Homework 4: Multinomial regression and data separation

TA : Arjama Das

Due: 04/07 at 11:59 PM

This homework set will cover problems concerning multinomial regression models and data separation. Point totals for specific problems are given, and 10 points will be reserved for correct submission of the homework assignment and professionalism.

Problem 1 [20 points]: Do the following regarding the Sabermetrics dataset (bball.csv),

- **part a** [5 points]: Fit a similar multinomial regression model to that fit to the Sabermterics data set in the course notes using the `multinom` function in the `nnet` package. Comment on the similarities and differences between the `nnet` and `VGAM` fits. Report interesting conclusions using either implementation.
- **part b** [15 points]: 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. You can use either the `nnet` or `VGAM` package for this question.

Solution:

- part a

```
library(VGAM)
library(nnet)
library(tidyverse)
setwd('/Users/arjam/Downloads') ## set your own WD
bball <- read.csv("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))
```

```
##    user  system elapsed
##    5.92    0.96    7.05
```

```
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
##    5.28    0.04    5.43
```

```
(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
##   14.52    1.40   16.17
```

```
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
##  114.25    0.32  116.41
```

```
(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  -0.0018891721 -0.024255607 -0.019049947  0.001527060      -0.003783049
## b3  -0.0008096977 -0.052739336 -0.003582374 -0.001547063      -0.004705272
## b4  -0.0029719995 -0.176115801 -0.028951744  0.001680411      -0.020233766
## out  0.0029097866  0.003629114  0.020215042  0.005739153       0.001484599
##     I(spray_angle^2) I(spray_angle^3) I(spray_angle^4) I(spray_angle^5)
## b2     -3.242103e-03     6.655960e-06     3.385413e-06    -2.551384e-09
## b3     -5.099407e-03    -1.743621e-05     4.660566e-06     8.468765e-09
## b4      7.844742e-05     2.240583e-05     9.616818e-07    -7.732372e-09
## out     2.037512e-04     6.540386e-06    -2.880517e-08    -7.583519e-10
##     I(spray_angle^6) I(spray_angle * launch_angle)
## b2     -6.365877e-10                  1.998358e-04
## b3     -9.736331e-10                 -4.953794e-04
```

```
## b4     -2.435075e-10                -1.231631e-05
## out    -1.394206e-12                -2.008000e-04
##     I(spray_angle * launch_speed) I(launch_angle * launch_speed)
## b2                   -0.0001676955                   0.0021287650
## b3                    0.0003269176                   0.0026887732
## b4                   -0.0002324824                   0.0126033765
## out                  -0.0001049875                  -0.0004168459
##
## Residual Deviance: 70193.49
## AIC: 70297.49
```

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 1 B, 2 B, 3 B. 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 1 B, 2 B and 3 B , 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 3 B , 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.

