M3/4S2 Spring 2019 - Assessed Coursework

Alexander Pinches CID:01201653

Normal Linear Modelling

Analysis of problem

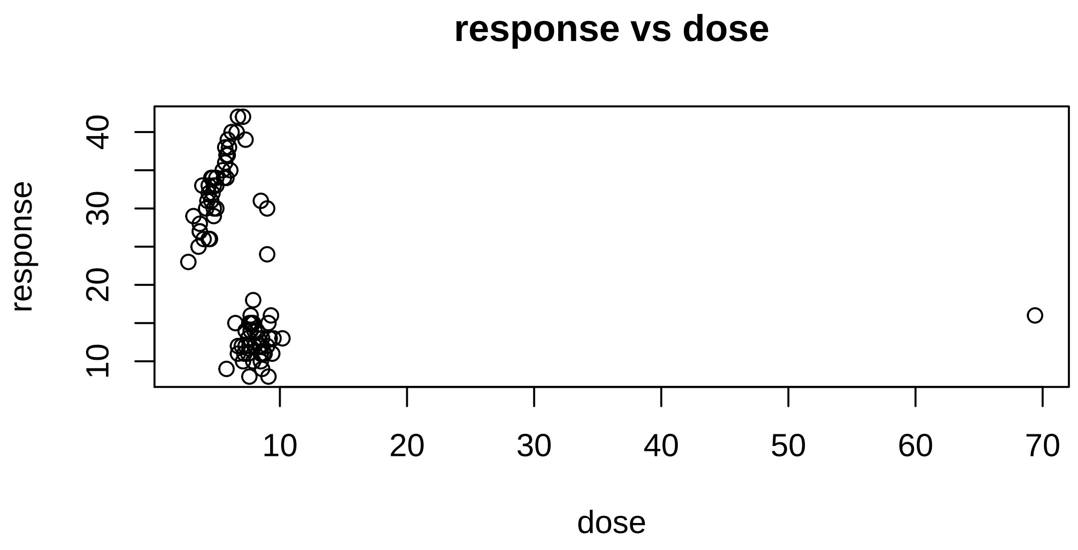
First we check the types of each variable in the data frame.

```
str(bp) # check types of each variable
```

```
## 'data.frame':    85 obs. of  3 variables:
##  $ response: num  33 42 34 40 30 31 38 23 38 26 ...
##  $ dose    : num  3.9 7.1 5 6.6 9 4.3 6 2.8 5.7 4.5 ...
##  $ female  : num  0 0 0 0 1 0 0 0 0 0 ...
```

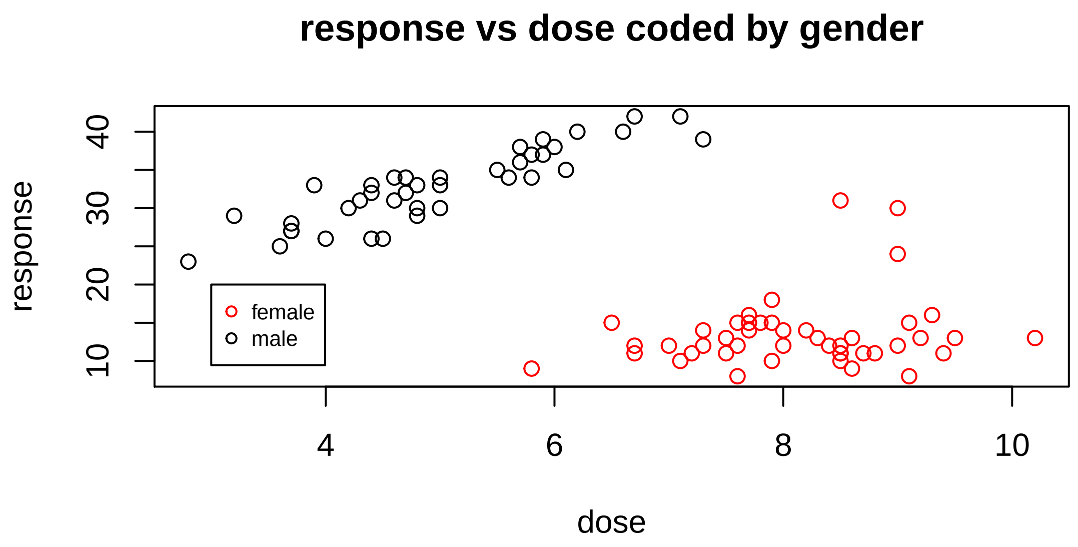
We change female to a factor variable as then R will automatically encode this variable by default with the treatment encoding. This encoding is likely the most practical for this dataset as it will allow us to seperate the data into two if gender effects the effectiveness of the drug. We plot response and dose to see if theres any obvious outliers.

```
bp$female <- as.factor(bp$female) # change to factor so R automatically encodes
plot(bp[,c(2,1)],main="response vs dose") # plot without female
```



We see one obvious outlier with a dose of 69.4 as linear models are influenced by extreme values we will remove it from the dataset. To see if using whether the patient is female or male may improve our model we plot response and dose again and colour code each point based on which one of the genders they are.

```
bp <- bp[which(bp$dose != 69.4),] # remove outlier
plot(bp[,c(2,1)],col=bp$female,main="response vs dose coded by gender") # plot colour coded by if female or male
legend(3,20,c("female","male"),col=c("red","black"),pch=1,cex=0.7) # add legend
```

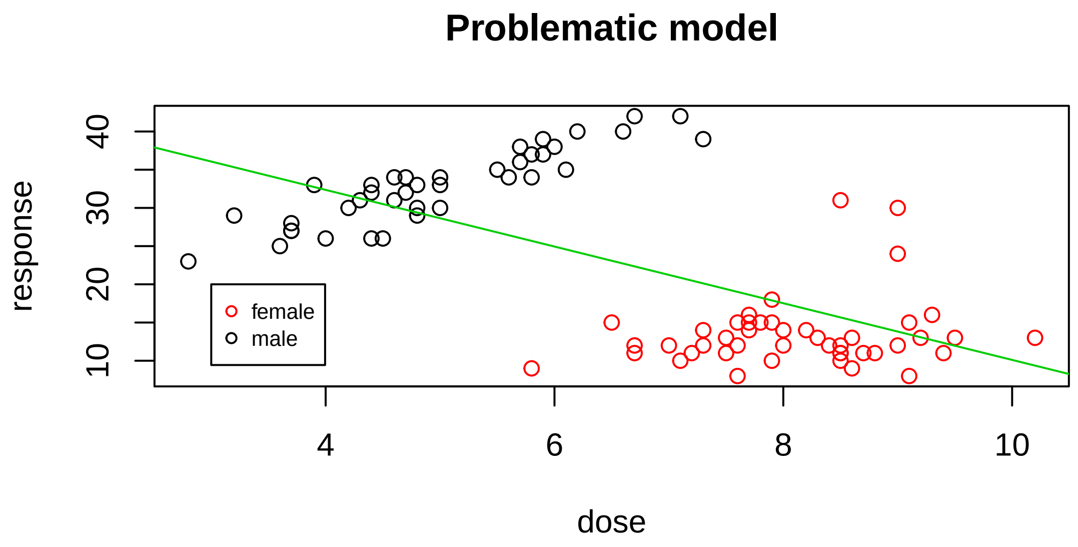


We see that responsiveness to the drug is clearly different based on whether or not the patient is female. Males being more responsive on the whole with lower doses than females with two clear clusters formed using the binary variable female will probably be useful in predicting response.

