Imperial College London

Coursework

Imperial College London

Department of Mathematics

Statistical Modelling 2

Author:
Ivan Kirev (CID: 01738166)

Date: March 11, 2022

1 Aims

The aim of this report is to provide a statistical analysis of the data from a clinical trial that will help study the effect of the clotting agent concentration on the time taken for blood to clot.

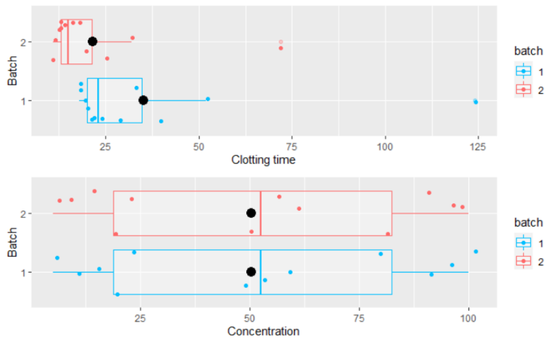
We first explore the given data-set and then we try determine the relationship between clotting time and concentration, by fitting two types of models. The first model is the suggested by the team normal linear model. We review the required assumptions, evaluate the fit and discuss the limitations when using this method. Then we propose a gamma generalised linear model (GLM), explain how it works and why it is more appropriate for the given data and describe our findings in detail.

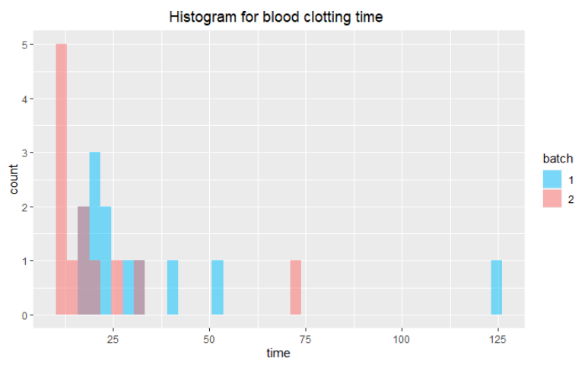
2 Data Exploration

Before we begin fitting our models, we perform an exploratory analysis of the data. The data contains the blood clotting time in seconds, the percentage of maximum clotting agent concentration to which each sample has been diluted and the batch of clotting factor that was used (1 or 2).

We create two box-plots for the clotting time and concentration for each batch. They give us important information for the data such as its mean (black dot), median and quartiles as well as the overall shape of the distribution.

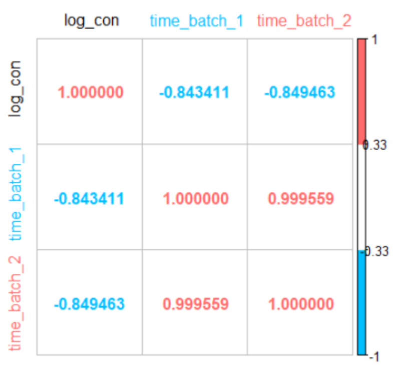
We can see that for each batch the concentration is taken to be the same, evenly spaced from 5 to 100. Also there is an equal number of data-points in each batch, i.e. we are working with a balanced data-set. Looking at the clotting time plots, we can see that while batch 1 appears to produce higher clotting times, both distributions seem very similar.





Since the models we are going to consider differ in the assumed distribution of the response variable, here we have plotted a histogram of the blood clotting time. There are some similarities between the data from the two batches - their distribution is skewed right and overall the time variable does not appear normally distributed. This is one of the reasons we consider the normal linear model to be inappropriate for this data-set. On the other hand, the gamma distribution seems to be a better fit which is why we will consider a gamma GLM.