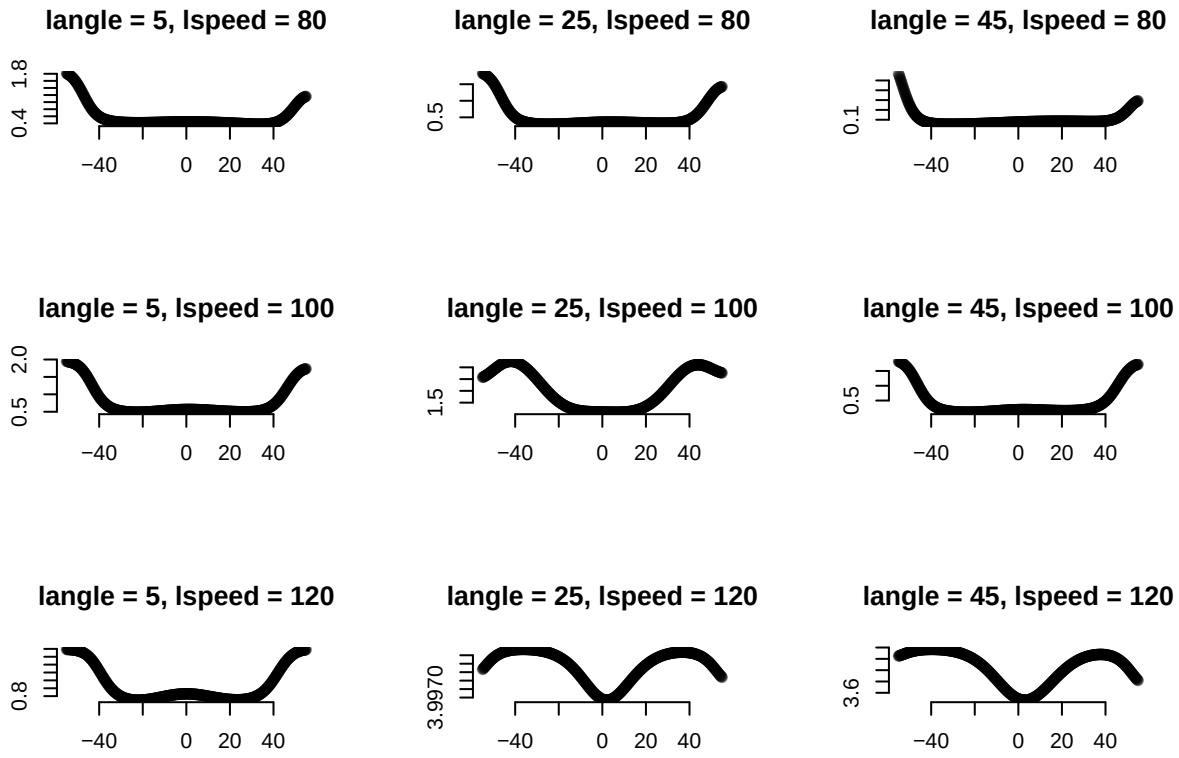
- part b

```
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 [10 points]: 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:

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

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

For j = 1 we have

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

But for j = 3

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

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

Problem 3 [10 points]: 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

- **part a** [4 points]: Fit the model

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

- **part b** [3 points]: Find the prediction equation for $\log(\pi_1/\pi_2)$.
- **part c** [3 points]: 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:

- part a

Want to fit the model

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

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

- part b

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

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

- part c

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(
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. $G^{-1}(P(Y \leq j \mid x)) = \alpha_j - x^T\beta$ {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. $G^{-1}(P(Y \leq j \mid x)) = \alpha_j - x^T\beta$ {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 4 [10 points]: 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:

The log-likelihood of the Bernoulli distribution is

$$l\left(p \mid y_i\right) = \log p \sum y_i + \left(n - \sum y_i\right) \log(1-p)$$

To find MLE,

$$\begin{aligned} \log \hat{p} \sum y_i + \left(n - \sum y_i\right) \log(1-\hat{p}) &= 0 \\ \implies \frac{\sum y_i}{\hat{p}} - \frac{(n - \sum y_i)}{1-\hat{p}} &= 0 \\ \implies \hat{p} &= \frac{1}{n} \sum y_i \end{aligned}$$

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

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

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

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

This means that the space of γ such that γ belongs to the null space of $\text{Var}\left(Y_i\right)$ 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

$$\begin{aligned} P_p\left(\sum y_i = 4\right) \geq \alpha &\implies p^{\sum y_i} (1-p)^{n - \sum y_i} \geq \alpha \\ &\implies p^4 (1-p)^{4-4} \geq \alpha \end{aligned}$$

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

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

Problem 5 [20 points]: Complete the following with respect to the `endometrial` example:

- **part a** [5 points]: Write your own Fisher scoring algorithm for this example. Argue that $\hat{\beta}$ diverges in some sense as the iterations of your algorithm increase.
- **part b** [5 points]: Show that the log likelihood has an asymptote in $\|\beta\|$.
- **part c** [5 points]: 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 $s \to \infty$.
- **part d** [5 points]: Explain why the likelihood asymptotes in $\tilde{\beta} + s\eta$ as $s \to \infty$.

Solution:

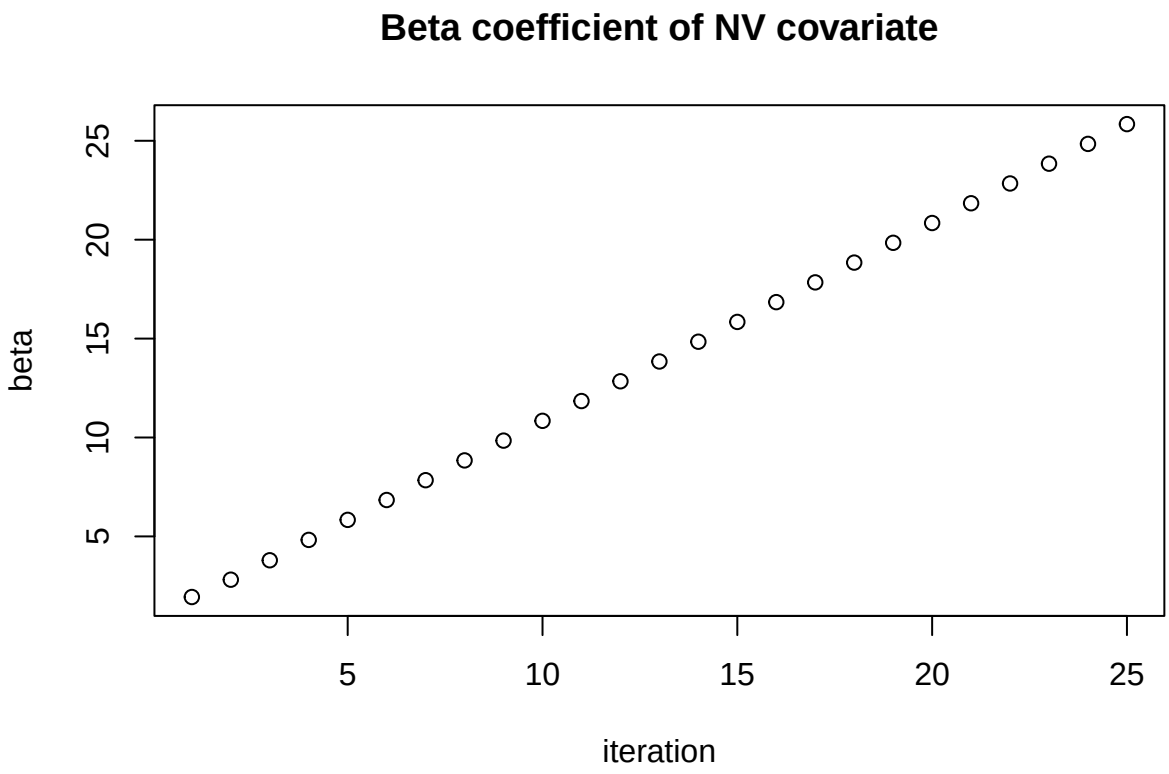
- part a