Models

We next recreate the model created by the clinical collaborator to see why it produced a model suggesting that dose had a negative impact on response contrary to what the drug should do and what the previous plots show. We suspect this is likely due to the model fitting incorrectly with the data despite being numerically the best fitting linear model. To check this we plot the line produced by the linear model and superimpose it on the dataset.

```
model0 <- lm(response~dose,data = bp) # make problamatic model
beta <- model0$coefficients # extract coeff
plot(bp[,c(2,1)],col=bp$female,main="Problematic model") # plot colour coded by if female or male
legend(3,20,c("female","male"),col=c("red","black"),pch=1,cex=0.7) # add legend
abline(coef = beta,col=3) # plot line created by lm
```



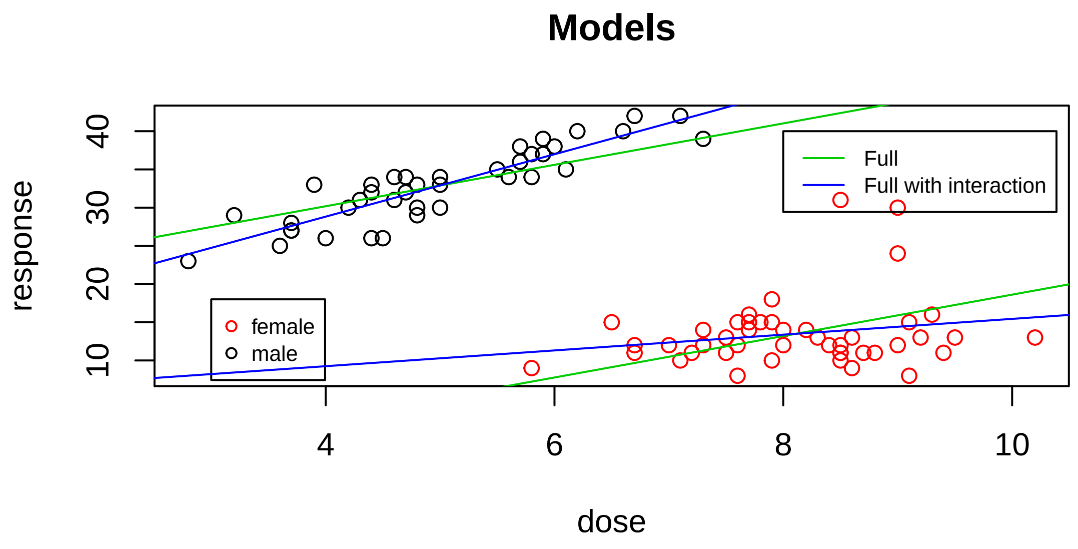
We see that the model made a negative line that fits the data well but incorrectly. This is because the linear model is fit by finding the line with the smallest least squares. We could likely fix this by including the female binary variable or the female variable and an interaction term between dose and female. As looking at the plots the lines that best fit the data for females and for male look like they should have a different gradient not just a different intercept. To help determine if this is likely true or us just overfitting a model to the data we can create the two models and compare their AICs which penalises extra paramaters.

```
model1 <- lm(response~.,data = bp)
model2 <- lm(response~.^2,data = bp)
sprintf("Full model AIC;%s with interaction terms ;%s",round(extractAIC(model1)[2],4),round(extractAIC(model2)[2],4))
```

```
## [1] "Full model AIC;234.2897 with interaction terms ;223.2592"
```

We see the AIC is lower for the model containing the interaction term so the model fits better and enough so that we can justify including the extra term as we are less worried about overfitting. As we included female this model now contains two lines one for female and one for male. We plot them below. In the full model we have two lines with seperate intercepts but the same gradient which we would intuitively think is unlikely to be true. In the full model with an interaction term we can take this into account and the model has two seperate lines of different gradients and intercepts. Although this data set is small and we may be overfitting the model.

```
beta1 <- model1$coefficients # extract coefficents
beta2 <- model2$coefficients # extract coefficents
plot(bp[,c(2,1)],col=bp$female, main="Models") # plot colour coded by if female or male
legend(3,18,c("female","male"),col=c("red","black"),pch=1,cex = 0.7) # add legend
# add model1 lines
abline(a=beta1[1],b=beta1[2],col=3)
abline(a=beta1[1] + beta1[3],b=beta1[2],col=3)
# add model2 lines
abline(a=beta2[1],b=beta2[2],col=4)
abline(a=beta2[1] + beta2[3],b=beta2[2]+beta2[4],col=4)
legend(8,40,c("Full","Full with interaction"),col=c(3,4),lwd=1,cex = 0.7) # add legend
```



We see for the full model we get two lines in the correct direction this time and look to fit the data well.In the next plot we see that the lines produced look to fit the data even better than when we didn't include the interaction term to further confirm this we can look at summaries of each model to see p values and standard errors for the models and diagnostic plots to help determine their performance.

