MATH501 Modelling and Analytics for Data Science Coursework

10684079

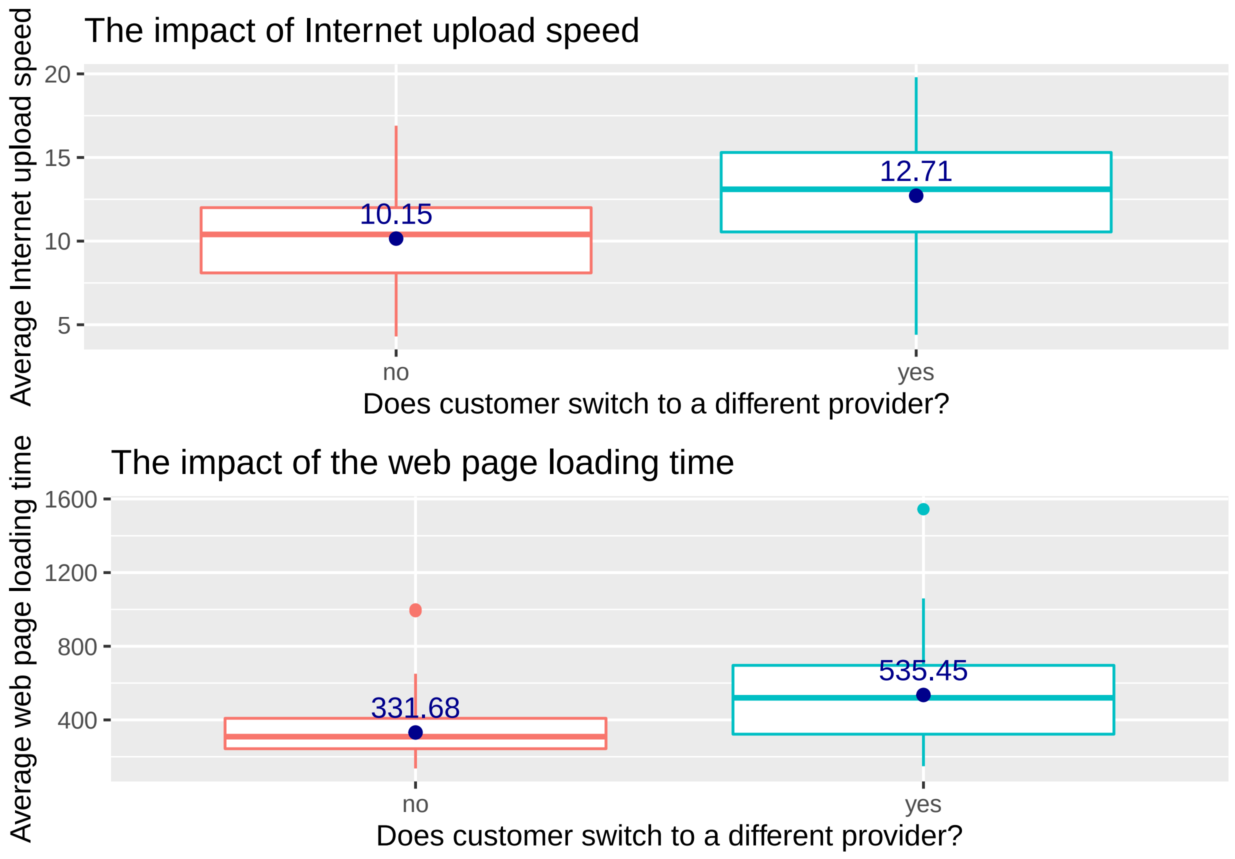
26/04/2021

1 Machine Learning Task

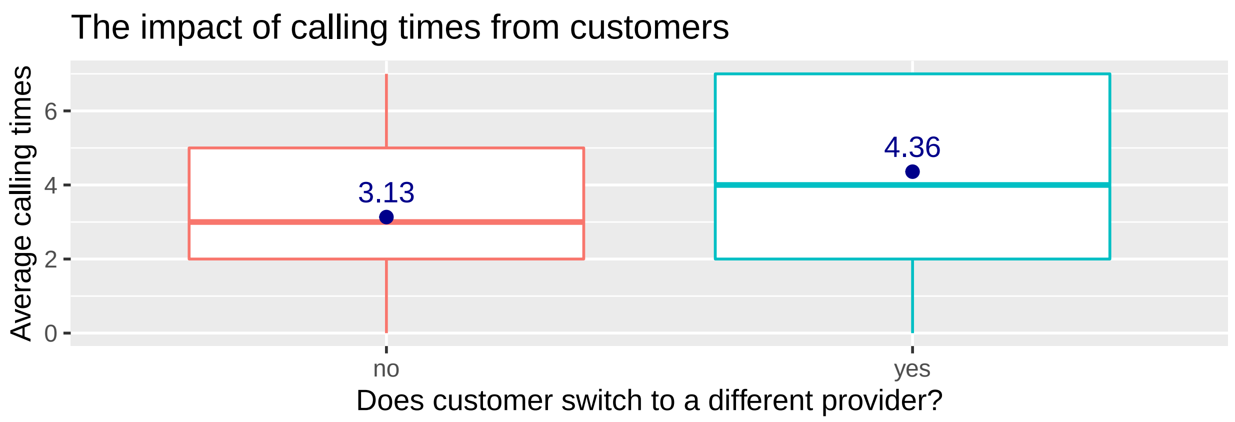
1.1 Machine Learning Part (a)**:

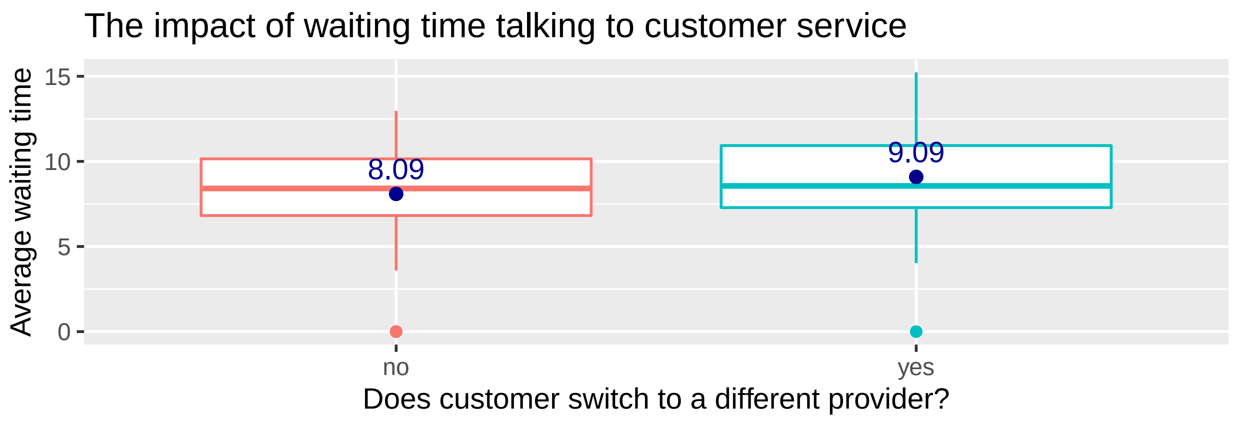
Present the data visually using box-and-whisker plots with a distinction for churn.

```r
setwd("/Users/xuehanyin/coursework/Math501/cw")

tele <- read.table("/Users/xuehanyin/coursework/Math501/cw/churndata.txt")

library(ggplot2)
library(ggpubr)

upload_churn <- ggplot(tele, aes(y = upload, x = churn, col = churn))+
  geom_boxplot() +
  labs(title = "The impact of Internet upload speed",
       x = "Does customer switch to a different provider?",
       y = "Average Internet upload speed") +
  stat_summary(fun = mean,
               color = "darkblue",
               geom = "point",
               shape = 20,
               size = 3,
               show.legend = FALSE) +
  stat_summary(fun = mean,
               color = "darkblue",
               geom = "text",
               show.legend = FALSE,
               vjust = -0.7,
               aes(label = round(..y.., digits = 2))) +
  theme(legend.position = "none")

webget_churn <- ggplot(tele, aes(y = webget, x = churn, col = churn))+
  geom_boxplot() +
  labs(title = "The impact of the web page loading time",
       x = "Does customer switch to a different provider?",
       y = "Average web page loading time") +
  stat_summary(fun = mean,
               color = "darkblue",
               geom = "point",
               shape = 20,
               size = 3,
               show.legend = FALSE) +
  stat_summary(fun = mean,
               color = "darkblue",
               geom = "text",
               show.legend = FALSE,
               vjust = -0.7,
               aes(label = round(..y.., digits = 2)))+
  theme(legend.position = "none")
ggarrange(upload_churn, webget_churn,
          nrow = 2, ncol = 1)
```



As we can see, the average Internet uploading speed is 12.71 for customers who switch to a different provider and is 10.15 for the other group of customers who stay with the current provider. It seems that the customers with faster internet upload speed are tended to switch to different providers. It may indicate that customers are less concerned with the Internet uploading speed. The customers with longer web page loading time are tended to switch to different providers.





Every customer who switched provider contacted the company via phone call more than 4 times. Each customer who did not switch company call the company approximately 3 times. It seems the customers who call the company more times are tended to switch provider. Furthermore, the customers who are waiting longer to talk to a customer service operator are tended to switch to a different provider. To conclude, customers are more concerned with the web page loading time and customer services.

1.2 Machine Learning Part (b)∗:

Create a training set consisting of 350 randomly chosen data points and a test set consisting of the remaining 150 data points.