```
#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

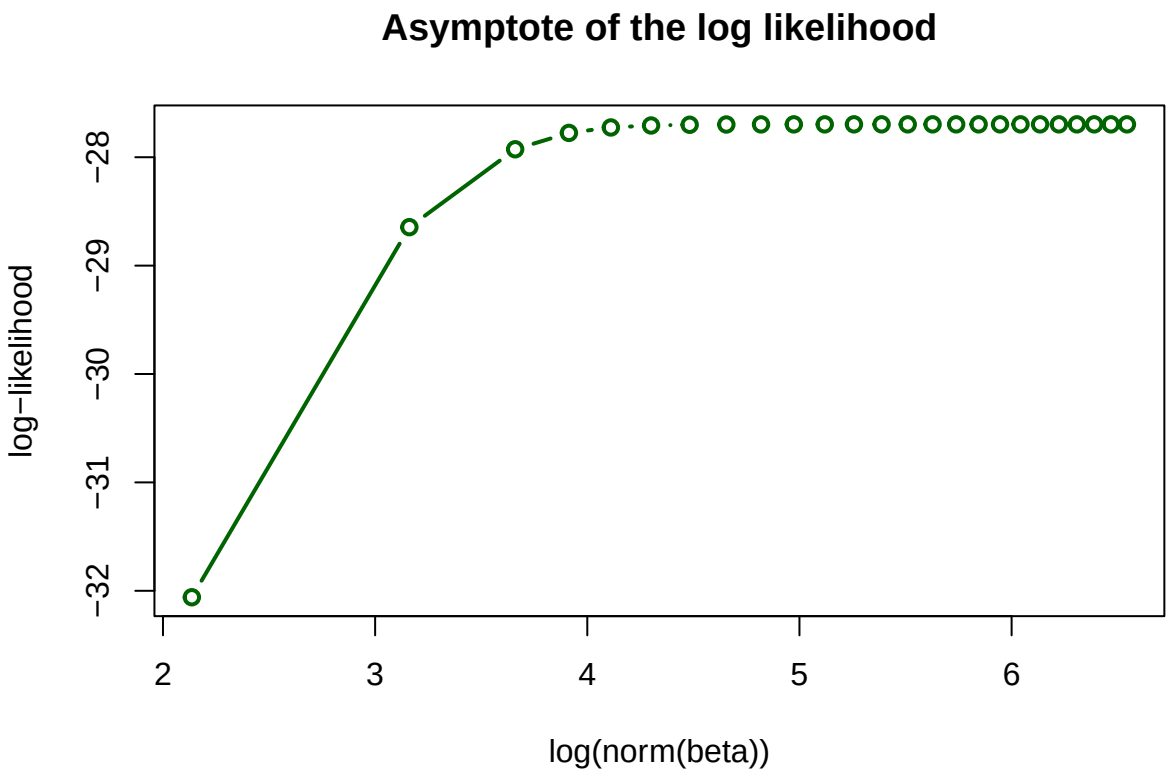
```r
plot(1:25,beta_list[2,],main = "Beta coefficient of NV covariate",xlab = "iteration",ylab = "beta")
```



Clearly it diverges.