Lastly, we consider the correlation between our variables. Since we will be working with the log of our concentration, we have plotted the correlation matrix between `log_con`, and the clotting times for each of the two batches. As we can see, the both correlations between `log_con` and `time_batch_1` and between `log_con` and `time_batch_2` are around -0.84 which is very high (in absolute value) suggesting that the relationship we are interested in does indeed exits. Further, we see that the clotting times from each batch are highly correlated (0.9996) indicating that the data-points from the two batches perform very similarly, so it is reasonable to assume they are identically distributed.



	log_con	time_batch_1	time_batch_2
log_con	1.000000	-0.843411	-0.849463
time_batch_1	-0.843411	1.000000	0.999559
time_batch_2	-0.849463	0.999559	1.000000

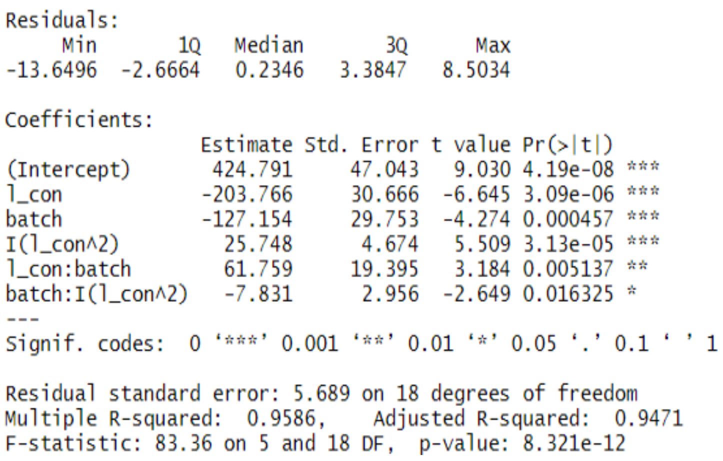
3 Normal Linear Model

We first discuss the initial normal linear model proposed by the experimental team. According to the assumptions of this model, the response variable is normally distributed and the observation errors are uncorrelated and normally distributed with zero mean and constant variance. As we have seen in the previous section, our response variables do not appear normally distributed, which is the first sign that this model is not appropriate. We will see below that the assumption of a constant variance of the errors is also violated.

Now, we fit our linear model by writing in R:

```
mylm <- lm(time ~ l_con*batch + I(l_con^2)*batch)
```

where `l_con` is the logarithm of the concentration variable given in the data and `*` denotes an interaction term. We obtain the following summary.

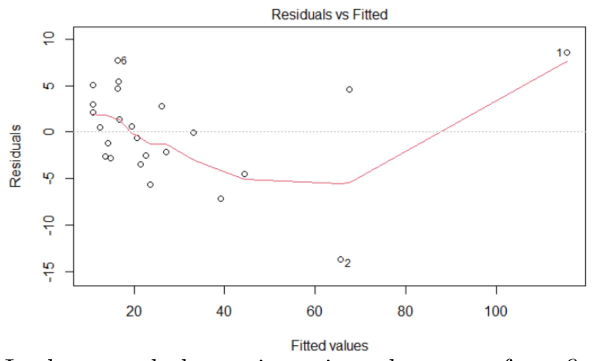


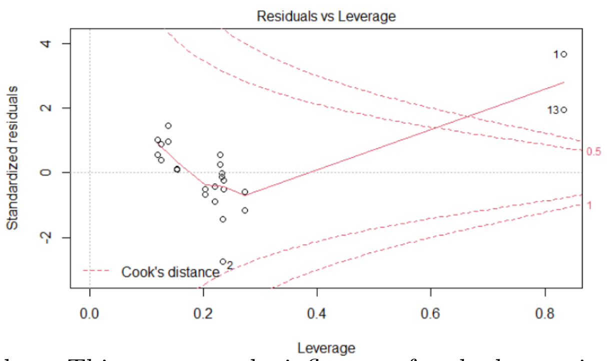
```
Residuals:
     Min       1Q   Median       3Q      Max 
-13.6496  -2.6664   0.2346   3.3847   8.5034 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)       424.791     47.043   9.030 4.19e-08 ***
l_con            -203.766     30.666  -6.645 3.09e-06 ***
batch            -127.154     29.753  -4.274 0.000457 ***
I(l_con^2)         25.748      4.674   5.509 3.13e-05 ***
l_con:batch        61.759     19.395   3.184 0.005137 ** 
batch:I(l_con^2)   -7.831      2.956  -2.649 0.016325 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.689 on 18 degrees of freedom
Multiple R-squared:  0.9586,	Adjusted R-squared:  0.9471 
F-statistic: 83.36 on 5 and 18 DF,  p-value: 8.321e-12
```

First, we notice that R automatically includes the variables `l_con`, `batch` and I(`l_con`^ 2) producing 6 estimated parameters. From their p-values, we can say that there is a strong evidence for an effect of each of the variables (slightly weaker for the interaction term batch:I(`l_con`^ 2), although still significant).