Model analysis

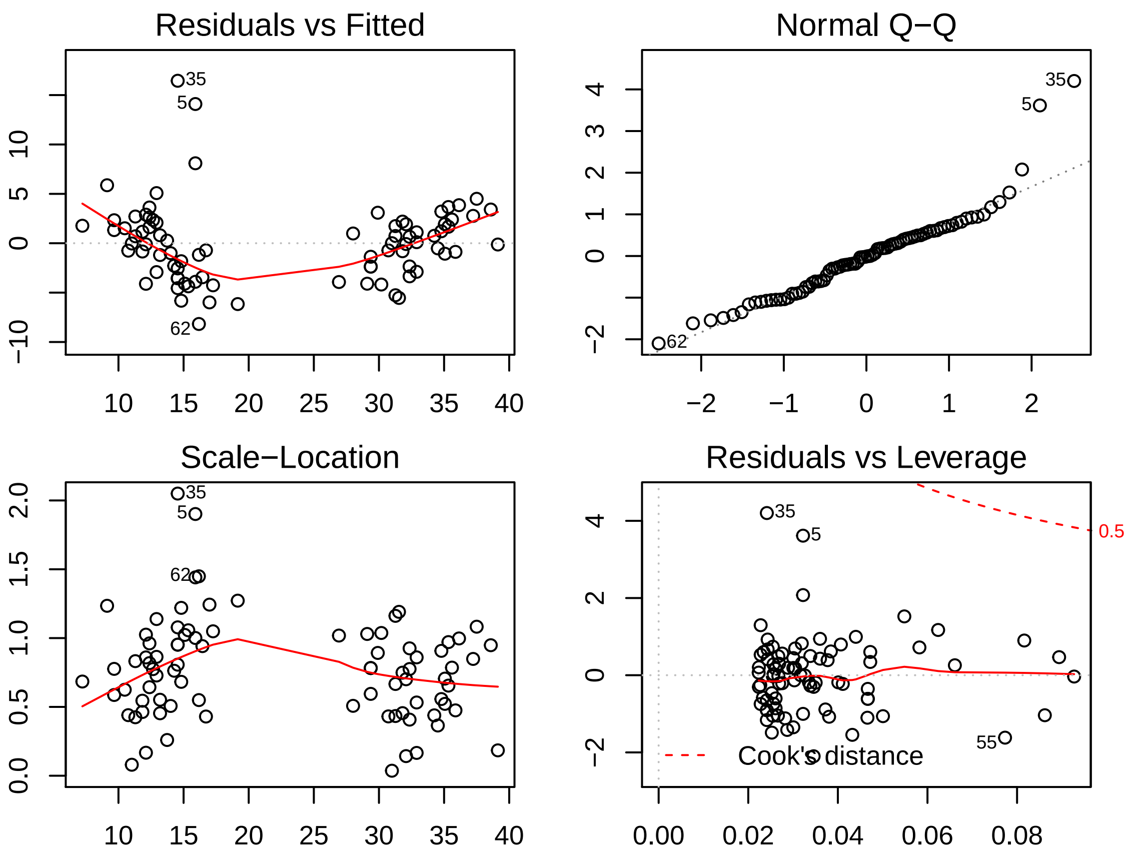
Firstly if we look at the summary of the full model we see that our estimates of the coefficents have small P values meaning we are confident in them being order 10^{-8} or much less and having low standard errors also means we are confident in them being what we estimate. The residual do have varying values indicating some values have a large influence on the estimates with a max of 16.451 and a min of -8.176. If we look at R squared and the adjusted R squared test statistics we see they are close to 1 indicating the model fits well as the data points are close to the regression line. This is further shown by the F-statistic having a very low p-value showing the model matches the data better tan the mean.

```
summary(model1) # summarise model

##
## Call:
## lm(formula = response ~ ., data = bp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.176 -2.635 -0.052  1.973 16.451
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.3370     2.3098   8.372 1.39e-12 ***
## dose          2.7111     0.4449   6.094 3.54e-08 ***
## female1     -27.8323     1.6353 -17.020  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 3.963 on 81 degrees of freedom
## Multiple R-squared:  0.8689, Adjusted R-squared:  0.8657
## F-statistic: 268.4 on 2 and 81 DF,  p-value: < 2.2e-16
```

Below we see 4 diagnostic plots of the full model. Firstly if we look at the standardised residual values against the fitted values we in a perfect model expect the line of best fit to be $y = 0$. We see this is not the case and the line and points form a parabola suggesting some non-linearity in the model that isnt accounted for. This could be improved by the inclusion of a interaction term. The Normal QQ plot plots quartiles of the data against standardised residuals. We want this to be close to $y = x$ we see this is the case for most points however at the higher quartiles points 5 and 35 deviate showing they deviate alot from the model and the model doesnt predict them well. Next if we look at the scale-location plot which plots the square root of the standardised residuals against the fitted values. We use this to check the assumption of homoscedasticity equal variance and we want as the points are here to be randomly spread. If we look at the final plot of residuals against leverage we see high residuals but low levage and a small cooks distance for all points indicating the model isnt being effected by any extreme outliers. As they have low leverage (potential to effect the model) but some points have high residuals (differing values from the model).

```
par(mfrow=c(2,2),mar=c(2,2,2,2)) # create space in plotting device
plot(model1) # plot diagnostic plots
```

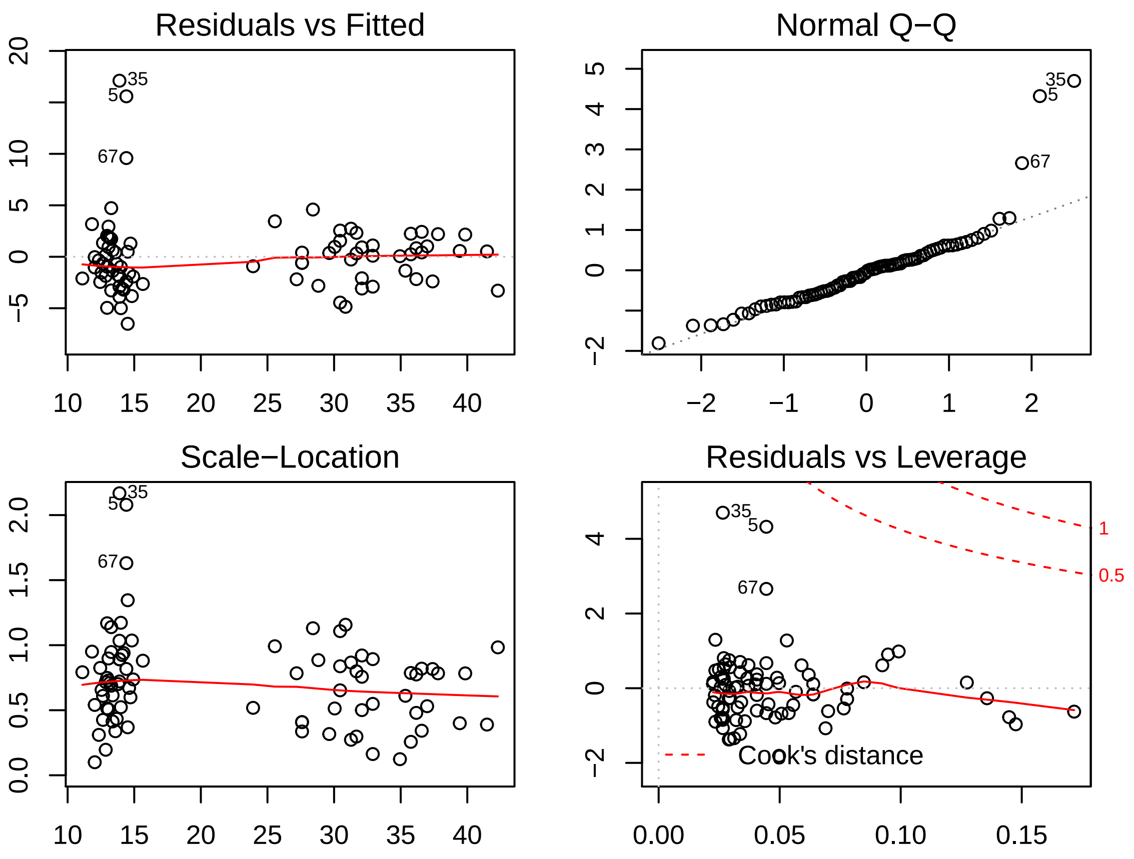


We can then compare this directly with the full model with interaction terms. We see the max residual is higher but the minimum is lower, the residual standard error is lower and the R squared and adjusted R squared values are higher suggesting a better fit. The F statistic also has a low pvalue suggesting the model fits the data better than the mean. However we achieve very low pvalues although not as low for the coefficents except for the is a female coefficent which has a pvalue of 0.2 and a large standard error suggesting we arent confiden't in the value of this coefficent.

```
summary(model2)

##
## Call:
## lm(formula = response ~ .^2, data = bp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5113 -2.1739 -0.1528  1.2981 17.1076
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   12.4900     2.8492   4.384 3.52e-05 ***
## dose           4.0826     0.5583   7.312 1.77e-10 ***
## female1       -7.3663     5.7896  -1.272 0.206940
## dose:female1  -3.0510     0.8327  -3.664 0.000445 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.69 on 80 degrees of freedom
## Multiple R-squared:  0.8877, Adjusted R-squared:  0.8835
## F-statistic: 210.9 on 3 and 80 DF,  p-value: < 2.2e-16
```