- part b

```r
#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))
#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 $\|\beta\|$

- part c

```
#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
```

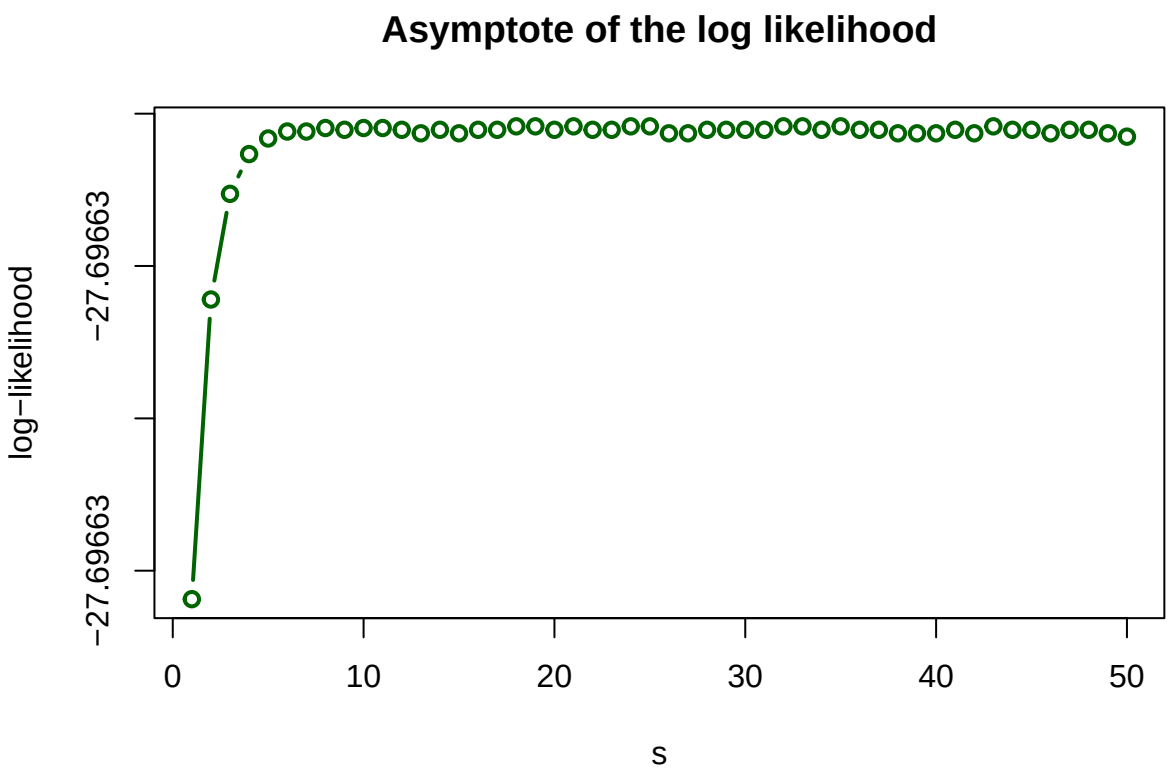
```
## (Intercept)          NV          PI          EH
##   4.3045178  26.1855559  -0.0421834  -2.9026056
```

```
#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 1.133857e-10
##
## $vectors
##               [,1]          [,2]          [,3]          [,4]
## [1,] -4.914836e-02  4.272808e-01  9.027821e-01  1.288818e-11
## [2,] -9.759971e-13 -1.266833e-11  2.021876e-11 -1.000000e+00
## [3,] -9.962646e-01 -8.522654e-02 -1.390049e-02  1.771028e-12
## [4,] -7.100158e-02  9.000931e-01 -4.298734e-01 -2.002498e-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",
```



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

- part 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 6 [10 points]: 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:

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 unbounded estimate intervals for separable variables.
- 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 bias to order $1/n^2$, while direct MLE only uses first order approximation, which has order $1/n$ bias.
- 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.
- One key point to note here is that in a logistic regression model with success probabilities defined to not include zero or one, it seems that one cannot be in asymptopia when separation is observed. Thus, the asymptotic bias reduction advantage of the Firth approach might not be realized in this setting.

Problem 7 [10 points]: 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 using a likelihood ratio test.

Solution:

```
library(nloptr)
library(mdscore)
library(glmdr)
#Loading the data
data=read.csv('catrec.csv',header = TRUE)
head(data)
```

```
##   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
```

```
#Using glmdr software to fit the 3rd order Poisson regression model
mod = glmdr(y ~ .^3,family="poisson",data)
summary(mod)
```

```
## MLE exists in Barndorff-Nielsen completion
## it is conditional on components of the response
## corresponding to object$linearity == FALSE being
## conditioned on their observed values

