \leq 3|G_1)) - logit(P(Y \leq 3|G_2)) = log\left(\frac{\pi_3(G_1)}{1-\pi_3(G_1)}\right) - log\left(\frac{\pi_3(G_2)}{1-\pi_3(G_2)}\right) \neq 0$$

Thus the cumulative logit model with proportional odds structure is not appropriate here. ###

Problem 4: Refer to the table below:

		Belief in Heaven		
Race	Gender	Yes	Unsure	No
Black	Female	88	16	2
	Male	54	17	5
White	Female	397	141	24
	Male	235	189	39

(a) Fit the model

$$\log(\pi_j/\pi_3) = \alpha_j + \beta_j^G x_1 + \beta_j^R x_2, \qquad j = 1, 2.$$

(b) Find the prediction equation for $\log(\pi_1/\pi_2)$.

(c) Treating belief in heaven as ordinal fit and interpret a cumulative logit model and a cumulative probit model. Compare results and state interpretations in each case.

Solution 4

(a) Want to fit the model

$$\log(\pi_j/\pi_3) = \alpha_j + \beta_j^G x_1 + \beta_j^R x_2$$

We consider belief in heaven to be a nominal variable and create a dataset which reflects the table

```
nominal_heaven = data.frame("Race" = c("Black","Black","White","White"),"Gender" =c("Female","Male","Fe
nominal_heaven
```

```
##    Race Gender Belief_Yes Belief_Unsure Belief_No
## 1 Black Female         88            16         2
## 2 Black   Male         54            17         5
## 3 White Female        397           141        24
## 4 White   Male        235           189        39
```

Now that we have the data we fit the baseline-category logistic model for the multinomial response data

```
library(VGAM)
mod_nom <- vglm(cbind(Belief_Yes, Belief_Unsure, Belief_No) ~ Race + Gender,
family=multinomial, data=nominal_heaven)
summary(mod_nom)
```

```
##
## Call:
## vglm(formula = cbind(Belief_Yes, Belief_Unsure, Belief_No) ~
##     Race + Gender, family = multinomial, data = nominal_heaven)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept):1   3.5318     0.4203   8.403  < 2e-16 ***
```

```
## (Intercept):2   1.7026     0.4483   3.798 0.000146 ***
## RaceWhite:1    -0.7031     0.4113  -1.710 0.087349 .
## RaceWhite:2     0.1056     0.4384   0.241 0.809613
## GenderMale:1   -1.0435     0.2587  -4.034 5.48e-05 ***
## GenderMale:2   -0.2545     0.2691  -0.946 0.344277
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: log(mu[,1]/mu[,3]), log(mu[,2]/mu[,3])
##
## Residual deviance: 0.6903 on 2 degrees of freedom
##
## Log-likelihood: -19.4657 on 2 degrees of freedom
##
## Number of Fisher scoring iterations: 3
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## '(Intercept):1'
##
##
## Reference group is level  3  of the response
```

Thus $\alpha_1 = 3.5318, \alpha_2 = 1.7026, \beta_1^G = -1.0435, \beta_2^G = -0.2545, \beta_1^R = -0.7031, \beta_2^R = 0.1056$

(b) Find the prediction equation for $\log(\pi_1/\pi_2)$

We use the fact that

$$\log(\pi_1/\pi_2) = \log(\pi_1/\pi_3) - \log(\pi_1/\pi_3)$$

Thus the prediction equation for $\log(\pi_1/\pi_2)$ is

$$\begin{aligned} log(\pi_1/\pi_2) &= log(\pi_1/\pi_3) - log(\pi_1/\pi_3) \\ &= \alpha_1 - \alpha_2 + (\beta_1^G - \beta_2^G)x_1 + (\beta_1^R - \beta_2^R)x_2 \\ &= (3.5318 - 1.7026) + (-1.0435 + 0.2545)x_1 + (-0.7031 - 0.1056)x_2 \\ &= 1.8292 - 0.789x_1 - 0.8087x_2 \end{aligned}$$

(c) Treating belief in heaven as ordinal fit and interpret a cumulative logit model and a cumulative probit model. Compare results and state interpretations in each case.