Looking at the diagnostic plots of this model we see that the residual verses fitted value plot the points are better distributed although some have large residuals. We do however see that the error seems to be linear so the non-linearity from the previous model is gone. The normal QQ plot has more values deviating from their expected residual than in the previous model. The scale location graph is similar to that of the previous model but slightly better seeming slightly more randomly distributed so the model is being effected lesss by extreme values. The residual leverage plot is almost the same but with more high residual low leverage points and thus meaning no points are massively effecting the model and have high cooks distance thus indicating this.

```
par(mfrow=c(2,2),mar=c(2,2,2,2)) # create space in plotting device
plot(model2) # plot diagnostic plots
```



So the best model has formula $Y = \hat{\beta}_0 + \text{dose}\hat{\beta}_1 + \text{female}\hat{\beta}_2 + \text{dose} \cdot \text{female}\hat{\beta}_3$. That has the lowest AIC and highest R^2 test staticties. To check it wasnt fitting incorrectly like the model the clinician created we plotted the regression lines. The model works by fitting a linear regressor for males of the form $Y_{\text{male}} = \hat{\beta}_0 + \text{dose}\hat{\beta}_1$ then when adding to this for female cases with the last two terms by using an encoding for if the input is female.

Proofs (2)

$X^T X$ is a nxn matrix

$$X_{(i)}^T X_{(i)} = \left(X^T X - x_i^T x_i\right)$$

Knowing this we can substitute into the given identity with $A = X^T X$,$u^T = -x_i^T$ and $v = x_i$ to get

$$X_{(i)}^T X_{(i)} = \left(X_{(i)}^T X_{(i)}\right)^{-1} + \frac{\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T x_i \left(X_{(i)}^T X_{(i)}\right)^{-1}}{1 - x_i^T \left(X_{(i)}^T X_{(i)}\right)^{-1} x_i} \tag{1}$$

$$= \left(X_{(i)}^T X_{(i)}\right)^{-1} + \frac{\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T x_i \left(X_{(i)}^T X_{(i)}\right)^{-1}}{1 - h_{ii}} \tag{2}$$

We then note that $\hat{\beta}_{(i)} = \left(X_{(i)}^T X_{(i)}\right)^{-1} X_{(i)}^T Y_{(i)}$ and $X_{(i)}^T Y_{(i)} = \left(X^T Y - x_i^T y_i\right)$

$$\implies \hat{\beta}_{(i)} = \left[\left(X_{(i)}^T X_{(i)}\right)^{-1} + \frac{\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T x_i \left(X_{(i)}^T X_{(i)}\right)^{-1}}{1-h_{ii}} \right] \left(X^T Y - x_i^T y_i\right) \tag{3}$$

$$\implies \hat{\beta}_{(i)} = \hat{\beta} - \left[\frac{\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T}{1-h_{ii}} \right] \left(y_i(1-h_{ii}) - x_i\hat{\beta} + h_{ii}y_i\right) \tag{4}$$

$$\implies \hat{\beta}_{(i)} = \hat{\beta} - \frac{\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T (y_i - \hat{y}_i)}{1-h_{ii}} \tag{5}$$

Which re arranges to the required result

$$\hat{\beta} - \hat{\beta}_{(i)} = \frac{\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T (y_i - \hat{y}_i)}{1-h_{ii}}$$

We substitute the negative of the result we just showed into the formula for cooks distance to get after cancelling inverses of each other. We note $e_i = y_i - \hat{y}_i$ and is a scalar.

$$C_i = \frac{\left(\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T (y_i - \hat{y}_i)\right)^T x_i(y_i - \hat{y}_i)}{p\hat{\sigma}^2(1-h_{ii})^2} \tag{6}$$

$$C_i = \frac{\left(\left(X_{(i)}^T X_{(i)}\right)^{-1} x_i^T\right)^T x_i e_i^2}{p\hat{\sigma}^2(1-h_{ii})^2} \tag{7}$$

Then as $r_i^2 = \frac{e_i^2}{\hat{\sigma}^2(1-h_{ii})}$ we factor this out and using the properties of the transpose we get.

$$C_i = \frac{r_i^2}{p(1-h_{ii})} x_i^T \left(X_{(i)}^T X_{(i)}\right)^{-1} x_i$$

The terms outside the fraction are just h_{ii} so we get the result

$$C_i = \frac{r_i^2}{p} \frac{h_{ii}}{1-hii}$$

Proofs (3)

Assume the residuals are not all identically zero. We know that if $e_i = \alpha + \beta x_i = y_i - \hat{y}_i$ then for the vector of residuals

$$\underline{e} = \alpha + \beta X = Y - \hat{Y}$$

then $\hat{Y} = Y - \hat{Y}$

$$\implies Y = 2\hat{Y}$$

Contradiction as this $\implies Y = 2\alpha + 2X\beta$ so the residuals must all be identically zero.

General linear modelling

Poisson model

We assume each individual test is independent and the mean for each independent observation is constant in time. We also assume the observations are continuous as poisson is continuous even though the problem is discrete. We initialise the problem by creating a full model without interaction terms we use the `step()` function in R to compare AIC's of models as we add and remove terms from the formula. We see that the model with the lowest AIC is the model with interaction terms.

```
model <- glm(y~x+q,data=read,family="poisson") # make basic model
step(model,scope = ~x+q+x*q,direction = "both") # test adding and removing variables from model by using AIC
```

```
## Start:  AIC=1185.52
## y ~ x + q
##
##        Df Deviance    AIC
## + x:q   1   856.48 1159.5
## <none>      884.49 1185.5
## - q     1   891.52 1190.5
## - x     1  1156.55 1455.6
##
```

```
## Step:  AIC=1159.51
## y ~ x + q + x:q
##
##        Df Deviance    AIC
## <none>      856.48 1159.5
## - x:q   1   884.49 1185.5
```

```
##
## Call:  glm(formula = y ~ x + q + x:q, family = "poisson", data = read)
##
## Coefficients:
## (Intercept)            x            q          x:q
##    4.097104    -0.011562    -0.146944     0.007311
##
## Degrees of Freedom: 51 Total (i.e. Null);  48 Residual
## Null Deviance:       1861
## Residual Deviance: 856.5     AIC: 1160
```