From the adjusted R^2 statistic we can deduce that 95% of the variance in the data is explained by the model, which suggest a good fit of the model. The R^2 statistic alone can be misleading, so now we will look at some plots of the model.

Below we have shown on the left a plot of the residuals against our fitted values. As per our normal theory assumptions, these should follow $N(\mathbf{0}, \sigma^2 I_n)$ for some σ^2 - constant. However, the variance of the residual does not appear to be constant indicating heteroskedasticity and suggesting an inaccuracy of our model.





In the second plot we investigate leverage of our fitted data. This measures the influence of each observation on our model fit using Cook's distance. As we can see, the observations with indices 1 and 13 have both high leverage and high residuals, meaning that they negatively influence our model and suggesting that these points could be outliers.

One could consider removing these two points and then fitting the model again, but this would be unreasonable since (1) the sample size of the data-set is very small with each data-point being important to the model and it is more sensible to think that our model is inappropriate rather than disregarding points as outliers; (2) as we saw in the data exploration part, the clotting time is skewed to the right and the concentration is evenly spaced, meaning that there is a rapid increase in the clotting times as we increase the concentration. Removing the high clotting times would result in disregarding this rapid increase and would render our model appropriate only for a small subset of values for the concentration.

Another violation of the assumptions of the normal linear model is the lack of normality of the distribution of the response variable, which we discussed in the previous section. With that in mind, we now propose a more appropriate model, namely a gamma GLM.

4 Description and Implementation of the GLM

Contrary to the normal linear model where we assumed that the response variable is normally distributed, we now suppose that it takes the more general form of a member of an exponential family, meaning that the distribution of $\boldsymbol{Y}$ is of the form

$$\exp\left\{\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right\}, \tag{1}$$

with $\mathbb{E}[\boldsymbol{Y}] = \boldsymbol{\mu}$. Further, instead of assuming that the response variable $\boldsymbol{Y}$ is linearly dependent on $\boldsymbol{\eta} \equiv X\boldsymbol{\beta}$, we suppose that $X\boldsymbol{\beta} = \boldsymbol{\eta} = g(\boldsymbol{\mu})$, where g is the link function. The general form of the log-likelihood in GLMs is

$$l(\boldsymbol{\beta};\phi,\boldsymbol{y}) = \sum_{i=1}^{n}\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i,\phi)\right\},$$

which depends on $\boldsymbol{\beta}$ through $\mu_i = b(\theta_i)$, $g(\mu_i) = \eta_i$, $\eta_i = \sum_{j=1}^{p} x_{ij}\beta_j$. To maximise the likelihood, we solve $\frac{\partial l}{\partial \beta} = 0$ by using the Iterative weighted least squares (IWLS) algorithm, which works as follows:

1. Given a current estimate $\hat{\beta}$ form the linear predictor $\hat{\eta}$ and the fitted values $\hat{\mu}$.
2. Form the adjusted dependent variable $z_i = \hat{\eta}_i + (y_i - \hat{\mu}_i)\frac{\partial \eta}{\partial \mu}\big|_{\mu=\hat{\mu}_i}$.
3. Form the estimated weights $\tilde{w}_{ii}^{-1} = \left(\frac{\partial \eta}{\partial \mu}\right)^2 b''(\theta)\big|_{\mu=\hat{\mu}_i}$.
4. Regress z_i on $\boldsymbol{x}_i$ with weights w_{ii} and obtain the new estimate $\hat{\beta}$.
5. Repeat steps $1-4$ until convergence.