We first create the data treating "Belief in heaven" as an ordinal variable.

```
rep.row<-function(x,n){
   matrix(rep(x,each=n),nrow=n)
}

data_heaven = rbind(rep.row(c("B","F",1),88),rep.row(c("B","F",2),16),rep.row(c("B","F",3),2),rep.row(c


colnames(data_heaven) = c("Race","Gender","Belief")

data_heaven = as.data.frame(data_heaven)
head(data_heaven)
```

```
##   Race Gender Belief
## 1    B      F      1
```

```
## 2    B      F      1
## 3    B      F      1
## 4    B      F      1
## 5    B      F      1
## 6    B      F      1
```

```
data_heaven$Race = factor(data_heaven$Race)
data_heaven$Gender = factor(data_heaven$Gender)
data_heaven$Belief = factor(data_heaven$Belief,ordered = T )
```

We fit the cumulative logit model i.e.

$G^{-1}(P(Y \leq j|x)) = \alpha_j - x^T\beta$ {where G^{-1} is the logit function}

```
mod <- vglm(Belief ~ Race+Gender, family=propodds(reverse=FALSE),
data=data_heaven)
summary(mod)
```

```
##
## Call:
## vglm(formula = Belief ~ Race + Gender, family = propodds(reverse = FALSE),
##     data = data_heaven)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept):1   1.6379     0.1910   8.576  < 2e-16 ***
## (Intercept):2   3.9138     0.2259  17.329  < 2e-16 ***
## RaceW          -0.7685     0.1911  -4.021 5.80e-05 ***
## GenderM        -0.8217     0.1215  -6.765 1.33e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: logitlink(P[Y<=1]), logitlink(P[Y<=2])
##
## Residual deviance: 1893.45 on 2410 degrees of freedom
##
## Log-likelihood: -946.7251 on 2410 degrees of freedom
##
## Number of Fisher scoring iterations: 4
##
## No Hauck-Donner effect found in any of the estimates
##
##
## Exponentiated coefficients:
##     RaceW   GenderM
## 0.4637078 0.4396769
```

We fit the cumulative probit model i.e.

$G^{-1}(P(Y \leq j|x)) = \alpha_j - x^T\beta$ {where G^{-1} is the probit function}

```
mod_prob <- vglm(Belief ~ Race+Gender, family=cumulative(link="probitlink",parallel=TRUE),
data=data_heaven)
summary(mod_prob)
```

```
##
## Call:
## vglm(formula = Belief ~ Race + Gender, family = cumulative(link = "probitlink",
```

```
##     parallel = TRUE), data = data_heaven)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept):1  0.95883    0.10703   8.959  < 2e-16 ***
## (Intercept):2  2.21185    0.12050  18.356  < 2e-16 ***
## RaceW         -0.43055    0.10809  -3.983 6.79e-05 ***
## GenderM       -0.47935    0.07147  -6.707 1.99e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: probitlink(P[Y<=1]), probitlink(P[Y<=2])
##
## Residual deviance: 1896.014 on 2410 degrees of freedom
##
## Log-likelihood: -948.0071 on 2410 degrees of freedom
##
## Number of Fisher scoring iterations: 4
##
## No Hauck-Donner effect found in any of the estimates
##
##
## Exponentiated coefficients:
##     RaceW   GenderM
## 0.6501506 0.6191833
```

We can see that the log-likelihood value (and thus the AIC) and the residual deviance for both the models is approximately the same and thus both the models have a similar performance despite the underlying models being different. Also note that the signs of the estimates of the coefficients are also the same in both models which means that the variables have the same relationship with the response in both models.