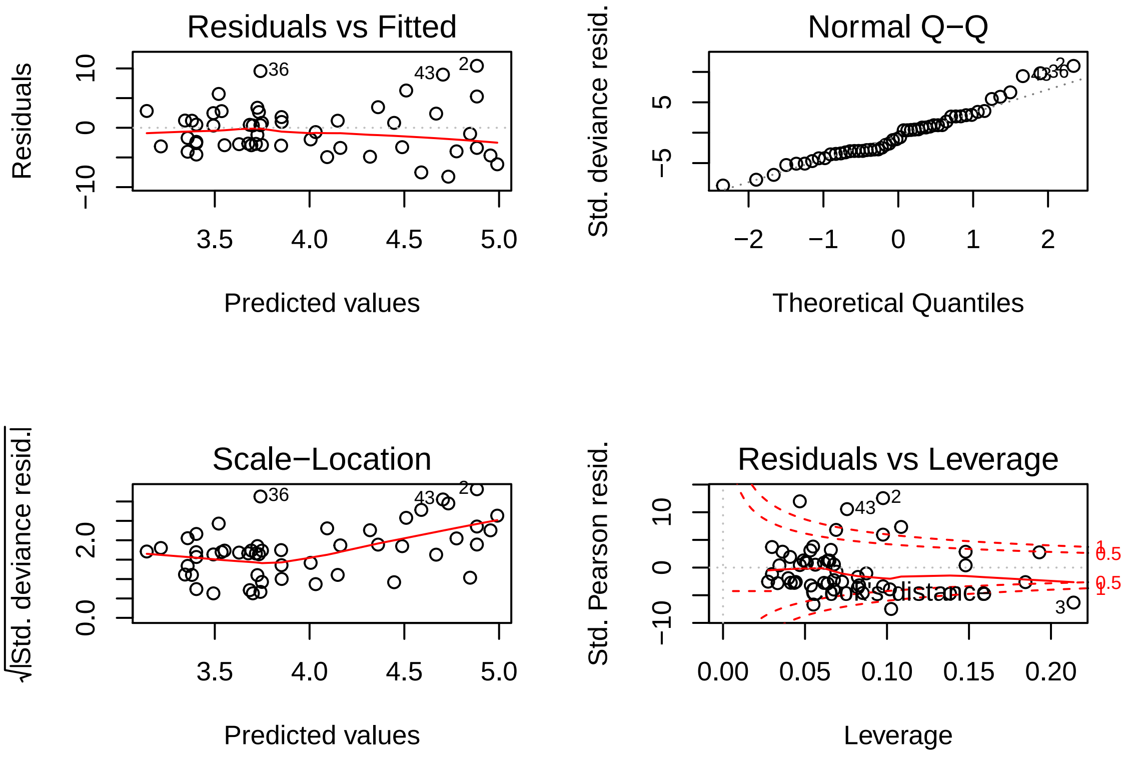
We see very low p values for all but our x coefficent which would suggest potentially removing this term. We see low standard errors for our estimates suggesting that we can be confident in the parameters of this model. The residual devience is significantly lower than the null showing this model fits much better then the null model.

```
model1 <- glm(y~.^2,data=read,family="poisson") # make model with lowest AIC
summary(model1) # summarise model
```

```
##
## Call:
## glm(formula = y ~ .^2, family = "poisson", data = read)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.2406  -3.0326  -0.8761   1.9484  10.4384
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.097104   0.676219   6.059 1.37e-09 ***
## x           -0.011562   0.032432  -0.357    0.721
## q           -0.146944   0.030531  -4.813 1.49e-06 ***
## x:q          0.007311   0.001361   5.371 7.84e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1861.21  on 51  degrees of freedom
## Residual deviance:  856.48  on 48  degrees of freedom
## AIC: 1159.5
##
## Number of Fisher Scoring iterations: 4
```

We see in the diagnostic plots that some values have a high cooks distance and thus have a high impact on the model. This could potentially be problamatic and reduces our confidence in this model. We could remove these values but as the data set is very small we risk reducing the quality of the fit of our model. This is also shown by some point being off the line in the QQ plot. The residual vs fitted plot suggests errors are constant through out the model but the residuals are large.

```
par(mfrow=c(2,2)) # create space in plotting device
plot(model1) # plot relevant graphs
```



We can calculate a 95%confidence interval for the value of the mean parameter from the fitted values.

```
# calculate CI for mean
ybar <- mean(predict(model1,read))
yvar <- var(predict(model1,read))
CI <- c(exp(ybar-1.96*sqrt(yvar/nrow(read))),exp(ybar+1.96*sqrt(yvar/nrow(read))))
sprintf("Mean confidence interval:%s",list(round(CI,4)))
```

```
## [1] "Mean confidence interval:c(45.5773, 61.2357)"
```

Negative binomial

$$\Pr(Y_i = y; p, r) = \binom{y+r-1}{y}(1-p)^r p^y$$

We take the natural log of the right hand side and the exponetial and seperate terms using the laws of logs to get

$$\Pr(Y_i = y; p, r) = \exp\left(\ln\binom{y+r-1}{y} + r\ln(1-p) + y\ln(p)\right)$$

All the terms in the logs are strictly positive so are well defined. So the equation is in exponetial form with $\theta = \ln(p)$,$\phi = 1$,$a(\phi) = 1$,$b(\theta) = -r\ln(1-p)$ and $c(y,\phi) = \ln\binom{y+r-1}{y}$.

IWLS Alogirthm

Using the definition of a pdf in exponential family for which we calculate.

$$\begin{aligned}
\mu_i &= E(Y_i) \\
&= \frac{rp_i}{1-p_i} \\
\theta_i &= \ln(p_i) \\
b(\theta_i) &= -r\ln(1-p_i) \\
b'(\theta_i) &= \frac{rp_i}{1-p_i} \\
b''(\theta_i) &= \frac{rp_i}{(1-p_i)^2} \\
w_{ii}^{-1} &= V(\mu_i)\left(\frac{\partial \eta_i}{\partial \mu_i}\right)^2 \\
&= \frac{1}{(1-p)\hat{\mu_i}}
\end{aligned}$$

$$\begin{aligned}
V(\mu_i) &= b''(\theta_i) \\
&= \frac{\mu_i}{1-p} \\
\eta &= \ln(\mu_i) \\
\frac{\partial \eta_i}{\partial \mu_i} &= \frac{1}{\mu_i} \\
z_i &= \hat{\eta_i} + \frac{y_i - \hat{\mu_i}}{\hat{\mu_i}} \\
&= \ln(\hat{\mu_i}) + \frac{y_i}{\hat{\mu_i}} - 1
\end{aligned}$$

We then repeatedly apply algorithm 3.1 the IWLS algorithm from the notes where we calculate the above for each iteration and fitting a linear model with formula z~X and weights as defined above updating beta for each iteration until a stopping criterion is fufilled.