Now we compute the needed parameters for $Gamma(\alpha_i,\gamma)$, using the canonical link, i.e. $\eta = \theta$, giving us

$$\mu_i = \alpha_i;\ \theta_i = -\frac{1}{\alpha_i};\ a(\phi) = \phi = \frac{1}{\gamma};\ b(\theta_i) = -\log(-\theta_i);\ b'(\theta_i) = -\frac{1}{\theta_i};\ b''(\theta_i) = \frac{1}{\theta_i^2};\ \eta_i = \theta_i = -\frac{1}{\mu_i};$$
$$\frac{\partial \eta_i}{\partial \mu_i} = \frac{1}{\mu_i^2};\ z_i = \eta_i + \frac{(y_i - \mu_i)}{\mu_i^2};\ \tilde{w}_{ii} = \frac{1}{\mu_i^2}.$$

As a stopping rule for the IWLS algorithm we will use the deviance, which is a measure of goodness of fit of a GLM. For a gamma distribution it is given by

$$D = 2\sum_{i=1}^{n}\left[y_i \log\left(y_i/\hat{\mu}_i\right) + (y_i - \hat{\mu}_i)/\hat{\mu}_i\right]$$

and we repeat our steps until the deviance varies by less than 1×10^{-8}.

We note that the algorithm requires an initial starting value of $\hat{\beta}$, which cannot be chosen at random as the algorithm does not converge for every $\hat{\beta}$. We pick our initial betas as the coefficients from a normal linear model where the response variable is transformed by the inverse link function. So for the canonical link, we use the linear model `lm`$(-1/y\ x)$. This is not an accurate model for our data, however the coefficients that are produced are good starting values for our algorithm.

We will fit several models with different linear predictors, so we create a function that takes as an input the response variable y, the predictor variables x and returns (1) the coefficients of the GLM model; (2) the produced deviance; (3) the weights which are needed for computing the standard errors of the coefficients; (4) $\hat{\mu}$ for estimating the parameter ϕ later. The implementation in R is shown below:

```
IWLS <- function(x, y){

  # Inverse link function
  inv_link <- function(u)
    return(-1/u)  # inverse canonical link

  # Compute starting betas
  beta <- lm(inv_link(y) ~ x)$coefficients

  # Deviance function
  deviance <- function(p){
    return(2*sum(-log(y/p) + (y-p)/p))
  }

  # Compute the initial deviance for the initial guess
  old_deviance <- deviance(inv_link(cbind(1,x)%*%beta))

  control <- 1
  while(control >= 1e-8){
    # Estimated linear predictor
    eta <- cbind(1,x)%*%beta
    # Estimated mean response
    mu <- inv_link(eta)
    # Form the adjusted variate
    z <- eta + (y-mu)/mu^2
    # Compute weights
    w = mu^2

    # Regress z on x with weights w
    linear_model <- lm(z ~ x, weights = w)

    # New beta and deviance
    beta <- as.numeric(linear_model$coefficients)
    new_deviance <- deviance(inv_link(cbind(1,x)%*%beta))

    # Compute the difference in deviances
    control <- abs(new_deviance - old_deviance)/(abs(new_deviance)+0.1)
    # Update deviance
    old_deviance <- new_deviance
  }
  # Returns the final coefficients, deviance, weights and mu
  return(list(beta, new_deviance, w, mu))
}
```

5 Evaluating the GLM Model

Now we will construct several models with different linear predictors and compute their parameters using our `IWLS` function. We then compare them using the Aikake's Information Criteria (AIC) and the Residual deviance. We will be working with models which have as covariates:

1. `l_con` and `batch`
2. `l_con`, `batch` and `l_con:batch`
3. `l_con`, `batch` and `I(l_con^ 2):batch`
4. `l_con`, `batch`, `l_con:batch` and `I(l_con^ 2):batch`

Model	1	2	3	4
Residual Deviance	0.43	0.059	0.15	0.054
AIC	130.1	84.7	106.32	84.26

We will first discard models 1 and 3 as both have relatively large deviance. Now looking at models 2 and 4, we can see that they have very similar deviances, but the AIC score for model 2 is slightly larger, indicating that the added variable in model 4, `I(l_con^ 2):batch`, does not have a valuable contribution. Therefore, we choose model 2 as the most optimal.