Problem 5: Suppose that you have a coin that when flipped has a probability $0 < p < 1$ of landing heads, and that we know nothing about p. Suppose that you flip the coin four times and all four flips resulted in heads. Derive the MLE of p and the MLE of $\text{Var}(Y_i)$ under the standard Bernoulli model. Now, for some error tolerance $0 < \alpha < 1$, derive a valid one-sided confidence interval for p making use of the statement $\mathbb{P}\left(\sum_{i=1}^{4} y_i = 4\right)$.

Solution 5:

The log-likelihood of the Bernoulli distribution is

$$l(p|y_i) = \log p \sum y_i + (n - \sum y_i) \log(1 - p)$$

To find MLE,

$$\log \hat{p} \sum y_i + (n - \sum y_i) \log(1 - \hat{p}) = 0$$

$$\implies \frac{\sum y_i}{\hat{p}} - \frac{(n - \sum y_i)}{1 - \hat{p}} = 0$$

$$\implies \hat{p} = \frac{1}{n} \sum y_i$$

Since we get four heads in four tosses $\sum y_i = 4$ and n = 4.

$$\implies \hat{p} = 1$$

Now $Var(Y_i) = p(1-p) \implies \hat{Var}(Y_i) = \hat{p}(1-\hat{p})$

$$\implies \hat{Var}(Y_i) = 0$$

This means that the space of γ such that γ belongs to the null space of $Var(Y_i)$ is $\mathbb{R}$

Thus the lower boundary of the Confidence interval for p is

$$\min_{P_p(\sum y_i=4)\geq\alpha} p$$

Now

$$P_p(\sum y_i = 4) \geq \alpha \implies p^{\sum y_i}(1-p)^{n-\sum y_i} \geq \alpha$$
$$\implies p^4(1-p)^{4-4} \geq \alpha$$

Now since the log likelihood is a concave function the min p which satisfies the constraint will satisfy

$$p^4 = \alpha$$
$$\implies p = \alpha^{1/4}$$

Thus the $100(1-\alpha)\%$ one sided confidence interval is

$$CI = (\alpha^{1/4}, 1)$$

Problem 6: Complete the following with respect to the `endometrial` example:

(a) Write your own Fisher scoring algorithm for this example. Argue that $\hat{\beta}$ diverges in some sense as the iterations of your algorithm increase.

(b) Show that the log likelihood has an asymptote in $\|\beta\|$.

(c) Code the likelihood function for this dataset, pick a value of $\tilde{\beta}$ that is in the LCM, find an eigenvector of estimated Fisher information η such that the likelihood asymptotes, and then show that the likelihood asymptotes in $\tilde{\beta} + s\eta$ as $s \to \infty$.

(d) Explain why the likelihood asymptotes in $\tilde{\beta} + s\eta$ as $s \to \infty$.

Solution 6:

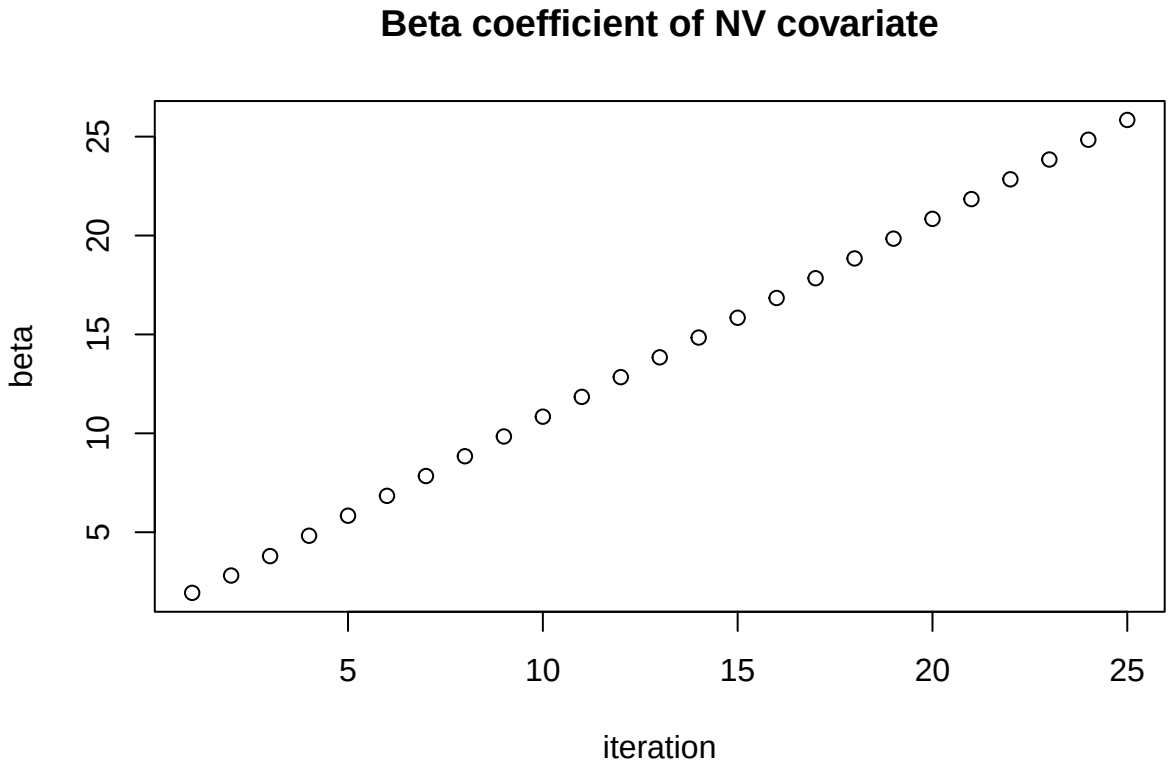
(a) Write your own Fisher scoring algorithm for this example. Argue that $\hat{\beta}$ diverges in some sense as the iterations of your algorithm increase.

```
#Creating a function to compute the log-likelihood
log_lik = function(X,Y,beta)
{
  t(Y)%*%X%*%beta - sum(log(1 + exp(X%*%beta)))
}
```

```
library(enrichwith)
data(endometrial)

#Creating the model matrix
X = model.matrix(HG ~ .,data = endometrial)
n = nrow(X)
p = ncol(X)
Y = endometrial$HG

# Initializing the beta
beta = matrix(rep(0,p))
beta_list = NULL
#Running the Fisher scoring iterations
for(t in 1:25)
{
  pi =  exp(X%*%beta)/(1+exp(X%*%beta))
  W = diag(c(pi*(1-pi)))
  beta = beta + solve(t(X) %*% W %*% X)%*%t(X)%*%(Y - pi)
  beta_list = cbind(beta_list,beta)
}
```

To show that the $\hat{\beta}$ diverges in some sense we plot the $\hat{\beta}$ corresponding to the NV variable over the iterations

```
plot(1:25,beta_list[2,],main = "Beta coefficient of NV covariate",xlab = "iteration",ylab = "beta")
```