Applying Algorithm

We will calculate the deviance of the model for comparison and for creating a stopping criteria for our algorithm.

$$D = 2\phi\left[\iota(\hat{\beta_{sat}}; y) - \iota(\hat{\beta}; y)\right]$$

Where ι represents the log likelihood function. We know the maximum log likelihood occurs under the saturated model when $y = \mu = \frac{rp}{1-p}$.

$$\iota(\hat{\beta_{sat}}; y) = \sum_{i=1}^{n}\left(\ln\binom{\frac{rp_i}{1-p_i} + r - 1}{\frac{rp_i}{1-p_i}} + r\ln(1-p_i) + \frac{rp_i}{1-p_i}\ln(p_i)\right)$$

$$\iota(\hat{\beta}; y) = \sum_{i=1}^{n}\left(\ln\binom{y_i + r - 1}{y_i} + r\ln(1-p_i) + y_i\ln(p_i)\right)$$

We substituting this in and simplifying by using the laws of logs and the definition of choose and cancel terms above and below from the factorials.

$$D = 2\sum_{i=1}^{n}\left[\ln\left(\frac{\prod_{k=1}^{r-1}\left(\frac{rp_i}{1-p_i} + k\right)}{\prod_{k=1}^{r-1}(y_i + k)}\right) + \left(\frac{rp_i}{1-p_i} - y_i\right)\ln(p_i)\right]$$

We can then use then use the deviance with a convergence criterion used by R.

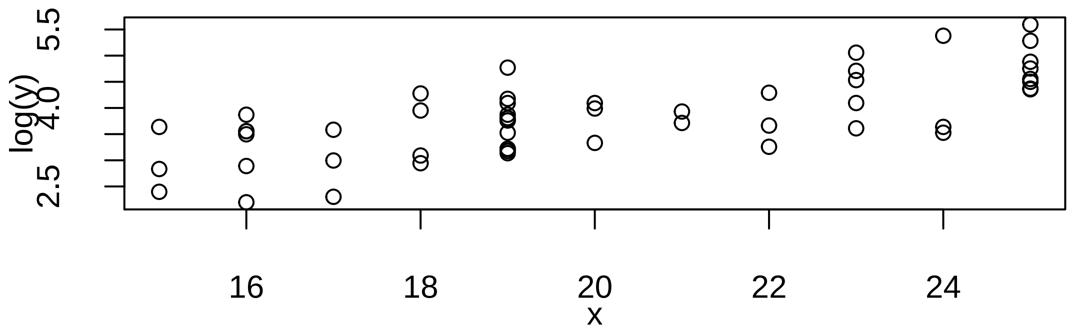
$$\frac{|D^{\text{new}} - D^{\text{old}}|}{|D^{\text{new}}| + 0.1}$$

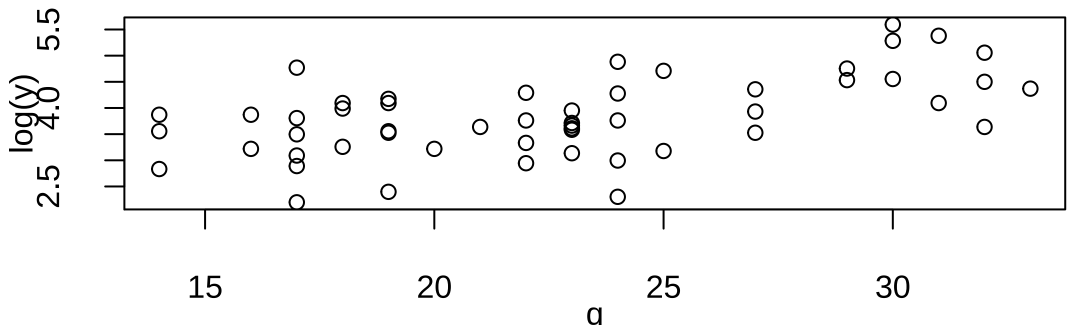
When this is less than 10^{-8} the model is deemed to have converged. This allows us to efficently determine a stopping point for the model based on the convergence of its deviance. So we create a function to calculate the devience and a helper function so we can vectorise the calculation of the products.

```r
seqvec <- Vectorize(seq.default,vectorize.args = c("from","to")) # creates sequences to do product sum over
#devience function
D <- function(n){ # n represents inputed mu's and p are the probabilities and y is the count
  a <- (n-y)*log(p) # term 1
  b <- apply(seqvec(from=n+1,to=n+r-1,by=1),2,prod)/apply(seqvec(from=y+1,to=y+r-1,by=1),2,prod) # term 2
  2*sum(a+b) # sum and calculate devience
}
```

From the problem we know $r = 3$ as r is the number of failures. We can create an initial beta by looking at plots of read and seeing how y changes with q and x. As we are using a log link function we need to estimate beta for $\ln(Y) = X\beta$ so we should plot y on the log scale.

```r
par(mfrow=c(2,1),mar=c(3,3.5,2,2),mgp=c(2.2,1.5,0))
plot(read$x,log(read$y),xlab="x",ylab = "log(y)")
plot(read$q,log(read$y),xlab="q",ylab = "log(y)")
```





Looking at the plots we see that if we were fitting a line we would probably want an intercept of 3 for both and a gradient that is small and close to zero. So a good initial beta might be $\beta = (3, 0, 0)^T$. We could formalise this by fitting a linear model to this and using these as our initial betas but by looking at the graph we should be able to get close enough for convergence.

We initially create a design for the formula $Y = \beta_0 + \underline{x}\beta_1 + \underline{q}\beta_2$, set the response vector, r and initial betas. We then apply the algorithm to fit a model using the values specified. We make the assumption that r is a constant, also p for a particular observation is constant (the probability of success doesn't change with time) and each observation is independent of one another. We print the deviance of the created model after creating it.

```
X <- cbind(1,read$x,read$q) # create design matrix
y <- read$y # response vector
beta <- c(3,0,0) # initial guess
r <- 3 # set r
terminate <- 10e-8
eta <- X%*%beta # calculate initial eta
mu <- exp(eta) # apply inverse of log to eta to get mu
p <- mu/(r+mu) # calculate p from mu using its definition
oldD <- D(exp(as.numeric(X%*%beta))) # calculate initial deviance of initial guess
control <- Inf # set control
while (control > terminate){ # loop until deviance criterion is less than terminate
  eta <- X%*%beta # calculate eta
  mu <- exp(eta) # apply inverse log to get mu
  p <- mu/(r+mu) # calculate p from mu
  detadmu <- 1/mu # calculate differential of eta wrt mu
  z <- eta + (y-mu)*detadmu # calculate z
  w <- (1-p)*mu # calculate weights
  lmod <- lm(z~X+0,weights=w) # fit model with weights to design matrix
  beta <- as.vector(lmod$coeff) # extract new betas
  newD <- D(exp(as.numeric(X%*%beta))) # calculate new deviance
  control <- abs(newD-oldD)/(abs(newD)+0.1) # calculate criterion value
  oldD <- newD # assign as oldD for next loop

}
dev <- newD
sprintf("Final deviance: %s",round(newD,4)) # print deviance
```

```
## [1] "Final deviance: 218.4923"
```

First we calculate the standard errors of each beta. The standard errors are small however we may be worried about the intercept term where the standard error is the same size as the intercept.

```
#standard error
J <- t(X)%*%diag(as.vector(w))%*%X
inv.J <- solve(J)
beta.sd <- sqrt(as.vector(diag(inv.J)))
sprintf("Standard errors: %s,Betas:%s",list(round(beta.sd,4)),list(round(beta,4)))
```