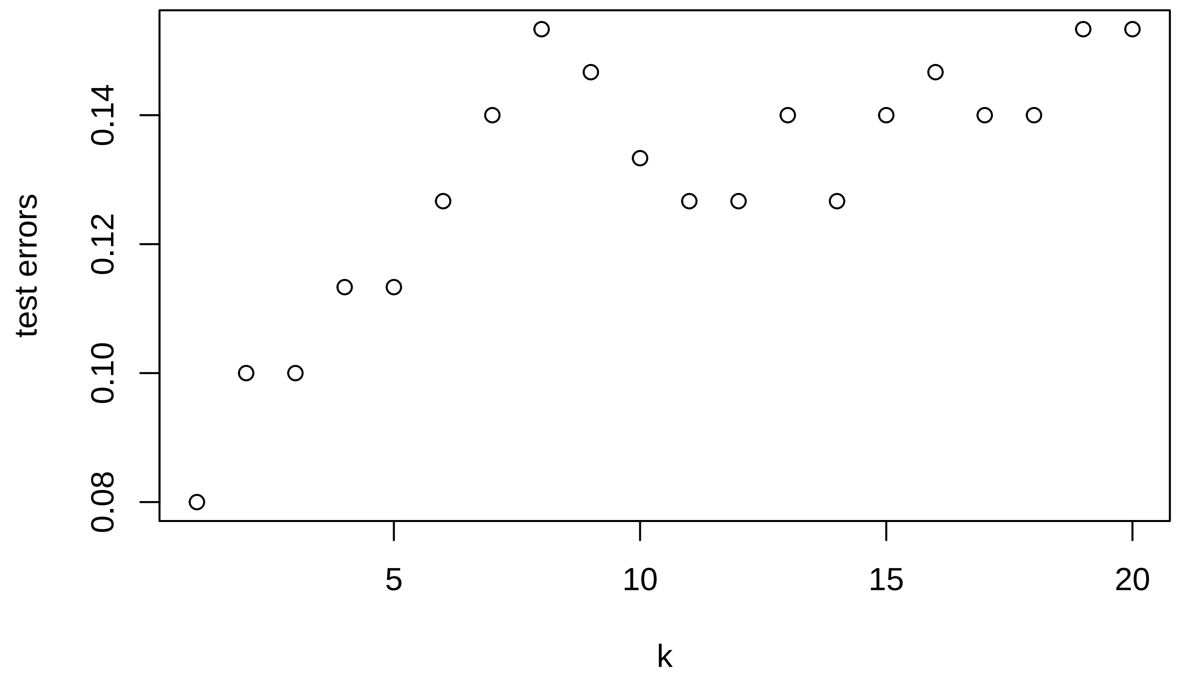
```r
library(dplyr)
# Make the the column churn factors
tele_1 <- tele %>% mutate(churn_1 = factor(tele$churn,
                                            levels = c("no","yes")))
# set class
cl <- tele_1$churn_1

# predictors
train <- data.frame(cbind(upload = tele_1$upload, webget =tele_1$webget,
               enqcount = tele_1$enqcount, callwait = tele_1$callwait))

# standardize the predictors
train <- scale(train)
```

```r
set.seed(1)
train_split <- sample(500,350) # randomly pick 350 numbers as train set

# train set predictors and class
train_pre <- train[train_split, ]
train_cl <- cl[train_split]

#test set predictors and class
test_pre <- train[-train_split,]
test_cl <- cl[-train_split]
```

1.3 Machine Learning Part (c)∗∗∗:

Using the training data set apply the K nearest neighbours method to construct a classifier to predict churn based on the four available predictors.

```r
library(class)

# fix the result every time run the code
set.seed(2)

test.error <- function(k){
  knn.k <- knn(train = train_pre, test = test_pre, cl = train_cl, k=k)
  tab <- table(knn.k, test_cl)
  error <- (tab[1,2] + tab[2,1]) / sum(tab)
  return(error)
}

set.seed(3)
errors <- rep(0,20)

for (i in 1:20) errors[i] <- test.error(k=i)

plot(errors, xlab = "k", ylab = "test errors")
```



```
# test error
t_1 <- test.error(k=1)
t_1
```

```
## [1] 0.08
```

```
t_2 <- test.error(k=3)
t_2
```

```
## [1] 0.1
```

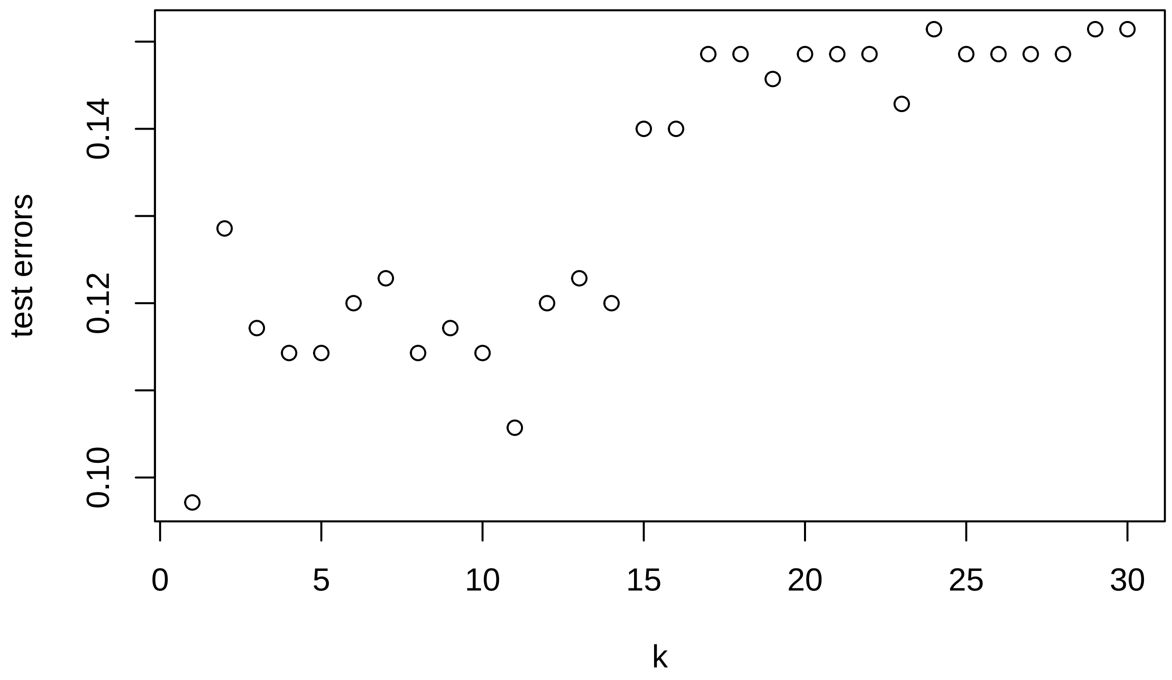
```
knn.k2 <- knn(train = train_pre, test = test_pre, cl = train_cl, k=3)
```

When k value is 1, it provides the lowest test error rate which is 8%. To prevent over-fitting, the better k value would be 3 with test error rate 10%.

Find the best k using leave-one-outcross-validation for the training data set.

```
set.seed(4)

# Use k-Nearest Neighbour Cross-Validatory Classification to find the best k
test.error.cv <- function(k){

  knn.cvv <- knn.cv(train = train_pre, train_cl, k=k)
  tab <- table(knn.cvv, train_cl)
  error.cv <- (tab[1,2] + tab[2,1]) / sum(tab)
  return(error.cv) # save all the results

}

# collect the results for 20 times
errors.cv <- rep(0,20)
for (i in 1:30) errors.cv[i] <- test.error.cv(k=i)
```

```r
# visualise the results
plot(errors.cv, xlab = "k", ylab = "test errors")
```



```r
t_3 <- test.error.cv(k = 11)
t_3 # The best k is 11 with test error 0.1057143
```

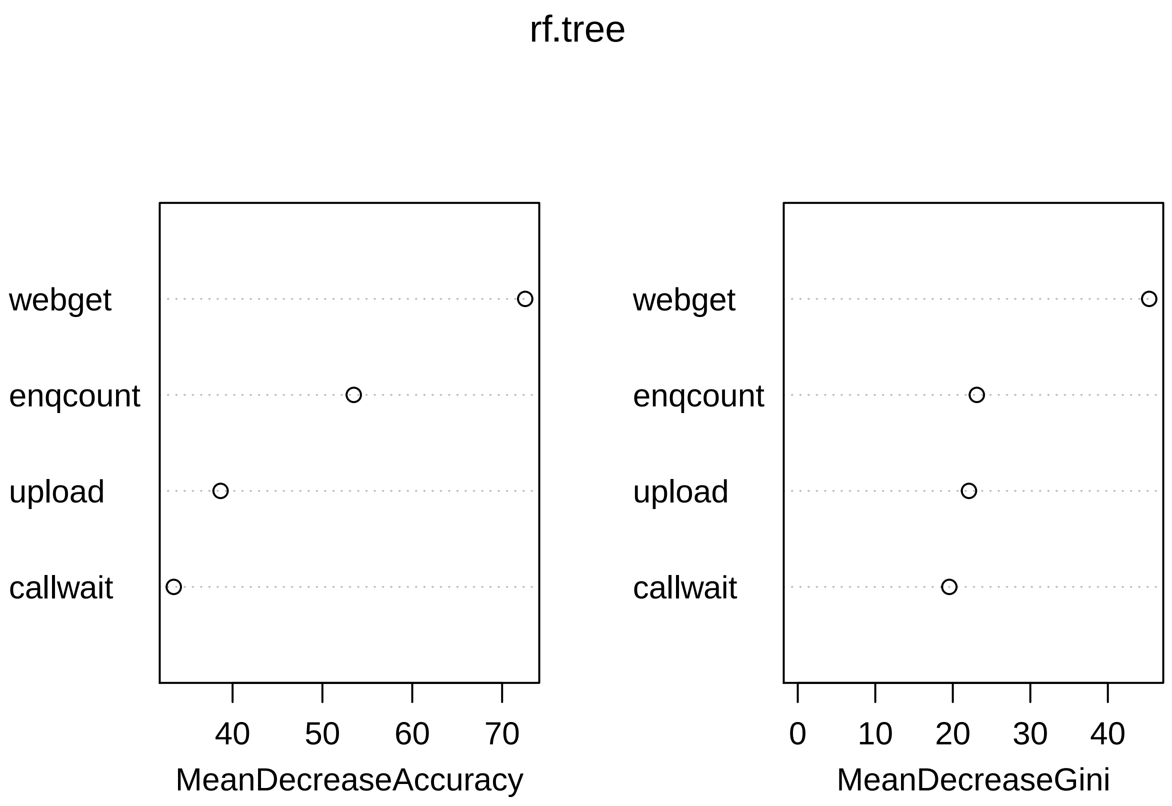
```
## [1] 0.1057143
```

In most cases, when k is 1 in k-NN method, it leads to over-fitting. As we can see from the graph, the test error rate is the lowest 11% when k is 11 apart from when k value is 1.

1.4 Machine Learning Part (d)**:

Using the training data set apply the random forest (bagging) method to construct a classifier to predict churn based on the four available predictors.

```r
df.tree <- tele_1[c(1,2,3,4,6)] # predictors with class labels

# test set
test.tree <- df.tree[-train_split,]
# class for test set
test.cl.tree <- df.tree$churn_1[-train_split]

library(randomForest)

set.seed(5)
rf.tree <- randomForest(churn_1 ~ ., data = df.tree, subset = train_split,
                        mtry = 2, importance = TRUE)

varImpPlot(rf.tree)
```



Importance of the four variables for predicting churn: web loading time is the most significant and Internet uploading speed is the least significant when it comes to classification of customers who switch the provider.

```
rf.pre <- predict(rf.tree, test.tree)
rf.tab <- table(rf.pre, test.cl.tree)
rf.tab
```

```
##        test.cl.tree
## rf.pre  no yes
##     no  114   3
##     yes   2  31
```

```
rf.error <- (rf.tab[1,2]+rf.tab[2,1]) / sum(rf.tab)
rf.error
```

```
## [1] 0.03333333
```

The **test error** is 3.33% by using random forest method which is lower than the test error of using k-NN method which is 10.57%.