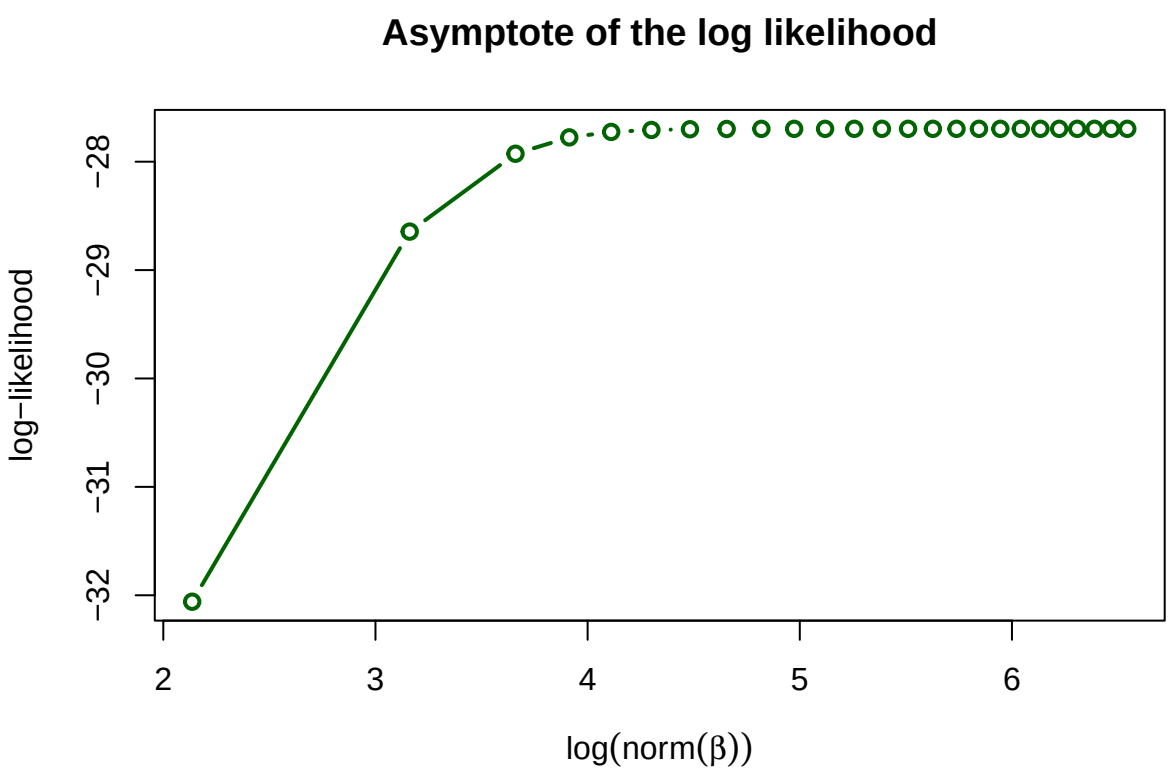


Clearly it diverges.

(b) Show that the log likelihood has an asymptote in $\|\beta\|$

```
#Computing the log likelihood for each of the beta values that we get
yvalues = apply(beta_list,2,function(t) log_lik(X,Y,t))
```

```r
#Computing the norm of the beta values
xvalues = log(apply(beta_list,2,function(t) sum(t^2)))

#Plotting
plot(xvalues,yvalues,ty = "b",lwd = 2,col = "darkgreen", main = "Asymptote of the log likelihood",xlab =
```



Clearly it aysmptotes in $\|\beta\|$

(c) Code the likelihood function for this dataset, pick a value of $\hat{\beta}$ that is in the LCM, extract the null eigen vector of estimated Fisher information η, and then show that the likelihood asymptotes in $\beta + s\eta$ as $s \to \infty$.

```r
#Creating a logistic model model for the endometrial data
mod <- glm(HG ~ ., family = "binomial",
control = list(maxit = 25, epsilon = 1e-100),data = endometrial)
summary(mod)
```

```
##
## Call:
## glm(formula = HG ~ ., family = "binomial", data = endometrial,
##     control = list(maxit = 25, epsilon = 1e-100))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5014  -0.6411  -0.2943   0.0000   2.7278
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.305e+00  1.637e+00   2.629 0.008563 **
## NV           2.619e+01  9.368e+04   0.000 0.999777
```

```
## PI          -4.218e-02  4.433e-02  -0.952 0.341333
## EH          -2.903e+00  8.456e-01  -3.433 0.000597 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 104.903  on 78  degrees of freedom
## Residual deviance:  55.393  on 75  degrees of freedom
## AIC: 63.393
##
## Number of Fisher Scoring iterations: 25
```