## $overview
## NULL
##
## $type
## [1] "lcm"
##
## $summary
##
## Call:
## stats::glm(formula = y ~ .^3, family = "poisson", data = data,
##     subset = c("2", "3", "4", "5", "6", "7", "8", "10", "11",
##     "12", "13", "14", "15", "16", "17", "18", "19", "21", "22",
##     "23", "24", "25", "26", "27", "29", "30", "31", "32", "34",
##     "35", "36", "37", "38", "39", "40", "42", "43", "44", "45",
##     "46", "47", "48", "49", "50", "51", "53", "54", "55", "56",
##     "57", "58", "59", "61", "62", "63", "64", "66", "67", "68",
##     "69", "70", "71", "72", "74", "75", "76", "77", "78", "79",
##     "80", "81", "82", "83", "85", "86", "87", "88", "89", "90",
##     "91", "93", "94", "95", "96", "98", "99", "100", "101", "102",
##     "103", "104", "106", "107", "108", "109", "110", "111", "112",
##     "113", "114", "115", "117", "118", "119", "120", "121", "122",
##     "123", "125", "126", "127", "128"), x = TRUE, y = TRUE)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.63571  -0.30009  -0.02353   0.27258   1.42540
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.150481   0.585423   3.673 0.000239 ***
## v1           0.069795   0.587067   0.119 0.905364
## v2          -0.524215   0.513583  -1.021 0.307396
## v3           0.052966   0.551965   0.096 0.923552
## v4          -0.709525   0.580147  -1.223 0.221326
## v5           0.243002   0.548686   0.443 0.657853
## v6          -1.163256   0.563668  -2.064 0.039044 *
## v7          -0.990704   0.597335  -1.659 0.097208 .
## v1:v2        0.384345   0.543024   0.708 0.479079
## v1:v3       -0.630375   0.570151  -1.106 0.268888
## v1:v4        0.008801   0.511458   0.017 0.986271
## v1:v5       -1.022805   0.570440  -1.793 0.072971 .
## v1:v6        0.540164   0.493879   1.094 0.274079
## v1:v7        0.097178   0.536628   0.181 0.856297
## v2:v3        0.602411   0.437371   1.377 0.168405
## v2:v4        0.748226   0.486811   1.537 0.124295
## v2:v5       -0.068926   0.428100  -0.161 0.872090
## v2:v6        0.297165   0.487409   0.610 0.542071
## v2:v7        0.274198   0.508369   0.539 0.589634
## v3:v4       -0.124465   0.541056  -0.230 0.818060
```

```
## v3:v5       -0.439354   0.468418  -0.938 0.348268
## v3:v6        0.024399   0.530220   0.046 0.963296
## v3:v7       -0.104400   0.556960  -0.187 0.851310
## v4:v5       -0.169421   0.521323  -0.325 0.745194
## v4:v6        0.756513   0.474213   1.595 0.110644
## v4:v7        0.780671   0.500911   1.559 0.119114
## v5:v6        1.245629   0.510770   2.439 0.014739 *
## v5:v7       -0.262620   0.523125  -0.502 0.615652
## v6:v7        0.697014   0.489957   1.423 0.154852
## v1:v2:v3    -0.349902   0.483330  -0.724 0.469102
## v1:v2:v4     0.101569   0.389778   0.261 0.794416
## v1:v2:v5     0.655208   0.493737   1.327 0.184496
## v1:v2:v6    -0.329286   0.390979  -0.842 0.399670
## v1:v2:v7    -0.520368   0.393042  -1.324 0.185520
## v1:v3:v4     0.353292   0.406623   0.869 0.384932
## v1:v3:v5     0.638711   0.484979   1.317 0.187843
## v1:v3:v6     0.352694   0.402715   0.876 0.381143
## v1:v3:v7    -0.001586   0.413554  -0.004 0.996941
## v1:v4:v5     0.664745   0.400212   1.661 0.096717 .
## v1:v4:v6    -0.463885   0.368214  -1.260 0.207732
## v1:v4:v7    -0.342583   0.372009  -0.921 0.357103
## v1:v5:v6     0.044968   0.399958   0.112 0.910481
## v1:v5:v7     0.447641   0.404364   1.107 0.268283
## v1:v6:v7     0.218868   0.371499   0.589 0.555763
## v2:v3:v4    -0.325914   0.404392  -0.806 0.420280
## v2:v3:v5           NA         NA      NA       NA
## v2:v3:v6    -0.247853   0.405621  -0.611 0.541168
## v2:v3:v7     0.028322   0.414520   0.068 0.945527
## v2:v4:v5     0.004655   0.394418   0.012 0.990583
## v2:v4:v6    -0.111152   0.373713  -0.297 0.766141
## v2:v4:v7    -0.148061   0.376692  -0.393 0.694279
## v2:v5:v6    -0.766051   0.394925  -1.940 0.052412 .
## v2:v5:v7     0.075213   0.399004   0.189 0.850482
## v2:v6:v7     0.460826   0.381109   1.209 0.226597
## v3:v4:v5    -0.063494   0.423318  -0.150 0.880771
## v3:v4:v6     0.357746   0.366298   0.977 0.328741
## v3:v4:v7    -0.106368   0.371567  -0.286 0.774672
## v3:v5:v6    -0.234816   0.422424  -0.556 0.578295
## v3:v5:v7     0.804923   0.423843   1.899 0.057550 .
## v3:v6:v7    -0.659090   0.371085  -1.776 0.075714 .
## v4:v5:v6    -0.427957   0.375755  -1.139 0.254734
## v4:v5:v7     0.125167   0.377356   0.332 0.740119
## v4:v6:v7     0.014192   0.370131   0.038 0.969413
## v5:v6:v7    -0.811516   0.377098  -2.152 0.031397 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 156.215  on 111  degrees of freedom
## Residual deviance:  31.291  on  49  degrees of freedom
## AIC: 526.46
##
## Number of Fisher Scoring iterations: 5
```