1.5 Machine Learning Part (e)**:

Using the entire data set (training set and test set combined), perform Principal Component Analysis for the four variables: upload, webget, enqcount and callwait.

```r
pca <- princomp(train, cor = TRUE)
summary(pca)
```

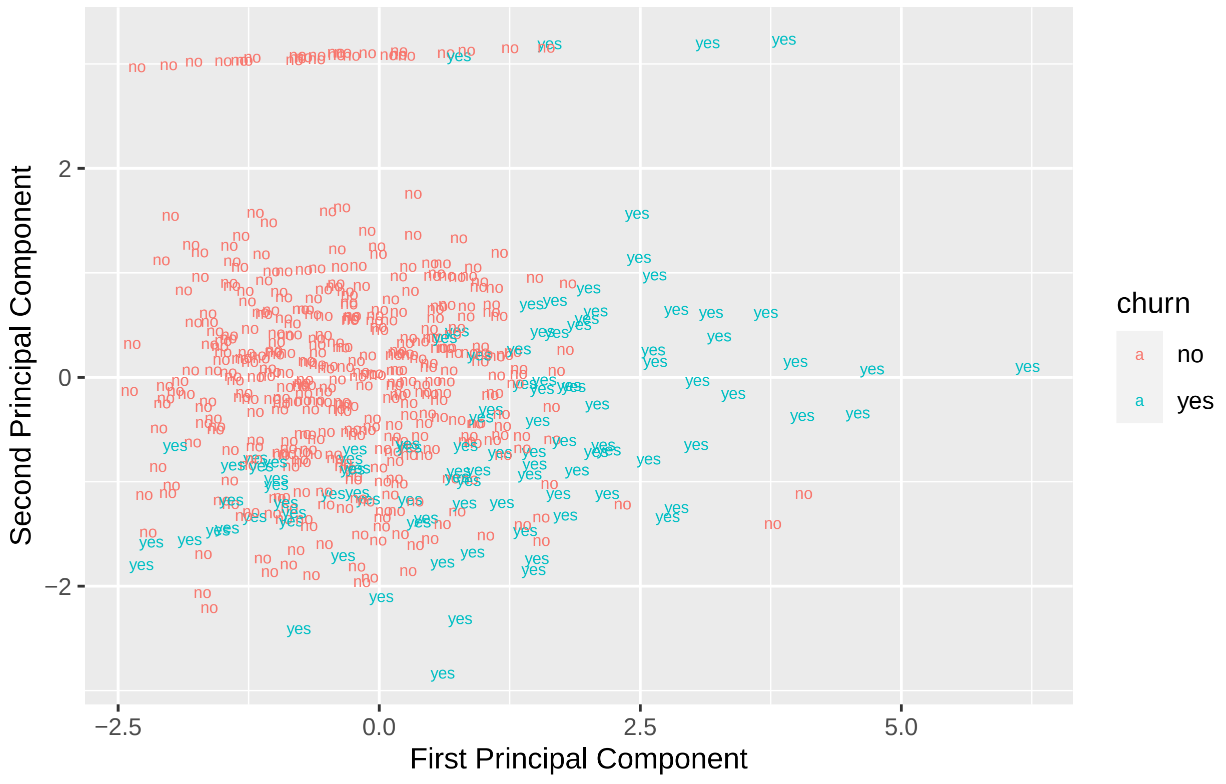
```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4
## Standard deviation     1.2891552 1.1307782 0.8497143 0.58086584
## Proportion of Variance 0.4154803 0.3196648 0.1805036 0.08435128
## Cumulative Proportion  0.4154803 0.7351451 0.9156487 1.00000000
```

```r
s <- pca$sdev^2/sum(pca$sdev^2)
```

The results tell us that the new variable in Component 1 accounts for 41.55% and Component 2 accounts for 31.97% for the information or variance in the data. Cumulative percentage is 73.51%.

Using principal components, create the "best" two dimensional view of the data set and use colour coding to indiciate the churn..

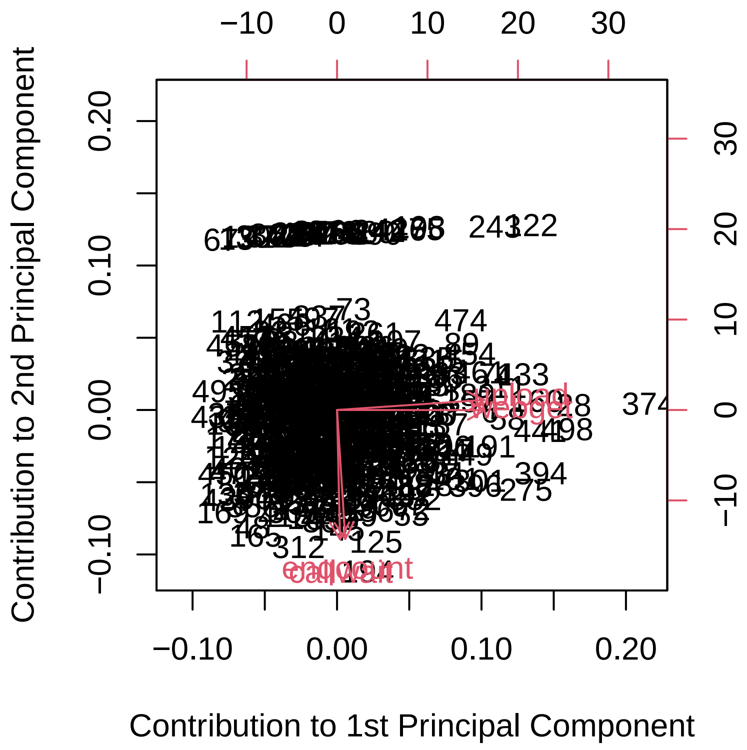
```r
# new variables
new_v <- data.frame(pca$scores)

#put column Churn into the comp dataframe
new_v$churn <- tele_1$churn_1

ggplot(new_v, aes(x = Comp.1, y = Comp.2, label = churn, col = churn))+
  geom_text(size = 2) +
  labs(x = "First Principal Component",
       y = "Second Principal Component") +
  coord_fixed(ratio = 1)
```



From the first principal component, factors make customers who switch provider varies more than the customers who stay with the current provider.

An interpretation of the first two principal components.

```r
biplot(pca,
       xlab = "Contribution to 1st Principal Component",
       ylab = "Contribution to 2nd Principal Component")
```



All the 4 variables makes a positive contribution to the 1st principal component. Factors of call waiting time and times make negative contributions to the 2nd principal component. Factors of uploading time and wed loading time contribute nearly 0 to the 2nd principal component.

1.6 Machine Learning Part (f)***:

Apply the random forest (bagging) method to construct a classifier to predict churn based on the two first principal components as predictors using the split of the data into a training and test set (the same indices as in part (b).

```r
pca_df <- data.frame(pca$scores[,c(1,2)], cl) # new variables

p_train <- pca_df[train_split, ] # train set with same indices as in part (b)

p_test <- pca_df[-train_split, ] # test set
p_cl <- cl[-train_split] # test class label

# Apply random forest method
p_rf <- randomForest(cl ~ ., data = p_train)
```

```
p_pre <- predict(p_rf,p_test)
p_tab <- table(p_pre, p_cl)
p_tab
```

```
##      p_cl
## p_pre no yes
##   no  99  13
##   yes 17  21
```

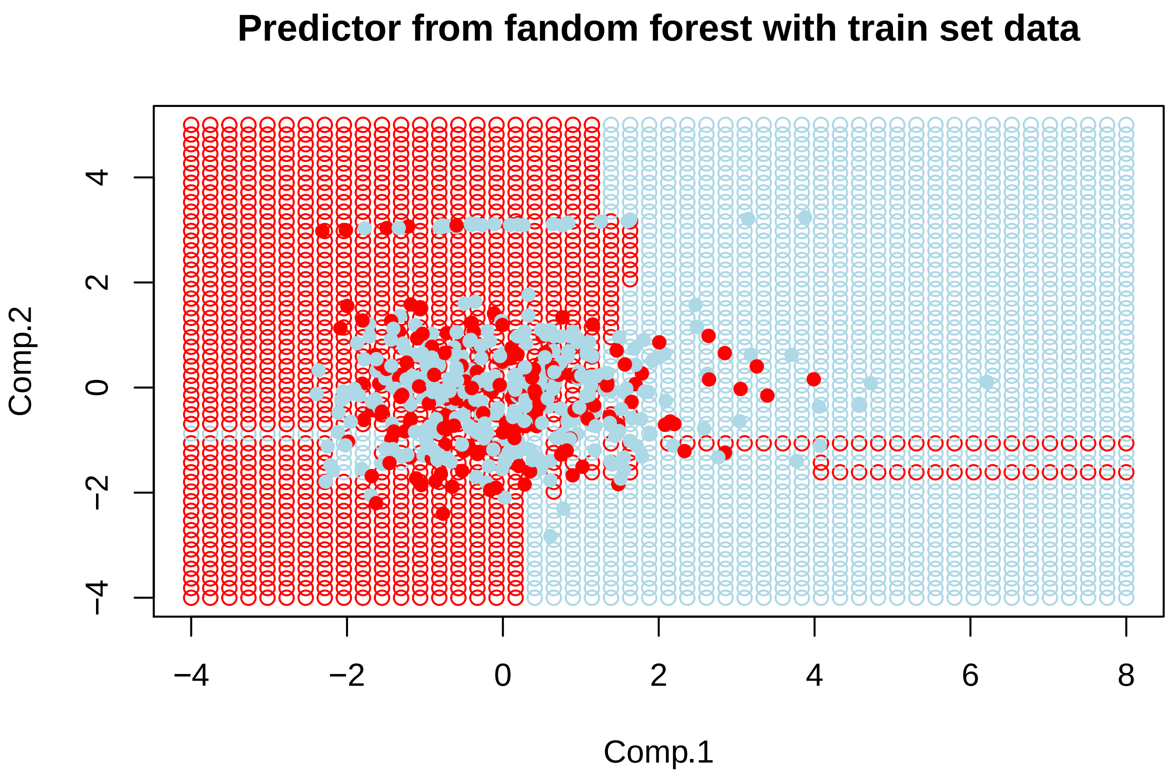
```
p_error <- (p_tab[1,2]+p_tab[2,1]) / sum(p_tab)
p_error
```

```
## [1] 0.2
```

The test error is 0.2 using two first principal components as predictors in random forest method which is higher than the test error 0.0333333 by only using random forest.