We can see that the Fisher Scoring algorithm goes through all the 25 iterations (which was specified as the max number of iterations). We have seen from part (b) that after 25 fisher scoring iteration the likelihood has already converged and thus the null space of the fisher scoring matrix obtained using the parameters of the OM at this stage estimate the null space of the fisher scoring matrix of the LCM well.

```
#Computing the beta belonging to the LCM
beta_tilde = mod$coefficients
beta_tilde
```

```
## (Intercept)          NV          PI          EH
##   4.3045178  26.1855559  -0.0421834  -2.9026056
```

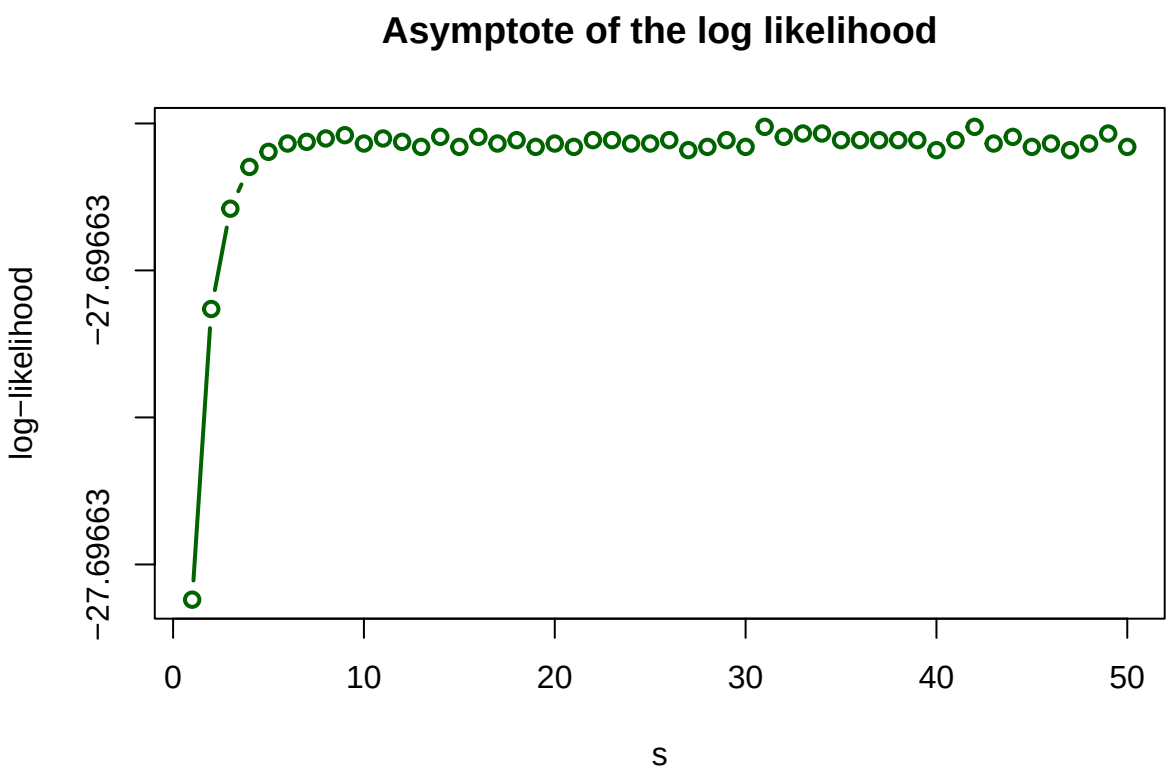
```
#Finding the null eigenvector of the Fisher scoring matrix
invFI <- vcov(mod)
FI <- solve(invFI)
eig = eigen(FI)
eig
```

```
## eigen() decomposition
## $values
## [1] 3.068079e+03 7.175314e+00 3.069389e-01 1.139389e-10
##
## $vectors
##               [,1]          [,2]          [,3]          [,4]
## [1,] -4.914836e-02  4.272808e-01  9.027821e-01  1.282627e-11
## [2,] -9.759971e-13 -1.266181e-11  2.014712e-11 -1.000000e+00
## [3,] -9.962646e-01 -8.522654e-02 -1.390049e-02  1.771594e-12
## [4,] -7.100158e-02  9.000931e-01 -4.298734e-01 -1.998847e-11
```