```
##
##
## $linearity
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
##    14    15    16    17    18    19    20    21    22    23    24    25    26
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##    27    28    29    30    31    32    33    34    35    36    37    38    39
##  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##    40    41    42    43    44    45    46    47    48    49    50    51    52
##  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##    53    54    55    56    57    58    59    60    61    62    63    64    65
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
##    66    67    68    69    70    71    72    73    74    75    76    77    78
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##    79    80    81    82    83    84    85    86    87    88    89    90    91
##  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##    92    93    94    95    96    97    98    99   100   101   102   103   104
## FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   105   106   107   108   109   110   111   112   113   114   115   116   117
## FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
##   118   119   120   121   122   123   124   125   126   127   128
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
##
## attr(,"class")
## [1] "summary.glmdr"
```

```
# One-sided Confidence Intervals
CIs = inference(mod)
CIs
```

```
##    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
```

In above we have used the inference function to obtain one-sided confidence intervals for mean-value parameters corresponding to components that are constrained on the boundary. Now we shall show the two sided intervals for responses that are unconstrained.

```r
mod1=glm(y ~ . ^3,family="poisson",data=data, x=TRUE,y=TRUE)
mod2=update(mod1,subset=mod$linearity)
summary(mod2)
```

```
##
## Call:
## glm(formula = y ~ .^3, family = "poisson", data = data, subset = mod$linearity,
##     x = TRUE, y = TRUE)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.63571  -0.30009  -0.02353   0.27258   1.42540
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.150481   0.585423   3.673 0.000239 ***
## v1           0.069795   0.587067   0.119 0.905364
## v2          -0.524215   0.513583  -1.021 0.307396
## v3           0.052966   0.551965   0.096 0.923552
## v4          -0.709525   0.580147  -1.223 0.221326
## v5           0.243002   0.548686   0.443 0.657853
## v6          -1.163256   0.563668  -2.064 0.039044 *
## v7          -0.990704   0.597335  -1.659 0.097208 .
## v1:v2        0.384345   0.543024   0.708 0.479079
## v1:v3       -0.630375   0.570151  -1.106 0.268888
## v1:v4        0.008801   0.511458   0.017 0.986271
## v1:v5       -1.022805   0.570440  -1.793 0.072971 .
## v1:v6        0.540164   0.493879   1.094 0.274079
## v1:v7        0.097178   0.536628   0.181 0.856297
## v2:v3        0.602411   0.437371   1.377 0.168405
## v2:v4        0.748226   0.486811   1.537 0.124295
## v2:v5       -0.068926   0.428100  -0.161 0.872090
## v2:v6        0.297165   0.487409   0.610 0.542071
## v2:v7        0.274198   0.508369   0.539 0.589634
## v3:v4       -0.124465   0.541056  -0.230 0.818060
## v3:v5       -0.439354   0.468418  -0.938 0.348268
## v3:v6        0.024399   0.530220   0.046 0.963296
## v3:v7       -0.104400   0.556960  -0.187 0.851310
## v4:v5       -0.169421   0.521323  -0.325 0.745194
## v4:v6        0.756513   0.474213   1.595 0.110644
## v4:v7        0.780671   0.500911   1.559 0.119114
## v5:v6        1.245629   0.510770   2.439 0.014739 *
## v5:v7       -0.262620   0.523125  -0.502 0.615652
## v6:v7        0.697014   0.489957   1.423 0.154852
## v1:v2:v3    -0.349902   0.483330  -0.724 0.469102
## v1:v2:v4     0.101569   0.389778   0.261 0.794416
## v1:v2:v5     0.655208   0.493737   1.327 0.184496
## v1:v2:v6    -0.329286   0.390979  -0.842 0.399670
## v1:v2:v7    -0.520368   0.393042  -1.324 0.185520
## v1:v3:v4     0.353292   0.406623   0.869 0.384932
## v1:v3:v5     0.638711   0.484979   1.317 0.187843
## v1:v3:v6     0.352694   0.402715   0.876 0.381143
## v1:v3:v7    -0.001586   0.413554  -0.004 0.996941
```