Visualise the resulting classification rule on the scatter plot of the two first principal components.

```
p_df <- data.frame(p_test, p_pre)
len <- 50
xp <- seq(-4, 8, length = len)
yp <- seq(-4, 5, length = len)
xygrid <- expand.grid(Comp.1 = xp, Comp.2 = yp)

grid.rf <- predict(p_rf, xygrid)

col3 <- rep("lightgreen", len*len)

for (i in 1:(len*len)){
  if (grid.rf[i]== 'no') col3[i] <- "red"
  else if (grid.rf[i]== 'yes') col3[i] <- "lightblue"
}
plot(xygrid, col = col3, main = "Predictor from fandom forest with train set data")
points(pca_df$Comp.1, pca_df$Comp.2, col = col3, pch = 16)
```


Predictor from fandom forest with train set data
Comp.2
Comp.1
4
2
0
−2
−4
−4
−2
0
2
4
6
8

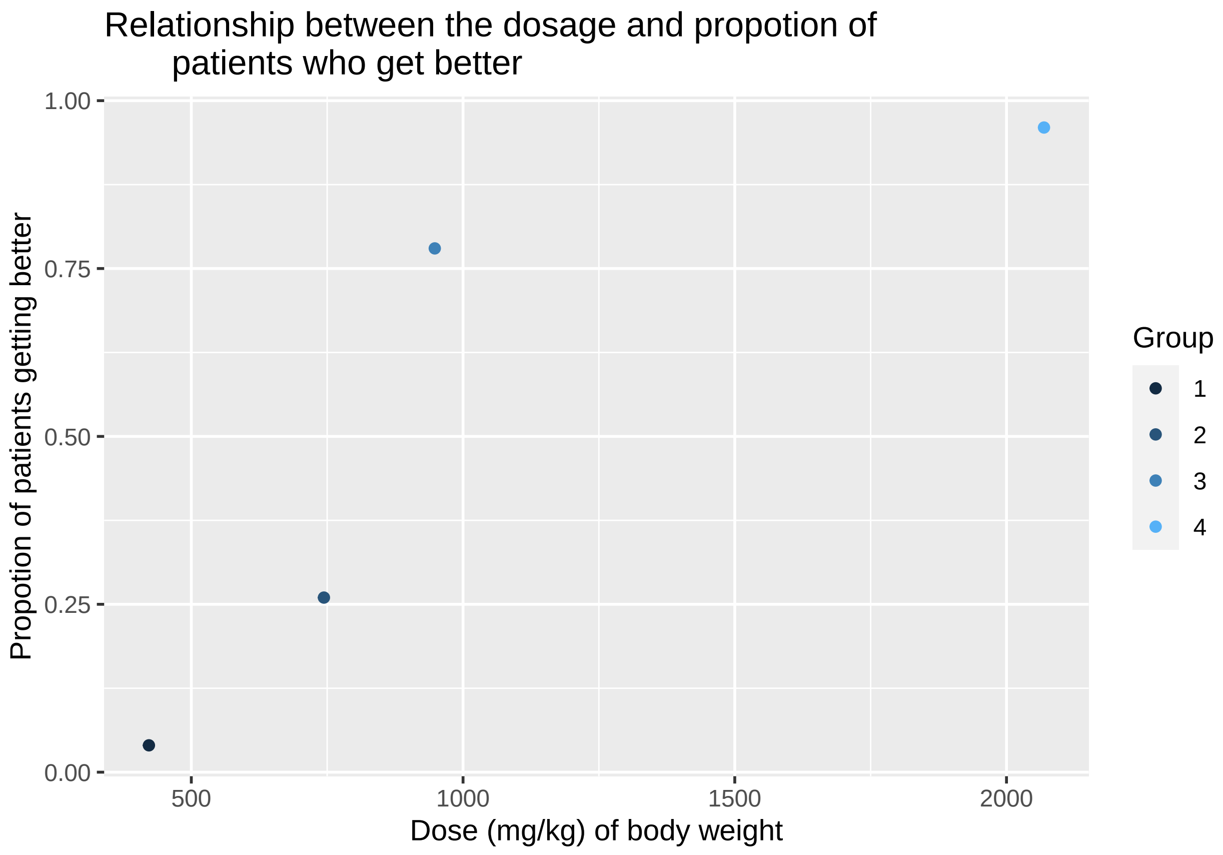
2 Statistical Modelling Task

2.1 First Sub-Task: Frequentist Binary Logistic Regression

2.1.1 Statistical Modelling Part (a)∗:

Calculate the proportion of patients who gets better with the new medicine and visualize these data.

```
Patient_group <- c(1,2,3,4)
Dose <- c(422,744,948,2069)
Number_treated <- c(50,50,50,50)
Number_recovered <- c(2,13,39,48)

#data frame of the patients
df_p <- data.frame(Patient_group, Dose, Number_treated, Number_recovered)

df_p$Propotion_recovered <- Number_recovered/Number_treated

ggplot(df_p, aes(x = Dose, y = Propotion_recovered, col = Patient_group))+
  geom_point()+
  labs(title = "Relationship between the dosage and propotion of
       patients who get better",
       x = "Dose (mg/kg) of body weight",
       y = "Propotion of patients getting better") +
  guides(col=guide_legend("Group"))
```



The proportion of patients who receive 422mg/kg of body weight and get better is subtle in group 1. Approximately a quarter of patients in group 2 get better with receiving doses of 744mg/kg of body weight.

More than 75% of patients recovered with receiving doses of 948mg/kg of body weight in group 3. Almost all patients get better with receiving doses over 2000mg/kg of body weight in group 4.

2.1.2 Statistical Modelling Part (b)*

An analyst adopts the following model for the Covid-19 data:

$$y_i \sim Bin(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\eta_i = \beta_0 + \beta_1 d_i$$

```
# use maximum likelihood function to estimate beta_0 and beta_1
# make a matrix of number recovered and number not recovered for using glm
Estimate_beta <- glm(cbind(Number_recovered,
                                    Number_treated-Number_recovered) ~ Dose,
                              family = binomial,
                              data = df_p)
Estimate_beta
```

```
##
## Call:  glm(formula = cbind(Number_recovered, Number_treated - Number_recovered) ~
##     Dose, family = binomial, data = df_p)
##
## Coefficients:
## (Intercept)         Dose
##   -4.559752     0.005272
##
## Degrees of Freedom: 3 Total (i.e. Null);  2 Residual
## Null Deviance:       133.6
## Residual Deviance: 19    AIC: 36.27
```

```
coef(Estimate_beta)
```

```
##  (Intercept)         Dose
## -4.559752119  0.005271615
```

The results show that $\widehat{\beta_0}$ is -4.5597521 and $\widehat{\beta_1}$ is 0.0052716.

Fit β_0 and β_1 into the model:

$$y_i \sim \text{Bin}(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\widehat{\beta_0} + \widehat{\beta_1} * Dose = -4.5597521 + 0.0052716 * Dose$$

2.1.3 Statistical Modelling Part (c)***:

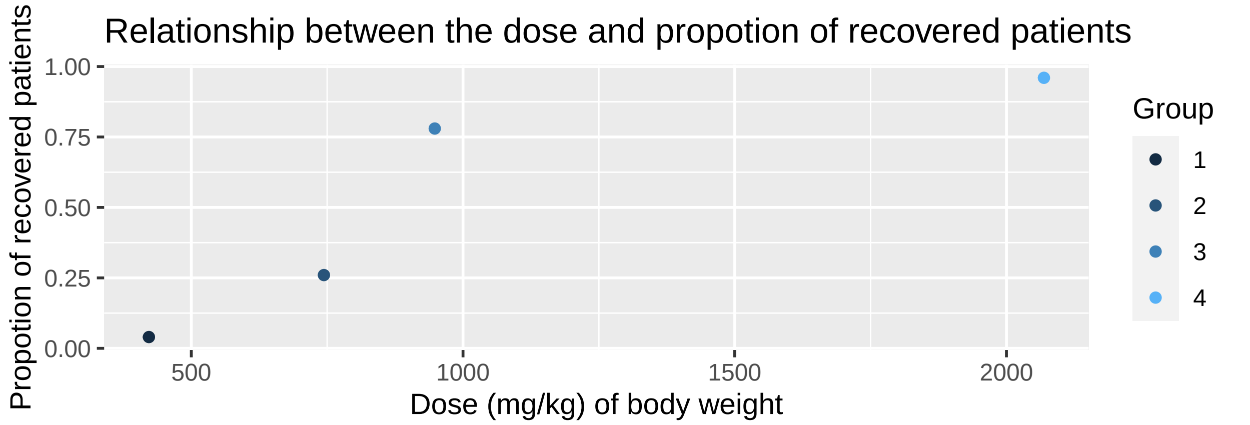
Another analyst adopts the following logarithmic model for the Covid-19 data:

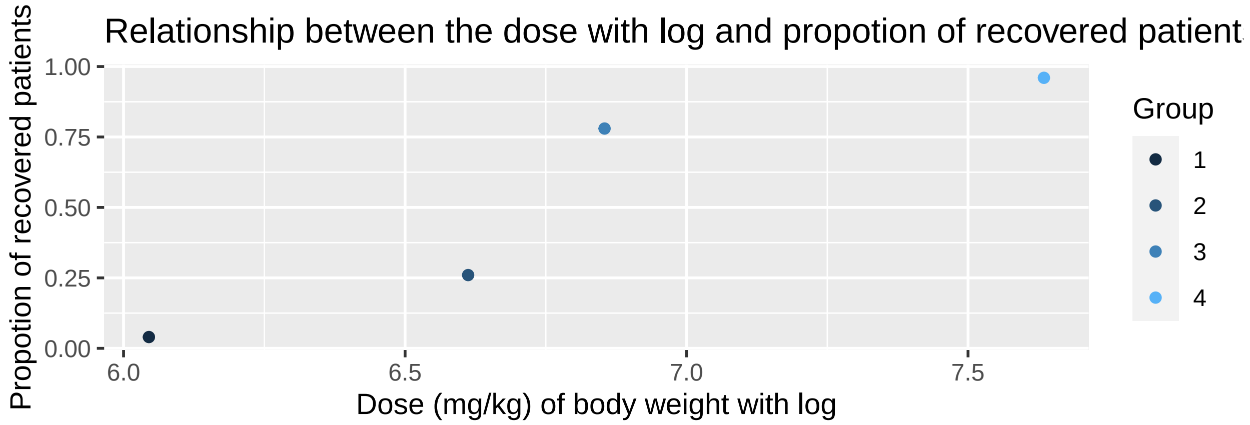
$$y_i \sim Bin(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\eta_i = \beta_0 + \beta_1 \log(d_i)$$

Visualise $\log(d_i)$ against the proportion of Covid-19 patients who gets better and compare the two plots.





The data points are better distributed with the logarithmic model.

Report $\widehat{\beta_0}$ and $\widehat{\beta_1}$.

```r
Estimate_beta_log <- glm(cbind(Number_recovered,
                               Number_treated-Number_recovered) ~ log(Dose),
                     family = binomial,
                     data = df_p)
coef(Estimate_beta_log)
```

```
## (Intercept)   log(Dose)
##  -32.638996    4.852825
```

$\widehat{\beta_0}$ is -32.6389963 and $\widehat{\beta_1}$ is 4.8528254.