We now evaluate the gamma glm model with $g(\mu) = X\boldsymbol{\beta}$, where g is the canonical link function and X contains an intercept column, `l_con`, `batch` and `l_con*batch`. First we calculate the standard errors of the fitted coefficients given by $\sqrt{\phi(X^TWX)^{-1}_{ii}} = \sqrt{\text{Var}(\hat{\beta}_i)}$. We estimate ϕ by $\hat{\phi} = \frac{X^2}{n-p}$, where $X^2 = \sum_{i=1}^{n} \frac{(y-\hat{\mu}_i)^2}{\hat{mu}_i^2}$ is the Pearson's statistic and $n-p$ are the degrees of freedom. We then perform a hypothesis test with null hypothesis $\hat{\beta}_i = 0$, using the fact that under H_0: $\frac{\hat{\beta}_i}{\sqrt{\text{Var}(\hat{\beta}_i)}} \sim N(0,1)$. We produce the results in the following table:

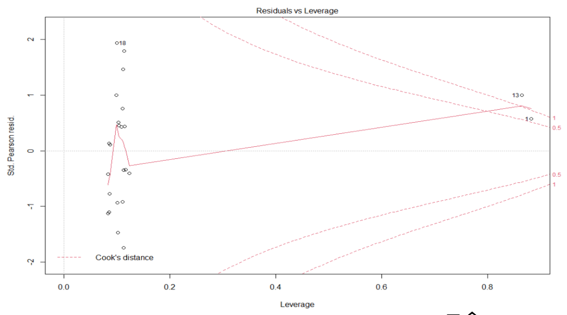
	β_0	β_1	β_2	β_3
Coefficients	0.0096	-0.0073	0.0072	-0.0082
Standard errors	0.0024	0.001002	0.0018	0.00074
p-values	7.67e-04	5.12e-07	6.91e-04	5.22e-10

We can see that the p-values are very low for all four coefficients, meaning that we reject the null hypothesis (i.e. we reject that $\hat{\beta}_i = 0$ for $i = 0, 1, 2, 3$), and more specifically, there is evidence that the relationship between concentration and clotting time is different between the two batches. The code producing the values is given in the Appendix, but can also be obtained automatically in R by using the glm function:

```
myglm <- glm(time ~ l_con+batch+l_con:batch, family = Gamma(link="inverse"))
summary(myglm)
```

We note that in the glm function we used the inverse link, whereas in our model we have used the canonical link. This is because for the Gamma distribution, the inverse link ($g(x) = 1/x$) is almost the same as the canonical link ($g(x) = -1/x$). This is why the coefficients from the summary of the function used above have opposite signs from our coefficients (the rest of the results remain the same).

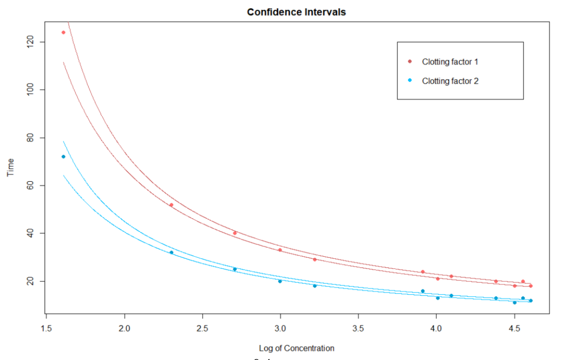
In the plot on the right we investigate the leverage of our fitted data. As in the normal linear model, we still have points with high leverage, however this time, they produce lower standardised residuals, indicating that the gamma GLM model is a good fit overall, whereas the normal linear one was appropriate only for points with high concentration and low clotting times.