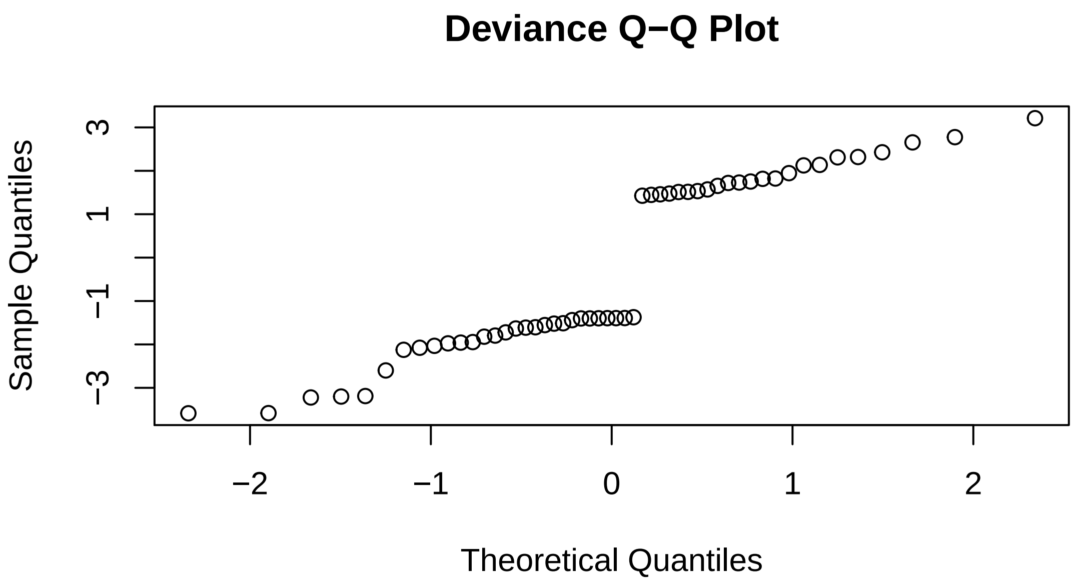
```
## [1] "Standard errors: c(0.5361, 0.0412, 0.0246),Betas:c(0.5898, 0.1644, 0.003)"
```

We can look at the deviance residuals and plot them we would expect them to form a curve and the summary seems to indicate this but on plotting the qqplot we see a large gap in the curve produced. This shows some unexpected discrepency in the model and the data produced as the deviance isn't a smooth curve.

```
#deviance residuals
n <- as.vector(exp(X%*%beta))
a <- (n-y)*log(p)
b <- apply(seqvec(from=n+1,to=n+r-1,by=1),2,prod)/apply(seqvec(from=y+1,to=y+r-1,by=1),2,prod)
d <- sign(y-mu)*sqrt(2*(a+b))
summary(d)
```

```
##        V1
##  Min.   :-3.5862
##  1st Qu.:-1.8025
##  Median :-1.3944
##  Mean   :-0.2525
##  3rd Qu.: 1.7222
##  Max.   : 3.2128
```

```
qqnorm(d,main = "Deviance Q-Q Plot")
```



We can also look at the pvalues of each of the coefficents. We see high pvalues for the intercept and coefficent of q. This might suggest q is not worth including in the model but the p value for x is very low.

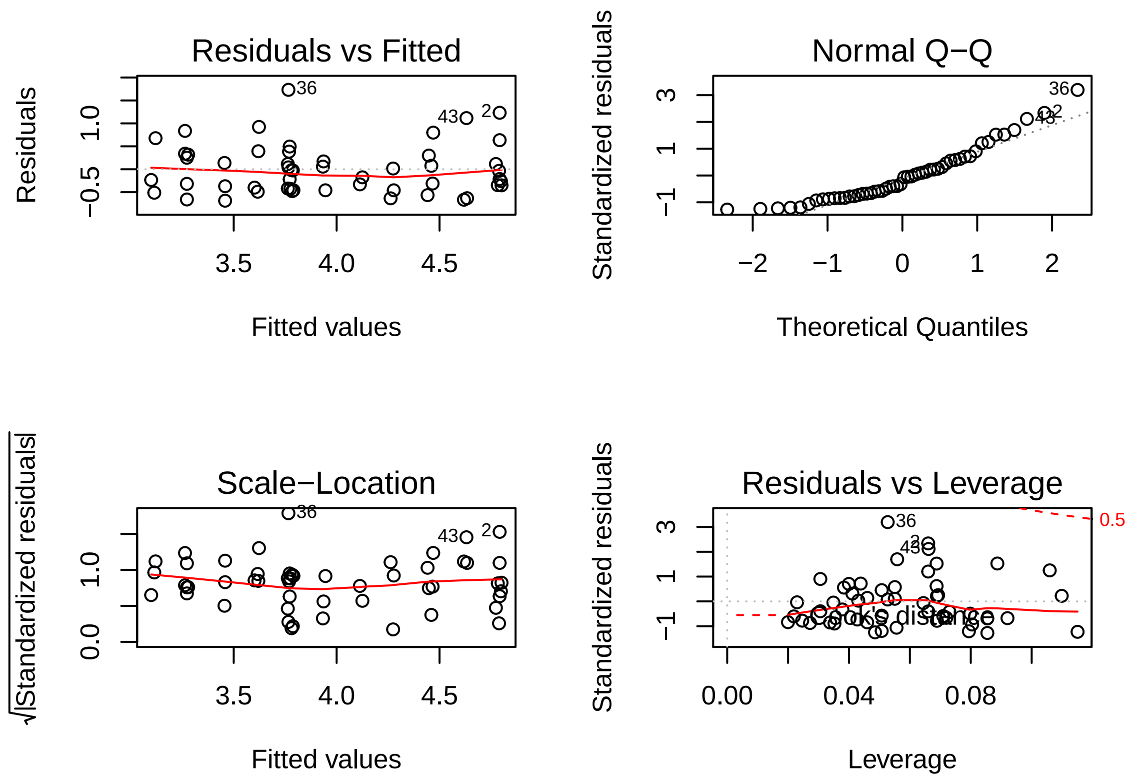
```
#pvalues
z <- beta/beta.sd
pvalues <- 2*(1-pnorm(abs(z),lower.tail = TRUE))
sprintf("Pvalues: %s",list(round(pvalues,4)))
```

```
## [1] "Pvalues: c(0.2713, 1e-04, 0.9017)"
```

```
# calculate 95% CI using standard errors for each beta
```

We see the residuls are randomly distributed against the fitted values as we expect the error to be random centered around zero. The normal Q-Q plot closely follows the line $x = y$ suggesting there aren't any extreme values. The scale-location plot seems to be randomly distributed suggesting equal variance homoscedasity and finally all the points in have a low cooks distance in the final pot so there are no points that are greatly effecting the model. Its also important to note cooks distance has a slightly different formula as it includes the weights here.

```
par(mfrow=c(2,2))
plot(lmod) # plot diagnostic plots
```