Clearly the last eigen value is 0 thus the corresponding eigen vector belongs to the null space of the Fisher information matrix. We can see that the null vector is $-e_2$ i.e $(0, -1, 0, 0)$. Thus we can consider $\eta = (0, 1, 0, 0)$

```
eta = -eig$vectors[,4]

#Considering s from 1 to 50
s = c(1:50)

#Creating a list of beta of the form beta_tilde + s*eta
vals = matrix(unlist(lapply(s,function(t) beta_tilde + t*eta)),nrow = 4)

#Computing the loglik values for these beta
yvalues = apply(vals,2,function(t) log_lik(X,Y,t))

#Plotting the loglik values against s
```

```
plot(s,yvalues,ty = "b",lwd = 2,col = "darkgreen", main = "Asymptote of the log likelihood",xlab = "s",y
```



Again clearly it asymptotes as $s \to \infty$

(d) LCM is the OM with lost dimensions. In other words, the sub-model canonical statistics of LCM is restricted to a hyper-plane in the support set. Since sub-model canonical statistics of LCM is constrained to the hyper-plane, it can not vary along the direction that is orthogonal to the hyper-plane, hence vectors η that are orthogonal to the hyper-plane span the null space of the Fisher information matrix of LCM. Recall that null space of Fisher information matrix can be approximated by Fisher information matrix of OM, so eigen vectors of OM are approximately orthogonal to the hyper-plane. Therefore, $\tilde{\beta} + s\eta$ is gradually moving $\tilde{\beta}$ towards to the orthogonal direction of hyper-plane. But sub-model canonical statistics can not vary along that direction. So, as $s \to \infty$, sub-model canonical statistics will gradually stop moving, leading to asymptote of likelihood.

Problem 7: Summarise the Firth approach mentioned in Section 7.4.7 and 7.4.8 of Agresti. Compare and contrast the Firth approach with the direct MLE approach outlined in the complete separation notes. What are the strengths and weaknesses of each approach?

Solution 7:

In the Firth approach we penalise the log-likelihood function to ensure that the MLE always exists. The penalized log-likelihood function utilizes the determinant of the information matrix $\mathcal{J}$,

$$L^*(\beta) = L(\beta) + \frac{1}{2}\log|\mathcal{J}|$$

It turns out that this approach coincides with the Bayesian approach with Jeffrey's prior.

Comparison:

- Even under the case of complete and quasi-separation, Firth's approach can still give finite and unique estimates of coefficients with decent certainty. On the other hand, the direct MLE approach(OM and LCM) can only give unbouded estimate intervals for separable variables. That being said, both approaches provide estimates for all mean-value parameters.
- However, the introduction of penalization makes Firth's approach tend to give larger estimated coefficients than MLE, leading to inaccurate estimation under certain cases. As for the direct MLE approach, estimated coefficients of non-separable covariates are generally reliable.
- Firth's approach uses second order approximation, hence can reduce asymptotic bias to order $1/n^2$, while direct MLE only uses first order approximation, which has asymptotic bias of order $1/n$. The observation of data separation may indicate that asymptotic approximation may be suspect, since $\hat{p}_i \in \{0, 1\}$ for some observations i.
- Note that Firth's approach actually falls into the category of Bayesian approaches, which come with the problem of choosing priors. It is shown that different priors have different merits, and can all give reasonable results. These facts make such approaches less objective. However, the frequentist approach, direct MLE, is always objective.
- It is interesting to note that there is debate among Bayesian authors about the appropriateness of Firth's approach when data separation is observed, and that switching to a Bayesian modeling framework itself is usually presented as an ad hoc fix to a breakdown in traditional MLE based inference. Different justifiable priors can yield different inferences about modeling variables as we saw in class.