We now produce 95% confidence intervals. Suppose we are given covariates $\boldsymbol{x}_*$, then we have $\hat{\eta}_* = \boldsymbol{x}_*^T \hat{\beta}_*$ with variance $\boldsymbol{x}_*^T \hat{\phi}(X^T W X)^{-1} \boldsymbol{x}_*$, and using that asymptotically we have $\frac{\hat{\beta}_i}{\sqrt{\mathrm{Var}(\hat{\beta}_i)}} \sim N(0, 1)$, we can construct an approximate confidence interval for η using the normal distribution. Then to obtain an approximate confidence interval for μ we use the inverse link function, giving us

$$\left(g^{-1}\left(\hat{\eta}_* - 1.96\sqrt{\boldsymbol{x}_*^T \hat{\phi}(X^T W X)^{-1} \boldsymbol{x}_*}\right), g^{-1}\left(\hat{\eta}_* - 1.96\sqrt{\boldsymbol{x}_*^T \hat{\phi}(X^T W X)^{-1} \boldsymbol{x}_*}\right)\right).$$

On the right we have produced the confidence interval for concentration in the range $(5, 100)$ for each of the two batches (the upper and lower red lines represent the upper and lower bounds for clotting factor 1 and similarly for clotting factor 2) together with the raw data. We notice that the width of the intervals is not constant, but increases as the time of clotting increases, which accounts for the heteroskedasticity in our data. This is one of the advantages of the GLM model - it does not assume constant variance.



In particular, if an individual receives a 50% dose of each of the two agents, the 95% confidence intervals for the clotting time is

$$\begin{aligned} &(22.07, 23.61) \quad \text{for clotting agent 1,} \\ &(14.06, 15.05) \quad \text{for clotting agent 2.} \end{aligned}$$

We now discuss some of the limitations of the model. The main concern is that the sample size is very small - only 24 data-points. This is a problem because we used many approximate distributions throughout the report for performing hypothesis tests and estimating the confidence intervals as well as the dispersion parameter, which are only accurate asymptotically, i.e. for larger data-sets.

6 Summary

Our conclusion is that the gamma GLM model, taking as covariates `l_con`, `batch` and `l_con*batch`, is a more appropriate model for the given data than a normal linear model. The latter model assumes normal distribution for the response variables and a constant variance of the residuals which are both seen to be inappropriate. The gamma distribution is a much better fit for the response variables and the GLM model copes with heteroskedasticity - we observe small variance of the response variable for small values of the clotting time and larger variance for larger times. We have put most of the code used for the evaluation of the model in the Appendix.

Appendix

In this appendix we include the code that we have used to produce the results in fitting and evaluating the gamma GLM model.

We first read-in the data

```
data <- read.csv("1738166.csv")
l_con = log(data$concentration)
batch = data$batch
time = data$time
```

Now we create our algorithm which takes as an input the response variable y, the predictor variables x and returns the coefficients of the GLM model, the produced deviance, the weights which are needed for computing the standard errors of the coefficients and $\hat{\mu}$ for estimating the parameter ϕ later.

```
IWLS <- function(x, y){

  # Inverse link function
  inv_link <- function(u)
    return(-1/u)  # inverse canonical link

  # Compute starting betas
  beta <- lm(inv_link(y) ~ x)$coefficients

  # Deviance function
  deviance <- function(p){
    return(2*sum(-log(y/p) + (y-p)/p))
  }

  # Compute the initial deviance for the initial guess
  old_deviance <- deviance(inv_link(cbind(1,x)%*%beta))

  control <- 1
  while(control >= 1e-8){
    eta <- cbind(1,x)%*%beta  # estimated linear predictor
    mu <- inv_link(eta)  # estimated mean response
    z <- eta + (y-mu)/mu^2  # form the adjusted variate
    w = mu^2  # compute weights
    linear_model <- lm(z ~ x, weights = w)  # regress z on x with weights w
    beta <- as.numeric(linear_model$coefficients)  # new beta
    new_deviance <- deviance(inv_link(cbind(1,x)%*%beta))

    # Compute the difference in deviances
    control <- abs(new_deviance - old_deviance)/(abs(new_deviance)+0.1)
    old_deviance <- new_deviance  # update deviance
  }
  # Returns the final coefficients, deviance and weights
  return(list(beta, new_deviance, w, mu))
}
```

Now we produce the four models with different covariates.

```r
# Create covariate matrices
x_1 = cbind(l_con, batch)
x_2 = cbind(l_con, batch, l_con*batch)
x_3 = cbind(l_con, batch, I(l_con^2)*batch)
x_4 = cbind(l_con, batch, l_con*batch, I(l_con^2)*batch)
```