Fit β_0 and β_1 into the model:

$$y_i \sim Bin(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\widehat{\beta_0} + \widehat{\beta_1} * \log(Dose) = -32.6389963 + 4.8528254 * \log(Dose)$$

Calculate the 95% confidence intervals for $beta_0$ and $beta_1$.

```
library(MASS)
Estimate_beta_log_confint <- confint(Estimate_beta_log)
Estimate_beta_log_confint
```

```
##                  2.5 %    97.5 %
## (Intercept) -44.701647 -23.66464
## log(Dose)     3.515515   6.64896
```

The logarithmic model is a better fit for the data according to the AIC value. The Akaike information criterion (AIC) is an estimator of prediction error. Lower AIC value indicates better fit for the model. The AIC for the logarithmic model is 25.8177548 which is lower than AIC of the standard model 36.2676404, that is, the logarithmic model is preferred.

2.1.4 Statistical Modelling Part (d)**:

Use the logarithmic model implemented in part (c) to predict the probabilities that Covid-19 patients who receive doses of 600, 800 and 1500 mg/kg of the medicine get better.

```
new_dose <- c(600, 800, 1500)

# Indirect way to get prediction and confidence intervals
# Get the value of eta_new
eta_hat <- predict(Estimate_beta_log,
                       newdata = data.frame(Dose = new_dose),
                  se.fit = TRUE) # to get standard error
eta_hat
```

```
## $fit
##          1          2          3
## -1.5958135 -0.1997426  2.8507855
##
## $se.fit
##         1         2         3
## 0.3148959 0.1997452 0.5177794
##
## $residual.scale
## [1] 1
```

```
p_new_dose <- exp(eta_hat$fit) / (1 + exp(eta_hat$fit))

# transform decimal number into percentage
p_1 <- paste(round(p_new_dose, digits = 4)*100, "%", sep='')
p_1
```

```
## [1] "16.86%" "45.02%" "94.54%"
```

The probabilities that Covid-19 patients who receive doses of 600, 800 and 1500 mg/kg of the medicine get better are 16.86%, 45.02% and 94.54%.

Calculate the 95% confidence intervals for each prediction by indirect method.

```
eta_confin <- data.frame(upper_limit = eta_hat$fit + 2 * eta_hat$se.fit,
                         lower_limit = eta_hat$fit - 2 * eta_hat$se.fit)

# to make eta into possibility between 0 and 1
p_cofin <- data.frame(new_dose,p_new_dose, exp(eta_confin) / (1 + exp(eta_confin)))
p_cofin
```

```
##   new_dose p_new_dose upper_limit lower_limit
## 1      600  0.1685676   0.2756742  0.09747458
## 2      800  0.4502297   0.5497716  0.35451920
## 3     1500  0.9453593   0.9798924  0.85999239
```

Calculate the 95% confidence intervals for each prediction by direct method.

```
direct_p <- predict(Estimate_beta_log,
                    newdata = data.frame(Dose = new_dose),
                    type = "response",
                    se.fit = TRUE)

direct_p_confin <- data.frame(new_dose,
                              direct_p$fit,
                              upper_limit = direct_p$fit + 2 * direct_p$se.fit,
                              lower_limit = direct_p$fit - 2 * direct_p$se.fit)
direct_p_confin
```

```
##   new_dose direct_p.fit upper_limit lower_limit
## 1      600    0.1685676   0.2568345  0.08030064
## 2      800    0.4502297   0.5491127  0.35134670
## 3     1500    0.9453593   0.9988512  0.89186736
```

```
# Probability by indirect method
p_2 <- paste(round(p_cofin$p_new_dose, digits = 4)*100, "%", sep='')
p_2
```

```
## [1] "16.86%" "45.02%" "94.54%"
```

```
# Probability by direct method
p_3 <- paste(round(direct_p_confin$direct_p.fit, digits = 4)*100, "%", sep='')
p_3
```

```
## [1] "16.86%" "45.02%" "94.54%"
```

```
# indirect confidence intervals for 1500mg/kg
p_4 <- paste(round(p_cofin[3,c(3,4)], digits = 4)*100, "%", sep='')
p_4
```

```
## [1] "97.99%" "86%"
```

```r
# direct confidence intervals for 1500mg/kg
p_5 <- paste(round(direct_p_confin[3,c(3,4)], digits = 4)*100, "%", sep='')
p_5
```

```
## [1] "99.89%" "89.19%"
```

The probabilities of patients get better by receiving 600, 800 and 1500mg/kg of medicine are 16.86%, 45.02%, 94.54% separately and they are the same by both direct and indirect methods. The confidence intervals of the probabilities of patients get better by receiving 600 and 800mg/kg are almost the same by both direct and indirect methods. The confidence intervals of the probabilities of patients get better by receiving 1500mg/kg are lower by using indirect method which are 97.99% and 86%. The figures by direct method are 99.89% and 89.19%.

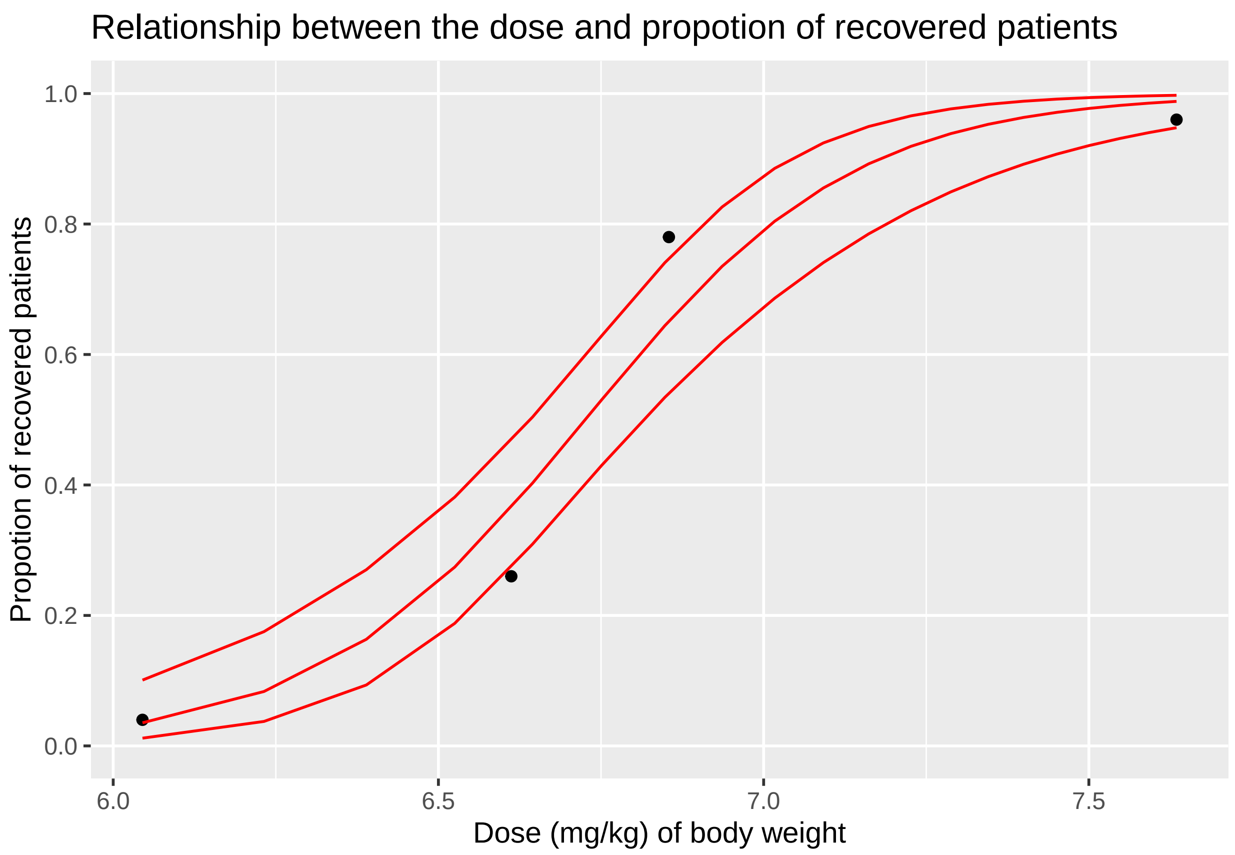
The indirect method is more recommended. It calculates the confidence intervals for $\hat{\eta}$ first and then transforms it into probabilities which prevents the upper limits from over 1. Within direct method, the result can be outside the confidence limit. In this example, the lower limit and upper limit by direct method are apparently higher than the result by using the indirect method.

2.1.5 Statistical Modelling Part (e)**:

Use the logarithmic model implemented in part (c) to produce the plot, with 95% confidence intervals obtained using the indirect method.

```r
N <- 20
Dose_seq <- seq(from = min(Dose), to = max(Dose), length = N)

# Predict eta with new doses
eta_seq <- predict(Estimate_beta_log,
                   newdata = data.frame(Dose = Dose_seq),
                   se.fit = TRUE)

# eta to probabilities
p_seq <- exp(eta_seq$fit) / (1 + exp(eta_seq$fit))

# confidence interval
eta_confin_seq <- data.frame(upper_limit = eta_seq$fit + 2 * eta_seq$se.fit,
                             lower_limit = eta_seq$fit - 2 * eta_seq$se.fit)

# to make eta into possibility between 0 and 1
p_confin_seq <- data.frame(log(Dose_seq),
                           p_seq,
                           exp(eta_confin_seq) / (1 + exp(eta_confin_seq)))
```

```r
ggplot(df_p, aes(x = log(Dose), y = Propotion_recovered)) +
  geom_point() +
  geom_line(aes(x = log.Dose_seq., y = p_seq), data = p_confin_seq,colour = "red")+
  geom_line(aes(x = log.Dose_seq., y = upper_limit), data = p_confin_seq,colour = "red")+
  geom_line(aes(x = log.Dose_seq., y = lower_limit), data = p_confin_seq,colour = "red")+
  labs(title = "Relationship between the dose and propotion of recovered patients",
       x = "Dose (mg/kg) of body weight",
       y = "Propotion of recovered patients") +
  scale_y_continuous(breaks = c(0, 0.2, 0.4, 0.6, 0.8, 1),
                     minor_breaks = NULL, limits = c(0, 1))
```



The probabilities of the patients who get better with receiving the doses of 422 and 2069mg/kg of body weight are in the range of 95% confidence intervals by the logarithmic model. The probabilities of the patients who get better with receiving the doses of 744mg/kg of body weight is lower than the lower limit of confidence interval. The probabilities of the patients who get better with receiving the doses of 948mg/kg of body weight is higher than the upper limit of confidence interval.

Perform a statistical test to check whether the logarithmic model is adequate.
State hypothesis:

Null hypothesis H_0 The model provides an adequate fit to the data.

Alternative hypothesis H_0 The model does not provides an adequate fit to the data.

```
p_6 <- pchisq(deviance(Estimate_beta_log),
              df.residual(Estimate_beta_log),
              lower = FALSE)
p_6
```

```
## [1] 0.01389397
```

The p-value is 0.013894 which is less than 0.05. Reject the H_0 hypothesis and accept hypothesis H_0 The model does not provides an adequate fit to the data.