```
## v1:v4:v5     0.664745   0.400212   1.661 0.096717 .
## v1:v4:v6    -0.463885   0.368214  -1.260 0.207732
## v1:v4:v7    -0.342583   0.372009  -0.921 0.357103
## v1:v5:v6     0.044968   0.399958   0.112 0.910481
## v1:v5:v7     0.447641   0.404364   1.107 0.268283
## v1:v6:v7     0.218868   0.371499   0.589 0.555763
## v2:v3:v4    -0.325914   0.404392  -0.806 0.420280
## v2:v3:v5           NA         NA      NA       NA
## v2:v3:v6    -0.247853   0.405621  -0.611 0.541168
## v2:v3:v7     0.028322   0.414520   0.068 0.945527
## v2:v4:v5     0.004655   0.394418   0.012 0.990583
## v2:v4:v6    -0.111152   0.373713  -0.297 0.766141
## v2:v4:v7    -0.148061   0.376692  -0.393 0.694279
## v2:v5:v6    -0.766051   0.394925  -1.940 0.052412 .
## v2:v5:v7     0.075213   0.399004   0.189 0.850482
## v2:v6:v7     0.460826   0.381109   1.209 0.226597
## v3:v4:v5    -0.063494   0.423318  -0.150 0.880771
## v3:v4:v6     0.357746   0.366298   0.977 0.328741
## v3:v4:v7    -0.106368   0.371567  -0.286 0.774672
## v3:v5:v6    -0.234816   0.422424  -0.556 0.578295
## v3:v5:v7     0.804923   0.423843   1.899 0.057550 .
## v3:v6:v7    -0.659090   0.371085  -1.776 0.075714 .
## v4:v5:v6    -0.427957   0.375755  -1.139 0.254734
## v4:v5:v7     0.125167   0.377356   0.332 0.740119
## v4:v6:v7     0.014192   0.370131   0.038 0.969413
## v5:v6:v7    -0.811516   0.377098  -2.152 0.031397 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 156.215  on 111  degrees of freedom
## Residual deviance:  31.291  on  49  degrees of freedom
## AIC: 526.46
##
## Number of Fisher Scoring iterations: 5
```

```
pred=predict(mod2,se.fit = TRUE, type = "response")
CIs_two=cbind(pred$fit-qnorm(0.975)*(pred$se.fit),
        pred$fit+qnorm(0.975)*(pred$se.fit))
CIs_two
```

```
##           [,1]      [,2]
## 2    4.1225794 14.297167
## 3    1.6553214  8.514389
## 4    3.3515950 12.663846
## 5    3.8973641 14.215000
## 6    1.8159636  8.523987
## 7    4.7546085 14.830931
## 8    2.2138503  9.358914
## 10   1.3002430  7.840116
## 11   1.7609413  8.810062
## 12   4.1471451 14.442860
## 13   0.8782774  6.988161
```

```
## 14   0.7813852  5.668862
## 15   2.6610345 10.315125
## 16   2.3788529  9.811419
## 17   5.3492799 16.553863
## 18   1.2217860  7.223596
## 19   2.1622088  9.941231
## 21   3.2718090 11.611529
## 22   0.7955744  4.991161
## 23   3.3100229 11.711938
## 24   1.9928071  9.647679
## 25   1.3337452  7.760888
## 26   0.7610721  6.116189
## 27   1.7134991  8.956378
## 29   0.5935347  4.527403
## 30   0.6511110  4.908353
## 31   1.2844978  6.636959
## 32   4.2062794 14.765606
## 34   1.5971712  8.281046
## 35   0.3040465  3.973260
## 36   1.1712670  7.146063
## 37   0.4357698  5.363538
## 38   1.2017812  6.883276
## 39   0.8234526  5.764531
## 40   0.9158568  5.923424
## 42   0.7981173  5.770057
## 43   1.2119806  7.265265
## 44   2.1133668  9.462096
## 45   0.8073299  6.867973
## 46   1.5877670  8.077067
## 47   2.2379602  9.663679
## 48   2.4495287  9.903330
## 49   5.9446631 17.839107
## 50   3.5085804 12.954586
## 51   1.0403033  7.183045
## 53   2.6027717 10.491818
## 54   2.6195220 10.385711
## 55   0.8175962  5.636603
## 56   1.2767174  7.914492
## 57   2.6313072 11.085386
## 58   2.1088480  9.600608
## 59   1.2835907  7.726240
## 61   1.4178212  7.531662
## 62   3.4004391 12.206277
## 63   0.7909976  5.258302
## 64   3.5400126 13.202993
## 66   0.8538205  6.683730
## 67   0.3978836  4.569568
## 68   0.4086172  4.714980
## 69   0.4249484  5.633720
## 70   0.2032951  3.602417
## 71   1.3896357  7.476210
## 72   0.2116017  3.214057
## 74   0.4991003  5.297443
## 75   1.5228062  8.197511
```