Print the deviances for the four models

```r
for (x in list(x_1, x_2, x_3, x_4)){
  print(IWLS(x, time)[2])
}
```

```
## [[1]]
## [1] 0.4291603
##
## [[1]]
## [1] 0.05966945
##
## [[1]]
## [1] 0.1468703
##
## [[1]]
## [1] 0.05393168
```

Now we work with model with covariate matrix x_2. We print its coefficients, compute their standard errors and p-values:

```r
# Model coefficients
betas = IWLS(x_2, time)[[1]]
print(betas)
```

```
## [1]  0.009561940 -0.007268502  0.007223176 -0.008225702
```

```r
# Standard errors

# Compute w and mu
weights = IWLS(x_2, time)[[3]]
mu = IWLS(x_2, time)[[4]]

# Compute X^TWX
X = cbind(1,x_2)
J = t(X)%*% diag(as.vector(weights))%*%X

# Estimate phi
phi = sum((time-mu)^2/mu^2)/20  # degrees of freedom = 20

# Compute variance-covariance matrix
invJ = phi*solve(J)

# Compute the standard errors
standard_errors_betas = sqrt(as.vector(diag(invJ)))
print(standard_errors_betas)
```

```
## [1] 0.0024128959 0.0010021385 0.0018022842 0.0007402506
```

Now we compute the p-values for the null hypothesis $\beta_i = 0$:

```r
# Under H_0: z follows a standard normal distribution
z <- betas/standard_errors_betas
print(2*pt(-abs(z), df = 20))
```

```
## [1] 7.673621e-04 5.120551e-07 6.908254e-04 5.222858e-10
```

Alternatively, we could have used the in-built glm function in R, which we check that produces the same results:

```r
myglm <- glm(time ~ x_2, family = Gamma(link="inverse"))
summary(myglm)
```

```
##
## Call:
## glm(formula = time ~ x_2, family = Gamma(link = "inverse"))
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.092857  -0.042716   0.006296   0.024047   0.097367
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0095619  0.0024129  -3.963 0.000767 ***
## x_2l_con     0.0072685  0.0010021   7.253 5.12e-07 ***
## x_2batch    -0.0072232  0.0018023  -4.008 0.000691 ***
## x_2          0.0082257  0.0007403  11.112 5.22e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.002994805)
##
##     Null deviance: 10.020445  on 23  degrees of freedom
## Residual deviance:  0.059669  on 20  degrees of freedom
## AIC: 84.685
##
## Number of Fisher Scoring iterations: 4
```

We have used the inverse link, but as explained in the report, our estimates are for the canonical link, which is the negative inverse link. This is why the coefficients have opposite signs (the models are equivalent since the negative signs cancel out when applying the canonical link to the response variables).

We now produce the confidence intervals:

```r
# Form the covariates for the two batches
x_star_1 = c(1, log(50), 1, log(50))  # covariate for 50% dose in batch 1
x_star_2 = c(1, log(50), 2, 2*log(50))  # covariate for 50% dose in batch 2

# Confidence interval for batch 1 for 50% concentration
lower_1 = -1/(t(x_star_1)%*%betas - 1.96*sqrt(t(x_star_1)%*%invJ%*%x_star_1))
upper_1 = -1/(t(x_star_1)%*%betas + 1.96*sqrt(t(x_star_1)%*%invJ%*%x_star_1))
CI_1 = c(lower_1, upper_1)

# Confidence interval for batch 2 for 50% concentration
lower_2 = -1/(t(x_star_2)%*%betas - 1.96*sqrt(t(x_star_2)%*%invJ%*%x_star_2))
upper_2 = -1/(t(x_star_2)%*%betas + 1.96*sqrt(t(x_star_2)%*%invJ%*%x_star_2))
CI_2 = c(lower_2, upper_2)
```

```
# Print CIs
print(CI_1)
```

```
## [1] 22.07125 23.61313
```

```
print(CI_2)
```

```
## [1] 14.06240 15.04723
```