2.1.6 Statistical Modelling Part (f)**:

A third analyst adopts the following quadratic model for the Covid-19 data:

$$y_i \sim Bin(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\eta_i = \beta_0 + \beta_1 \log(d_i) + \beta_2 [\log(d_i)]^2$$

```r
quadratic_m <- glm(cbind(Number_recovered,
                   Number_treated-Number_recovered) ~ log(Dose) + I(log(Dose)^2),
             family = binomial,
             data = df_p)
summary(quadratic_m)$coefficient
```

```
##                   Estimate Std. Error  z value   Pr(>|z|)
## (Intercept)     -96.845292  56.912698 -1.701646 0.08882166
## log(Dose)        23.729478  16.453203  1.442241 0.14923451
## I(log(Dose)^2)   -1.385234   1.188927 -1.165113 0.24397325
```

Report β_0, β_1 and β_2:

$$\beta_0 = \text{-96.8452925}$$

$$\beta_1 = 23.7294775$$

$$\beta_2 = \text{-1.3852338}$$

Fit the quadratic model in the frequentist framework:

$$y_i \sim Bin(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\eta_i = -96.8452925 + 23.7294775 \log(d_i) + -1.3852338[\log(d_i)]^2$$

Perform a frequentist hypothesis test of size 0.05 of whether β_2 is statistically significant.
Hypotheses:

$$H_0{:}\beta_2 \text{ is not statistically significant}$$

$$H_1{:}\beta_2 \text{ is statistically significant}$$

The p-value for β_2 is 0.2439732 which is lager than 0.05. We do not reject H_0, so β_2 is not statistically significant.

Report the 95% confidence interval for β_2.
State hypotheses:

$$H_0{:}\ \beta_2 = 0$$

$$H_1{:}\ \beta_2 \text{ is not equal to } 0$$

The 95% confidence interval for β_2 is -4.0288859, 0.9388293. Therefore, the 95% confidence interval for β_2 contains zero, we would accept the null hypothesis $\beta_2 = 0$. This result confirms the conclusion of the hypothesis test.

2.1.7 Statistical Modelling Part (g)∗:

Use the analysis of Deviance method to compare the logarithmic model fitted in part (c) with the quadratic model fitted in part (f).

```
# deviance of the logarithmic model
Estimate_beta_log$deviance
```

```
## [1] 8.552601
```

```
# deviance of the quadratic model
quadratic_m$deviance
```

```
## [1] 7.13771
```

```
# differences in deviances
p_7 <- Estimate_beta_log$deviance - quadratic_m$deviance
p_7
```

```
## [1] 1.414891
```

We refer the deviance of the logarithmic model as D_ω and the deviance of the quadratic model as D_Ω. So $D_\omega = 8.5526008$ and $D_\Omega = 7.1377097$.

The deviance D_ω of the small model (logarithmic model) is always larger than the deviance of the large model (quadratic model), as the large model cannot fit the data worse than the small model. Hence, in general, $D_\omega - D_\Omega$ 0. Here, $D_\omega - D_\Omega = 1.4148912$.

State hyphotheses by approximate p-value:

Null hypothesis H_0: logarithmic model D_ω is adequate compared to quadratic model D_Ω.

Alternative hypothesis H_1: logarithmic model ω is not adequate compared to quadratic model D_Ω.

```
p_8 <- anova(Estimate_beta_log, quadratic_m, test = "Chisq")["2","Pr(>Chi)"]
p_8
```

```
## [1] 0.2342461
```

The p-value is 0.2342461 which is larger than 0.05. Do not reject the H_0 hypothesis: logarithmic model is adequate compared to quadratic model.

2.2 Second Sub-Task: Bayesian Binary Logistic Regression

2.2.1 Statistical Modelling Part (h)∗∗∗:

Write jags/BUGS code to perform inference about the following Bayesian logarithmic model.

$$y_i \sim Bin(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\eta_i = \beta_0 + \beta_1 \log(d_i)$$

$$\beta_0 \sim N(0, \text{precision} = 0.0001)$$

$$\beta_1 \sim N(0, \text{precision} = 0.0001)$$

```
Baye_Bi_Lo_Re <- function(){

  for (i in 1:n_obs){

    y[i] ~ dbin(p[i], n[i])

    logit(p[i]) <- eta[i]

    eta[i] <- beta_0 + beta_1 * log(x[i])

  }

  beta_0 ~ dnorm(0.0, 1.0E-4)
  beta_1 ~ dnorm(0.0, 1.0E-4)


}
# prepare the data for the list
n_obs <- length(Dose)
y <- Number_recovered
x <- Dose
n <- Number_treated

data_list <- list("n_obs","x","y","n")

library(R2jags)

Baye_Bi_Lo_m <- jags(data = data_list,
                        parameters.to.save = c("beta_0",
                                               "beta_1",
                                               "p"),
                        n.iter = 100000,
                        n.chains = 3,
                        model.file = Baye_Bi_Lo_Re)
```

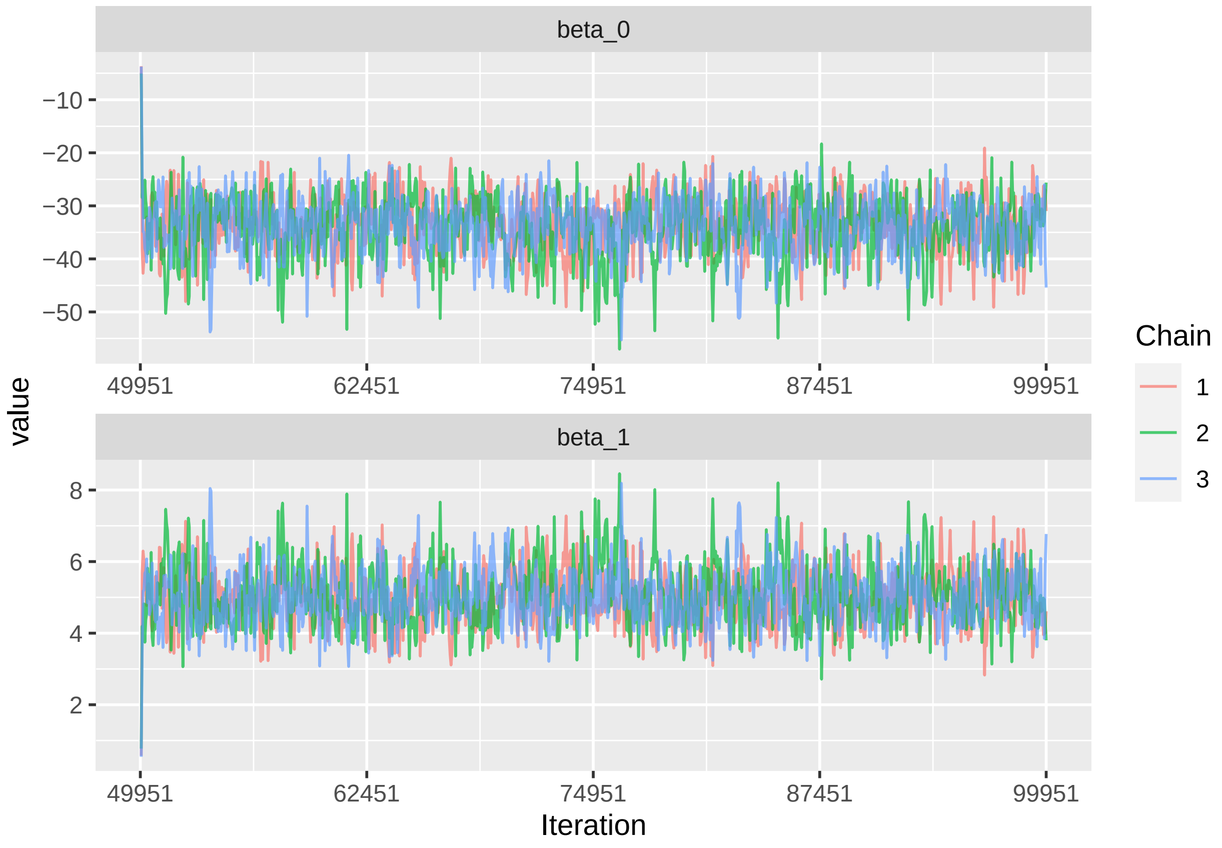
```
print(Baye_Bi_Lo_m, intervals = c(0.025, 0.5, 0.975))
```

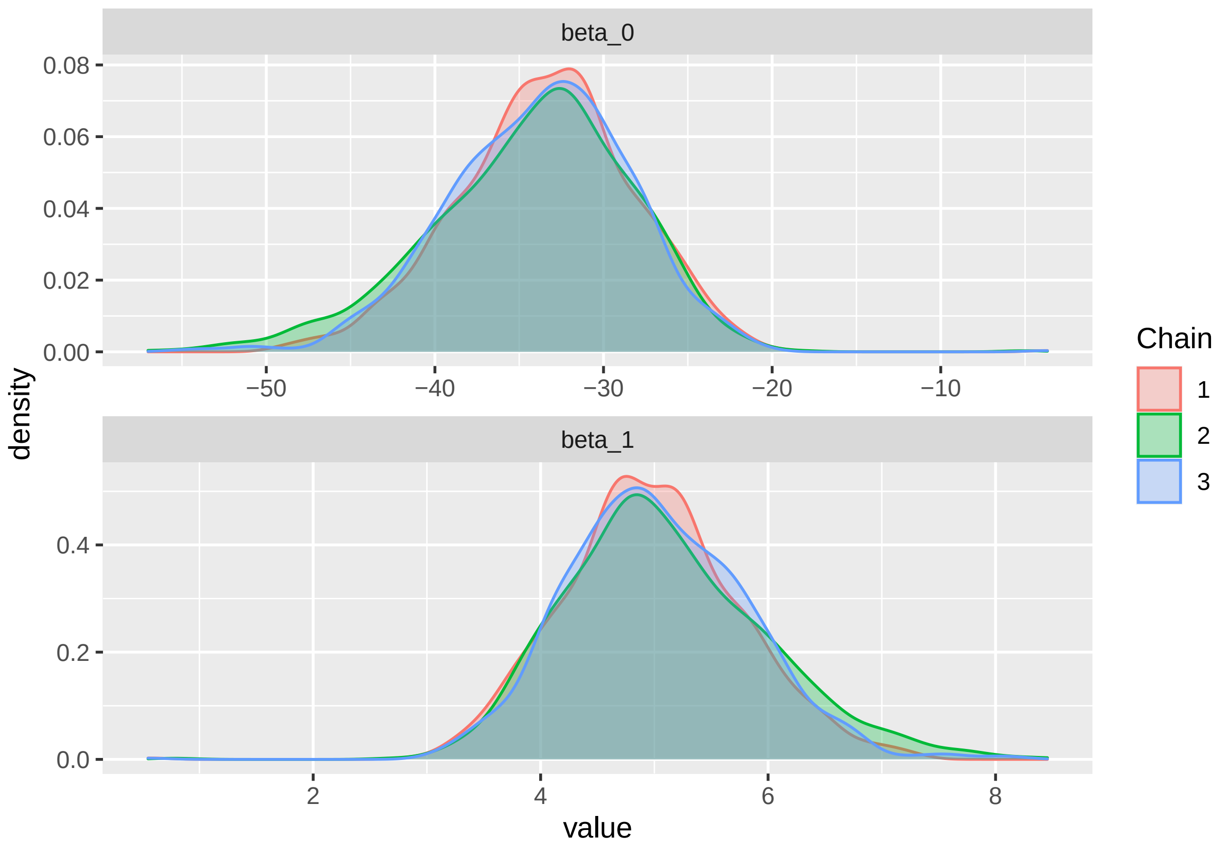
```
## Inference for Bugs model at "/var/folders/vh/sfz9s_dj40j7s7kpz2hjsk8m0000gn/T//RtmpmL2Gg1/model12ca43
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 50
##  n.sims = 3000 iterations saved
##           mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## beta_0    -33.816   5.544 -45.400 -33.387 -24.027 1.007   590
## beta_1      5.029   0.826   3.568   4.965   6.762 1.003   820
## p[1]        0.037   0.023   0.009   0.033   0.081 1.005   720
## p[2]        0.363   0.050   0.269   0.363   0.462 1.001  3000
## p[3]        0.655   0.054   0.544   0.656   0.756 1.002  1300
## p[4]        0.986   0.015   0.959   0.989   0.998 1.006  3000
## deviance   23.974   3.264  21.868  23.262  29.371 1.003   940
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 5.3 and DIC = 29.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
Baye_Bi_Lo_m$BUGSoutput$summary[c(1,2), c("2.5%", "97.5%")]
```

```
##              2.5%      97.5%
## beta_0 -45.400336 -24.026893
## beta_1   3.568174   6.762107
```