We can also calculate it's AIC compare it with including an interaction term to determine if that model is better and to compare it to the poisson model.

```
#AIC
AIC <- -2*sum(log(choose(y+r-1,y)*((1-p)^r)*(p^y))) + 2*length(beta)
sprintf("AIC without interaction terms:%s",round(AIC,4)) # print
```

```
## [1] "AIC without interaction terms:490.4491"
```

We can calculate a 95% confidence interval for the mean count and use this also calculate the confidence interval for the probability as r is fixed. We see the confidence interval for both the mean and probability are small. The fitted values have already been scaled before calculating the confidence intervals here.

```
ybar <- mean(n) # calculate mean of fitted values
yvar <- var(n) # calculate variance of fitted values
# CI for mean value
CImean<-c(ybar-1.96*sqrt(var(n)/nrow(read)),ybar+1.96*sqrt(var(n)/nrow(read)))
# CI for probability
CIp<-c(CImean[1]/(r+CImean[1]),CImean[2]/(r+CImean[2]))
sprintf("Confidence interval mean:%s,probability:%s",
        list(round(CImean,4)),list(round(CIp,4)))
```

```
## [1] "Confidence interval mean:c(52.1334, 70.3488),probability:c(0.9456, 0.9591)"
```

To compare this same as before we create a model but now including an aditional beta for the interaction term. As we thought that the coefficents of x and q would be close to zero a good initial guess for their interaction would be also zero. So with $\beta = (3, 0, 0, 0)^T$ as our initial guess and adding the interaction term to the design matrix we produce the model as before.

```
X <- cbind(1,read$x,read$q,read$x*read$q) # create design matrix
y <- read$y # response vector
beta <- c(3,0,0,0) # initial guess
r <- 3 # set r
terminate <- 10e-8
eta <- X%*%beta # calculate initial eta
mu <- exp(eta) # apply inverse of log to eta to get mu
p <- mu/(r+mu) # calculate p from mu using its definition
oldD <- D(exp(as.numeric(X%*%beta))) # calculate initial deviance of initial guess
control <- Inf # set control
while (control > terminate){ # loop until deviance criterion is less than terminate
  eta <- X%*%beta # calculate eta
  mu <- exp(eta) # apply inverse log to get mu
  p <- mu/(r+mu) # calculate p from mu
  detadmu <- 1/mu # calculate differential of eta wrt mu
  z <- eta + (y-mu)*detadmu # calculate z
  w <- (1-p)*mu # calculate weights
  lmod1 <- lm(z~X+0,weights=w) # fit model with weights to design matrix
  beta <- as.vector(lmod1$coeff) # extract new betas
  newD <- D(exp(as.numeric(X%*%beta))) # calculate new deviance
  control <- abs(newD-oldD)/(abs(newD)+0.1) # calculate criterion value
  oldD <- newD # assign as oldD for next loop

}
sprintf("Final deviance: %s",round(newD,4)) # print deviance

## [1] "Final deviance: 214.6723"
```

We get a slightly lower deviance. We next calculate the AIC to see if this improvement of model fit is worth the use of an extra parameter and the extra risk of overfitting to the data. We see that the AIC is slightly higher suggesting including the term is not worth it it may be worth like with the poisson model testing the AIC of each model as we add and remove variables to determine an optimal model. This also confirmed by a F test on the deviances of the two models.

```
#AIC
AIC <- -2*sum(log(choose(y+r-1,y)*((1-p)^r)*(p^y))) + 2*length(beta)
sprintf("AIC with interaction terms:%s",round(AIC,4)) # print

## [1] "AIC with interaction terms:491.0938"

pf(((dev-newD)/3-2)/(newD/(nrow(read)-3)),3-2,nrow(read)-3) # perform F test

## [1] 0
```

Conclusion

The best model using the negative binomial and the log link function fits the equation $g(Y) = \hat{\beta}_0 + \underline{x}\hat{\beta}_1 + \underline{q}\hat{\beta}_2$ where g is the link function the natural log in this case and Y is assumed to be distributed according to the negative binomial.

The negative binomial model performs much better than the poisson model. We see this in the much lower AIC of the negative binomial model and the much lower deviance. This would be expected as the poisson model models the rate of something occuring in this case every third failure. However the model is the same in the case of another number of failures just the mean changes to reflect this. Whereas the negative binomial handles this with the variable r to control the number of failures as we would assume that the probability of success is constant for each individual and dependent on x and q. The poisson model is also a continuous model where as the negative binomial and the problem itself are discrete. We can also probably attribute some of the loss in performance due to this incorrect assumption made in the poisson case. So the negative binomial more accurately represents what we are trying to predict so a model based on this would likely perform better.

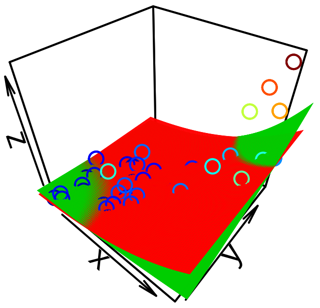
The confidence intervals of the means of the two overlap however the negative binomials is slightly larger however we can except this as the model appears to fit better than the poisson model and intuitively should as the model more accurately represents the problem.

However performing chi-squared goodness of fit test we get very low pvalues suggesting even though the negative binomial model fits much better than the poisson model both fit the data badly. This could be rectified by a larger sample as a smaple of 52 is very small. Due to the nature of the problem we would expect large errors in count as each persons for the same x and q could be expected to vary lots as people have bad days or other external factors. Using a link function that has additive errors instead of multiplicitive like the log does could reduce the size of the residuals and make a better model in both cases.

```
# perform chi^2 test
pchisq(dev,nrow(read)-3,lower = FALSE)

## [1] 2.890972e-23

pchisq(model1$deviance,nrow(read)-4,lower = FALSE)

## [1] 1.439607e-148
```

To illistrate what the models produce the surfaces produced are shown below with the poisson model as the green surface and the negative binomial as the red surface and the data points as points on the scatterplot. The x axis represents x the y axis q and the z axis is count. We can use this to quickly check that we don't run into the same error as the linear model by looking at only numerical measures of fit and not how the model looks against the points. Also looking at the surfaces the negative binomial model appears to fit better than the poisson model.

```
library(plot3D) # load in 3D plotting library
scatter3D(read$x,read$q,read$y,colkey = FALSE) # make 3D scatterplot
# create surface plots of models
M <- mesh(seq(min(read$x)-1,max(read$x)+1,length=100),seq(min(read$q)-1,max(read$q)+1,length=100))
# calculate x,y,z
x <- M$x
y <- M$y
z <- exp(model1$coefficients[1]+x*model1$coefficients[2]+y*model1$coefficients[3]+x*y*model1$coefficients[4])
# plot surface
surf3D(x, y, z, col=3, colkey = FALSE,add = TRUE)
#calculate z
z <- exp(lmod$coefficients[1]+x*lmod$coefficients[2]+y*lmod$coefficients[3])
# plot other surface
surf3D(x, y, z, col=2, colkey = FALSE,add = TRUE)
```


Z
X
Y