```
## 76   1.0893371  6.858939
## 77   0.2472298  4.917246
## 78   0.1508936  3.157049
## 79   1.7689727  8.173054
## 80   0.4423221  3.893612
## 81   0.6371389  5.617391
## 82   0.2271642  3.931172
## 83   0.3583352  4.543338
## 85   1.2940929  7.268820
## 86   0.6034011  5.128230
## 87   2.4191916 10.190232
## 88   1.4709159  8.526093
## 89   0.6437610  5.781252
## 90   0.5280266  5.419476
## 91   1.3359634  7.883246
## 93   0.8062256  5.747319
## 94   1.2744985  7.421137
## 95   2.5465551 10.208352
## 96   5.3111995 16.877472
## 98   1.5297011  8.571835
## 99   0.6635226  5.986745
## 100  1.6346324  8.907213
## 101 -0.1427022  2.157578
## 102  0.2214164  3.625596
## 103  0.3957688  4.514910
## 104  0.2985760  3.850724
## 106  1.7305100  8.827363
## 107  6.3944030 18.772296
## 108  4.6806200 15.207423
## 109  0.2597288  5.048838
## 110  0.8071847  5.692058
## 111  3.2387386 11.991389
## 112  1.5722143  7.561718
## 113  0.6057880  5.450983
## 114  1.2924829  7.705369
## 115  0.2796504  4.428939
## 117  0.2232701  3.252460
## 118  1.0149417  6.381940
## 119  0.2857476  3.676834
## 120  0.7615243  6.424554
## 121  1.2532205  7.512227
## 122  2.0453060  9.355738
## 123  1.8875546  9.276477
## 125  0.6060110  4.753452
## 126  2.9814260 11.236278
## 127  1.0949753  6.131196
## 128  4.0175398 14.064114
```

Now we shall verify whether the third order model is appropriate using a likelihood ratio test.

```
mod.simple=glmdr(y ~ .,family="poisson",data)
mod1.simple=glm(y ~ .,family="poisson",data=data, x=TRUE,y=TRUE)
mod2.simple=update(mod1.simple,subset=mod.simple$linearity)
mod.2=glmdr(y ~ .^2,family="poisson",data)
```

```
mod1.2=glm(y ~ .^2,family="poisson",data=data, x=TRUE,y=TRUE)
mod2.2=update(mod1.2,subset=mod.2$linearity)
mod.4=glmdr(y ~ .^4,family="poisson",data)
mod1.4=glm(y ~ .^4,family="poisson",data=data, x=TRUE,y=TRUE)
mod2.4=update(mod1.4,subset=mod.4$linearity)
lr.test(mod2.2,mod2)
```

```
## $LR
## [1] 160.3381
##
## $pvalue
## [1] 1.729207e-13
##
## attr(,"class")
## [1] "lrt.test"
```

```
lr.test(mod2.simple,mod2)
```

```
## $LR
## [1] 246.5368
##
## $pvalue
## [1] 3.160119e-21
##
## attr(,"class")
## [1] "lrt.test"
```

```
lr.test(mod2.simple,mod2.2)
```

```
## $LR
## [1] 86.19868
##
## $pvalue
## [1] 7.245628e-10
##
## attr(,"class")
## [1] "lrt.test"
```

```
lr.test(mod2,mod2.4)
```

```
## $LR
## [1] 15.22438
##
## $pvalue
## [1] 0.9921224
##
## attr(,"class")
## [1] "lrt.test"
```

From the above results of likelihood ratio tests, we can see that when we compare our model with 3rd order terms, 'mod2' with other models with lower orders ('mod2.simple', 'mod2.2'), then the p-values are very low

(almost zero). So, we can say that the 3rd order terms are significant. Moreover, the likelihood ratio test between 'mod2' and the model with 4 th order terms ('mod2.4') shows that p-value is not significant (LR follows chi-square). So, the 4th order terms are not necessary in this model. Therefore, we can conclude that the third order model is appropriate.