$\widehat{\beta_0}$ is -33.8156303 and $\widehat{\beta_1}$ is 5.028522 by Bayesian logarithmic model. They are both slightly higher than the results in logarithmic frequenstic model which $\widehat{\beta_0}$ is -32.6389963 and $\widehat{\beta_1}$ is 4.8528254.





The model results suggest that the chains mixed well and scatter around the mean value in traceplots. The probability density does not include 0 which means they are significant.

2.2.2 Statistical Modelling Part (i)∗:

95% confidence intervals in infrequentist logarithmic model:

```
confint(Estimate_beta_log)
```

```
## Waiting for profiling to be done...
```

```
##                  2.5 %    97.5 %
## (Intercept) -44.701647 -23.66464
## log(Dose)     3.515515   6.64896
```

95% confidence intervals in Bayesian logarithmic model:

```
Baye_Bi_Lo_m$BUGSoutput$summary[1:2, c("2.5%", "97.5%")]
```

```
##              2.5%      97.5%
## beta_0 -45.400336 -24.026893
## beta_1   3.568174   6.762107
```

95% confidence intervals of β_0 and β_1 obtained using the frequentist logarithmic model and with the corresponding 95% credible intervals obtained using the Bayesian logarithmic model are similar.

2.2.3 Statistical Modelling Part (j)**:

Using the Bayesian logarithmic model implemented in part (h), estimate approximate 95% credible intervals for for the probability that Covid-19 patients who receive doses of 550, 780 and 1900 mg/mL of the medicine get better.

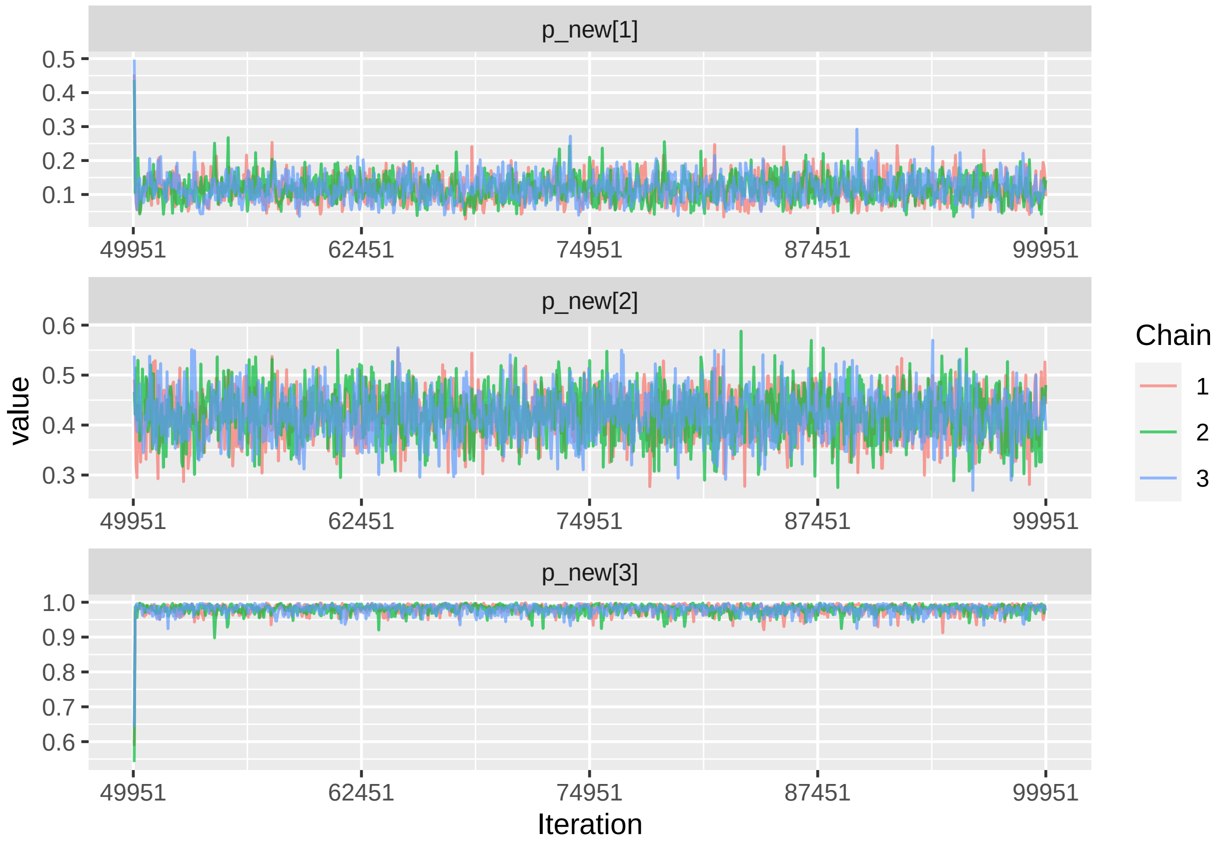
```
Baye_Bi_Lo_Re_new <- function(){

  for (i in 1:n_obs){

    y[i] ~ dbin(p[i], n[i])

    logit(p[i]) <- eta[i]

    eta[i] <- beta_0 + beta_1 * log(x[i])

  }

  beta_0 ~ dnorm(0.0, 1.0E-4)
  beta_1 ~ dnorm(0.0, 1.0E-4)


for (d in 1:n_new_dose) {

  eta_new[d] <- beta_0 + beta_1 * log(x_new[d])

  p_new[d] <- exp(eta_new[d]) / (1 + exp(eta_new[d]))
}


}

# prepare for the list
x_new <- c(550, 780, 1900)
n_new_dose <- length(x_new)

data_list_new <- list("n_obs","x","y","n","x_new","n_new_dose")

Baye_Bi_Lo_m_new <- jags(data = data_list_new,
                     parameters.to.save = c("beta_0",
                                            "beta_1",
                                            "p_new"),
                     n.iter = 100000,
                     n.chains = 3,
                     model.file = Baye_Bi_Lo_Re_new)
```

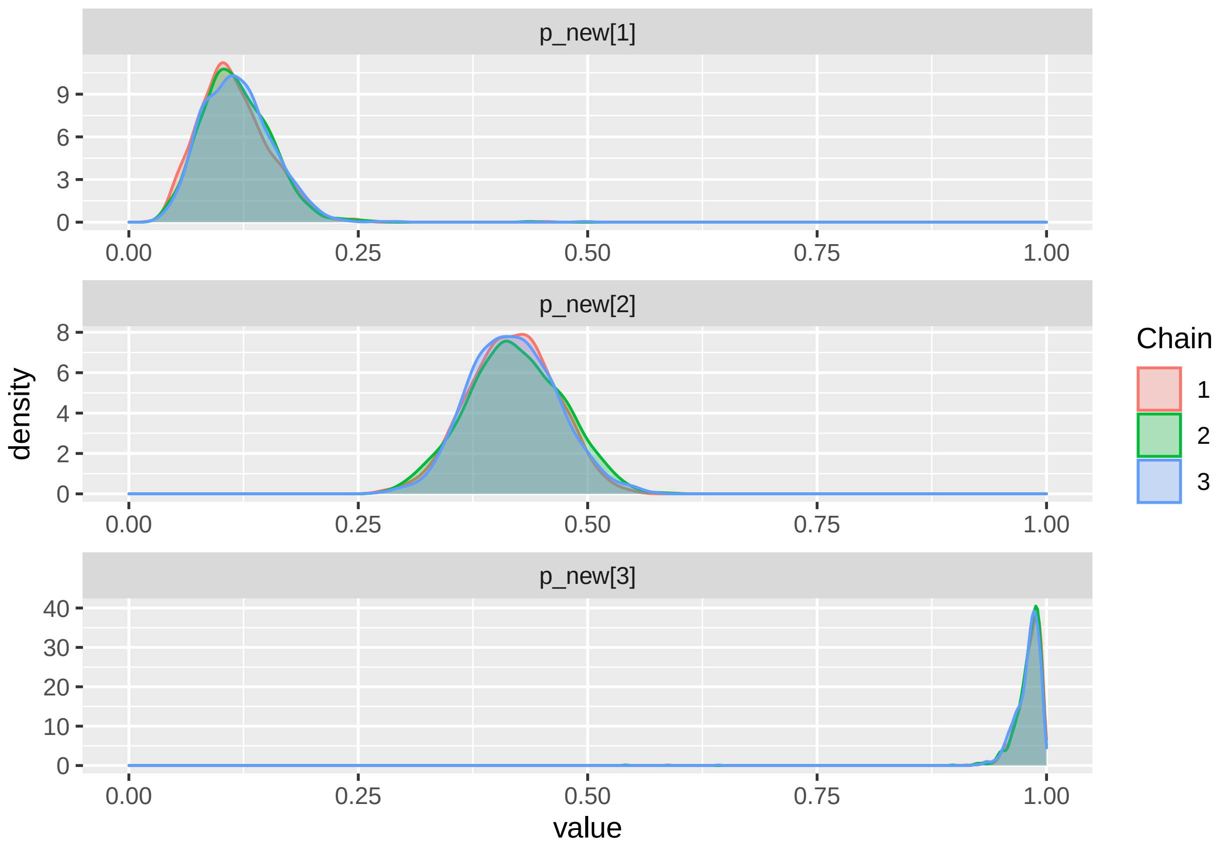
```
p_9 <- Baye_Bi_Lo_m_new$BUGSoutput$summary[4:6, c("2.5%", "97.5%")]
p_9
```

```
##                 2.5%     97.5%
## p_new[1] 0.05283768 0.1954792
## p_new[2] 0.32479440 0.5139807
## p_new[3] 0.94944826 0.9964488
```

95% credible intervals for the probability that Covid-19 patients who receive doses of 550, 780 and 1900 mg/kg of body weight of the medicine get better are 5.28% to 19.55%, 32.48% to 51.4% and 94.94% to

99.64%. It is estimated that patients who take 1900 mg/ml have more probability to get better.