Problem 8: Use `glmdr` software to analyze the `catrec.txt` data using Poisson regression. Specifically, fit a third order model and provide confidence intervals for all mean-value parameter estimates, both one-sided intervals for responses that are constrained on the boundary and two-sided intervals for responses that are unconstrained. Also verify that the third order model is appropriate.

First, we see that a third order model fits this data well.

```
library(glmdr)
dat <- read.table("catrec.txt", header = TRUE)
head(dat)
```

```
##   v1 v2 v3 v4 v5 v6 v7 y
## 1  0  0  0  0  0  0  0 0
## 2  1  0  0  0  0  0  0 8
## 3  0  1  0  0  0  0  0 7
## 4  1  1  0  0  0  0  0 8
## 5  0  0  1  0  0  0  0 9
## 6  1  0  1  0  0  0  0 7
```

```
m2 <- glm(y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^2,
          family = "poisson", data = dat)
m3 <- glm(y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3,
          family = "poisson", data = dat)
anova(m2, m3, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^2
## Model 2: y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        99    191.629
## 2        64     31.291 35   160.34 < 2.2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We fit the third order model using glmdr software and obtain one-sided confidence intervals for responses that are constrained on the boundary.

```
mod <- glmdr(y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3,
          family = "poisson", data = dat)
inference(mod)
```

```
##    lower      upper
## 1      0 0.28630975
## 2      0 0.14082947
## 3      0 0.21996698
## 4      0 0.42095569
## 5      0 0.08946242
## 6      0 0.09376644
## 7      0 0.19302341
## 8      0 0.28869769
## 9      0 0.10631113
## 10     0 0.11415033
## 11     0 0.09128766
## 12     0 0.26461097
## 13     0 0.06669488
## 14     0 0.15477613
## 15     0 0.14096916
## 16     0 0.32392015
```

Standard two-sided confidence intervals for mean-value parameters corresponding to responses that are unconstrained are provided below.

```
mod_lcm <- glm(y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3,
          family = "poisson", data = dat[mod$linearity, ] )
p_lcm <- predict(mod_lcm, type = "response", se.fit = TRUE)
CIs <- cbind(p_lcm$fit + qnorm(0.025) * p_lcm$se.fit,
             p_lcm$fit + qnorm(0.975) * p_lcm$se.fit)
head(CIs)
```

```
##       [,1]      [,2]
## 2 4.122579 14.297167
## 3 1.655321  8.514389
## 4 3.351595 12.663846
## 5 3.897364 14.215000
## 6 1.815964  8.523987
## 7 4.754609 14.830931
```

Problem 9: Do you think that the `glm` function can be used to provide an appropriate value of the Akaike information criterion (AIC) when complete separation or quasi-complete separation exists? Why or why not?

Yes, glm software can be used to calculate an appropriate value of the Akaike information criterion (AIC) when complete separation or quasi-complete separation exists. The reason for this is that the Fisher scoring algorithm implemented by glm nearly maximizes the log likelihood even though submodel canonical parameter estimates are infinitely far away from their values in the completion of the exponential family.

An acceptable counter answer is: When complete separation exists, the MLE is at the boundary of the closure of its parameter space, MLEs are "at infinity" and the log-likelihood will converges and asymptotes to 0.

Thus the the AIC we get from glm asymptotes 2K. In that case, we will always get a smaller AIC by fitting a smaller model, thus we could not identify which model is better or more adeqate by using AIC from glm function. Although it should be noted that this answer indicates a problem with AIC, and not whether glm can be used to calculate an appropriate value of AIC.