The probability density does not include 0 which means they are significant. The model results suggest that the chains mix well in each case especially for the probability that patients who receive doses of 1900 mg/kg of body weight of medicine.

2.2.4 Statistical Modelling Part (k)∗∗:

Perform inference about the parameters of the following quadratic Bayesian model for the Covid-19 data.

$$y_i \sim Bin(n, p_i), i = 1, \cdots, 4, \text{ independently}$$

$$\log(\frac{p_i}{1-p_i}) = \eta_i \text{ logit link function}$$

$$\eta_i = \beta_0 + \beta_1 \log(d_i) + \beta_2[\log(d_i)]^2$$

$$\beta_0 \sim N(0, \text{precision} = 0.0001)$$

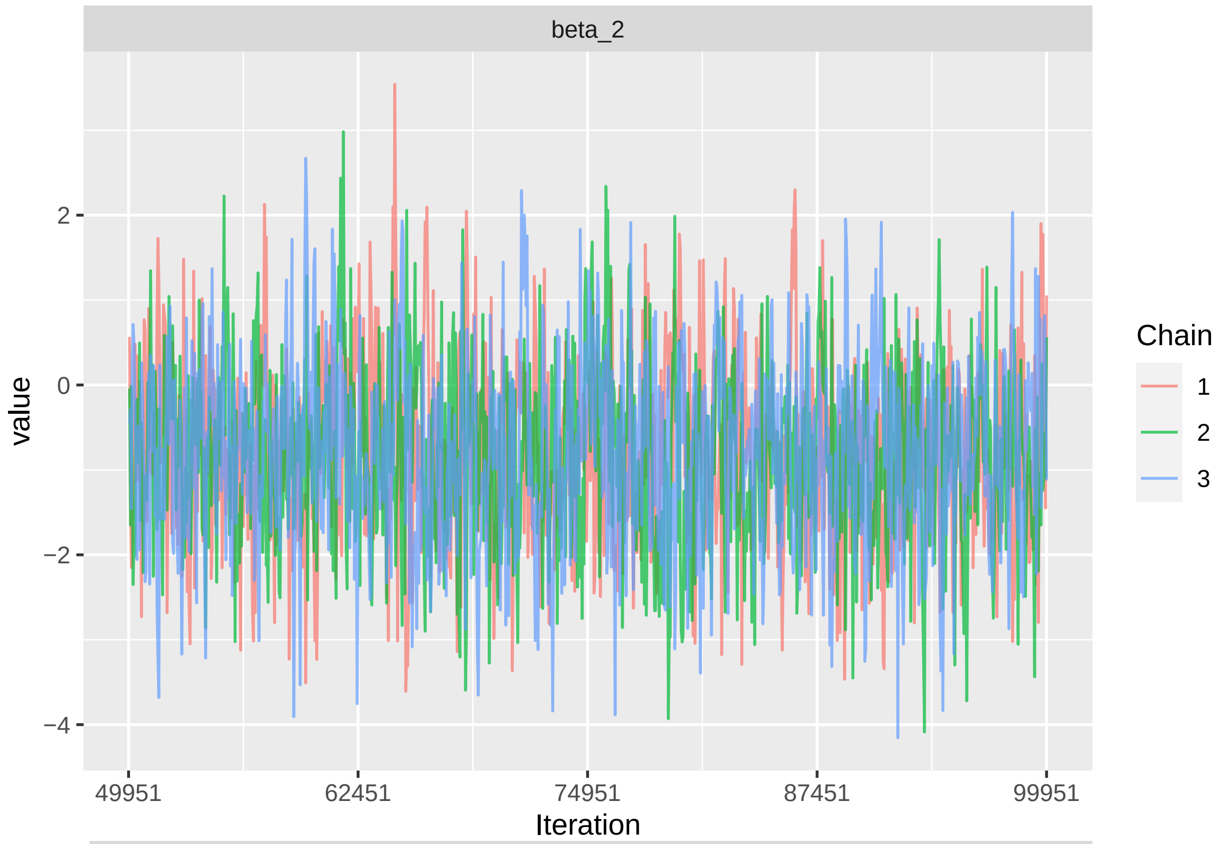
$$\beta_1 \sim N(0, \text{precision} = 0.0001)$$

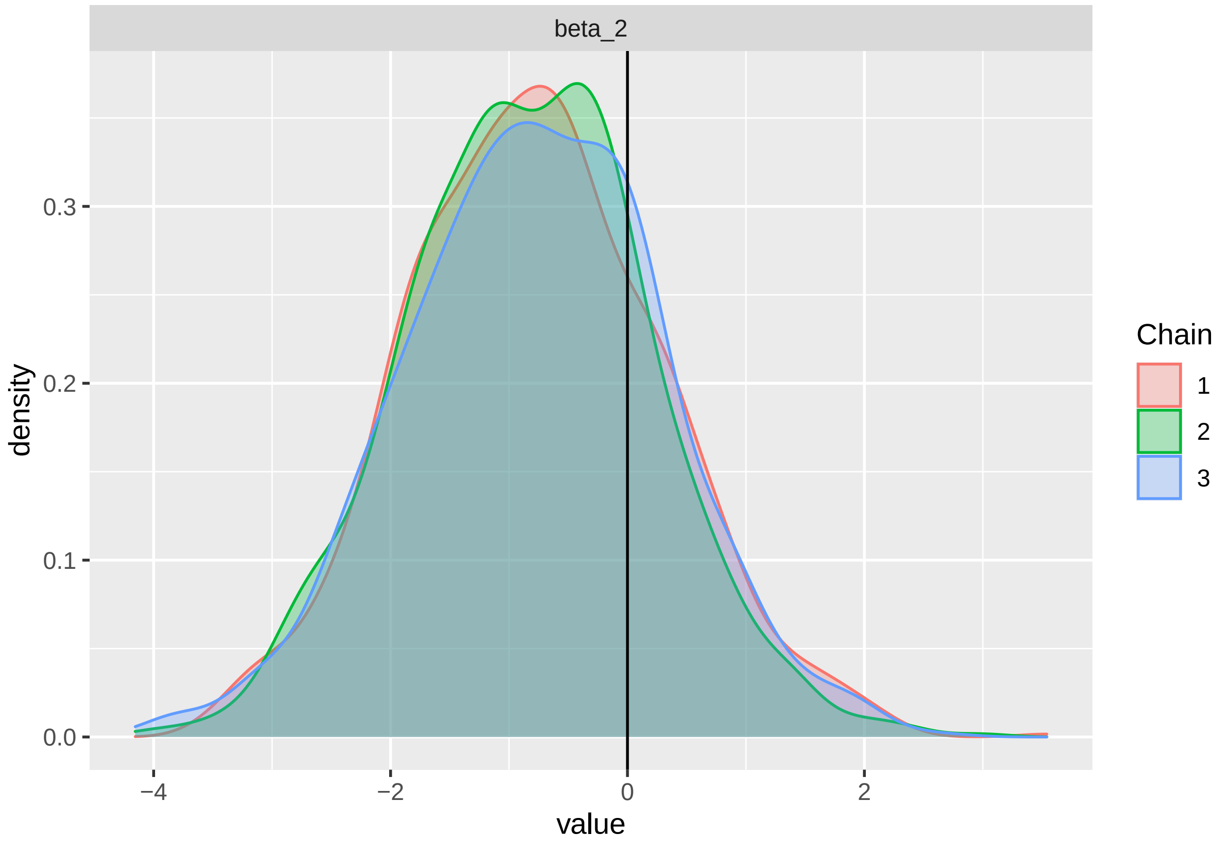
$$\beta_2 \sim N(0, \text{precision} = 0.0001)$$

```
Baye_Bi_Lo_Re_quadratic <- function(){

  for (i in 1:n_obs){

    y[i] ~ dbin(p[i], n[i])
    logit(p[i]) <- eta[i]

    eta[i] <- beta_0 + beta_1 * log(x[i]) + beta_2 * (log(x[i]))^2

  }

  beta_0 ~ dnorm(0.0, 1.0E-4)
  beta_1 ~ dnorm(0.0, 1.0E-4)
  beta_2 ~ dnorm(0.0, 1.0E-4)


}


Baye_Bi_Lo_m_quadratic <- jags(data = data_list,
                     parameters.to.save = c("beta_0",
                                            "beta_1",
                                            "beta_2",
                                            "p"),
                     n.iter = 100000,
                     n.chains = 3,
                     model.file = Baye_Bi_Lo_Re_quadratic)
```

```
print(Baye_Bi_Lo_m_quadratic, intervals = c(0.025, 0.5, 0.975))
```

```
## Inference for Bugs model at "/var/folders/vh/sfz9s_dj40j7s7kpz2hjsk8m0000gn/T//RtmpmL2Gg1/model12ca4
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 50
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect     2.5%     50%  97.5%  Rhat n.eff
## beta_0   -71.592  49.515 -171.453 -71.234 25.381 1.001  3000
## beta_1    16.058  14.508  -13.007  16.001 45.068 1.001  3000
## beta_2    -0.804   1.065   -2.884  -0.799  1.364 1.001  3000
## p[1]       0.027   0.023    0.004   0.021  0.077 1.001  3000
## p[2]       0.367   0.052    0.268   0.365  0.474 1.001  3000
## p[3]       0.670   0.052    0.566   0.672  0.766 1.001  2000
## p[4]       0.977   0.022    0.926   0.983  0.998 1.002  3000
## deviance  23.536   3.623   20.611  22.801 29.940 1.002  1400
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 6.6 and DIC = 30.1
## DIC is an estimate of expected predictive error (lower deviance is better).
```





There is considerable posterior support for values of β_2 around zero. This suggests that the quadratic term should not appear in the model. The logarithmic model in (h) is more appropriate than the quadratic model in (k).

2.2.5 Statistical Modelling Part (l)∗:

The Deviance Information Criterion or DIC means the badness of the model. The model with the lower value of DIC is more preferred.

```
Baye_Bi_Lo_m$BUGSoutput$DIC
```

```
## [1] 29.29695
```

```
Baye_Bi_Lo_m_quadratic$BUGSoutput$DIC
```

```
## [1] 30.09726
```

DIC in the logarithmic model (h) is 29.2969459 which is smaller than 30.0972573 the DIC in the quadratic model(k), so the logarithmic model is preferred.